ETHIOPIAN COFFEE LEAF DISEASE DETECTION

USING DEEP LEARNING

A Thesis Presented

By

Biniyam Yoseph Mamo

To

The Faculty of Informatics

Of

St. Mary's University

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In

Computer Science

January, 2024

ACCEPTANCE

ETHIOPIAN COFFEE LEAF DISEASE DETECTION USING DEEP LEARNING

By

Biniyam Yoseph Mamo

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

_____

Internal Examiner

Shimeles Tamiru (Ph.D.)

_____

External Examiner

Minale Ashagrie (Ph.D.))

_____

Dean, Faculty of Informatics

{Date of Defense}

Feb, 2024

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Biniyam Yoseph Mamo

_____

Signature

Addis Ababa
Ethiopia

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

Alembante Mulu (PhD)
Advisor

_____

Signature

Addis Ababa, Ethiopia
Feb, 2024

ACKNOWLEDGMENT

This thesis would not have been possible without the support of many people and some organizations. First of all, I would like to express my deepest and sincere gratitude to my advisor Dr. Alembante Mulu for the continuous support of my M.Sc. thesis and his patience and motivation. I also want to thank him for his comprehensive lectures on image augmentation which helped me solve the problem of the thesis. I would like to forward my special thanks to Dr Kifle Belachew Bekele, a plant science expert, and researcher at the Jimma Agricultural Research Center, for providing expert advice on the disease that affects coffee leaves and staff members of the organization for their kind cooperation and for giving me all the necessary information during my work. Last but not least, I want to thank everyone who has helped me over the past two years in any way they can. Above all, I would like to express my deepest gratitude to the Almighty God, for granting me the courage and unwavering support, not only during moments of joy but also through the challenges in my life and providing the strength to complete this endeavor.

To my beloved family and friends, your strong values, beliefs, and love make this thesis possible!!!

Table of Contents

List of Table

List of Figures

List of Acronyms and Abbreviations

ANN: Artificial Neural Network

BARC: Bonga Agricultural Research Center

CBD: Coffee Berry Disease

CLR: Coffee Leaf Rust

CLS: Cercospora leaf spot

CNN: Conventional Neural Network

CPU: Central Processing Unit

CWD: Coffee Welt Disease

DIP: Digital Image Processing

DL: Deep learning

DLDR: Deep learning Disease Recognition

GLCM: Gray Level Co-occurrence Matrix

JARC: Jimma Agricultural Research Center

KNN: K-Nearest Neighbors

MLP: Multi-Layer Perceptron

PLS: phoma life spot

RBF: Radial Basis Function Unit

ReLu: Rectify Linear Unit

RGB: Red, Green and Blue Color Mode

SVM: Support Vector Machines

TBDR: Texture Based Disease Recognition

ABSTRACT

Ethiopia is Africa's biggest coffee exporter, accounting for 22% of total commodity exports. Coffee is an important agricultural crop in the world economy, especially in Ethiopia. Diseases are now recognized by manual assessment by professionals based on visual inspection. However, this presents issues because expertise may not be available in all industrial regions. Additionally, researchers may have difficulty to detect disease such as Cercospora leaf spot. To solve these challenges, the goal of this study is to use digital image processing and deep learning techniques for automated detection of coffee leaf disease.

This study uses a dataset of 4000 coffee leaf images from the Jimma and Bonga agricultural Research Centers to identify specific diseases such as coffee leaf rust, Phoma Life Spot, Brown Eye Spot and healthy. The dataset allows for comprehensive training and evaluation of the Convolutional Neural Network (CNN) model, ensuring its effectiveness in accurately identifying different coffee leaf diseases. The CNN model underwent extensive training and refinement to improve its ability to detect subtle patterns and distinguish traits associated with different diseases. The proposed methodology achieved an impressive 95.3% accuracy rate, demonstrating the efficacy and dependability of the CNN-based strategy.

This achievement has enormous agricultural uses. A detailed investigation of the CNN's decision-making process provides useful insights into the key qualities required for accurate identification of coffee leaf diseases. Our findings contribute to a better knowledge of how to employ deep learning techniques in complicated contexts, showing CNNs' capacity to successfully solve issues associated with the exact and efficient identification of coffee leaf disease. In future work, we would like to recommend the model's ability to identify patterns in the leaf parts of the coffee plant, including skeletonized patterns, and citrus leaf miner with large images.

Keywords: Conventional Neural Network, Digital Image processing. Coffee Leaf Disease, Augmentation.

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the study

Ethiopia is Africa's largest coffee grower and the world's fifth-largest exporter of Arabica coffee. The nation, known as the origin of Arabica coffee, cultivates just this coffee varietal. Coffee's roots may be traced back to Ethiopia's southern highlands, notably the Buno area in the Kaffa region. Coffee production thrives in the country's hilly terrains, particularly in the Oromia and Southern areas. Notably, the Jimma, Sidamo, Yirgacheffe, Harrar, and Limu districts are known for their specialty coffees, which each have distinct features, excellent quality, and different flavor profiles. Ethiopia consumes over half of its entire coffee production, making it Africa's biggest coffee consumer.

Coffee is important in Ethiopian culture, with many people starting their day with one or two cups of coffee in the morning. The country's coffee ceremony and distinctive coffee-serving traditions emphasize its cultural significance even further. Ethiopia had an anticipated coffee yield of over 450,000 tons and a marketable supply of 334,000 metric tons in the 2012/13 agricultural year [1].

Ethiopian coffee production has increased significantly in recent years and is expected to continue growing. Forecasts for 2023/24 predict a total coffee production of 8.35 million 60 kg bags, or around 501,000 metric tons. Domestic coffee consumption in Ethiopia is expected to reach 3.5 million bags, or around 210,000 tons, in fiscal year 2022/23. The coffee business is vital to Ethiopia's economy, having a considerable effect on its socio-cultural and spiritual components. Approximately 25% of Ethiopia's population relies directly or indirectly on the coffee value chain for a living, a trend that is anticipated to continue shortly [2]. Ethiopia is the leading coffee exporter in Africa which accounts for 22% of the country’s commodity exports. Ethiopia is Africa's biggest coffee exporter, accounting for 22% of total commodity exports. According to Post Addis, Ethiopia's coffee output is expected to exceed 8.27 million 60 kg bags, or 496,200 metric tons, from October 2022 to September 2023[2].

The table below shows a constant increase trend in yearly coffee productivity.

item	2020/21	2021/22	2022/23	2023/24(Forecast)
Production (000 tons)	478	489	496	501
Area (1000 ha)	542	585	590	600
Productivity (ton/ha)	0.82	0.84	0.84	0.83

Table 1-1. Ethiopia's Coffee Productivity estimate concerning the area (MT/ha) [2].

According to the depicted Figure 1-1, the primary cultivation areas for coffee Arabica are concentrated in the forested regions of Jimma, Kaffa, Harrar, Bale, Sidamo, and Buno districts within the southwestern highlands of Ethiopia. Coffee cultivation spans more than approximately 400,000 hectares of land in the country.

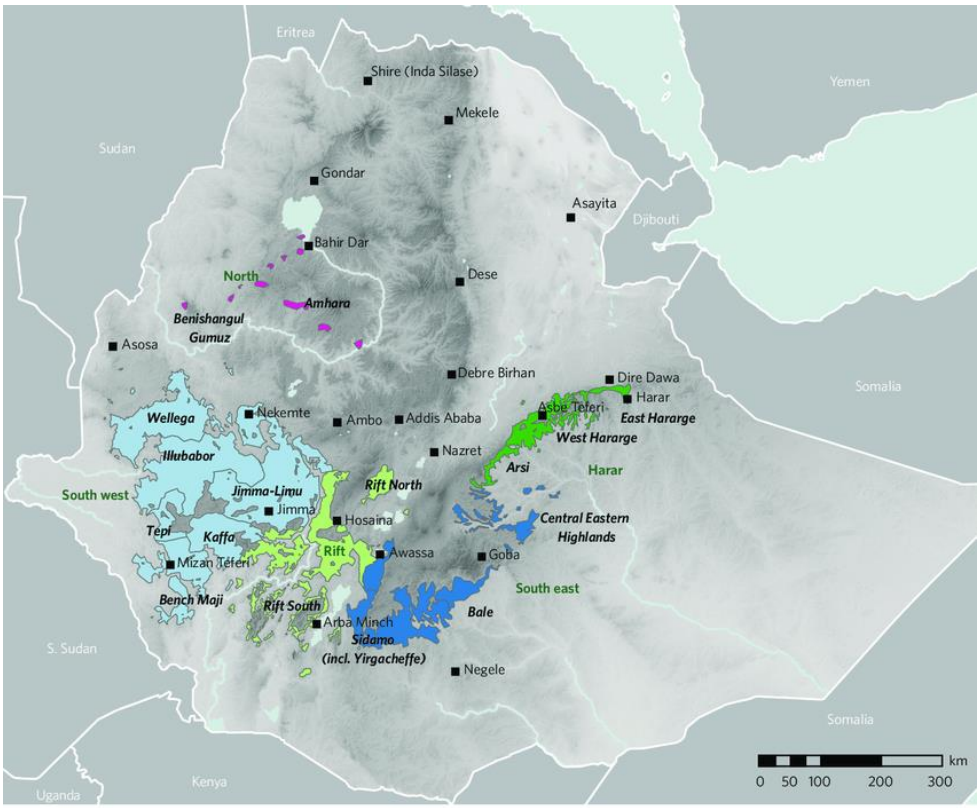


Figure 1-1. Coffee growing areas in Ethiopia [3].

Ethiopian coffee production is often measured in 60 kilogram bags rather than metric tons. Ethiopia is well-known for producing high-quality Arabica coffee, and yearly coffee output is estimated to be between 7 and 9 million 60 kilogram bags [2].

1.2. Motivation of the Study

The detection of coffee leaf diseases holds significant importance in the coffee industry, contributing to disease management, crop yield prediction, and quality control. Conventional methods, such as visual inspection by human experts, are often time-consuming, subjective, and prone to errors. The advent of deep learning techniques, specifically convolutional neural networks (CNNs), has revolutionized image detection and identification. These approaches, trained on extensive image datasets, can autonomously and accurately identify coffee leaf diseases. In the realm of coffee leaf disease detection, employing deep learning methods offers a faster, more objective, and accurate means of identifying diseases from images, thereby addressing some of the limitations associated with traditional methods.

1.3. Statement of the problem

Coffee diseases pose a significant threat to coffee production, potentially leading to up to a 57% reduction in yield [4]. Common diseases in Ethiopia include coffee berry disease (CBD), coffee wilt disease (CWD), and coffee leaf rust (CLR), caused by pathogenic organisms like fungi and bacteria. Additionally, physiological disorders like CBD and branch dieback can result in substantial yield losses. The current method of disease detection relies on visual observation by experts, which is time-consuming and prone to inaccuracies. In extreme cases, entire coffee trees are often cut down or burned to prevent disease spread [4].

Emerging technologies, such as machine learning and image recognition, offer the potential for automating coffee disease detection. Analyzing images of coffee leaves through these technologies can enhance accuracy and efficiency in disease detection, minimizing damage caused by diseases. However, the existing disease identification process in Ethiopia is iterative and time-consuming, involving repeated laboratory testing until a confident diagnosis is reached. This prolonged process, taking days or weeks, increases the likelihood of errors and inefficiencies, significantly impacting coffee production efficiency. The economic implications are profound, as the threat to foreign currency earnings from decreased coffee production is substantial.

In Ethiopian coffee production, the following diseases commonly affect the cultivation areas:

- Coffee Leaf Rust Disease (CLR)

- Phoma Life Spot (PLS)
- Brown Eye Spot Disease (BES)s
- Insects
- pests

Several researchers have conducted studies to address the topic of coffee disease, each with their own distinct perspective. H. Tadese [5] created a CNN model that can identify and categorize common coffee disease as Phoma Life Spot disease (PLS), coffee bean disease (CBD), and Brown Eye Spot (BES). Its coverage, however, did not include the major coffee leaf rust disease.

According to, Eyob Birhanu [6] concentrated on three classes of damaged coffee beans and one healthy class. Recognizing the presence of more coffee leaf disease outside the examined classes, Birhanu advised future researchers to widen the dataset by adding new Coffee Leaf Disease classes to improve model performance and use small data.

Abraham Debasu [7] assessed four classifications (ANN, KNN, Nave, and a combination of SOM and RBF) for diagnosing Ethiopian coffee disease from coffee leaves, including Coffee Leaf Rust, Coffee Berry Disease, and Coffee Wilt Disease. Their findings underscored the importance of more study and breakthroughs in detecting various forms of coffee disease in Ethiopia. They advised looking into other properties of coffee plant leaves, stems, and roots to improve identification, highlighting the need of having a larger knowledge base.

Based on the experimental results, the researchers recommend that there is a need for additional research and advancements in identifying the different types of coffee diseases in Ethiopia. They propose exploring more characteristics and features of coffee plant leaves to enhance the identification process. By expanding the knowledge base and understanding of the unique traits associated with specific diseases, more accurate and efficient identification methods can be developed.

The researchers also acknowledged a limitation in their study, where an effort was made to identify coffee wilt disease (CWD) from coffee leaf samples, despite the fact that CWD primarily affects the coffee stem. Similarly, there was an inadvertent reference to coffee berry disease (CBD) as coffee leaf disease. This underscores the crucial need for precision in

identifying and categorizing various coffee diseases to ensure alignment with the specific diseases under investigation.

The primary objective of this study is to develop a model for the detection and identification of coffee leaf diseases, utilizing deep learning and digital image processing technologies. While prior research has employed image processing techniques to assess whether coffee images are infected or healthy, this study goes a step further by constructing a model capable of identifying and detecting various coffee leaf diseases specifically from images of coffee leaves. Leveraging deep learning methods, the model aims to enhance the accuracy and effectiveness of disease identification and detection in coffee leaves.

1.4 Research Questions

This research addressed the following research questions and proposed solutions.

- What is the primary coffee leaf disease that affects coffee leaves?
- How accurate is the deep learning model in predicting occurrences of coffee disease?
- Does the implementation of CNN techniques enhance the performance of models for coffee leaf disease detection?

1.5. Objective of the study

1.5.1. General objective

The general objective of this thesis is to develop deep learning model for detecting and identifying Ethiopian coffee leaf diseases by using the CNN approach.

1.5.2. Specific Objectives

The following specific objectives will be accomplished to achieve the general objective of the study.

- To review related literature to understand the domain of research,
- To collect coffee leaf images and create a suitable dataset for training and testing the model.
- To labeled the data with experts when collecting the diseases images
- To develop a CNN model for Ethiopian coffee leaf disease.
- To evaluate the performance of the proposed model by using unseen data.

1.6. Scope and Limitation of the Work

This study focuses on using deep learning techniques to develop a robust model for detecting and identifying coffee leaf diseases common in Ethiopia. The scope of the research is only detection and identification of four types of disease infecting Arabica coffee plants in the region, coffee leaf rust, Brown Eye Spot, Phoma life spot disease, and healthy leaf identification. The research did not try to detect and identification all the diseases rather we chose the most common diseases seen on the leaves of the coffee tree.

This research focused exclusively on:-

- The diseases that only affect coffee leaves.
- The model testing and training process utilized both infected and healthy coffee leaf images as input.
- The study employed Convolutional Neural Network (CNN) architecture for the learning process and the detection of coffee diseases.

1.7. Significance of the study

This research is focused on employing deep learning techniques for the detection and identification of coffee leaf diseases. Deep learning, as a machine vision technology, offers a viable solution to this challenge. Its implementation aims to support farmers and experts in Ethiopia by enabling early detection and identification of coffee leaf diseases. This proactive approach serves to enhance the quality of coffee production, thereby positioning the country as a key player in the global coffee export market, leading to increased foreign currency earnings. By leveraging deep learning in this context, farmers can achieve higher-quality coffee yields, thus fostering economic growth and sustainability within the coffee industry.

- **Customizing a moderate model for Ethiopia**

  Deep learning models can be customized to the specific needs of Ethiopia. This means that the model can be trained on a dataset of coffee leaf images from Ethiopia, which will help to improve the accuracy of the model.

- **Support the expert and the agriculture**

DL can be used to support experts and farmers in Ethiopia. This means that the model can be used to help experts diagnose coffee leaf disease, and it can also be used to provide farmers with information about how to prevent and control the disease.

- **Reduce the expense of disease**

Deep learning can help to reduce the expense of coffee leaf disease. This is because the model can help to detect the disease early stage, which can prevent the disease from spreading and causing more damage.

- **Increase survival rate of disease**

Deep learning can help to increase the survival rate of coffee leaf disease. This is because the model can help diagnose the disease early, which allows farmers to take steps to control the disease and prevent it from killing the coffee plants.

1.8. Methodology of the Study

Methodology refers to the systematic procedure used to carry out a certain research project. The selection of proper techniques is critical to the success of the current inquiry. The methodologies and procedures used to perform the current investigation are outlined below.

1.8.1. Research Design

The research design employed in this study encompasses a comprehensive approach aimed at developing a robust model for coffee leaf disease detection and identification. Utilizing a Convolutional Neural Network (CNN) architecture, the research focuses on leveraging deep learning techniques to accurately identify various types of coffee leaf diseases. The design involves extensive data collection from diverse coffee-growing regions, Jimma and Bonga Agricultural Research Centers, the research methodology incorporates rigorous training and refinement techniques to enhance the CNN model's capability in detecting subtle patterns indicative of different diseases. Through experimental experimentation and evaluation using unseen data, the research design aims to validate the effectiveness of the proposed model in automating the detection and identification of coffee leaf diseases.

1.8.2. Data Preparation

The dataset used to detect coffee leaf diseases using a CNN technique comes from the Jimma and Bonga Agricultural Research Centers. This study used a dataset that includes four unique categories of coffee leaf disease and healthy samples, with each class including a large collection of over 2500 photos. This dataset provides a complete representation of several coffee leaf diseases, promoting variety in training, validation, and tasting sets. Notably, a fraction of these photos comes directly from the Jimma Research Center, adding real-world differences and features to the collection. 90% of the data is set aside for training, and the presence of an agricultural research center increases the dataset's authenticity and usefulness to coffee leaf disease detection tasks. The large size of each class, together with the use of agricultural imagery, provides a solid foundation for training and assessing the leaf identification model.

1.9. Organization of the Thesis

This thesis comprises into five chapters, each of which serves a specific function.

The second chapter is a comprehensive review that incorporates insights from worldwide journals, thesis papers, books, and other relevant sources. This section looks into coffee history, coffee leaf diseases, and digital picture processing. In addition, a thorough review of linked tasks is conducted to assess progress in fixing the stated problem.

Chapter Three provides an in-depth explanation of the datasets used in the research, outlines the hyperparameter ranges for the deep learning algorithms, and elucidates the features extracted and utilized in this thesis.

Chapter Four delves into the detection framework and presents the experimental findings of the proposed model for identifying Ethiopian coffee leaf disease, accompanied by thorough analysis and interpretation.

The last chapter, Chapter Five, presents a summary of results draws conclusions and recommendations from the study for users of the research, and provides future work for Ethiopian coffee disease.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1. Overview

This chapter focuses on the literature on the identification of coffee leaf diseases. It contains a thorough examination and analysis of relevant literature within the framework of the thesis, as well as a background discourse. The chapter explores the prevalence and kinds of coffee disease, providing a succinct review of the supporting facts. Furthermore, it fully explores the technologies critical in identifying coffee disorders, including digital image processing, machine learning, computer vision, and deep learning principles. This chapter also incorporates key works linked to the issue, ensuring full and unique treatment of the subject area.

2.2. Ethiopian Coffee

The coffee plant's global distribution spans several locations, with its major origin in Ethiopia, recognized as the birthplace of coffee. This country is well-known for the thriving Arabica coffee types found in various places. The agricultural sector has a huge impact on Ethiopia's economy and social structure, with around 80% to 85% of the people reliant on it for survival. Coffee cultivation accounts for approximately 40% of this demographic's agricultural sector [8].

According to legend, Kaldi, an Ethiopian goat herder, discovered the stimulating effects of coffee. The story recounts how Kaldi saw his goats displaying increased vigor and restlessness after taking berries from a certain tree, which led to the discovery of coffee's distinctive characteristics.

At heights ranging from 1000 to 2000 meters above sea level, Ethiopia provides perfect growth conditions for coffee, with coffee plants even growing wild in several locations. Sidamo, Ilubabor, Jimma, Welega, Bonga, Harerge, and the southern and western parts of Keffa all make substantial contributions to Ethiopian coffee farming. Coffee cultivation covers roughly 500,000 hectares and is mostly handled by smallholder farmers (98%) with plots smaller than a hectare in size. The remaining 2% of coffee output is from commercial plantations, both public and private sectors [9].

Ethiopia's distinct rainfall patterns contribute to the country's capacity to harvest coffee at various times of the year. The rainfall distribution is bimodal in the southern and eastern areas, but monomodal in the western region. This diversified rainfall pattern assures a year-round supply of fresh coffee.

Ethiopian farmers heavily employ external high inputs such as fertilizers, pesticide, herbicides, and fungicides in the coffee production process to increase crop yields and protect against diseases and pests. Furthermore, the rich cultural tradition surrounding Ethiopian coffee rituals, as well as the particular tastes of Ethiopian coffee, add to the country's global appeal and renown in the coffee market.

Coffee, formally known as Coffea spp., is farmed in tropical locations and includes more than 70 distinct varieties, many of which originated in Africa. Arabica (Coffea arabica) and Robusta (Coffea canephora, var. Robusta) are the two most commercially significant kinds currently, with Arabica accounting for 64% of global output and Robusta accounting for 35%. Coffee is grown on around 10.3 million hectares globally, with key producing nations including Brazil, Vietnam, Colombia, Indonesia, and Ethiopia. Coffee is one of the most profitable cash crops in the developing world, with more than 60% of these nations involved in both production and export [10].

Ethiopian coffee, technically known as Coffea arabica, is renowned for its peculiar characteristics, which include flowery, fruity, and wine-like notes. Ethiopia's many regions contribute to the diversity of flavor profiles, which are impacted by elements such as soil composition, altitude, and temperature. Ethiopian coffee varietals such as Sidamo, Yirgacheffe, Harrar, and Limu are popular among coffee fans and experts due to their bright acidity, flowery smells, and complex taste profiles.

Ethiopia is recognized by its distinctive coffee ceremony, a culturally significant event that respects the skill of coffee preparation and hospitality. The laborious procedure of roasting and grinding coffee beans is followed by ceremonial brewing and serving. It is a social gathering that promotes community and connection among participants. Ethiopia's rich coffee heritage, unique tastes, and cultural relevance continue to enchant coffee enthusiasts worldwide. Ethiopia is firmly established as a top producer and exporter of this treasured beverage due to its remarkable quality and significant contribution to the global coffee industry.

The major goal of this review is to improve our understanding of Ethiopia's coffee production and value chain system. The emphasis is on the evaluation and potential improvement of many aspects of the value chain, including production levels, local and international marketing tactics, and consumer trends. The goal of this research is to actively contribute to Ethiopia's ongoing development and sustainability in the global coffee market. Furthermore, there is opportunity for growth through the integration of technology to expedite and improve numerous areas of the coffee value chain.

2.3. Coffee Leaf Disease

2.3.1. Coffee leaf rust (CLR)

Coffee leaf rust, caused by the bio-trophic fungus *Hemileia vastatrix*, is one of the main limiting factors in Arabica coffee production and is considered the most significant pathogen of coffee plants worldwide [11]. The pathogen is an obligate parasite that infects leaves causing severe defoliation and leading to branch drought [12]. Coffee leaf rust causes not only a reduction in primary productivity in a single year but also secondary loss in subsequent years due to branch drought [13]. The pathogen requires an average of 33 days to begin the sporulation process from the deposition of spores (urediniospores) on the leaf, and 40 days for 50% of the pustules (where the spores are located) to develop [14].

Figure 2-1. Coffee leaf rust Disease.

2.3.2. Phoma Leaf Spot

Phoma is caused by the fungus that called *Phoma tarda (Stewart) Boerema & Bollen*, and poses a significant threat to coffee crops in Brazil, leading to substantial losses in both quality and productivity. This disease manifests through various symptoms such as leaf tip and branch

dieback, rosette necrosis, and to the leaf spots. To combat this issue, several control methods have been recommended. These include selecting planting sites that are less prone to cold winds, implementing windbreaks to minimize the impact of strong gusts, adopting a well-balanced fertilization approach incorporating nitrogen (N), calcium (Ca), and micronutrients, and utilizing using chemical control measures when necessary. By implementing these strategies, coffee growers can mitigate the negative effects of Phoma leaf spots and safeguard the health and yield of their crops [15].

Figure 2-2. Phoma Leaf Spot Disease.

2.3.3. Brown Eye Spot (Cercospora Leaf Spot)

Brown Eye Spot, caused by the fungus Cercospora, is a common source of coffee leaf diseases and a major problem in Arabica coffee cultivation. This pathogen is an obligate parasite that predominantly targets coffee plant leaves. The infection causes severe defoliation and frequently results in branch drought, causing in considerable global Arabica coffee yield losses. Cercospora is an obligatory parasite that needs a living host to exist and reproduce. The infection process starts with the deposition of urediniospores (spores) on coffee leaves. Following that, the pathogen takes an average of 33 days to start sporulating. Notably, 50% of typical pustules containing spores require around 40 days to fully develop [16]. Brown Eye Spot's influence goes beyond a single growing season. Aside from the acute decline in primary output, the pathogen's effect causes secondary losses in following years, mostly due to branch dryness [17]. This dual impact highlights the complexities and durability of Cercospora difficulties in the context of coffee agriculture.

Figure 2-3. Coffee Brown Eye Spot Disease.

2.4. Lists of Common Coffee Diseases and Pests

The table below provides information on common pests and diseases affecting coffee, including their respective categories, categories of symptoms, symptoms, causes, and management.

NO	Pest/Disease	Category Symptoms	Symptoms	Cause	Management
1	Coffee Leaf Rust	Fungal Disease	Yellow-orange powdery lesions on leaves	Fungus Hemileia vastatrix	Regular monitoring, fungicide application, resistant varieties
2	Coffee Berry Disease	Fungal Disease	Dark, sunken lesions on coffee berries	Fungus Colletotrichum kahawae	Proper pruning, removal of infected berries, fungicide use
3	Cercospora Leaf Spot	Fungal Disease	Circular brown spots on	Fungus Cercospora coffeicola	Good sanitation, pruning, and

			leaves		fungicide application
4	Coffee Wilt Disease	Bacterial Disease	Wilting of leaves and branches	Bacteria Xylella fastidiosa	Removal and destruction of infected plants, resistant varieties
5	Coffee Berry Borer	Insect Pest	Presence of small holes in coffee berries	Insect Hypothenemus hampei	Good farm hygiene, proper harvesting, and processing techniques
6	Aphids	Insect Pest	Curling leaves, stunted growth	Insect infestation	Insecticidal soaps, biological control methods
7	Coffee White Stem Borer	Insect Pest	Tunneling inside coffee stems, wilting branches	Insect Xylotrechus quadripes	Regular pruning, removal of infested branches, insecticide treatment
8	Coffee Root-Knot	Nematode	Stunted growth, root galls,	Nematode Meloidogyne spp.	Crop rotation, soil fumigation,

	Nematode		yellowing leaves		resistant varieties
9	Coffee Berry Moth	Insect Pest	Webbing and feeding damage on coffee berries	Insect Hypocala andremona	Proper sanitation, harvest timing, insecticide treatment
10	Coffee Black Twig Borer	Insect Pest	Drying and blackening of coffee twigs	Insect Xylosandrus compactus	Pruning, removal of infested twigs, insecticide treatment

Table 2- 1. Common coffee diseases and pests [18].

This detailed chart provides further information on the many pests and diseases that coffee plants are susceptible to. With this information and the use of appropriate management measures, coffee producers can protect their crops, assuring the continued health and production of their farms.

2.5. Digital Image Processing

A digital image is made up of a finite number of elements known as pixels, each with its own location and value. These pixels are the basic building blocks of a digital image, reflecting an important feature of its composition. Pixels, often known as the building elements of digital pictures, produce a two-dimensional representation of a visual scene by collecting attributes such as brightness or color. In essence, an image is a function that maps the properties of a scene onto an organized grid.

"Image processing" refers to the use of digital computers to edit and improve digital photographs by removing noise and abnormalities [19]. With the increased availability and use of digital computers in a variety of applications during the last few decades, the topic of digital image

processing has become especially fascinating. Its purpose is to improve visual data for human interpretation and to process picture data for storage, transmission, and machine perception [19].

Image processing methods are critical in refining raw pictures collected from cameras, sensors on airplanes, satellites, or ordinary gadgets for a variety of purposes. This field has seen considerable breakthroughs, with applications in a variety of scientific and technical sectors. Picture capture, picture pre-processing, image segmentation, feature extraction, and image detection and identification are all steps of image processing [20] [21]. In the next sections, we'll look at specific image processing techniques such picture capture, image pre-processing, and feature extraction.

The continuous development and implementation of image processing techniques has triggered a revolution in domains such as medical imaging, computer vision, remote sensing, and digital photography. These approaches make it easier to extract useful information from pictures, which helps with analysis, interpretation, and decision-making. Image processing is also important in sectors like object recognition, pattern recognition, and image-based machine learning methods. Image processing techniques are expected to evolve continuously as technology advances, opening up new possibilities and uses in a variety of sectors. The incorporation of cutting-edge concepts and methodologies drives image processing's revolutionary potential in defining the future of technological innovation even further.

2.5.1. Image Acquisition

The first stage of image processing begins with the capture of an image using devices such as cameras or scanners. Image acquisition entails using these devices to capture a digital representation of a real-world item and converting it to the required output format [20]. The picture acquisition method, on the other hand, is prone to random variations in pixel values, needing normalization to assure consistency [21]. These oscillations are caused by a variety of reasons, including air conditions, camera or object motion, scanner quality, and the distance between the camera lens and the item being photographed.

To improve the accuracy and dependability of captured pictures, it is critical to address the reasons of pixel value fluctuations. Image stabilization techniques can reduce the impact of camera or object motion while keeping pixel values. The optimal quality and calibration of the

scanner or camera aid to reduce distortions or fluctuations in pixel values. Controlling the distance of the camera lens and the itemt is critical for accurate focus and preventing loss of detail or pixel value disparities. To get the required image quality, elements like as lighting conditions and exposure settings must be considered during image acquisition [21].

2.5.2. Image Pre-processing

This step involves a procedure to transform an input image into a digital format and carry out various processes, with the main aim of obtaining an enhanced image or extracting valuable information from it. It can be considered as a form of signal processing where the input is an image, such as a video frame or photograph, and the output can be an improved image or specific characteristics associated with that image. The process of image pre-processing encompasses tasks like filtering, converting colors, and enhancing image details [21]. Image pre-processing plays a crucial role in enhancing the quality of an image, enabling the extraction of relevant features for tasks like object recognition. The primary objective of pre-processing is to suppress noise (often introduced during digitization and transmission) and eliminate distortions caused by scanning devices. Additionally, it may involve the suppression or highlighting of other attributes that are important for subsequent tasks such as segmentation, edge detection, and feature extraction [21].

2.5.3. Image Segmentation

Image segmentation is a challenging task in the field of digital image processing, as it involves dividing an image into meaningful regions or objects. It serves as a crucial step in various applications where the identification and extraction of specific objects or regions of interest are required. Segmentation can be achieved by grouping together image components that share similar characteristics, such as color, texture, or shape. The goal of segmentation is to partition an image into distinct parts or objects that exhibit consistency within themselves. This can be achieved through different algorithms and techniques. Some commonly used methods include Otsu's methodology, k-means clustering and conversion of RGB image into HIS mode, and multi-level threes holding based on pattern recognition [14] [21]. These algorithms analyze the image's properties and assign pixels or regions to different segments based on their similarities.

Segmentation can be a challenging task due to various factors such as variations in lighting conditions, complex object boundaries, occlusions, and noise. To overcome these challenges,

researchers continue to develop and refine advanced segmentation techniques that utilize sophisticated algorithms and machine-learning approaches. The importance of accurate image segmentation lies in its ability to enable further analysis and processing of specific regions or objects within an image. It plays a crucial role in applications such as medical imaging, object recognition, scene understanding, and image-based measurements. Precise segmentation results in improved object detection, tracking, and identification, contributing to the overall success and reliability of image processing tasks.

2.5.4. Feature extraction

Feature extraction plays a pivotal role in image identification by capturing the key parameters that contribute to image characterization.

Texture Extraction*:* Texture extraction involves analyzing the patterns and structures within an image. It provides valuable information about object attributes such as size, shape, thickness, and characterization. Texture component extraction is a fundamental step in gathering these features. It holds significant importance in diverse image processing applications, including remote sensing, biomedical imaging, and object-based analysis [22].

Color Extraction: Color extraction is a crucial factor in distinguishing different classes or categories in an image. Digital image processing techniques enable precise color estimations, which are particularly useful for early diagnosis in various domains. Typically, images are represented in the RGB colors space, where each pixel's color is defined by a combination of red, green, and blue values. Other color spaces, such as HIS and CIE, are extensively used in various segmentation procedures, considering their advantages and limitations. The CIE Lab color space, for instance, provides a Euclidean distance that correlates with the perceived color variation by the human visual system [23].

Shape Extraction: Shape extraction involves utilizing descriptors such as object count, shape boundaries, image dimensions, and area to characterize the shape of an image. These descriptors are crucial in extracting relevant information, such as identifying the extent of lesions or evaluating the severity of injuries. Mass analysis techniques can be employed to compute statistics for marked regions in de-noised data, including the number of objects, area, and edges [24].

Edge Detection*:* Edges in an image correspond to areas with sharp boundaries, and they significantly impact the overall image quality. Edge detection, a widely used image processing technique, focuses on identifying these boundaries by detecting discontinuities in brightness. It serves important purposes in image segmentation and information extraction in domains such as image processing, computer vision, and machine vision [24]. Hence, the precise detection and extraction of edges play a crucial role in delineating object boundaries within a scene.

In summary, feature extraction techniques, including texture, color, shape, and edge analysis, are instrumental in capturing the distinctive characteristics of images. These techniques provide crucial information for image identification and facilitate various image-processing applications.

2.6. Deep learning

DL techniques have emerged as a powerful subset of machine learning, particularly effective in handling unstructured data. Compared to conventional machine learning methods, deep learning techniques have demonstrated superior performance. Deep learning enables computational models to progressively learn features from data at various levels [25]. As access to high-performance computing has become more widespread, deep learning methods utilizing deep neural networks have become significant popularity. In the realm of unstructured data, DL exhibits exceptional strength and flexibility, owing to its ability to analyze a vast number of features. The deep learning algorithm involves passing data through multiple layers, where each layer progressively extracts features and passes them to the subsequent layers. The initial layers capture low-level features, while subsequent layers combine these features to construct a comprehensive representation [25].

2.6.1. Artificial Neural Network (ANN)

The utilization of Artificial Neural Networks (ANNs) is a fundamental approach in deep learning. ANNs are designed to mimic the learning process of the human brain, providing a brain-inspired system [26]. These networks consist of artificial neurons, which are interconnected units or nodes that loosely resemble the neurons found in the human brain. Neurons, the cells found in the central nervous system, are connected through axons and dendrites, with synapses acting as the connection points between them. Within the network, each neuron receives input data, performs mathematical calculations, and transmits the results through

synapses, represented by weights. The first Artificial Neural Network, based on this concept, was developed by psychologist Frank Rosenblatt in 1958 [27].

Artificial Neural Networks (ANNs) resemble their biological counterparts in the sense that they are composed of multiple interconnected nodes. These nodes are organized into several hidden layers, forming a network structure. Data is initially received by the input layer, then sequentially passed through one or more hidden layers, and ultimately predicted by the output layer. This sequential flow allows ANNs to process and learn from the input data, enabling them to make predictions or perform desired tasks [27].

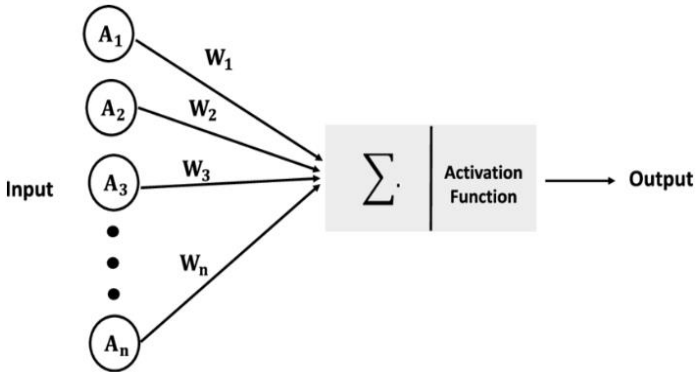


Figure 2-4. Sketch of an artificial neuron [52].

2.6.2. Multi-layer perceptron (MLP)

The multi-layer perceptron (MLP) is a fundamental type of artificial neural network (ANN) consisting of an input layer, one or more hidden layers, and an output layer [30]. Each layer is composed of a set of perceptron's, and the input layer receives the data. The hidden layers contain neurons that process specific features of the input data before passing it to the next hidden layer. The output layer receives data from the last hidden layer and produces the final output. The concept of a neuron similar to those used in MLPs was first described by Warren McCulloch and Walter Pitts in the 1940s. Perceptron's can be activated or deactivated based on a threshold value, and various values can be used to determine their activation state. Training of perceptron's can be achieved using the delta rule, which is a basic learning algorithm.

2.6.3. Activation function

The choice of activation functions in a neural network is crucial as it affects the network's learning capability. The activation function used in the hidden layer determines how effectively the model learns the training data, while the activation function in the output layer determines the type of predictions the model can make. Each node in the network receives multiple inputs and computes a weighted sum to generate a single output. The perceptron, invented by Frank Rosenblatt in 1957, exemplifies this simple yet effective model [31]. By comparing the weighted sum to a threshold, the perceptron produces a discrete output. Activation functions play a vital role in neural network implementation [32]. They serve as transformation functions that convert the input signal of a node into an output signal. Linear activation functions lack the complexity to capture intricate relationships among variables, making them less suitable for datasets with nonlinear characteristics such as images, voice recordings, and videos. Additionally, differentiability is essential for activation functions to facilitate the backpropagation optimization technique, which adjusts the weights of neurons by computing gradients of the loss function [32]. The vanishing gradient problem is a well-known challenge encountered by many activation functions. As more layers with a specific activation function are added, the gradients of the loss function tend to approach zero, impeding effective training. This issue becomes particularly problematic during backpropagation, where the gradients of each layer are multiplied down the network. A small gradient can hinder the accurate updating of weights and biases in the earlier layers, leading to a loss of accuracy as the model fails to capture essential features from the input data.

2.6.4. Convolutional Neural Network (CNN or ConvNet)

Convolutional Neural Networks (CNNs), also known as ConvNets or CNNs, are one of the most popular types of Artificial Neural Networks (ANNs) used in deep learning [33]. Convolutional Neural Networks (CNNs) are a specific type of neural network designed to process data with a grid-like structure, such as images or time series data [34]. They are inspired by the biological structure and functioning of the visual cortex, and they have proven to be highly effective in computer vision tasks like image identification and object detection [35].

CNNs utilize a multi-layered architecture with specialized layers, including convolutional layers, pooling layers (such as max pooling), and fully connected layers. These layers are responsible

for extracting relevant features from the input data and making predictions. The core concept behind CNNs is the mathematical operation of convolution, which helps in detecting patterns and features within the data.

Training a CNN involves optimizing its learnable parameters, such as the weights and kernels, using a loss function and the gradient descent optimization algorithm. Through a process called forward propagation, the model's performance is evaluated by comparing its predictions to the ground truth labels. The learnable parameters are then adjusted during the training process to minimize the loss and improve the model's accuracy [35].

Overall, CNNs have revolutionized the field of computer vision (CV) by automatically learning and extracting meaningful features from raw image data. Their ability to capture spatial relationships and hierarchical representations has made them a fundamental tool in various applications, including image recognition, object detection, and visual understanding [35].

The structure consists of three layers there are:-

- Convolution layer
- Pooling layer
- Fully connected layer

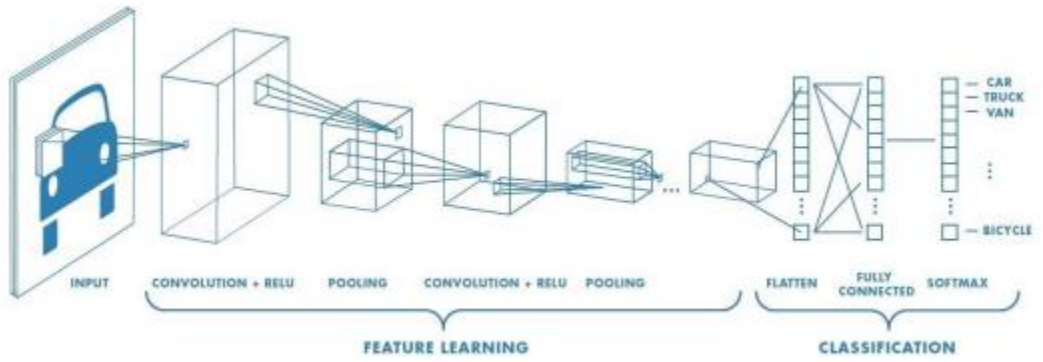


Figure 2-5. ConvNet Architecture [35].

Convolution Layer: The Convolutional Layer is a fundamental component in Convolutional Neural Networks (CNNs) used for feature extraction in image processing tasks. Images are represented as matrices of pixels, typically with three channels (Red, Green, and Blue) in the case of RGB images. Each channel is a 2D matrix, and the three matrices are stacked together.

Convolutional layers consist of a group of convolutional filters, also referred to as kernels or Feature detectors. These filters are small matrices, often with dimensions like 3x3 or 5x5, and contain learnable weights [36]. The filters are applied to the input image by sliding them across the image callculating the dot product between the filter values and corresponding the image patch. This procedure generates a feature map that accentuates particular patterns or characteristics within the image.

The primary objective of the convolutional layer is to extract meaningful and relevant features from the input image. By learning the optimal weights of the filters during the training process, the convolutional layer can capture distinctive patterns such as edges, textures, or shapes. These features are crucial for subsequent layers to make accurate predictions or identifications. In addition to RGB images, convolutional layers can also be applied to other types of data with grid-like structures, such as time series or spatial data. The convolutional operation allows the network to efficiently capture local dependencies and spatial relationships in the data.

Overall, the convolutional layer plays a vital role in CNNs by performing local feature extraction through the application of convolutional filters. By effectively learning and combining these filters, [37] [38] the network can extract informative features from images and enable high-performance tasks such as object recognition, image identification, and image segmentation.

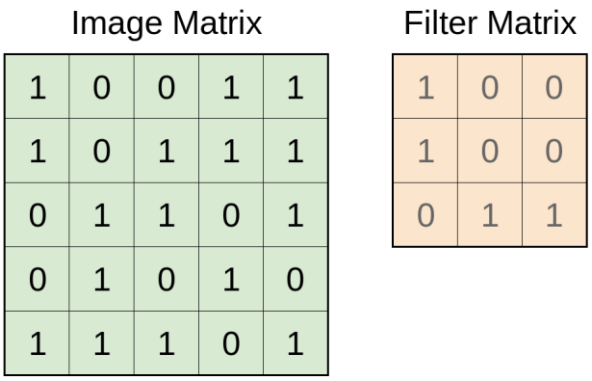


Figure 2-6. Filter matrix and input image example [51].

Pooling layer: The Pooling Layer is positioned immediately after the Convolutional Layer and serves the purpose of reducing input size by eliminating irrelevant information, thereby enhancing computational efficiency. It achieves this through a down-sampling process that reduces the dimensionality of feature maps, introducing translation invariance to accommodate

minor shifts and distortions while simultaneously reducing the number of learnable parameters for subsequent layers. Notably, pooling layers do not possess any learnable parameters, unlike convolutional layers [39]. Instead, hyper-parameters such as filter size, stride, and padding govern the pooling operations, similar to convolution operations.

One common pooling operation is max pooling, where a predefined filter size (e.g., 2x2) slides over the input feature map, selecting non-overlapping patches. Within each patch, only the maximum value is retained, discarding all other values. This down-sampling process reduces the in-plane dimensionality of the feature map based on the chosen stride value (e.g., a stride of 2 results in a down-sampled output by a factor of 2) Pooling layers offer several advantages. Firstly, they effectively reduce computational complexity by decreasing the number of parameters and operations required in subsequent layers. Secondly, they introduce translation invariance, enabling the network to recognize patterns or features irrespective of their precise locations within the input. This property proves valuable in tasks like object detection and image identification. Figure 2-7 [40] displays an example of max pooling process.

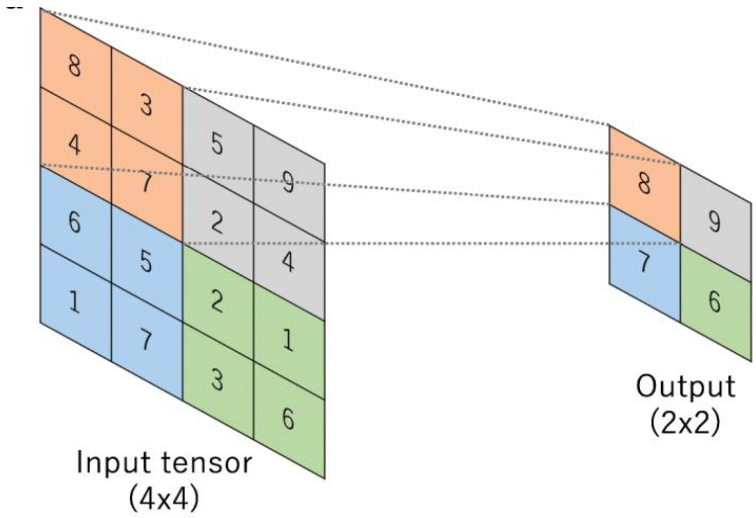


Figure 2-7. An example of max pooling operation [40].

Here is an example of a max pooling operation to illustrate its functionality: Consider a max pooling operation with a filter size of 2x2, no padding, and a stride of 2. This operation involves extracting 2x2 patches from the input tensors and outputting only the maximum value within each patch while discarding all other values. As a result, the input tensor's in-plane dimension is down-sampled by a factor of 2. In this example, the max pooling operation acts as a down-

sampling technique by reducing the spatial resolution of the feature maps. By selecting the maximum value within each patch, the pooling layer retains the most prominent features while discarding less significant information. This down-sampling process not only reduces the computational burden but also introduces translation invariance, allowing the network to recognize patterns or features regardless of their precise locations within the input.

The Pooling Layer is an essential component of ConvNet. Positioned after the Convolutional Layer, to down-sample feature maps, discard unnecessary information, and enhance computational efficiency. While pooling layers lack learnable parameters, their hyper-parameters control filter size, stride, and padding, similar to convolutional operations. This layer's role includes reducing dimensionality, introducing translation invariance, and improving the network's ability to extract meaningful features. The provided example demonstrates a max pooling operation with specific parameters, showcasing how it extracts patches, selects the maximum value, and discards the remaining values. This down-sampling process contributes to dimensionality reduction and translation invariance, both essential aspects of the pooling layer in convolutional neural networks.

There are three types of pooling operations commonly used in convolutional neural networks: Max pooling, Average pooling, and Sum pooling. Among them, Max pooling is the most widely used. In Max pooling, the output is the maximum value within the filter region. Average pooling calculates the average of all values within the filter region, while Sum pooling computes the sum of all values. Pooling operations serve two purposes: dimensionality reduction and, in the case of Max pooling, noise reduction. Max pooling retains the most prominent features by selecting the maximum value, making it effective for both dimensionality reduction and noise suppression. On the other hand, Average pooling is primarily used for dimensionality reduction, as it calculates the average value within the filter region. Sum pooling simply sums all the values within the filter region [37] [38].

Fully Connected Layer (Dense layer): In a convolutional neural network (CNN), the final pooling layer's output is flattened, meaning it is transformed into a one-dimensional array of numbers. This flattened output is then passed on to one or more fully connected layers, also known as dense layers. The fully connected layers connect every input to every output node through learnable weights.

The purpose of the fully connected layers in a CNN is to act as identifiers. They receive the features extracted by the preceding convolution and pooling layers and map them to the final outputs of the network, such as the probabilities for each class in a identification task. Typically, the number of nodes in the final fully connected layer corresponds to the number of classes in the problem. After each fully connected layer, a nonlinear function, often ReLU (Rectified Linear Unit), is applied to introduce nonlinearity and enhance the network's representational capacity.

Training a CNN is similar to training a fully connected neural network and involves the use of the backpropagation algorithm. The input data is divided into groups and processed through a series of convolutional and pooling layers before reaching the fully connected layers for identification.

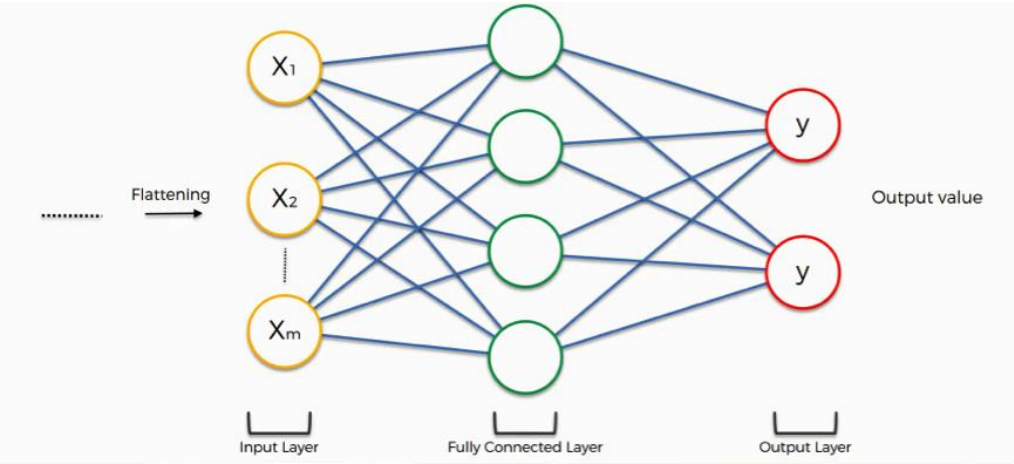


Figure 2-8. Fully Connected Layer Example [37].

2.6.5. Evolution of Convolutional Neural Network Models

Convolutional Neural Networks (CNNs) have undergone significant evolution and development since their inception. Here is an overview of the key milestones and advancements in CNN models.

LeNet-5: Introduced by Yann LeCun in 1998, [41] LeNet-5 was one of the earliest successful CNN models. It was primarily designed for handwritten digit recognition and consisted of multiple convolutional and pooling layers.

AlexNet: In 2012, AlexNet emerged as a groundbreaking CNN model. Developed by Alex Krizhevsky, it significantly outperformed traditional methods in the ImageNet Large-Scale

Visual Recognition Competition (ILSVRC). AlexNet featured a deeper architecture with multiple convolutional layers, pooling layers, and fully connected layers. It also introduced the use of ReLU activation and GPU acceleration, leading to improved accuracy and faster training.

ZFNet: Introduced in 2013, ZFNet was an improved version of CNN architecture that won the ILSVRC competition that year. It incorporated changes such as reducing filter sizes and adjusting strides to enhance feature distinctiveness and reduce the number of dead features.

VGGNet: In 2014, the VGGNet model, developed by the Visual Geometry Group (VGG) at Oxford, demonstrated the significance of network depth in CNNs. VGGNet increased the network's depth by utilizing multiple stacked convolutional layers with small 3x3 filters. This deeper architecture improved accuracy, especially in object recognition tasks.

GoogLeNet (Inception v1): Developed by Google in 2014, GoogLeNet introduced the inception module, which allowed for parallel connections of different-sized convolutional filters and pooling layers. This architecture reduced the number of parameters while maintaining or improving performance. GoogLeNet won the ILSVRC competition in 2014.

ResNet: proposed in 2015, revolutionized CNNs with the introduction of residual connections. These connections allowed information to flow directly from one layer to another, mitigating the vanishing gradient problem and enabling the training of extremely deep networks. ResNet's deep architecture with 152 layers achieved superior performance and won the ILSVRC competition in 2015.

Inception-ResNet: In 2017, the Inception-ResNet model combined the benefits of the inception module and residual connections. This hybrid model improved training speed and achieved slightly better performance than pure ResNet models.

MobileNet: is a lightweight convolutional neural network architecture optimized for mobile devices, balancing between accuracy and efficiency in image recognition tasks.

Xception: Introduced in 2017, Xception employed depth-wise separable convolutions, which separate spatial and cross-channel operations, resulting in more efficient and parameter-reduced networks. Xception performed slightly better than Inception V3 on the ImageNet dataset.

DenseNet: Introduced in 2016, DenseNet proposed a novel architecture that focused on improving feature reuse and alleviating the vanishing gradient problem. DenseNet's key innovation was the dense connectivity pattern, where each layer was connected to every other layer in a feed-forward fashion. This dense connectivity facilitated information flow and gradient propagation, leading to improved accuracy and reduced parameter count.

EfficientNet: In 2019, the EfficientNet model was proposed to address the trade-off between model size and performance. EfficientNet introduced a compound scaling method that uniformly scaled the depth, width, and resolution of the network in a balanced manner. By leveraging this scaling approach, EfficientNet achieved state-of-the-art performance on various image identification tasks while maintaining efficient resource utilization.

These are just a few notable milestones in the evolution of CNN models. Since then, researchers have continued to develop and refine CNN architectures to tackle various computer vision tasks, leading to continuous advancements in accuracy, speed, and efficiency.

Model	Top 1	Top 5 accuracy
Inception	78.8%	91.8%
MobileNet	82.4%	93.3%

Table 2- 2 Classification performance

2.7. Related works

In [42], a study on automatic coffee plant disease recognition is presented, focusing specifically on Brown Eye spot and coffee leaf rust diseases. The research explores two methodologies TBDR and DLDR. The experimental output classes include Cercospora, Rust, and healthy leaf. The statistical attribute extraction algorithms employed in the study rely on the GLCM, which records the count of adjacent pixels with identical gray levels as a designated reference pixel. The GLCM is utilized to compute texture attribute vectors, which are then fed into a neural network classifier in the TBDR approach. This fusion of statistical and local binary characteristics

contributes to the identification process. In TBDR, the local binary pattern (LBP) technique is utilized, where a 3x3 portion surrounding the central pixel is examined. LBP produces binary values that represent the pattern, and a uniform pattern is characterized by a sequence of LBP patterns with just one transition in the binary values sequence. On the other hand, DLDR directly applies DL techniques, employing a convolutional neural network (CNN) to analyze the sample photos. Performance evaluation measures such as the Kappa coefficient and sensitivity rates are computed for each experiment to facilitate comparisons.

In [43], a machine learning technique is presented for the recognition of three coffee diseases CLR, CBD, and CWD. The study utilizes a dataset consisting of 9,100 coffee leaf samples, with 6,370 samples used for model training and 2,730 samples for testing. The data is collected from various regions in Ethiopia, The research incorporates eleven features extracted from the coffee leaves, comprising five Gray Level Co-Occurrence Matrix (GLCM) features and six color features. These features serve as inputs for the identification models. Four different classifiers are evaluated in the study: Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Naïve Bayes, and a combination of Radial Basis Function (RBF) and Self-Organizing Map (SOM). In the RBF and SOM approach, RBF computation is performed, and the output is then fed into the SOM for further processing. This combined approach demonstrates superior performance compared to the other classifiers tested. The results indicate that the RBF and SOM algorithm achieves an impressive accuracy rate of 90%.

In [44], the study addresses coffee plant disease identification using texture features. The methodology includes the extraction of features using Gray Level Co-Occurrence Matrix (GLCM) and identification using Support Vector Machines (SVM). The reported accuracy is 87%. The research gap may lie in exploring alternative texture feature extraction methods for improved identification performance.

In [45], the paper presents automated coffee plant disease detection and identification. The methodology involves using color and texture features combined with a Convolutional Neural Network (CNN). The achieved accuracy is reported as 91.7%. A research gap in this study may be comparing the performance of the proposed approach with other existing methods for detection and identification.

In [46], conducted by Arivazhagan and Ligi (2018), focused on the automated detection of mango leaf diseases using a deep learning technique. In this study, CNN was used for feature extraction and identification. The scientists reported an impressive overall accuracy of 96.67% in diagnosing mango leaf diseases. These researches demonstrate the usefulness of deep learning techniques, particularly transfer learning and CNNs, in accurately and automatically identifying plant diseases, highlighting its potential for increasing precision agricultural operations.

In [47], Sorte et al. (2019) developed an automated approach for detecting coffee plant diseases, specifically Cercospora and coffee leaf rust. They employed texture attribute extraction techniques, especially GLCM and LBP, to generate texture vectors for a group of photographs used as inputs to a feed-forward neural network. The study sought to evaluate the efficacy of these texture extraction methods to a Convolutional Neural Network (CNN) that did not use texture properties. The CNN methodology outperformed texture extraction approaches, especially with a large training dataset, highlighting CNNs' better efficacy in disease detection tasks, particularly in circumstances with a lot of training data.

In [48], Jadhav et al. (2020) introduced an automated method for soybean disease diagnosis that combined transfer learning approaches with pre-trained deep learning models, notably AlexNet and GoogleNet, for identification. To improve identification performance, the authors significantly modified GoogleNet's last three layers and altered critical convolutional neural network (CNN) parameters such as learning rate, number of epochs, and iterations. The classifier's performance was evaluated using a fivefold-cross validation approach, which yielded excellent accuracy of 98.75% and 96.25% for AlexNet and GoogleNet, respectively.

NO	**Problem statement**	**Methodology**	**Results**	**Research Gap**
[42]	Automatic coffee disease recognition	Text-Based TBDR, Deep Learning Disease Recognition	Kappa coefficient, sensitivity rates	Focuses only two namely Cercospora and Rust with only two approaches.

[43]	Coffee disease recognition using classifiers	Combination of RBF and SOM	90% accuracy	Identify CWD from coffee leaf, while it affects coffee stem not the leaf.
[44]	Coffee plant disease identification using texture features	RBF and SOM	90.07%accuracy	Exploration of different texture feature extraction methods
[45]	Automated coffee plant disease detection and identification	Color and texture features, CNN	91.7% accuracy	Comparing the performance with other detection and identification methods
[46]	automated detection of mango leaf diseases	CNN model	96% accuracy	Use a limited data for detecting for Five different leaf diseases
[47]	Coffee leaf disease recognition based on deep learning and texture attributes	GLCM and LBP	97% accuracy	They focus on the 2 coffee leaf brown eye and leaf rust the coffee leaf there are more leaf dissases

[48]	automated soybean disease detection	combined transfer learning approach AlexNet and GoogleNet	96% and 98% accuracy	They Use a small data for detecting for three different leaf diseases and they see only 3 leaf disease and they Not including other part of soya bean diseases

Table 2- 3. Related works Summery.

CHAPTER THREE

3. METHODOLOGY AND ARCHITECTURE

3.1. Introduction

In this chapter, we will elucidate the methodology and architecture employed in my project, shedding light on the techniques and data sources utilized. The data collection strategy adopted a combination of primary and secondary data to ensure a comprehensive dataset for the study. I used a deep learning approach, namely a Convolutional Neural Network (CNN), to create the model.

This CNN toe is critical for efficiently training the model for the coffee leaf disease detection challenge. The software and hardware setup of the system are critical factors to consider. This section describes the operating system, programming language, machine learning library, and computer hardware, offering insight into the technological foundation utilized for model training and assessment.

The coffee leaf disease detection architecture follows a conventional machine-learning approach designed for picture detection problems. During the first training phase, a wide set of photos for detection training is collected. Following that, preparatory procedures such as cleaning and resizing are carried out on the gathered data. The dataset is then divided into training and test sets, and additional photos are created artificially to supplement the training dataset. Finally, significant characteristics from the photos are retrieved to improve the detection process. In the testing step, the retrieved characteristics are used to detect new photos. This two-step process guarantees that the model is not only efficiently trained but also capable of appropriately detecting unknown data based on the learned characteristics.

In this thesis, using a selection of photos from the dataset, an experimental research technique is used to investigate the identification of coffee leaf diseases. Several controlled tests are carried out in the study to examine the performance of a convolutional neural network (convNet) in diagnosing coffee leaf diseases. Using an experimental technique, the dataset is purposefully partitioned into training, validation, and test sets to rigorously analyze the CNN's performance. Stratified sampling enables a balanced representation of several disease classes, each folder of

This is limited to a minimum of 2000 photos. The following stage goes into important data preparation stages, experimenting with several approaches to preprocess raw coffee leaf photos before they are fed into the neural network.

3.2. The proposed architecture

The system design approach begins with the creation of training and validation data, with experts categorizing disease pictures throughout the gathering phase. Following that, the raw data passes through a critical data preparation stage in which photos are normalized, cropped, and identified according to their particular disease. The preprocessed photos, each measuring 256x256x3, vividly highlight the region of focus while capturing key details.

The CNN stacked layers are critical in extracting essential characteristics from these preprocessed pictures, which will serve as the foundation for categorization throughout the model development phase. Throughout the training phase, the model is focused on extracting characteristics unique to leaf diseases. The identifier, using the retrieved characteristics, distinguishes between benign and sick leaves.

During the training phase, the model's performance is evaluated with the validation dataset. In essence, when the model is trained, its capacity to recognize disease patterns is constantly evaluated using the validation set. To attain optimal performance, numerous models are tested, including those generated from scratch and those that have been pre-trained. The most effective model is chosen based on its performance indicators after a thorough review.

Moving on to the testing step, the selected top-performing model is evaluated using unseen (new) and unlabeled data. The model makes class predictions, indicating the likelihood that the picture belongs to one of the predefined classes created during the training phase. This extensive system architecture guarantees that the model is both resilient and effective in properly identifying coffee leaf diseases.

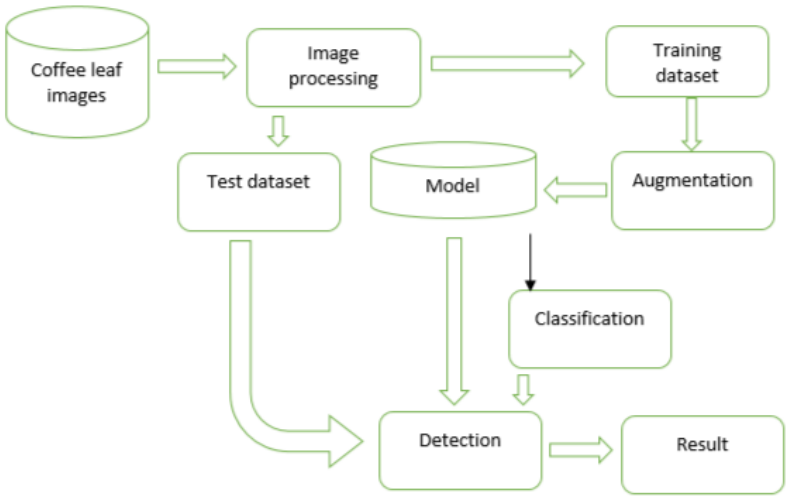


Figure 3-1. The Proposed Architecture.

3.3. Research design

To achieve the aims mentioned in this thesis, this study employs an experimental research approach. Experimental research is a scientific technique that aims to establish a link between two fundamental variables: independent and dependent variables. The primary goal is to look into the relationship between a certain property of an object and the variable under consideration. The purpose of this thesis is to determine the presence or absence of disease in mammography pictures.

The three essential processes in the research process. Initially, the domain of the problem is determined, with an exhaustive literature review aiding comprehension. Following that, the thesis's broad and particular objectives are defined. The first part of thesis design includes data preparation, whereas the second phase includes both data preparation and overall thesis design. a collection of coffee leaf photos is methodically collected from the Jimma and bonga research facilities throughout the data preparation phase. This dataset is then divided into three parts: training, validation, and testing. Following data preparation, the model design phase begins, during which the model's architecture and specifications are developed to meet the study's objectives. The third part entails putting the thesis into action. The defined model is applied at this step using appropriate tools and approaches. The model is trained and tested using relevant data, and its performance is regularly reviewed during the training process to ensure it predicts

properly. The model's efficacy is determined during the assessment phase, and it is then evaluated with new data to confirm its predictive capabilities. This comprehensive technique guarantees that the created model's performance is thoroughly examined, confirming its capacity to provide correct forecasts.

3.4. Software Tools

A detailed analysis of existing tools and their libraries was undertaken in the discovery of potential software tools for implementing the CNN algorithm for inset picture identification. The investigation indicated a discrepancy between tools that support both deep learning and machine learning algorithms and those that focus just on one of them. Several factors were reviewed throughout the tool selection process to facilitate the selection of acceptable software tools and libraries. The programming language used for algorithm implementation was the most important of these factors. Furthermore, aspects such as the availability of learning tools, such as free video lessons and prior expertise, as well as compatibility with computers with restricted resources (e.g., CPU-only settings), were considered.

Python was chosen as the programming language for implementing the CNN algorithm in the Anaconda environment, with the TensorFlow and Keras modules. Anaconda, a free and open-source Python and R distribution, focuses on package management and deployment. It offers several integrated development environments (IDEs) for code authoring, with Jupyter Notebook being used for coding due to its user-friendliness and web browser capability.

Tensor Flow, a free and open-source deep learning framework built by Google, emerged as the dominant deep learning library for this study. Tensor Flow, known for its speed and adaptability, works with cloud-based services, mobile devices (including iOS and Android), and desktop PCs running Windows, macOS, or Linux. The design of the library is excellent in managing data preparation, model creation, training, and estimation. Tensors are used for data representation in calculations, and a graph structure is used for graphical representation during training. It provides CPU and GPU distributions. But we use only CPU.

Keras, free and open-source Python, and TensorFlow in data science and machine learning applications. Jupyter Notebook was used for coding again as part of the toolset, delivering a user-friendly experience within a web browser. These chosen tools together match the given

criteria, coinciding with Python's familiarity and allowing for the smooth implementation of the CNN algorithm for inset picture identification.

3.5. Data Partitioning

The dataset is divided into two unique pieces during the first data preparation phase: a training set and a test set. The training set is used to train the model, whereas the test set is left unchanged throughout training and is only used to evaluate the model's performance after training. To test the model's competency throughout the training phase and fine-tune parameters for best performance, an extra validation split is implemented. Surprisingly, a portion of the test data equal to 10% is set aside for validation.

It's worth noting that the collection includes 4000 distinct photos obtained at Jimma and Bonga research facilities. And we augmented the data in to 4000 photos to 8000 and in total we have 12000 data. The literature proposes a wide range for the training split, ranging from 60% to 90% of the whole dataset, with the remaining 20% put aside for test. In this work, an experimental strategy is used using a split ratio of 9:1, which deviates somewhat from the standard ratios. The decision to keep an equal number of photos in each class for testing and training prevents overfitting while also assuring impartial weight updates throughout training, resulting in a more balanced and robust model.

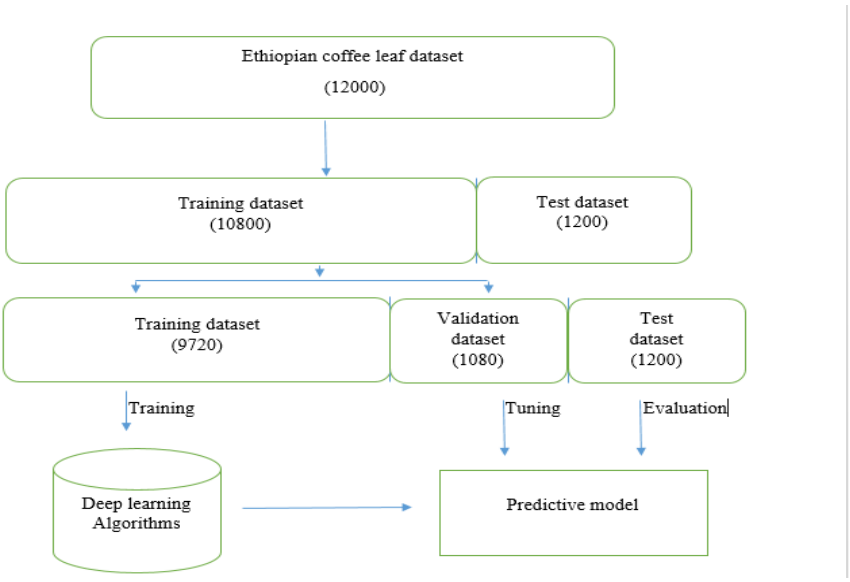


Figure 3-2. Data Partitioning.

3.6. Data Augmentation

Large research datasets, especially those utilized in picture detection and deep learning applications, sometimes need considerable data augmentation approaches. Data augmentation is an important procedure for increasing the amount of training data points in a dataset by creating extra data from current training samples [49]. Even though the data collection is currently large, increasing the number of data points is critical. This method helps the neural network extract more complex characteristics from the data while addressing the issue of overfitting [50].

The annotated dataset was incorporated into the Rob flow workspace in this study, we collect data from JARC and BARC over 4000 images and augmented to 8000 images totally we have 1200images, where automatic augmentation techniques such as 40-degree rotation and zoom_range flipping were used. These strategies were designed to diversity the dataset, which is critical for Convolutional Neural Networks (CNNs) with many parameters. The importance of a large dataset is highlighted in order to avoid overfitting, a problem explored in the literature.

The study uses augmentation to boost the quantity of photos. To augment data, we use an image data generator. And the photos supplemented with preprocessed data.

Image Data Generator	**values**
Augmentation techniques	
• Rotation range	40
• Width shift range	0.2
• High shift range	0.2
• Sheer range	0.2
• Zoom range	0.2
• Horizontal flip	True
• Fill mode	nearest

3.6.1. Data splitting after augmented images for the Proposed Model

Image Data Generator Augmentation methods are used in the investigation. These strategies were employed to enrich our dataset, to counteract overfitting, a train-test split technique was

used, with 90% of the data allocated for training and 10% for testing. The training dataset was used for model creation, whereas the test dataset was used for model assessment.

This research employed a 90% training and 10% testing dataset split, along with the introduction of a validation dataset to address potential of overfitting issues. Subsequently, 90% of the initial training data was subdivided into 80% for training and 10% for validation. As a result, the final dataset distribution consisted of 80% for training, 10% for validation, and 10% for testing, facilitating robust model evaluation.

3.7. Model Selection

Data augmentation plays a crucial role in preventing overfitting in deep learning models. This technique involves manipulating existing training images (scaling, rotating, flipping,) to create virtually new examples without altering their semantic meaning. For instance, rotating a coffee leaf disease image doesn't change the fact that it depicts a diseased leaf. These augmented images effectively increase the training dataset size, promoting the model's ability to generalize better to unseen (new) data. This section dives deeper into various geometric transformations and their application in data augmentation while preserving the semantic integrity of the labels.

- **Flipping**, while a common data augmentation technique, requires careful consideration of its label-preserving nature. While horizontally flipping images often maintains semantic meaning (e.g., a flipped image of a plant is still a plant), vertical flipping can potentially alter the label in certain datasets. For instance, vertically flipping a picture of a coffee leaf with disease symptoms on its upper side could incorrectly suggest the disease is present on the underside, leading to misinterpretation. Therefore, the suitability of flipping, particularly vertical flipping, should be evaluated based on the specific dataset and its labeling characteristics.
- **Zoom Range,** 0.2 specifies that photos can be arbitrarily zoomed in or out by up to 20% during data augmentation. This implies that the model will be exposed to different picture size and detail changes, allowing it to learn features more robustly and avoid remembering specific pixel patterns. To put it another way, consider photographing a dog and then digitally zooming in to show more of its face or zooming out to show more of its body. The model better learns the dog's overall form and characteristics by viewing it from several "zooms," independent of the specific viewpoint in a fresh photograph.

- **Shear Range,** of 0.2 in data augmentation indicates that pictures might be slightly distorted horizontally or vertically (up to 20%). Consider tilting a picture frame at an angle. This distortion aids the model in learning features over a wide range of viewing angles and viewpoints.
- **Horizontal flip,** is set to True, the original photos will be mirrored horizontally with a 50% chance during data augmentation. This implies that the model will view a reversed version of the picture alongside the original half of the time. This approach diversifies the training data by generating fresh "views" of existing photos that do not change the semantic meaning of the labels.

3.8. Training Components of the Proposed Model

To solve the unique challenge of coffee leaf disease detection, a new Convolutional Neural Network (CNN) model is created from scratch. The collection contains four unique classes of photos of healthy and diseased coffee leaves.

The model's architecture comprises a series of convolutional layers succeeded by pooling layers. Convolutional layers detect spatial patterns within input images, while pooling layers help reduce spatial dimensions and control computational complexity.

The structure consists of multiple convolutional layers with rectified linear unit (ReLU) activation functions. To down sample the spatial dimensions while retaining critical information, these convolutional layers are alternated with max-pooling layers. This step is repeated for clarity, ensuring that the model learns hierarchical representations properly.

Following the convolutional and pooling layers, two fully connected layers with the ReLU activation function are implemented. These layers aid in the discovery of intricate linkages in data. Finally, the output layer makes use of a sigmoid activation function, which is appropriate for binary identification issues.

A second testing set is also employed to evaluate the model's performance on unknown data. To maximize the model's performance, hyperparameters like the quantity of filters within the convolutional layers, the dimensions of the filters, and the count of neurons in the fully connected layers are explored with different values. This iterative method allows the model's

architecture to be fine-tuned in order to get improved outcomes in recognizing different picture classes linked to coffee leaf disease.

Input layer: is designed to handle RGB pictures with 256x256 pixel dimensions representing four separate classes: 'Cerscospora,' 'Healthy,' 'Leaf rust,' and ' Phoma life spot' This layer, specified by input_shape = (256, 256, 3), acts as the first point of picture input, simplifying the following feature extraction procedure. It is important to note that it does not add any learnable parameters, such as weights or biases, and instead works as a conduit, effortlessly passing the input data to the first convolutional layer. During the model training phase, its major job is to set the scene for the convolutional and pooling layers to distinguish and learn hierarchical features from the input pictures.

Convolutional layers: There are three layers in our scratch model (Conv2D_1, Conv2D_2, and Conv2D_3). The input picture has dimensions of (None, 256, 256, 3), and each convolutional layer conducts operations on the input, gradually extracting features. Each convolutional layer's output shapes and number of parameters as follows:

1. Conv2D_1: Output shape ((None, 254, 254, 32)), Parameters: 896
2. Conv2D_2: Output shape (None, 125, 125, 64), Parameters: 18,496
3. Conv2D_3: Output shape (None, 60, 60, 128), Parameters: 73,856

```
# Create a Sequential model
model = Sequential()

# Add Convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten layer
model.add(Flatten())

# Dense layers
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))

model.summary()
```

Figure 3-3. CNN model summary.

As Figure 3-3 shows in our model, these convolutional layers use the default values of 32 filters, 3x3 kernel size, and 1 stride and dilation rate. The formula (Image Height - Kernel Height + Stride, Image Width - Kernel Width + Stride, Filter) is used to generate the output forms, which results in the required dimensions for each layer. The formula (Kernel Height * Kernel Width * Input Channels * Output Channels + Output Channels if bias is used) determines the parameter count for each Conv2D layer, and the overall number of the parameters is the sum of these counts, producing 896 parameters for Conv2D_1.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 14,839,492
Trainable params: 14,839,492
Non-trainable params: 0
```

Figure 3-4. CNN model.

Pooling layer for spatial reduction, are appropriately positioned after each of the three convolutional layers. Following the initial convolutional layer, conv2d, comes a max-pooling layer, max_pooling2d, which employs a 3x3 filter with a stride of one. This procedure reduces the output size of the relevant convolutional layer significantly. The output of the second convolutional layer, conv2d_1, is then routed into the second max-pooling layer,

max_pooling2d_1, which employs 2x2 filters with a stride of 1. The third convolutional layer, conv2d_2, is followed by the third max-pooling layer, max_pooling2d_2, to which employs two-stride 2x2 filters. These max-pooling layers contribute to spatial down sampling along their respective input volume spatial axes. It is important to note that these max-pooling layers are just intended for spatial reduction and do not provide any extra learnable properties into the overall model architecture.

Fully Connected (FC) layer: There are three levels in total: dense, dense_1, and output. The dense layer, the first fully connected layer, comprises 128 neurons and can capture detailed patterns and characteristics from the flattened vector created after the convolutional and pooling layers. Following that, the second completely linked layer, dense_1, consists of four neurons and serves as the last layer for multi-class categorization. The output layer, represented by dense_1, is the last layer and comprises of a single neuron for each unique class in the identification job. This design improves the network's capacity to learn and extract relevant information, resulting in improved identification performance.

Output layer is the dense_1 layer depicts the third completely linked layer. This layer is made up of a single neuron that uses the softmax activation function. This output layer's major function is to categorize the input into many classes, in accordance with the overall model architecture created for categorizing photos into separate and preset categories. The softmax activation function guarantees that the model assigns probabilities to each class, allowing for a more complete and probabilistic approach to categorization. This model setting improves its capacity to produce anuanced and accurate predictions across all classes given.

3.9. Hyperparameter Adjustment

Hyperparameters, which are specified settings independent of the deep learning method, play an important part in our code. These values are established prior to the start of the training procedure. There is no defined method for identifying the best hyperparameters for a given issue, unlike the core algorithm. As a result, significant testing is carried out to fine-tune these hyperparameters. The next paragraphs go over the hyperparameters we picked for our model and discuss them in the paragraphs that follow.

The optimization method, especially the Adam optimizer with a learning rate of 0.001, is critical in neural network training. During training, the model's weights are iteratively updated via backpropagation of mistakes. Adam, a popular optimization technique, is evaluated across several learning rates (0.001 in this case) to improve learning efficiency and reduce mistakes. Gradient descent is still a fundamental tactic in deep learning, and the selected framework, Keras, includes such optimization algorithms. Based on the computed gradients and the loss function, the Adam optimizer dynamically modifies parameters and updates weights. This repeated procedure improves the model's capacity to learn and generalize. The selected loss function ('categorical crossentropy') and accuracy metric, as well as Adam optimization, are crucial in building the model before starting training epochs.

Learning rate, also known as the learning rate, is a hyperparameter that controls the size of the steps that are made throughout the optimization process. The learning rate for the Adam optimizer is explicitly set at 0.001 in your code. This implies that the model's parameters (weights) are updated in the direction that minimizes the loss during each training iteration, and the magnitude of these modifications is regulated by the learning rate of 0.001. The learning rate is an important hyperparameter whose ideal value might influence the model's convergence and overall performance during training.

Activation function: Softmax in the output layer and ReLU (Rectified Linear Unit) at the hidden layers are the activation functions used in the experimental model. For multiple identification issues, the Softmax activation function, selected for the output layer, proved to be the best option. Softmax gives probabilities to each class, allowing the model to provide reliable and mutually exclusive predictions for numerous disease categories in the context of detecting coffee leaf disease. ReLU is also used in the hidden layers to provide non-linearity and improve the model's ability to learn detailed patterns from input data.

Epoch: The epoch parameter specifies how many times the neural network is given with training material throughout the training phase. The model is trained for 5 epochs in the context of the.

Batch size: The batch size in the provided code refers to the number of sub samples dispatch to the neural network before a parameter update during the training phase. The most typical batch

sizes are 32, 64, 128, or equivalent amounts. However, I found that a batch size of 32 produced the greatest results in terms of training efficiency and model correctness.

3.10. Model Visualization

The visual depiction of the model structure provides a simple overview of the architecture, demonstrating the sequential flow of data via each tier. This visualization may be accomplished easily using Python graph visualization frameworks such as Matplotlib's pyplot or Keras' plot_model. We may obtain a full grasp of the convolutional neural network's architecture by utilizing these tools, which provide a visual guide to the difficult process of feature extraction, pooling, and identification. Matplotlib's pyplot (import matplotlib.pyplot as plt) This graphical depiction is a helpful resource for model interpretation and helps to communicate the model's complexity more understandably.

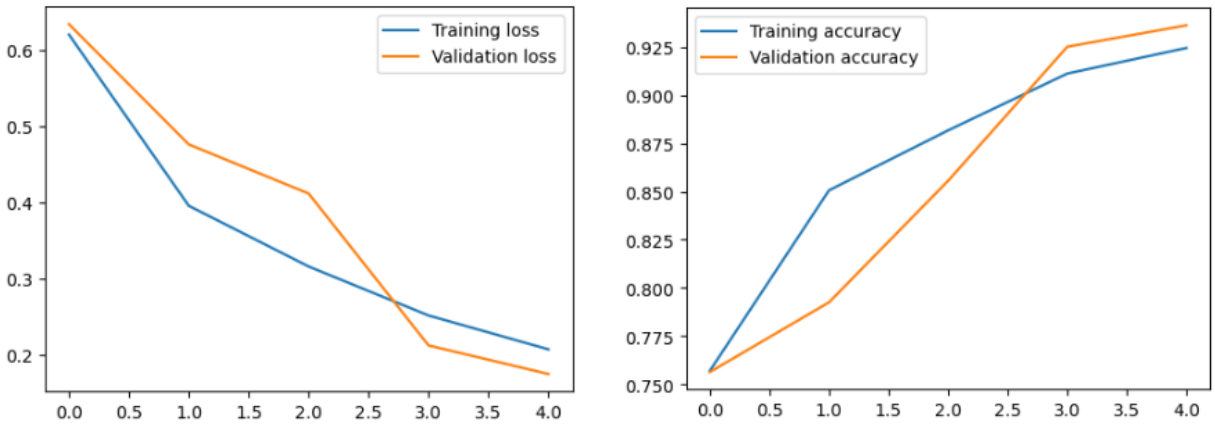


Figure 3-5. Model plot Output.

3.11. Hardware Tools

A computer with modest specs was used to perform the deep learning algorithm using the selected software tools. The hardware setup comprises an Intel(R) Core(TM) i3-8130U CPU @ 2.20GHz 2.21 GHz, 8 GB of memory, and a 500 GB storage capacity.

CHAPTER FOUR

4. EXPERIMENT AND DISCUSSION

4.1. Overview

This chapter delves into the detection and categorization of coffee leaf disease using deep learning. The selected Technology Stack, which includes TensorFlow, OpenCV, Pandas, and NumPy, serves as the basis for model creation. The Model Performance section describes the architecture and training technique, as well as showcasing the model's excellent accuracy in diagnosing several coffee leaf diseases. The Evaluation Methods enable a complete evaluation by using various training, validation, and testing sets that show the model's generalization capabilities. Furthermore, by providing real-time predictions, the inclusion of Single Image Prediction improves user involvement.

4.2. Model Selection

The selected model is a Convolutional Neural Network (CNN), a strong architecture for picture categorization. The sequential model architecture enables obvious and uncomplicated layer stacking. Multiple convolutional and pooling layers allow the model to learn hierarchical features from input photos, capturing spatial patterns associated with various disease. Early Stopping serves as a callback to check validation loss and halt training when overfitting is observed. The assessment measures, such as accuracy, confusion matrix, and identification report, show the model's capacity to properly diagnose coffee leaf disease. Data augmentation improves the model's performance by producing different training examples, hence preventing overfitting. The Adam optimizer and categorical cross-entropy loss function are suitable choices for multiclass identification tasks.

The chosen CNN architecture, together with proper preprocessing and regularization approaches, performs well for the job of identifying Ethiopian coffee leaf diseases.

4.3. Comparison of CNN Model

This study compares the pre-trained architectures Inceptionv3 and MobileNetV2, focusing on variables like computational efficiency, model accuracy, and adaptation to varying picture complexity. Inceptionv3, with its complex inception modules, contrasts with MobileNetV2, which is noted for its lightweight architecture. The comparison seeks to provide insights for

researchers and practitioners looking for the best architecture for their individual application needs.

Inceptionv3 (GoogLeNet)

InceptionV3, commonly known as GoogLeNetv3, is a convolutional neural network (CNN) architecture from Google's Inception model family. It is a huge step forward in deep learning for image categorization problems. InceptionV3 was launched as the successor to the original Inception (GoogLeNet) and InceptionV2 models, with the goal of improving both accuracy and computing efficiency.

It has performed exceptionally well on a variety of image identification benchmarks, demonstrating its ability to capture complex patterns and characteristics inside pictures. InceptionV3's architectural improvements and efficient design have led to applications not just in image identification but also in transfer learning situations, where pre-trained models on huge datasets may be fine-tuned for specific tasks with less data.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 flatten (Flatten)           (None, 73728)             0

 dense (Dense)               (None, 128)               9437312

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 31,240,612
Trainable params: 9,437,828
Non-trainable params: 21,802,784
```

Figure 4-1. InceptionV3 model.

- Top-1: Accuracy: 78.8%
- Top-5: Accuracy: 91.8%
- Epoch 5

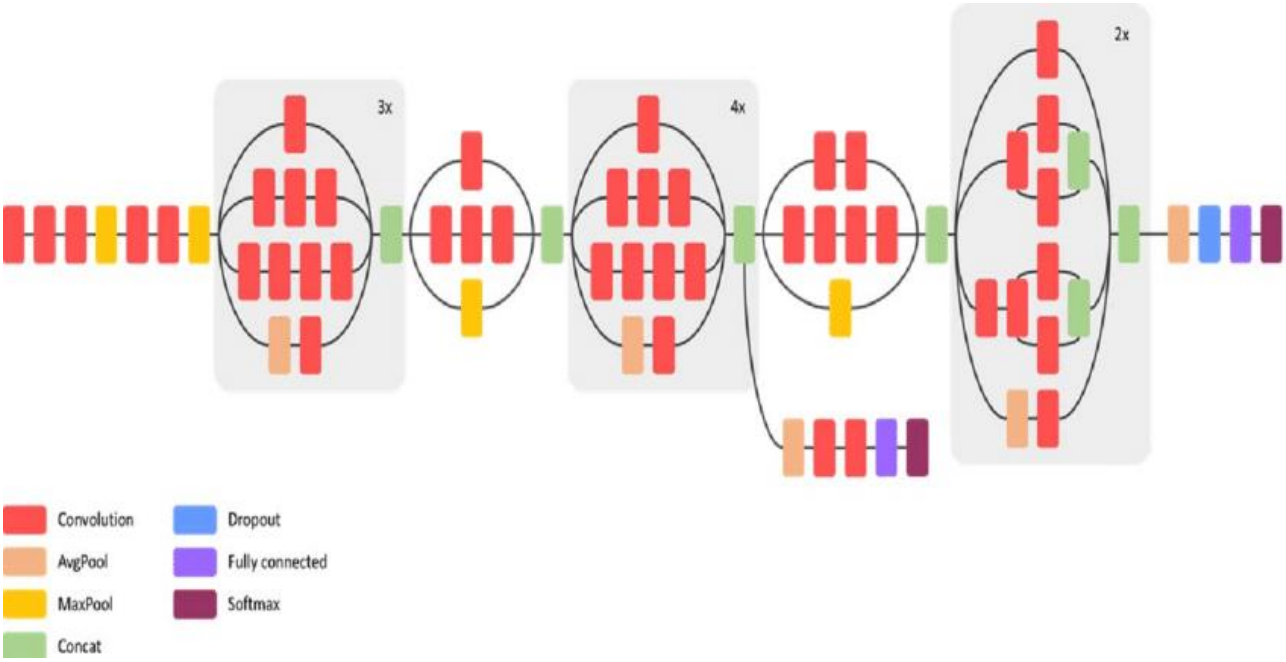


Figure 4-2. Architecture for Inceptionv3.

MobileNetV2

MobileNetV2 is a convolutional neural network (CNN) architecture that is lightweight and efficient, specifically developed for mobile and edge devices that have limited resources. It is an extension of Google's original MobileNet, which was expressly designed to give great accuracy while lowering computing needs. MobileNetV2 improves on the success of its predecessor by providing new design aspects that improve efficiency and efficacy in real-world applications.

It has shown outstanding performance on a variety of benchmarks, demonstrating its ability to strike a compromise between model accuracy and computational efficiency. Its design principles make it ideal for use in mobile and edge computing settings when resources are limited. As a result, MobileNetV2 has become a commonly used architecture in computer vision, allowing deep learning models to be deployed on a variety of devices.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224 (Funct  (None, 8, 8, 1280)       2257984
 ional)

 flatten (Flatten)           (None, 81920)             0

 dense (Dense)               (None, 128)               10485888

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 12,744,388
Trainable params: 10,486,404
Non-trainable params: 2,257,984
```

Figure 4-3. MobileNetV2 model.

- Top-1: Accuracy: 82.4%
- Top-5: Accuracy: 93.3%
- Epoch 5

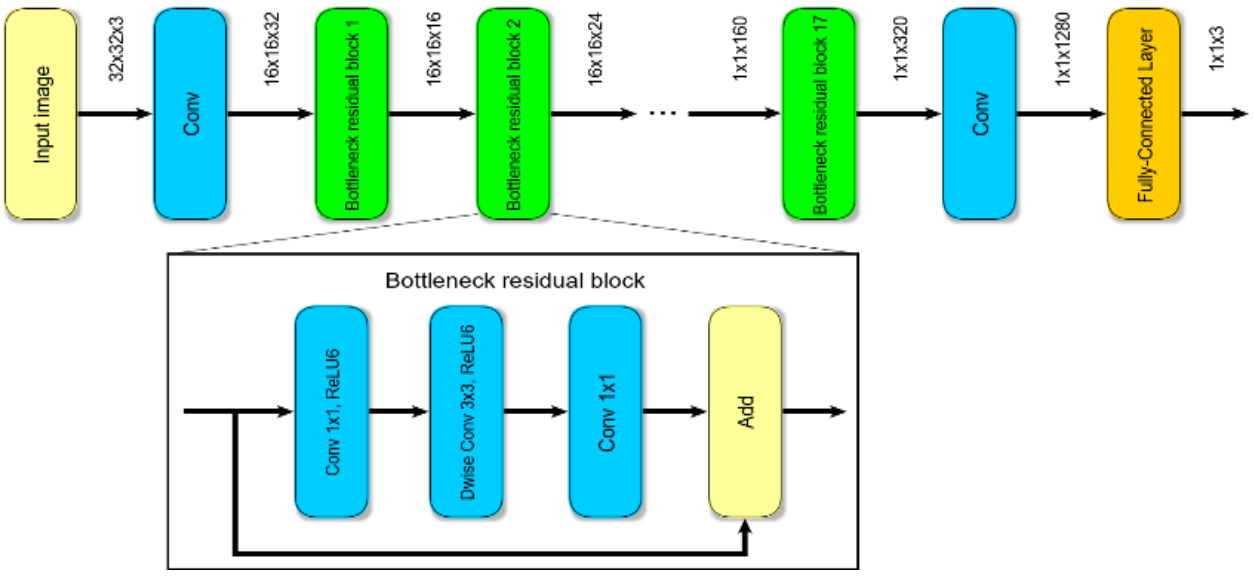


Figure 4-4. Architecture for MobileNetV2.

4.4. CNN Architectures Result Analysis

CNN Architectures	Number of Parameters	Number of epochs	Accuracy	loss	Remark
InceptionV3	31,240,612	5	93%	0.1886	
MobileNetV2	12,744,388	5	91%	0.2075	
Scratch model	14,839,492	5	95%	0.1346	My Proposed Architecture

Table 4- 1. Result of the Study by Comparing CNN Architectures.

4.5. Experiment Duration

The deep learning experiment for coffee leaf disease detection and identification involved a lengthy training procedure covering 5 epochs. Each epoch represents to a full loop of the whole dataset, helping to enhance the model's weights and improve its ability to reliably diagnose coffee leaf disease. The model's iterative learning approach, in which it constantly modified its parameters to reduce categorical cross-entropy loss, affected the training length. The model was subjected to multiple cycles of forward and backward passes in order to improve its forecasting skills.

The period of the experiment was carefully set to create a compromise between obtaining adequate accuracy and avoiding overfitting, ensuring that the model generalized well to previously unknown data. This repeated training method, in conjunction with the given number of epochs, was critical in generating the outstanding model performance shown in the findings.

4.6. Technology Stack

This code's technological stack consists of a broad mix of tools and frameworks designed for constructing and implementing a strong image identification model for coffee leaf diseases. At its heart is the usage of Python, a flexible and widely utilized programming language, as well as well-known frameworks like as TensorFlow and Keras. TensorFlow serves as the foundation for

building and training neural network models, while Keras offers a high-level interface for defining and customizing model architecture.

TensorFlow: stands as a pivotal deep learning framework appreciated for its adaptability and scalability. Crafted by Google Brain, its primary aim is to streamline the process of crafting, refining, and deploying machine learning models across diverse platforms and devices. Offering a comprehensive ecosystem, TensorFlow caters to users across proficiency levels, facilitating seamless trial-and-error sessions and the integration of sophisticated neural networks and other machine learning methodologies into real-world applications. This holistic approach not only encourages exploration but also supports the robust execution of machine learning tasks, ensuring accessibility and efficiency across the development spectrum.

Keras: as a high-level neural networks API, provides a user-friendly interface built atop TensorFlow, making it accessible for developers and researchers to quickly prototype and deploy deep learning models. With its simplicity and modularity, Keras abstracts away the complexities of building neural networks, allowing users to focus on model architecture and experimentation. It serves as an ideal tool for rapid development and experimentation in the field of deep learning.

Matplotlib: Matplotlib is renowned as a potent and adaptable Python library dedicated to crafting static, interactive, and animated visualizations. Its rich repertoire of plotting functions and customizable features equips users with the tools to generate a diverse spectrum of visual representations, encompassing line plots, scatter plots, histograms, and beyond. This flexibility and user-friendly interface render Matplotlib indispensable for various domains, spanning scientific research to data-driven decision-making, where data exploration, analysis, and presentation are paramount. Its versatility serves as a cornerstone for empowering users to convey insights effectively and engage with data intuitively, thereby amplifying the impact of visual storytelling and analysis.

Seaborn: an extension built upon the foundation of Matplotlib, introduces a sophisticated interface tailored for crafting insightful and aesthetically pleasing statistical graphics. Through its streamlined API, Seaborn facilitates the creation of intricate visualizations with minimal coding overhead, rendering it especially conducive to exploratory data analysis and presentation endeavors. Its forte lies in illuminating statistical relationships and patterns within datasets,

boasting a repertoire of specialized plots meticulously designed for tasks ranging from distribution visualization to categorical data analysis and regression analysis. Leveraging Seaborn, users can navigate complex datasets with ease, uncovering meaningful insights and communicating findings with clarity and precision.

NumPy: serves as the fundamental library for numerical computing in Python, providing support for efficient array operations and mathematical functions. Its multidimensional array object, ndarray, enables users to perform vectorized computations and manipulate large datasets with ease. NumPy's extensive collection of mathematical functions and tools for linear algebra, Fourier analysis, and random number generation make it indispensable for scientific computing, data manipulation, and machine learning tasks. Its seamless integration with other libraries in the Python ecosystem, such as TensorFlow and Matplotlib, further enhances its utility in data analysis and visualization workflows.

OpenCV: a computer vision package, is essential in picture preparation, including features such as image reading and scaling. Furthermore, NumPy and Pandas aid in efficient data processing and array operations, boosting the code's ability to handle big datasets.

The code makes use of Matplotlib and Seaborn, two popular Python libraries for constructing visual representations of model performance measures like as accuracy and loss during training epochs, enabling viewing and analysis. The addition of scikit-learn extends the code's capabilities by providing necessary methods for creating a confusion matrix and a full identification report, allowing for a detailed examination of the model's performance.

In summary, the technological stack consists of Python, TensorFlow, Keras, OpenCV, NumPy, Pandas, Matplotlib, and Seaborn, which enable the development, training, assessment, and visualization of a complex image identification model for coffee leaf disease.

4.7. Training Strategy

The training technique used to construct the coffee leaf disease identification model is an experimental and successful way to optimize model performance across 5 epochs. The training procedure, which used a convolutional neural network design, had several critical components. For feature extraction, the Sequential model used convolutional layers with max-pooling,

followed by flattening and dense layers for identification. The Adam optimizer with a categorical cross entropy loss function was used to construct the model.

A complete assessment technique comprised of training the model over numerous epochs, as well as a rigorous study of training and validation loss and accuracy. This repeated training method enabled the model to adapt to and learn nuanced patterns from the dataset, enhancing its capacity to make correct predictions on previously unknown cases. Using data augmentation techniques such as picture sharing, zooming, and horizontal flipping during training improved the model's capacity to generalize and robustly categorize various occurrences of coffee leaf disease. Overall, the training technique exhibits a well-thought-out mix of design, optimization, and assessment methodologies to produce a high-performing and adaptable model over 5 epochs.

4.8. Model Performance

The associated validation accuracy is 75.65%, establishing the groundwork for future advancements. Subsequent epochs showed considerable improvements, with training accuracy rising progressively to 85.06%, 88.17%, 91.11%, and ultimately stabilizing at an astonishing 92.44% in the fifth epoch. Concurrently, validation accuracy improved significantly, reaching 79.26%, 85.56%, and 93.61%. The linked loss measures decreased throughout training, demonstrating the model's capacity to reduce mistakes and increase prediction accuracy. The final assessment on an independent test set yielded a loss of 0.1346 and an accuracy of 95.33%, demonstrating the model's resilience and ability to generalize successfully to new data. Both the training and validation loss measurements showed a decreasing trend, suggesting satisfactory convergence and effective learning. The detection or identification report and confusion matrix offered a thorough analysis of the model's prediction performance across several classes, indicating high accuracy and recall for each class. The robustness of the model is underlined further by the interactive depiction of random test set photos, which allows viewers to actively examine predictions alongside real identifications. Overall, the model's efficiency in categorizing diverse coffee leaf disease is confirmed by this extensive study, opening the door for its prospective implementation in real-world settings. Continued modification and optimization of the model's performance may reveal new chances for improvement.

```
Epoch 1/5
304/304 [==============================] - 1645s 5s/step - loss: 0.6201 - accuracy: 0.7571 - val_loss: 0.6337 - val_accuracy: 0.7565
Epoch 2/5
304/304 [==============================] - 2474s 8s/step - loss: 0.3952 - accuracy: 0.8506 - val_loss: 0.4758 - val_accuracy: 0.7926
Epoch 3/5
304/304 [==============================] - 1970s 6s/step - loss: 0.3155 - accuracy: 0.8817 - val_loss: 0.4113 - val_accuracy: 0.8556
Epoch 4/5
304/304 [==============================] - 1596s 5s/step - loss: 0.2512 - accuracy: 0.9111 - val_loss: 0.2118 - val_accuracy: 0.9250
Epoch 5/5
304/304 [==============================] - 1177s 4s/step - loss: 0.2067 - accuracy: 0.9244 - val_loss: 0.1743 - val_accuracy: 0.9361
```

Figure 4-5. Model performance and accuracy.

4.9. Evaluation Methods

The coffee leaf disease detection model's assessment metrics show that it performs well in properly The assessment metrics for the coffee leaf disease detection model demonstrate its high accuracy in predicting various disease classes. On a specific test dataset, the model obtains a high overall accuracy of over 95%. The thorough confusion matrix breaks down the model's predictions into four categories: Cerscospora, Healthy, Leaf Rust, and Phoma life spot. Notably, the model excels at properly recognizing cases of certain disease, with high precision and recall in each class.

This matrix, together with a thorough identification report that includes accuracy, recall, and the F1-score, offers a comprehensive picture of the model's generalization skills and suggests possible areas for improvement. The interactive predict_and_print function improves the assessment process by allowing users to see real-time predictions on individual photographs and identify the model's strengths and weaknesses. Overall, the assessment metrics, including accuracy, confusion matrix, and identification report, affirm the model's robust performance and suggest its potential applicability in real-world scenarios.

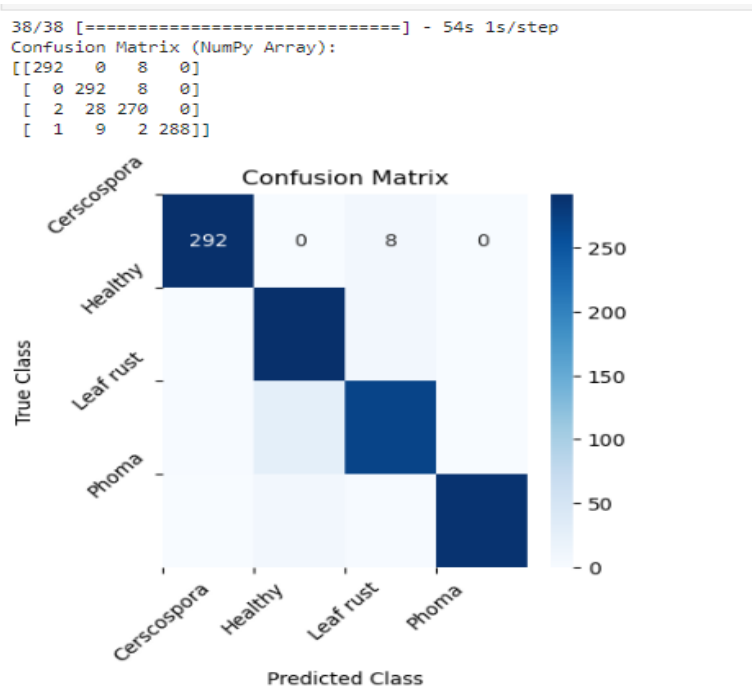


Figure 4-6. Proposed model Confusion matrix.

4.10. Single Image Prediction

The inclusion of a single-image prediction capability is a remarkable element of the offered code's experimental design. The predict_and_print function, which allows users to enter a picture and obtain real-time predictions from the trained model, is the focus point. Loading the picture, turning it into an array, and completing the appropriate preprocessing procedures to conform to the model's input needs are all part of the process. The model was trained using a convolutional neural network architecture and a dataset including numerous types of coffee leaf disease.

The model was trained using a Sequential model that included convolutional layers with max-pooling, flattening, and dense layers. The Adam optimizer and categorical crossentropy loss function were used to optimize the built model. Over numerous epochs, the model's performance was thoroughly evaluated, with visualizations displaying training and validation loss, as well as training and validation accuracy.

The attained accuracy was determined to be around 95% after model assessment using a second test dataset. A confusion matrix and a complete identification report were also created to

evaluate the model's performance across several classes of coffee leaf diseases. These assessment metrics offer useful information about the model's accuracy, recall, and F1-score for each class.

Identification report

	precision	recall	f1-score	support
Cerscospora	0.99	0.97	0.98	300
Healthy	0.90	0.97	0.94	300
Leaf rust	0.93	0.92	0.92	300
Phoma	1.00	0.95	0.98	300
accuracy			0.95	1200
macro avg	0.95	0.95	0.95	1200
weighted avg	0.95	0.95	0.95	1200

Figure 4-7. Detailed Identification Report.

A function called predict_and_print was added to improve user engagement and comprehension. Users may provide an image file path into this method, and the model will predict the coffee leaf disease class, presenting the picture alongside the predicted and real classes. Finally, the model was stored and later loaded for possible future usage, demonstrating a comprehensive approach to model development, assessment, and deployment.

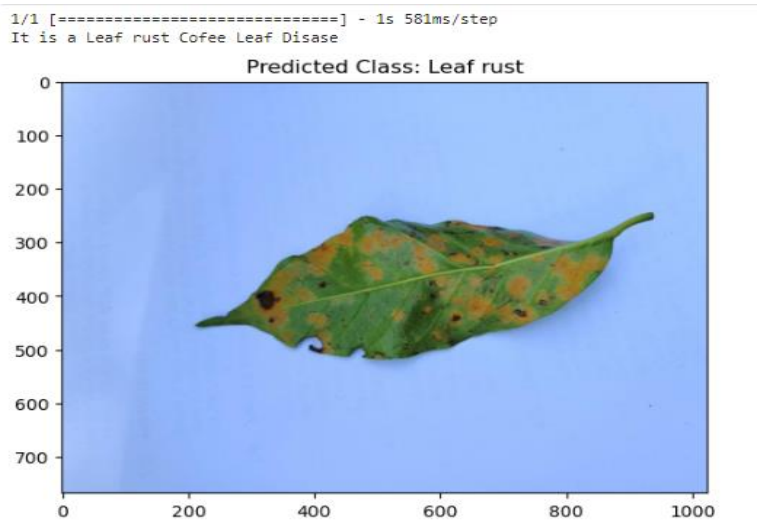


Figure 4-8. Predicted Image.

4.11. Discussion of the Result

In our experimental design, we created particular scenarios to thoroughly test the recognition performance of characteristics taken from photos of Ethiopian coffee leaves and beans. The feature extraction procedure includes essential features such as image color, texture, and size, with a focus on utilizing RGB color characteristics from coffee leaf pictures. This thorough technique seeks to capture complex nuances and variances in photographs. To distinguish the sections of a coffee leaf, we use the particular RGB color properties, which allow for exact delineation and segmentation within the photos. The detection and identification performances are evaluated using the Convolutional Neural Network (CNN) technique. In our comparison study, Table 4 to analyze the architectural advantages of InceptionV3, scratch model, and MobileNetV2. This comparison analysis serves as the foundation for selecting the most optimum model, which is governed by strict model assessment criteria. The CNN method was chosen strategically because of its intrinsic appropriateness for processing visual data, which perfectly matches the properties of our datasets. The dataset contains four unique classes: Brown Eye, phone, life rust, and healthy coffee. This variety guarantees that the model is properly trained to recognize and categorize varied coffee leaf and bean states.

Our assessment findings demonstrate the superiority of the suggested model, which is both time-efficient and successful in coffee disease detection and identification, with an astounding accuracy of 95.33%. This accomplishment emphasizes the model's robust performance and prospective use as a dependable tool for tackling the intricacies of coffee disease identification in real contexts.

CHAPTER FIVE

5. CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

Diseases such as Brown Eye Spot, Phoma life spot, and other coffee leaf disase represent serious dangers to the coffee industry today, resulting in lower yields and coffee quality. The absence of diagnostic tools, particularly in emerging countries such as Ethiopia, impedes the overall expansion of the coffee industry. To address this difficulty, we developed a low-cost and user-friendly deep learning system based on the Convolutional Neural Network (CNN) method for early identification of coffee leaf disease. Our CNN model, which we created from scratch, is a reliable and practical tool for reliably diagnosing and categorizing CLS, PLS, and other prevalent coffee leaf diseases. This novel strategy enables early intervention and tailored agricultural initiatives, which aligns with the overarching goal of empowering coffee producers and supporting sustainable growth in developing nations.

This thesis makes major advances to successfully identifying and categorizing common coffee leaf disase, namely CLD, CLS, and PLS, using a well-designed Convolutional Neural Network (CNN) model. The model was created primarily to solve real-world issues like as complicated backdrops, dynamic picture resolutions, shifting lighting, and orientations. The second significant contribution is the systematic arrangement and maintenance of a large dataset containing photographs of coffee leaves.

Two tests using pre-trained models and a proposed model. The photos utilized in the experiment were acquired directly from coffee leaves, with the assistance of agricultural specialists, as well as from existing databases. The two modified pre-trained models, InceptionV3 and MobileNetV2, as well as a scratch model, were trained and produced impressive identification results. InceptionV3 achieved 93% accuracy, while the scratch model scored 95% and MobileNetV2 achieved 91%. Furthermore, the suggested model achieved a high accuracy of 95.3%.

Importantly, the suggested scratch model emerged as a substantial contributor to successful disease identification and identification, with less computing effort and fewer photos than traditional deep learning techniques. This is especially notable because many deep learning

techniques require large datasets and significant processing resources. The results of these trials are positive, spurring a continuous commitment to fine-tune and test the model with new coffee disorders in the future.

In Section 1.4 of Chapter One formulated research questions. The study solved all of the questions posed by the research questions. The following are the answers to the questions.

- What is the primary coffee leaf disease that affects coffee leaves?

Coffee Leaf Rust (CLR), caused by the fungus Hemileia vastatrix, is the principal coffee disease that seriously damages coffee leaves. CLR appears as orange-yellow powdery lesions on the undersides of coffee leaves, which contain spore masses. This widespread and destructive disease impairs photosynthesis, resulting in defoliation and, in extreme cases, premature leaf shedding. Reduced photosynthetic ability jeopardizes coffee leaf health and output, resulting in lower bean quality and quantity. CLR is especially prevalent in warm and humid conditions, offering a constant challenge to coffee farmers across the world.

- How accurate is the deep learning model in predicting occurrences of coffee disease?

The deep learning model, as implemented in the given code, predicts the occurrence of coffee dis ease with noteworthy accuracy. The model steadily improves its accuracy during 5 epochs of trai ning, reaching high levels. To improve its performance, the model is tested using a different data set, yielding a loss of 0.1346 and an accuracy of 0.9533%. This implies that the model retains its usefulness when applied to previously unknown data. In summary, based on the supplied code, t he deep learning model predicts coffee leaf diseases with good accuracy.

- Does the implementation of CNN techniques enhance the performance of models for coffee leaf disease detection?

Yes, Convolutional Neural Network (CNN) approaches have shown great success in boosting model performance for a variety of image-based applications, including disease detection in plants like coffee leaf disease. CNNs are especially well-suited for image analysis jobs due to their capacity to learn hierarchical features from pictures automatically. Feature Extraction: CNNs are built to automatically extract meaningful information from pictures using convolutional layers. This is critical for recognizing trends and distinguishing features associated

with various stages or kinds of coffee leaf diseases. Traditional machine learning techniques may struggle to create such complicated characteristics manually.

In the case of detecting coffee leaf disease, feature extraction using CNNs allows the model to learn visual traits and patterns associated with healthy and sick leaves automatically. CNNs' hierarchical and localized feature learning makes them adept in capturing the subtle visual signals found in agricultural photos, resulting in reliable and robust disease detection models.

5.2. Recommendations

Ethiopia's agricultural dependency and the importance of agriculture to the country's economy, it is critical for the agricultural sector to emphasize crop preservation, particularly disease detection. Recognizing the need of early detection of agricultural diseases, image analysis techniques have emerged as critical tools. While previous research in Ethiopia has focused on certain elements of coffee disease, such as coffee leaves, there is still an untapped need to expand these efforts to include disease detection in coffee beans. This thesis acts as a possible spark, encouraging academics to explore further into this domain, which includes regions such as coffee plant stems and roots, to strengthen the agricultural sector's ability to protect crops from disease and contribute to the nation's economic development.

For the future, the study recommended to the following two recommendations:-

- Diversifying disease detection for coffee leaf disease is crucial, acknowledging that their causes may extend beyond fungi or bacteria to include issues induced by insects and pests. Incorporating elements like recognizing skeletonized patterns and identifying insect-induced disorders such as citrus leaf miner enhances the model's versatility and applicability in addressing a broader range of coffee plant health challenges. Additionally, it is recommended to train and test the model specifically for disease detection in coffee stem and root components, further advancing its capabilities for comprehensive plant health assessment.

REFERENCE

[1] Alemseged Assfa;, and G. Arega; "Agricultural Economics, Research Extension and Farmers' Linkage Coordination Research Directory," Ethiopian Coffee Exporters Association (ECEA), Jun. 2013.

[2] A. Tefera; "Coffee Annual Report," United States Department of Agriculture Foreign Agricultural Service, Ethiopia, Sep. 09, 2022.

[3] Justin moat;, Jenny Williams;, Susana baena;, Tim Wilkinson; "Resilience Potential of the Ethiopian coffee sector under climate change" June 2017

[4] T. Tadesse, B. Tesfaye, and G. Abera, "Coffee Production Constraints and Opportunities at Major Growing Districts of Southern Ethiopia," Cogent Food & Agriculture, pp. 1-36, Apr. 20, 2020.

[5] H. Taddese; "Coffee Disease Detection through Convolutional Neural Networks: An Image Processing Approach" November, 2021.

[6] Eyobe Birhanu; Paulos ; Michael Melese Woldeyohannis; "Detection and Classification of Coffee Leaf Disease using Deep Learning," November 2022, on 20, January 2023.

[7] Abrham Debasu; Dagnachew, Melesew Alemayehu; Seffi Mengistu, Gebeyehu Mengistu;, "Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques," International Journal of Database Theory and Application, 2016.

[8] S. N. Ghaiwat and P. Arora,"Detection and Classification of Coffee diseases Using Image processingTechniques", A Review, vol. 2, no. 3,(2014).

[9] S.K.Mangal, "Coffee Planting, Production and Processing," Gene-Tech Books, pp 51-59 2007.

[10] F. R. Ervine, "West African Agriculture: West African Crops," 3rd ed., vol. 2. 1996.

[11] V. der Vossen, H. Bertrand, B. Charrier, A. Next generation variety development for sustainable production of arabica coffee (*Coffe*a arabica L.): A review. Euphytica, vol. 204, pp. 243–256, 2015 [Google Scholar] [CrossRef]

[12] Motisi, N.; Bommel, P.; Leclerc, G.; Robin, M.; Aubertot, J.; Butron, A.; Treminio, E.; Avelino, J. Improved forecasting of coffee leaf rust by qualitative modeling: Design and expert validation of the ExpeRoya model. *Agric. Syst.* 2021, *197*, 103–129. [Google Scholar] [CrossRef]

[13] Cerda, R.; Avelino, J.; Gary, C.; Tixier, P.; Lechevallier, E. Primary and secondary yield losses caused by pests and diseases: Assessment and modeling in coffee. *PLoS ONE* 2017, *12*, e0169133. [Google Scholar] [CrossRef][Green Version]

[14] De Moraes, S.A. *A Ferrugem do Cafeeiro: Importância, Condições Predisponentes, Evolução e Situação no Brasil*; Instituto Agronômico: Londrina, Brazil, 1983; p. 50. [Google Scholar].

[15] Catarino, Aricléia de Moraes; Pozza, Edson Ampélio ; Pozza, Adélia Aziz Alexandre; Santos, Leone Stabile dias; Vasco, Gabriel Brandão ; Souza, Paulo Estevão de;, "Calcium and potassium contents in nutrient solution on Phoma leaf spot intensity in coffee seedlings," ResearchGate, 25 March 2017.

[16] Hermann A. Jürgen pohlan; Marc JJ Janssens;, " Growth and Production Dynamics of Coffee," vol 3 june 2015.

[17] Linigerew Mengstie Shita, Prof. Mogalla Shashi; ” Ethiopian Coffee Leaf Diseases Classification using Deep Learning Features,” October 2021, Vol 8, Iss 10,

[18] “Common pests and diseases of coffee” online Available https:// plantwiseplusknowledgebank.org/doi/pdf/10.1079/pwk"b.20137804172

[19] Z. Zhang, X. He, X. Sun, L. Guo and J. Wang, "Image Recognition of Maize Leaf Disease Based on GA-SVM," 2015. [Online].

[20] V. Dixit and J. Pruthi, "Review of Image Processing Techniques for Automatic Detection of Tumor in Human Liver," International Journal of Computer Science and Mobile Computing, vol. 3, p. 371–378, 2014.

[21] Vijai Singh [a], A.K. Misra “Detection of plant leaf diseases using image segmentation and soft computing techniques” Vol 4, Is 1, March 2017, P 41-49

[22] T. Berhanu, "Survey of diseases of major crops in Darolebu district,West Hararge, Ethiopia," International Journal of Plant Breeding and Crop Science, vol. 2, p. 43–45, 2015.

[23] Sanyal, P., Bhattacharya, U., Parui, S. K., Bandyopadhyay, S. K. and Patel, S., 2007. Color Texture Analysis of Rice Leaves Diagnosing Deficiency in the Balance of Mineral Levels towards Improvement of Crop Productivity, 10th International Conference on Information Technology (ICIT 2007), Orissa, doi: 10.1109/ICIT.2007.40, 85-90.

[24] Arivazhagan, S., Shebiah, R.N., Ananthi, S. and Varthini, S.V., 2013. Detection of unhealthy region of plant leaves and Classification of plant leaf diseases using texture features. Agricultural Engineering International: CIGR Journal, 15(1), 211-217.

[25] Gajanan, D.E., Shankar, G.G. and Keshav, G.V., 2018. Detection of Leaf Disease Using Feature Extraction for Android Based System.

[26] Mathew, Amitha; P. Amudha; S. Sivakumari;, "Deep Learning Techniques: An Overview," researchgate.net/publication/341652370, 02 August 2020.

[27] A. K. Jain, J. Mao and K. M. Mohiuddin, Artificial neural networks, A tutorial In: Computer 29.3, 1996, p. 31–44.

[28] Patterson, Josh; , Adam Gibson;, "Deep Learning," in Deep Learning, O'Reilly Media, Inc., 1005 Gravenstein Highway North,Sebastopol, CA 95472, 2017.

[29] Yamashita, Rikiya; , Mizuho Nishio; Gian Do, Richard Kinh ; , Kaori Togashi;, "Convolutional neural networks: an overview and application in radiology," 22 June 2018.

[30] A. K. Jain, J. Mao and K. M. Mohiuddin, Artificial neural networks, A tutorial In: Computer 29.3, 1996, p. 31–44.

[31] S. Sharma, "Activation Functions in Neural Networks," 6 9 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks.

[32] A. Fernandez, "Data Science in Aviation," September 29, 2020. [Online]. Available: https://datascience.aero/aviation-function-deep-learning/.

[33] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev and N. Chervyakov, Application of the residue number system to reduce hardware costs of the convolutional neural network implementation, Mathematics and Computers in Simulation, 2020, pp. 232-243.

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley and S. C. Ozair, Advances in neural information processing systems, Generative adversarial nets, 2014, pp. 3,29,31.

[35] Shaily, Tumun, "Bacterial Image Classification Using Convolutional Neural Networks," 2020 IEEE 17th India Council International Conference, INDICON 2020, vol. 3, no. 9, 2020.

[36] F. Li, J. Justin; , Y. Serena;, "Convolutional Neural Networks for Visual Recognition," 20 January 2019. [37] F. Li, J. Justin; , Y. Serena;, "Convolutional Neural Networks for Visual Recognition," http://cs231n.stanford.edu/index.html, 20 january 2019.

[37] F. Li, J. Justin and Y. Serena, "Convolutional Neural Networks for Visual Recognition," http://cs231n.stanford.edu/index.html, 20 january 2019.

[38] D. A. Khan, "Introduction to Deep Convolutional Neural Networks," ResearchGate, 08 November 2016.

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, Going deeper with convolutions, In: Cvpr, 2015.

[40] R. Yamashita, M. Nishio, . R. K. G. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," Insights Imaging, vol. 9, p. 611–629, 2018.

[41] Geoffrey Hinton;, Yosua Yann;, Lecun; “Deep Learning,” ResearchGate;, Volume 52, May 2015

[42] L. Ximenes, F. Paz, C. Toledo, F. Fambrini, R.R. Goulart, J.H. Saito, "Coffee Leaf Disease Recognition Based on Deep Learning and Texture Attributes," in Proceedings of the 2019 16th IEEE International Conference on Computational Science and Engineering (CSE), pp. 135-144, 2019.

[43] A. Debasu; Dagnachew, M. Alemayehu; S. Mengistu, Gebeyehu Mengistu;, "Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques," International Journal of Database Theory and Application, 2016.

[44] Biniyam Mulugeta Abuhayi, Abdela Ahmed Mossa "Coffee disease Classification using Convolutional Neural Network based on feature concatenation" Volume 39, 2023

[45] Yamashita, Rikiya; , Kaori Togashi;, " Coffee plant disease Classification using texture features," 22 June 2018.

[46] Arivazhagan, S., & Ligi, S. V. (2018). Mango leaf diseases Classification using convolutional neural network. Int. J. Pure Appl. Math, 120(6), 11067-11079.

[47] Sorte, L. X. B., Ferraz, C. T., Fambrini, F., dos Reis Goulart, R., & Saito, J. H. (2019). Coffee leaf disease recognition based on deep learning and texture attributes. Procedia Computer Science, 159, 135-144.

[48] Jadhav, S. B., Udupi, V. R., & Patil, S. B. (2020). Classification of plant diseases using convolutional neural networks. International Journal of Information Technology, 1-10.

[49] Connor Shorten, Taghi M. Khoshgoftaar;, "A survey on Image Data Augmentation for Deep Learning," Journal of Big Data, July 2019.

[50] Fatih Ertam, g. aydin "Data classification with deep learning using tensorflow," 2nd Int. Conf. Comput. Sci. Eng. UBMK 2017, p. 755–758, 2017.

[51] Austin, "Convolutional Neural Networks (CNN) to Classify Sentences" Convolutional Neural Networks (CNN) to Classify Sentences - Austin G. Walters (austingwalters.com) January 2, 2019January 4, 2019

[52] Faris g, javad s, sina b, hossein s, "A Deep Learning Approach for Inverse Design of the Metasurface for Dual Polarize Wave" November 2021.

[53] M. Alemayehu worki, "Ethiopian Highlands: Home for Arabica Coffee (Coffea Arabica L.)," 30 May 2017.

Appendix

#Import libraries

```
# Importing all the libraries needed.

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
import pandas as pd
import os, requests, cv2, random

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam

from tensorflow.keras import models
from tensorflow.keras import Sequential, layers
from keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from sklearn.metrics import confusion_matrix,classification_report
from tensorflow.keras.losses import SparseCategoricalCrossentropy
```

load data

```
# Loading a diverse set of coffee leaf images showcasing various diseases.

train_data_dir =  r'C:\Users\Admin\OneDrive\Desktop\ethiopian cofee leaf dataset\train aug'
test_data_dir = r'C:\Users\Admin\OneDrive\Desktop\ethiopian cofee leaf dataset\test' #this folder will be used for evaluating model's perfomance
```

```
IMG_SIZE = (256, 256)  # specify your desired image size

train_datagen = ImageDataGenerator(
                                   rescale=1/255.0,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   validation_split=0.1)  # specifying the validation split inside the function

test_datagen = ImageDataGenerator(
                                  rescale=1/255.0,
                                  shear_range=0.2,
                                  zoom_range=0.2,
                                  horizontal_flip=True)
```

```
train_gen = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    shuffle=True,
    class_mode='categorical',
    subset='training')
```

```
Found 9720 images belonging to 4 classes.
```

```
val_gen = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    shuffle=True,
    class_mode='categorical',
    subset='validation')
```

```
Found 1080 images belonging to 4 classes.
```

```
test_gen = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='categorical',
    shuffle=False)  # shuffle will not affect the accuracy of the model, but will affect the computation of some metrics that depend on the order of the
```

```
Found 1200 images belonging to 4 classes.
```

#Display random image

```
def display_random_images(data, class_names, num_images=12, figsize=(10, 6)):
    plt.figure(figsize=figsize)

    # Get an iterator for the data generator
    data_iter = iter(data)

    for i in range(num_images):
        images, labels = next(data_iter)
        plt.subplot(3, 4, i + 1)
        plt.imshow(images[0])
        plt.title(class_names[np.argmax(labels[0])])
        plt.axis('off')  # Turn off axis labels and ticks

    plt.show()
```

```
display_random_images(train_gen, class_names=C_N)
```

Build model

```
# Create a Sequential model
model = Sequential()

# Add Convolutional layers
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

# Flatten layer
model.add(Flatten())

# Dense layers
model.add(Dense(128, activation='relu'))
model.add(Dense(4, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 14,839,492
Trainable params: 14,839,492
Non-trainable params: 0
_________________________________________________________________
```

Train model

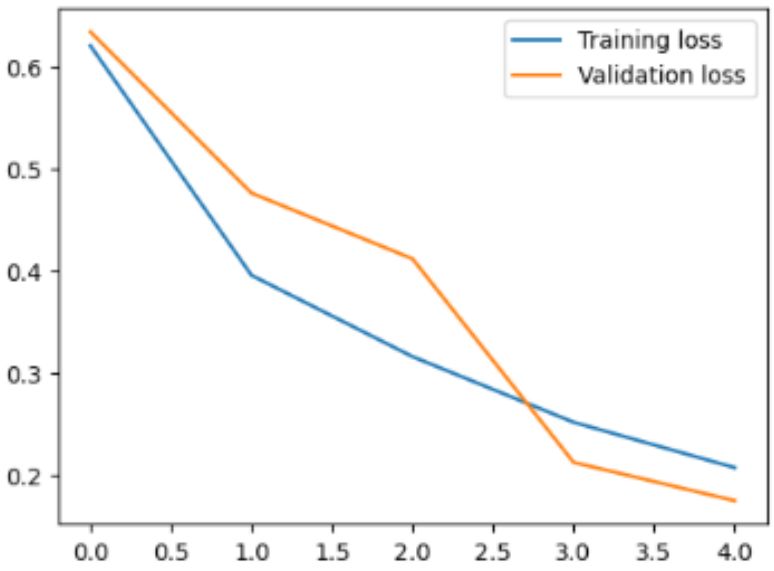
```
# Creating the optimizer
opt = keras.optimizers.Adam(learning_rate=0.001)

# Compile the model with the specified optimizer and loss function

model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

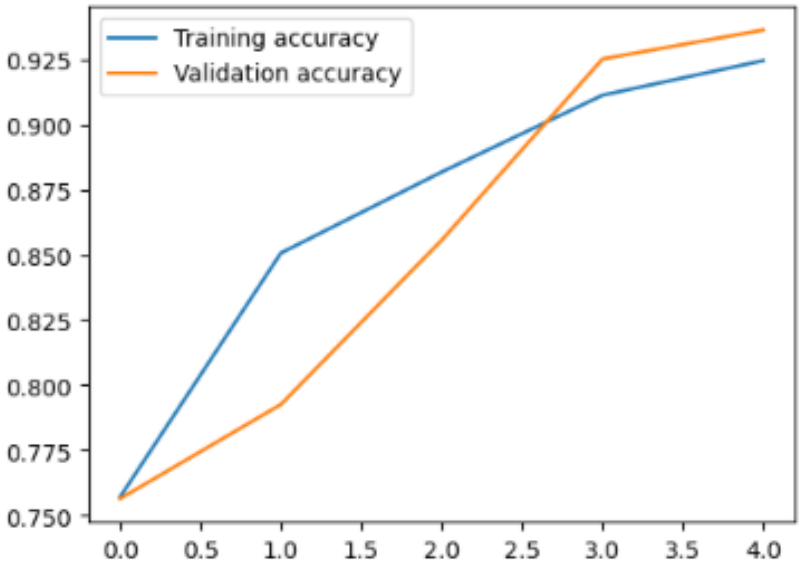
```
import warnings
from PIL import Image
from keras.callbacks import EarlyStopping

# Set a higher threshold for the decompression bomb warning
warnings.filterwarnings("ignore", category=Image.DecompressionBombWarning)

# Define the EarlyStopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)

history = model.fit(
    x=train_gen,
    epochs=5,
    verbose=1,
    steps_per_epoch=len(train_gen),
    validation_data=val_gen,
    validation_steps=len(val_gen),
    callbacks=[early_stopping]  # Add the EarlyStopping callback
)
```

```
Epoch 1/5
304/304 [==============================] - 1645s 5s/step - loss: 0.6201 - accuracy: 0.7571 - val_loss: 0.6337 - val_accuracy: 0.7565
Epoch 2/5
304/304 [==============================] - 2474s 8s/step - loss: 0.3952 - accuracy: 0.8506 - val_loss: 0.4758 - val_accuracy: 0.7926
Epoch 3/5
304/304 [==============================] - 1970s 6s/step - loss: 0.3155 - accuracy: 0.8817 - val_loss: 0.4113 - val_accuracy: 0.8556
Epoch 4/5
304/304 [==============================] - 1596s 5s/step - loss: 0.2512 - accuracy: 0.9111 - val_loss: 0.2118 - val_accuracy: 0.9250
Epoch 5/5
304/304 [==============================] - 1177s 4s/step - loss: 0.2067 - accuracy: 0.9244 - val_loss: 0.1743 - val_accuracy: 0.9361
```

Compare the accuracy and ploting the graph training amd validation

```
# plot the loss

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.legend()
plt.show()

# Plot the accuracy
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training accuracy')
plt.plot(history.history['val_accuracy'], label='Validation accuracy')
plt.legend()
plt.show()
```





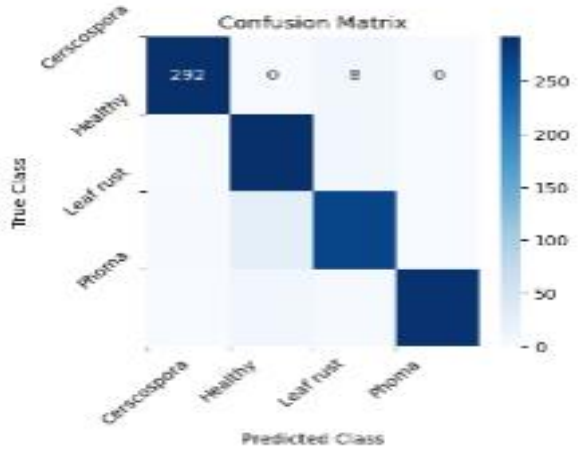
save model

```
]: model.save('final code.h5')
```

Tast accuracy and classfication report

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Assuming we have predictions and true labels from our model
y_pred = model.predict(test_gen)
y_true = test_gen.classes

# Calculate the confusion matrix
cm = confusion_matrix(y_true, np.argmax(y_pred, axis=1))

# Display the confusion matrix as a NumPy array
print("Confusion Matrix (NumPy Array):")
print(cm)

# Create a visualization of the confusion matrix with class names
plt.figure(figsize=(4, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap=plt.cm.Blues)

# Set class names as tick labels
class_names = ["Cerscospora", "Healthy", "Leaf rust", "Phoma"]
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names, rotation=45)
plt.yticks(tick_marks, class_names, rotation=45)

plt.xlabel("Predicted Class")
plt.ylabel("True Class")
plt.title("Confusion Matrix")

plt.show()
```

```
38/38 [==============================] - 54s 1s/step
Confusion Matrix (NumPy Array):
[[292   0   8   0]
 [  0 292   8   0]
 [  2  28 270   0]
 [  1   9   2 288]]
```




```
# Generate a detailed classification report
detailed_report = classification_report(true_classes, predicted_classes, target_names=class_labels)

# Print the report for analysis
print(detailed_report)
```

```
              precision    recall  f1-score   support

  Cerscospora       0.99      0.97      0.98       300
      Healthy       0.90      0.97      0.94       300
    Leaf rust       0.93      0.92      0.92       300
        Phoma       1.00      0.95      0.98       300

     accuracy                           0.95      1200
    macro avg       0.95      0.95      0.95      1200
 weighted avg       0.95      0.95      0.95      1200
```

Random predict images

```
def predict_images(data, class_names,
                   model, num_images=12,
                   figsize=(10, 6)):

    # Visualizes a sample of images along with their actual and predicted classes, aiding model evaluation and understanding.


    plt.figure(figsize=figsize)

    for i in range(num_images):
        images, labels = next(iter(data))  # Efficiently get next batch
        id = np.random.randint(len(images))
        img = tf.expand_dims(images[id], axis=0)

        plt.subplot(3, 4, i + 1)  # Subplot with 3 rows and 4 columns
        plt.imshow(img[0])

        predicted = class_names[np.argmax(model.predict(img))]
        actual = class_names[np.argmax(labels[id])]

        plt.title(f"Actual:  {actual} \nPredicted:  {predicted} ")
        plt.axis('off')

    plt.show()
```

```
predict_images(test_gen, class_names, model)
```

```
1/1 [==============================] - 1s 594ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 47ms/step
```

Load model

```
# To Load the model
from tensorflow.keras.models import load_model

loaded_model = load_model('final code.h5')
```

predict image

```
from PIL import Image
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt

# Load the pre-trained model
model = load_model('final code.h5')

# Function to preprocess the image for prediction
def preprocess_image(img_path):
    img = Image.open(img_path).convert('RGB')
    img = img.resize((256, 256))  # Updated size
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalize the image
    return img_array

# Function to predict the disease and print the image
def predict_and_print(image_path, model):
    img = preprocess_image(image_path)
    prediction = model.predict(img)

    # Assuming you have a list of class names
    class_names = ["Cerscospora", "Healthy", "Leaf rust", "Phoma"]  # Replace with your actual class names
    predicted_class = class_names[np.argmax(prediction)]

    # Print the predicted class
    print(f"It is a {predicted_class} Cofee Leaf Disase")

    # Display the image
    plt.imshow(Image.open(image_path))
    plt.title(f"Predicted Class: {predicted_class}")
    plt.axis('on')
    plt.show()
```

```
image_path_to_predict = r'C:\Users\Admin\OneDrive\Desktop\ethiopian cofee leaf dataset\test\Leaf rust/20231123_103017.jpg'
predict_and_print(image_path_to_predict, model)
```

```
1/1 [==============================] - 1s 581ms/step
It is a Leaf rust Cofee Leaf Disase
```