ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCES

SCHOOL OF INFORMATION SCIENCE

WHEAT LEAF DISEASE CLASSIFICATION FROM IMAGES USING DEEP-LEARNING TECHNIQUES

By

TATEK CHANE MENGESTU

ID: GSR/3922/15

A THESIS SUBMITTED TO ADDIS ABABA UNIVERSITY, COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCES, SCHOOL OF GRADUATE STUDIES, SCHOOL OF INFORMATION SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE AND SYSTEMS (INFORMATION SYSTEMS SPECIALIZATION)

JANUARY/2025

ADDIS ABABA, ETHIOPIA

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCES

SCHOOL OF INFORMATION SCIENCE

By Tatek Chane Mengistu

Names and Signatures of the Examining Board Members

Title	Name	Signature	Date
Advisor	Melkamu Beyene (PhD)	_____	_____
Internal Examiner	Million Meshesha (PhD)	_____	_____
External Examiner	Wondwossen Mulugeta (PhD)	_____	_____

DECLARATION

This is to certify that the thesis prepared by Tatek Chane, titled *"Wheat Leaf Disease Classification from Images Using Deep Learning Techniques,"* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Systems and Systems (Information Systems Specialization), complies with the regulations of the University and meets the accepted standards of originality and quality.

DEDICATION

This research is dedicated to my beloved family for their steadfast support and encouragement throughout this journey. I also dedicate this work to the farmers and agricultural scientists in my country, whose tireless efforts to improve crop health and productivity have been a source of inspiration.

ACKNOWLEDGMENT

First and foremost, I would like to thank Almighty God for blessing every moment of my life and guiding me in all my endeavors. Nothing would have been possible without His support. I extend my heartfelt gratitude to my advisor, Dr. Melkamu Beyene, for his unwavering commitment, enthusiastic encouragement, valuable feedback, and constructive suggestions from the early stages of conceptual inception through to the completion of my thesis. His guidance helped me overcome the challenges I faced and successfully finish my work. I am grateful to the experts in wheat cultivation for their willingness to assist me by providing comprehensive information about wheat leaf diseases. Their dedication in explaining disease symptoms and categorizing disease classes was invaluable to my research. I thank my teachers, who appreciated my efforts and motivated me to persevere until the end. I would also like to thank my beloved family members and friends for their love, encouragement, and support, which were instrumental in accomplishing this research. Your steadfast assistance and insightful criticism have significantly enhanced my writing and study. I have great respect for the farmers in my country, especially those who work tirelessly to improve crop health and increase yields. Lastly, I would like to express my deepest gratitude to everyone who helped and supported me throughout my thesis work.

Thank You All.

Tatek Chane

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acronyms and Abbreviations

AI …..…..…..…..…..…..…..…..…..…..…..…..…. Artificial Intelligence
ANN …..…..…..…..…..…..…..…..…..…..…..…... Artificial Neural Network

API …..…..…..…..…..…..…..…..…..…..…..…..... Application Programming Interface

CNN …..…..…..…..…..…..…..…..…..…..…..…..… Convolutional Neural Network

CPU …..…..…..…..…..…..…..…..…..…..…..…..… Central Processing Unit

DL …..…..…..…..…..…..…..…..…..…..…..…..… Deep Learning

FC …..…..…..…..…..…..…..…..…..…..…..…..…. Fully connected

FN…..…..…..…..…..…..…..…..…..…..…..…..… False Negative

FP …..…..…..…..…..…..…..…..…..…..…..…..... False Positive

GPU …..…..…..…..…..…..…..…..…..…..…..…..… Graphics Processing Unit

IoT …..…..…..…..…..…..…..…..…..…..…..…..…. Internet of Thing

IoU …..…..…..…..…..…..…..…..…..…..…..…...Intersection over Union

LR …..…..…..…..…..…..…..…..…..…..…..…...Learning Rate

ML …..…..…..…..…..…..…..…..…..…..…..…...Machine Learning

ML …..…..…..…..…..…..…..…..…..…..…..…. Machine learning

NN …..…..…..…..…..…..…..…..…..…..…..…. Neural Network

ReLU …..…..…..…..…..…..…..…..…..…..…...Rectified Linear Unit

RESNET …..…..…..…..…..…..…..…..…..…..... Residential Energy Services Network

RGB …..…..…..…..…..…..…..…..…..…..…..... Red, Green, Blue

RNN …..…..…..…..…..…..…..…..…..…..…...Recurrent Neural Network

TN …..…..…..…..…..…..…..…..…..…..…..…True Negative

TP …..…..…..…..…..…..…..…..…..…..…..…. True Positive

VGG …..…..…..…..…..…..…..…..…..…..…...Visual Geometry Group

ABSTRACT

Ethiopia, the largest wheat-producing country in sub-Saharan Africa, relies heavily on wheat as a staple crop essential for food security and economic stability. However, wheat cultivation faces significant challenges due to various biotic and abiotic stressors, including diseases caused by pathogenic fungi, viruses, and adverse environmental conditions. Traditional methods to detecting and classifying illnesses of wheat leaves are labor-intensive, time-consuming, and often inaccurate, necessitating the development of more efficient and reliable approaches. This study explores the application of deep learning techniques for the early detection and classification of wheat leaf diseases, focusing on healthy leaves, brown rust, orange rust, and yellow rust. A custom convolutional neural network (CNN) model, along with pre-trained models such as VGG16, VGG19, InceptionV3, ResNet152, DenseNet201, and EfficientNetB7, was developed and evaluated using a dataset of 4,397 labeled wheat leaf images. Data preprocessing techniques, including resizing, normalization, and augmentation, were employed to improve the model's robustness. The research compared the performance of custom CNN architectures and pre-trained models using accuracy, loss, precision, recall, and inference speed metrics. Fine-tuning the pre-trained models enhanced their performance, with DenseNet201 achieving the highest accuracy of 99.80%. The study found that selecting a model involves balancing accuracy and computational efficiency and addressing challenges such as class confusion through dataset diversity and architecture optimization. The findings demonstrate the effectiveness of transfer learning models for wheat disease detection and offer valuable insights for agricultural applications, suggesting that future research should focus on expanding the dataset, exploring additional architectures, and implementing advanced augmentation methods to further improve model performance and adaptability. The experimental results demonstrate that the developed model can accurately identify and classify wheat leaf diseases. These experiments suggest that deep learning, particularly through transfer learning, can significantly improve the efficiency and precision of wheat disease management.

Keywords: Wheat leaf disease, transfer learning, VGG16/19, InceptionV3, ResNet152, EfficientNetB7, and DenseNet201classification, agricultural technology

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

Ethiopia is the largest wheat-producing country in sub-Saharan Africa[1]. Rainfed wheat is farmed in the mountains of much of Ethiopia by smallholder farmers. The timing of the wheat-growing season varies among wheat-producing regions. Wheat is one of the main crop varieties for the food security of the globe and our country. However, the emergence and spread of wheat leaf and spike diseases have caused serious problems for the agricultural industry. For this reason, several biotic and abiotic stressors can affect wheat grain crops[2]. Pathogenic fungi and viruses, which cause a variety of leaf and root diseases in wheat, are examples of biotic stressors. They can also be nonparasitic illnesses. Plants employ induced systemic resistance and systemic acquired resistance to fend against these diseases, but these are insufficient when stress reaches an excessive level. Abiotic stresses include drought, salinity, and high temperatures. The most common threats to wheat crops can be caused by climate change, pollen, and plant diseases. Agriculture is Among the most important sectors of human life in terms of food, business, and job opportunities[3]. The most widely farmed crop in agriculture is wheat, but every year, several diseases significantly negatively impact its final output. An accurate and timely diagnosis and classification of these disorders are essential for the implementation of targeted interventions aimed at reducing crop losses.

Crop diseases were traditionally identified and detected through a manual inspection conducted by farmers, scientists, and breeders. This process necessitated expertise and knowledge to ensure accurate detection. Over time, conventional crop disease detection methods have become increasingly time-consuming and imprecise, relying heavily on manual observation. The fact that different diseases can induce identical circumstances and have comparable effects on plants further supports the inefficiency of human inspection. Because of this, people required a more appropriate method that would enable them to identify plants more quickly. Accurate identification of plant diseases is crucial for developing effective and sustainable disease management strategies. This guide aims to inform readers on how to recognize wheat diseases and explore potential management options [4]. Therefore, deep learning (DL) involves solving such problems in a more reliable, fast, accurate, and precise way. Because of its excellent performance in image analysis and classification, DL is an end-to-end method that is

extremely necessary for a variety of classification jobs. So, deep learning algorithms is used to detect and classify various wheat diseases. For wheat biotic diseases, deep-learning approach is trained on diverse datasets comprising images of healthy wheat plants and those infected with various diseases. This training process enables the model to learn distinctive features associated with different diseases, facilitating robust and reliable identification during real-time applications.

Plant diseases can occur throughout the different stages of plant development, including seed development and seedling growth[3]. When diseased, plants go through different mechanical, morphological, and biochemical changes. Crop diseases affecting wheat can be broadly divided into two categories: biotic and abiotic[5]. Living things like bacteria, fungi, viruses, and nematodes are the source of biotic illnesses. The following are a few instances of biotic wheat crop diseases: Fusarium ear blight, ergot, black chaff, common bunt, crown rot, and bacterial leaf blight. Conversely, non-living variables such as nutrient shortages, high or low temperatures, dryness, and compacted soil are the root causes of abiotic diseases.

Agriculture plays a crucial role in the national economy, but plant diseases threaten crop quality and incur significant costs for farmers; early diagnosis is essential to maintain crop health and reduce pesticide use, necessitating innovative farming practices. Recent advancements in machine learning, particularly through the use of the EfficientNetB0 deep learning model, have improved the accuracy of diagnosing and classifying wheat plant diseases, showcasing enhanced detection results in ongoing studies. [6]. For agricultural stakeholders, the implementation of this deep learning approach holds great promise, providing advantages like early disease detection, accurate disease classification, and proactive management strategies. Wheat crop illnesses less likely to strike farmers who employ technology to quickly detect and neutralize risks.

wheat production in the highlands of Eastern Ethiopia, specifically focusing on the challenges posed by rust diseases and the strategies to address them[7]. It highlights the importance of wheat for food security and livelihoods in the region. Opportunities such as favorable climatic conditions and market access are discussed alongside challenges including rust diseases, climate variability, and limited resources. Coping strategies encompass various approaches, including breeding for resistant varieties, cultural practices, and policy interventions aimed at enhancing wheat productivity and resilience.

Wheat is a staple crop that is crucial to the food security of humanity. The rise of wheat leaf and stem diseases, however, has made it increasingly difficult for the agriculture sector to sustain high crop

yields. One of the most pressing challenges confronting agriculture and its associated trade is the prevalence of crop diseases. Infectious pests significantly impact agricultural production, leading farmers to closely monitor plants for disease detection, a process that can be time-consuming, costly, and often inaccurate. This paper aims to develop a disease recognition model based on leaf image classification, utilizing image processing techniques in conjunction with a convolutional neural network [8]. Wheat cultivation is the mainstay of agriculture; however, biotic diseases are a threat to crop quality and yield. New biotic stresses have emerged around the globe over the last decades, threatening food safety and security[9].

Deep learning (DL) is a powerful subset of machine learning that employs neural networks with multiple layers to model complex patterns and representations in data. It has gained significant traction across various fields, including computer vision, natural language processing, and speech recognition, primarily due to its remarkable performance. A notable feature of deep learning is its ability to automatically extract relevant features from raw data, eliminating the need for manual feature engineering. For instance, in image recognition tasks, lower layers of a convolutional neural network (CNN) might detect edges, while higher layers can identify shapes or objects. The success of deep learning models often hinges on the availability of large labeled datasets for training, which, combined with advancements in computational power like GPUs, has accelerated the growth of DL applications.

In the context of wheat leaf disease detection, deep learning has proven to be a transformative tool, leveraging its strengths in image analysis. CNNs, in particular, excel at classifying images, enabling models to differentiate between healthy and diseased wheat leaves by learning from labeled images. This automated feature learning capability is particularly beneficial, as traditional methods often rely on manual feature extraction, which can be labor-intensive and subjective. Moreover, deep learning models are adept at handling the variability found in wheat leaves, including differences in color and texture due to environmental factors and disease progression. Techniques such as transfer learning further enhance this application by allowing researchers to fine-tune pre-trained models on large datasets for specific tasks, achieving high accuracy even with relatively smaller datasets of wheat images.

Integrating deep learning into wheat disease detection facilitates real-time analysis and allows for predictive analytics by combining DL with other data sources like weather and soil health data. This multifaceted approach enables the development of models that can forecast disease outbreaks and

inform management practices, ultimately leading to improved crop monitoring and timely interventions. As agricultural challenges intensify due to climate change and food security concerns, incorporating deep learning into crop management strategies promises to enhance productivity and contribute to sustainable farming practices. this research aims to address challenges in the agricultural sector by focusing on automated detection of wheat diseases to improve production. Early and accurate identification of plant diseases on large farms is crucial and can be achieved through image classification powered by deep learning and image processing techniques. Consequently, this study concentrates on developing an automated system that can classify wheat leaf images as either healthy or diseased.

1.2. Motivation

Wheat is one of the principal crops that are staples in this country; wheat supplies a large amount of people's daily caloric intake. Diseases caused by bacteria, viruses, and fungi have a major impact on Ethiopia's wheat yield and productivity. Early identification and categorization of these diseases is crucial for implementing timely preventive measures to safeguard the crop. Consequently, the approach to monitoring crop health may evolve if this technology is leveraged within the agricultural sector for disease detection and classification. The problem is that wheat leaves suffer from the listed diseases. Researchers use digital image processing to identify wheat diseases based on images of infected wheat leaves. This type of disease makes wheat production less than expected from agriculture. The researcher needs to find a solution that can improve the quality of wheat production. Furthermore, wheat is essential to the nation's economic stability and food security. The nation's recent initiatives to increase wheat output through irrigation, mechanized farming, and better seed varieties are encouraging steps toward lowering import dependency and guaranteeing self-sufficiency. Utilizing cutting-edge technologies like deep learning in the agriculture industry can greatly increase production and resilience as the nation works toward wheat self-sufficiency. In this sense, the quick developments in deep learning methods have the potential to be revolutionary. High levels of accuracy can be attained while automating the identification process by using deep neural networks for the detection and categorization of wheat leaf diseases. The successful application of deep learning to wheat leaf disease detection aligns with national efforts to modernize agriculture but also contributes to the development of smart farming systems, which are critical for sustainable growth.

1.3. Statement of the Problem

Ethiopia is one of the largest wheat producers in Africa, as wheat is a vital agricultural commodity fundamental to the nation's food supply and economic stability[10]. However, several illnesses, such as yellow leaf rust, brown leaf rust, and orange leaf rust, pose serious risks to Ethiopia's wheat production. Significant crop losses result from the nation's reliance on antiquated farming practices and restricted access to cutting-edge disease detection technologies

Conventional approaches to wheat plant disease identification and categorization are frequently labor-intensive, time-consuming, and prone to human error. Furthermore, traditional methods might not detect diseases early enough, which would cause reactions to be delayed and disease care to be less effective. More accurate, timely, and scalable solutions are needed to guarantee the nation's food security. It is critical to identify these illnesses as soon as possible to take prompt action to stop productivity loss.

Manual inspection, visual symptom identification, or molecular techniques are the main methods currently used to detect wheat-biotic diseases. Although these techniques can be successful, they are limited in their ability to quickly and precisely process large amounts of intricate agricultural imagery. Our capacity to proactively detect and manage biotic diseases in wheat crops is thus severely lacking, which results in significant financial losses and reduced food production. Furthermore, an advanced and flexible strategy is required due to the dynamic nature of biotic diseases and the growing diversity of pathogens. It is crucial to investigate cutting-edge and technologically advanced solutions because conventional approaches find it difficult to keep up with the rapidly changing field of wheat diseases. While some automated systems for detecting crop diseases already exist, their effectiveness can be constrained by the intricacy of picture processing, the unpredictability of environmental factors, and the requirement for substantial feature engineering. These drawbacks highlight the need for cutting-edge technology that can reliably identify and categorize wheat crop illnesses.

Recent advances in deep learning have made it an increasingly popular approach for detecting and classifying wheat diseases due to its ability to automatically learn features from complex datasets without the need for manual feature engineering. Convolutional Neural Networks (CNNs), in particular, have demonstrated remarkable success in image classification tasks, making them well-suited for agricultural applications, including disease identification. Unlike traditional methods, deep learning models can efficiently process vast amounts of image data, capturing intricate patterns and

subtleties in the visual symptoms of wheat diseases. Furthermore, these models are highly adaptable and can continuously improve as they are exposed to new and diverse datasets, making them ideal for handling the dynamic and evolving nature of biotic diseases in wheat crops. The adoption of deep learning techniques is transforming the field of agricultural disease management, offering the potential for more precise, scalable, and automated solutions.

Creating a successful diagnostic system for detecting and classifying wheat biotic diseases requires carefully selecting the appropriate deep-learning techniques. With numerous options available, it can be challenging to choose the algorithm that best meets the necessary criteria of accuracy, speed, and scale. Obtaining representative training datasets for wheat biotic diseases is particularly difficult but essential for the performance of deep learning models. To address potential imbalances and biases arising from variations in disease prevalence across regions, seasons, and wheat cultivars, the research aims to develop methods that ensure the inclusion of diverse wheat cultivars and disease profiles in the training data. While the introduction of deep learning for wheat leaf biotic disease detection and classification may necessitate significant effort, a key obstacle in this field is the lack of well-annotated datasets, particularly for different wheat species. Incorporating deep learning models into real-world applications can lead to false positives, which can impact the model's accuracy during evaluation. Despite these challenges, convolutional neural networks have demonstrated impressive performance in image classification tasks. The goal of this research is to utilize deep learning techniques to identify and categorize wheat leaf diseases, overcoming the shortcomings of current technologies and conventional approaches.

Wheat disease detection in this case is distinct because of the complexity and variability of biotic diseases affecting the crops. The diseases often manifest as subtle, overlapping, or visually similar symptoms, making manual or traditional methods less effective. Additionally, the high volume of agricultural imagery, often affected by unpredictable environmental factors such as lighting, weather, and soil conditions, further complicates the detection process. The approach also emphasizes real-time, scalable, and precise detection, which is crucial for proactive disease management to minimize crop loss and enhance productivity. Moreover, the constantly evolving nature of wheat pathogens necessitates advanced systems that can adapt to new disease strains, a challenge that traditional methods struggle to address effectively.

Traditional machine learning methods have several limitations when it comes to wheat disease detection. One major drawback is the reliance on manual feature extraction, which requires domain expertise and may fail to capture the full complexity of the disease symptoms present in images. These methods are also typically less efficient at handling large and diverse datasets, which are common in agricultural applications. Additionally, traditional machine learning models often struggle to generalize across different environmental conditions or regions, limiting their robustness and accuracy. They may lack the ability to dynamically adapt to new or emerging diseases, which is critical in the fast-evolving field of plant pathology.

1.4. Research Questions

1. Which pre-trained models are most suitable for classifying wheat leaf diseases through transfer learning?
2. How well-suited is the constructed model for disease classification?

1.5. Objectives

1.5.1. General Objective

The main objective of this study is to construct an optimal deep learning-based model for disease classification.

1.5.2. Specific Objectives

To accomplish the general objective of the study, the following specific objectives are identified:

1. Organize and label a collection of wheat leaf photos that depict both healthy and unhealthy circumstances.
2. Pre-processing the image, including resizing, cropping, augmentation, and normalization.
3. Selecting an appropriate deep learning model that is reliably identify and categorize diseases in wheat crops.
4. Fine-tuning model's adaptability to detect and classify emerging or previously undocumented wheat leaf diseases.
5. To evaluate the performance of the developed model using the test data set.

1.6. Scope and Limitation of Study

This study is confined to the categorization and identification of diseases, focusing solely on the four most prevalent classes: these are healthy wheat leaves, yellow rusty wheat leaves, brown rusty wheat leaves, and orange rusty wheat leaves. By employing advanced image processing and deep learning methodologies, the research encompasses the complete process from image capture to the application of various deep learning models for effective classification. Utilizing frameworks like convolutional neural networks, the study evaluates model efficacy in disease classification, detailing the training, validation, and testing phases to provide insights into performance metrics. However, the study acknowledges certain limitations, including its narrow focus on specific diseases and potential geographical constraints that may affect the generalizability of its findings. The only objective of this experiment is to classify photos of these leaves, while the categorization of roots, spikes, or other disorders is not addressed. Additionally, estimating the severity of a detected disease and recommending the best course of treatment are beyond the scope of this research project.

1.7. Significance

This study seeks to develop and implement a deep learning methodology tailored to the complex of biotic diseases affecting wheat leaves, to address these challenges.

This research provides data-driven insights that enhance the understanding of detecting and classifying wheat leaf biotic diseases. Farmers can leverage the deep learning-based experiments to identify and mitigate crop losses caused by diseases. Additionally, the proposed technique offers farmers a reliable and efficient method for recognizing wheat illnesses in real-world applications. Farmers able to detect and prevent crop losses caused by diseases thanks to deep learning-based research on the detection and identification of wheat diseases. Furthermore, farmers have a trustworthy and efficient method of identifying wheat illnesses with the help of the suggested technique when it comes to real-world applications. Furthermore, by facilitating the early and precise identification of diseases in wheat crops, the deep learning approach for the classification of wheat biotic diseases has the potential to transform agricultural practices, improve crop management techniques, and support food security. For experts, it lessens confusion, enhances expertise, and lightens the workload of professionals who work with wheat biotic crops. The study provide access to researchers interested in the field of agricultural technology.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

Accurate and efficient identification of diseases is critical for effective crop management. Consequently, there has been significant interest in utilizing deep learning techniques to detect and classify wheat leaf diseases. This literature review examines the most important research and advancements in this area. As scientific investigations require empirical evidence, the study involved reviewing relevant journal articles, papers, and previous theses related to the topic.

Primarily, this chapter explores the diseases that affect the leaves of wheat crops and the digital image processing techniques used to detect these diseases. Also, this chapter covers a wide range of symptoms and offers a comprehensive analysis of diseases that impact wheat leaves.

Wheat is one of the most extensively grown and significant cereal crops globally. It is highly versatile, with the grain utilized in various applications, including baking flour, pasta, breakfast cereals, and beer production. In addition to its nutritional value, wheat also serves as an essential ingredient in animal feed and various industrial processes.

Throughout the world, wheat is a major cereal crop that is extensively grown and consumed[10]. Wheat is grown in a wide range of regions, including tropical and temperate ones. To reach its maximum potential, this cool-season crop must be subjected to cold temperatures for an extended period of time throughout the verbalization process. In contrast, a number of challenges, chief among them the consequences of climate change and the reduction of input needs in other sectors, impede worldwide wheat productivity. wheat (Triticum aestivum) is considered a vital food crop globally and its significance is increasing due to political issues between Russia and Ukraine[11].

Figure 2.1 Wheat sprouts North Shewa Zone Irrigation and Lowland Bureau

2.2 Wheat in Ethiopia

Among Africa's top producers of wheat crop is Ethiopia. In actuality, it is sub-Saharan Africa's second-largest producer of wheat, only ahead of South Africa. For the Ethiopian government, the wheat subsector holds strategic importance as it may be used for many culinary applications using both contemporary and traditional processing methods [12]. Widely grown in the central and highland regions of Ethiopia, wheat is a significant cereal crop. the most significant cereal crops, wheat has significant economic potential to provide food security; nonetheless, a number of production issues have led to low wheat production among smallholder farmers.

2.3 Wheat disease

Diseases affecting wheat production include several issues, such as pests, water scarcity, climate change, and sustainable agricultural practices. Droughts and floods are among the extreme weather phenomena that are becoming more frequent in Ethiopia. The region's agricultural output and food security could be threatened by variations in the weather and climate, particularly considering the unpredictable[13]. Efforts to create climate-resilient wheat varieties, increase resource efficiency, and support sustainable agricultural practices are ensuring the supply and production of the future wheat

crop. In addition, improved research and breeding activities, including high production potential, disease resistance, drought tolerance, and nutritional quality, are being done to create improved wheat varieties, but early detection and prevention of various late diseases should be done in depth. Therefore, leaf rust is one of the wheats rust a global wheat disease. This is an extremely dangerous disease that can cause significant crop losses, especially in environmentally favorable wheat-growing regions of the world. In Ethiopia, one of the major wheat-producing nations in Africa, wheat rust is a significant biological barrier to wheat production. Smallholder farmers' livelihoods are threatened by fungal infections, which result in financial losses. Although wheat rust epidemics are known to have occurred in Ethiopia, there has been no systematic long-term analysis of past outbreaks[1]. In Ethiopia, wheat leaf rust is one of the most problematic diseases in wheat-growing areas, resulting in yield losses of up to 75% of wheat varieties found in hot areas[14]. This review examines the economic significance, epidemiological characteristics, geographic range, life cycle, and recent research on wheat leaf rust disease. It also discusses management approaches including cultural, chemical, biological, and host resistance strategies. Additionally, it provides information on the defense system, including details on the types and sources of defenses.

2.4 Wheat Leaf rust disease

Among the first crop diseases identified, wheat rust is still an issue. Though we still have localized crop failures, scientific research provides information that helps us better control the illness and prevent global crop losses. Rust refers to several fungal diseases that can infect wheat. These diseases are some of the most difficult wheat diseases worldwide, causing significant yield losses. The two most common types of wheat rust are leaf rust and stem rust. These fungi can spread by wind, water (from irrigation or rain), insects, animals, humans, or soil. They can also spread by seed or soil.

This fungus spreads through airborne spores, making it highly contagious and difficult to control. The fungus produces different types of spores during its complex life cycle, ensuring its survival and continued threat to wheat crops. The vulnerability of the host, the inoculum's density, the surrounding temperature, and other environmental variables all affect the likelihood of fungal pathogen infection. It's normally necessary to have free water on the surface of the host plant. Some fungi target a small number of host species, whereas others target numerous host species. The host-parasite relationship determines the disease's course and its symptoms. Depending on the fungus implicated, symptoms can be similar or different. Fungi should consequently be identified mostly by their morphology, while

some groups benefit from the use of molecular techniques. Affected triticale, durum wheat, and bread wheat diseases are caused by the fungus listed in this field handbook, unless otherwise specified.

Wheat leaf diseases are called thus because the diseased have tiny, rounded, yellow, brown, or orange spots that frequently resemble rust. The degree of severity of a specific variety of leaf rust might vary depending on wheat seeds, resistance genes, and many environmental conditions. There are various ways in which wheat types might be impacted by the leaf rust disease. crop breeders so attempt to develop rust-resistant cultivars. Leaf rust must first be recognized as a distinct disease type before integrated rust management techniques can be used to manage it and lessen its detrimental effects on wheat productivity.

Wheat leaf rust diseases included in this antiquity are yellow, brown, orange, which show the following symptoms:

2.4.1. Yellow rust of wheat

Wheat producers around the world are very concerned about yellow rust, particularly in areas with moderate weather. To lessen the damage yellow rust does to wheat production, careful monitoring and control are needed. Yellow rust is a devastating disease that affects wheat harvests. Yellow-colored streaks in the veins of the leaves are indicative of the disease and produce yellow. Significant yield losses may result from it, particularly in areas with mild winters and cool, wet summers. Generally, varietal resistance and fungicide treatments are used in conjunction to manage yellow rust. Since the disease is extremely adaptive and can elude resistance genes, it's critical to monitor the emergence of new racial groups. Farmers need to keep a close eye on their crops and take precautions to protect the wheat seeds from this disease.

2.4.2. Brown rust of wheat

Wheat leaf brown rust is a fungal disease of wheat caused by Puccinia horde. Epidemics of brown rust in wheat typically begin sooner in the spring than in other crops. Winter wheat is typically more susceptible to brown rust than spring wheat, particularly in harvests grown early in warm winters. Humidity and moderate temperatures (15–22°C) are ideal for brown rust growth. Because brown rust is hindered at temperatures over 25 °C and killed at temperatures below 5 °C, inoculum is reduced by hot summers and freezing winters. On dry, windy days, spores may multiply more easily. Although severe winters and scorching summers might help reduce the inoculum (the amount of infectious material), fungicides and resistant wheat cultivars are the main ways to prevent brown rust. In terms of

how it spreads, brown rust differs greatly from yellow rust. Since the spores that are germinating include sensing and orienting systems that help them locate the best entrance places into the leaf, they are around eight times more efficient at entering the plant than yellow rust. It's usually more difficult to control brown rust than yellow rust.

2.4.3. Orange rust of wheat

Among the three main rust diseases that impact wheat crops are orange rust. It primarily affects wheat crops, leading to significant yield losses. This disease is difficulties because, depending on the variety's sensitivity, the race of the pathogen present, the timing of infection, and the weather, it can spread quickly and lower wheat output and quality. In epidemic years, it can seriously reduce yield as it exclusively targets foliage. Anywhere wheat, barley, and other cereal crops are cultivated, the disease is present. Russet-colored pustules that protrude through the crop surface are a common indicator of all rusts. By rubbing or spreading the rust spores with your finger over the leaf surface, you can differentiate them from other leaf diseases. Differences in the color, size, and distribution of the pustules on the plant's surface, as well as the typical location of the infection on the crop, can be used to differentiate between leaf rust, stripe rust, and stem rust on wheat.

2.4.4. Wheat Health leaf

Wheat health leaf could refer to various aspects related to the health and well-being of wheat plants. Healthy leaves are typically green and free from discoloration, spots, or damage caused by pests or diseases. Environmental factors are drought, heat stress, or excessive moisture can impact the health of wheat leaves. Symptoms of stress may include wilting, leaf curling, or necrosis. Monitoring the health of wheat leaves is essential for diagnosing any issues that may affect plant growth and yield. Evaluating wheat leaves' health requires an understanding of their structure and function. The primary purpose of a leaf is to use photosynthesis to create nourishment for the crops. The green hue of crops is attributed to the absorption of light energy by chlorophyll. The stem epidermis and leaf epidermis are connected, protecting the leaf's internal structure. The functions of leaves in a plant include photosynthesis, nutrition intake, and water management. Wheat yield and health may be significantly impacted by any changes to the structure or function of the leaves.

Figure 2.2 Sample image of Wheat crop Leaves[15]

2.5 Artificial intelligence

Like biology or mathematics, artificial intelligence is a branch of science. It has long been seen as a human right and investigates methods of creating machines and intelligent algorithms that are capable of coming up with original solutions to issues. The process of transferring human intelligence into machines (algorithms) is known as artificial intelligence[16]. Artificial intelligence is the field that investigates the development of machines capable of exhibiting intelligent behavior akin to human cognition and reasoning. It combines the concepts of "artificial" or man-made with "intelligence," which refers to the capacity to comprehend and think effectively. Fundamentally, AI research focuses on training machines to emulate the cognitive abilities and thought processes of the human brain.

Artificial intelligence has become an essential tool in agricultural practices, enabling farmers to effectively identify and manage crop diseases, including wheat leaf spot. By examining this context, we can understand the pivotal role of AI in the pre-diagnosis and diagnosis processes. AI-powered systems possess the capability to analyze images of wheat leaves, detecting disease symptoms at an early, often imperceptible stage to the human eye. This timely diagnosis facilitates prompt interventions to curb the spread of the disease and minimize crop losses. Through the application of machine learning

algorithms trained on extensive datasets of both healthy and diseased leaves, AI systems can discern subtle signs and patterns associated with various plant diseases[16].

Precision agriculture; To control crop health at scale, AI technologies, such as machine learning and deep learning, can be linked with precision agriculture systems. Farmers can observe their crops' health in real time, including the detection of illnesses on wheat leaves, by using sensors, drones, or other imaging devices with AI algorithms installed. This targeted approach allows farmers to apply pesticides or fungicides only where they are needed, reducing costs and reducing environmental impact. Decision support systems; AI can help farmers make informed decisions about disease management strategies. Artificial Intelligence (AI) systems analyze a range of data, including weather, soil moisture content, past disease status, and crop health information, to suggest the best ways to manage and stop the spread of diseases in wheat crops. These decision support tools aid in raising overall agricultural production, maximizing crop yields, and allocating resources optimally. Data-driven understanding: Research and policymakers can gain important insights from AI-driven analysis of agricultural data, including information on crop genetics, environmental factors, disease incidence, and management techniques. Through extensive data collection and analysis, AI can discover trends, relationships, and predictive patterns that can help build a better understanding of wheat crop diseases and inform disease prevention and mitigation strategies. Mainly, AI technologies have great potential to revolutionize the detection, management and prevention of diseases in wheat crops. It uses advanced machine learning and data analysis techniques to provide farmers with timely, accurate and actionable information to protect their crops and optimize agricultural practices.

2.6 Machine Learning

Artificial intelligence featuring machine learning enables computer systems to learn from data and improve performance without explicit programming. Leveraging statistical models and algorithms, this approach facilitates in-depth data analysis, pattern recognition, and predictive capabilities that are highly valuable for managing wheat crop leaf diseases.

Machine learning algorithms can scrutinize expansive datasets of visual information showcasing both healthy and diseased wheat foliage to precisely pinpoint symptoms linked to diverse maladies. By learning from labeled exemplars, ML models can differentiate between unaffected and infected leaves, facilitating the early identification of diseases prior to the manifestation of overt symptoms. This preemptive detection empowers farmers to swiftly intervene, impeding the proliferation of diseases and

minimizing crop yield detriments. Furthermore, ML-based systems can eliminate the requirement for manual inspection and intervention by automating the monitoring and surveillance of wheat crops for disease indicators. Farmers can continuously monitor vast agricultural areas and identify anomalies suggestive of disease outbreaks by deploying sensors, cameras, drones, or satellite imagery outfitted with machine learning algorithms. In order to customize treatment plans for particular disease outbreaks, machine learning approaches can evaluate a variety of datasets that include data on crop genetics, environmental factors, soil characteristics, and disease prevalence. ML-based optimization techniques can be assisted in scheduling preventive measures, such as crop rotation or planting disease-resistant varieties, to mitigate the risk of future outbreaks. ML models can continuously learn and adapt to evolving disease patterns and environmental conditions through feedback loops and ongoing training

2.7 Deep learning

Deep learning, also known as deep neural learning, is a branch of machine learning that leverages neural networks to examine various elements through a structure resembling that of the human brain. Because of its capacity to absorb vast amounts of data, uncover complex patterns, and generate highly accurate predictions, deep learning (DL) is therefore very important in the field of wheat crop leaf disease control.

Deep Learning models, particularly Convolutional Neural Networks, excel at image recognition tasks, rendering them well-suited for accurately identifying subtle visual cues associated with various wheat crop leaf diseases. By being trained on extensive datasets of labeled images depicting healthy and diseased leaves, DL models can learn to differentiate between distinct types of diseases and precisely classify infected leaves, even in cases where symptoms are not visually apparent to human observers. In the context of wheat crop leaf disease detection, DL models can identify distinctive patterns, textures, and spatial relationships within leaf images that are indicative of specific diseases, without requiring explicit human-defined features. Consequently, this high level of accuracy in disease identification enables prompt and targeted interventions to control disease spread and minimize crop losses.

Deep Learning algorithms are highly scalable and capable of handling large-scale datasets and complex computational tasks. it can process vast amounts of image data collected from extensive agricultural areas, facilitating comprehensive monitoring and surveillance of wheat crops for disease outbreaks. Moreover, Deep Learning models can undergo continual learning and adaptation to evolving disease patterns and environmental conditions. By leveraging techniques such as transfer learning and online

learning, DL models can incorporate new data and insights over time, improving their accuracy and predictive performance in detecting and diagnosing wheat crop leaf diseases. This adaptability ensures that DL-based systems remain effective and reliable even as disease dynamics and agricultural practices evolve. Therefore, Farmers can mitigate the economic and environmental impacts of crop diseases, optimize resource utilization, and enhance food security for communities worldwide.

Accurate plant disease identification is the first step in designing effective and sustainable disease management programs[4]. Plant diseases can significantly impact the development of their host species, thus underscoring the importance of early disease detection. Numerous Machine Learning techniques have been utilized to identify and classify various plant diseases. However, recent advancements in a specific branch of ML, known as Deep Learning, have shown great promise in enhancing the accuracy of these disease detection and

classification efforts. [17]. Wheat diseases pose significant challenges to wheat production, but certain algorithms can effectively identify common afflictions of wheat leaves. The wheat disease spectrum encompasses a range of viral, bacterial, fungal, insect-borne, and rust-based pathologies. Numerous disease types are prevalent in wheat foliage[18]. The proposed deep learning framework was executed on the Jetson accelerator to minimize the computational complexity involved in processing image data. Image-based plant disease recognition assists agricultural experts in the early identification of diseased crops, thereby mitigating the challenges associated with yield loss. Diseases like leaf rust, stem rust, and strip rust negatively impact wheat crop production and yield, while manual identification of these diseases is time-consuming and relies heavily on expert knowledge. to address these challenges, deep learning models (Inceptionv3, Resnet50, VGG16/19) were developed in collaboration with the Bishoftu Agricultural Research Institute, achieving an impressive 99.38% accuracy in classifying wheat diseases using RGB image data. However, the current crop disease monitoring and management systems face several limitations in providing input for decision-making processes [19].

Agriculture in contemporary farming practices is one of the domains where the integration of IoT and automation can have substantial influence. Preserving the health of plants and closely monitoring their environmental conditions to identify or detect any diseases is essential for maximizing crop yields. The adoption of cutting-edge technologies, such as artificial intelligence, machine learning, and deep learning, has proven to be extremely valuable in modern agriculture as a means of advanced image analysis and processing [3]. Conventional methods employed for identifying and categorizing diseased

plants rely on field surveys, which are labor-intensive, monotonous, and can lead to numerous errors and unreliable outcomes [20].

Wheat production is significantly impacted by fungal diseases known as wheat rusts. Leaf rust can lead to crop losses of 45-50%, while stem and stripe rust have the potential to destroy up to 100% of the crop under favorable environmental conditions.[21]. Owing to population expansion, the demand for sustenance is consistently rising. To address this exigent requirement, it is imperative to augment agricultural productivity and safeguard crops. However, crops remain highly vulnerable to diverse maladies due to the plethora of pathogens present in their milieu [22]. Agriculture represents a significant revenue stream and contributes substantially to national economies in numerous regions globally. A key priority for this industry is mitigating or eradicating plant diseases, which not only diminish crop quality but also impose financial burdens on farmers [6]. Wheat is a crucial agricultural commodity, supplying sustenance for billions globally. However, an array of pathogenic afflictions jeopardize its cultivation and output, potentially precipitating considerable declines in wheat productivity [23].

Identifying crop diseases is a critical task that farmers routinely undertake and address through appropriate actions, as these diseases can have detrimental impacts not only on the crops themselves but also on farmers, consumers, and the environment. Ensuring the quality and safety of agricultural products is a paramount concern in the current context [24]. Wheat is widely regarded as one of the most valuable food resources and staple crops in numerous nations, particularly throughout Africa and Asia [25]. Pathogenic fungal diseases affecting cereal crops can substantially diminish yields. Many populations are susceptible to these diseases. Controlling such diseases on a large scale is challenging; therefore, one relevant approach is monitoring crop fields, which facilitates early disease detection and the implementation of preventive measures. One effective control method involves disease identification through the analysis of digital images, which can be captured in field settings using mobile devices [26].

The necessity of automatically identifying wheat illnesses using deep learning techniques based on this paper, we propose and compare models based on a convolutional neural network (CNN) for wheat disease detection and recognition. The convolutional layers of a CNN can be considered as matching filters derived directly from data images (images of healthy and unhealthy wheat)[27]. Wheat is the predominant cereal grain globally in terms of production and consumption. However, a substantial

portion of the wheat crop is compromised due to disease pressures. With more than two dozen detrimental wheat diseases, the manual identification of these conditions poses a significant challenge. Consequently, an automated wheat disease classification system could be instrumental in enhancing the quantity and quality of wheat yields [28]. Plants and crops are a vital energy source and a great answer to the global warming problem, and such plants are affected by various diseases that cause high damage to the economy, society, and ecology. Hence accurate and timely identification of the disease is the most significant issue[29]. Various deep learning architectures exist. For the task of wheat leaf disease detection and classification, the researcher has chosen to employ convolutional neural networks.

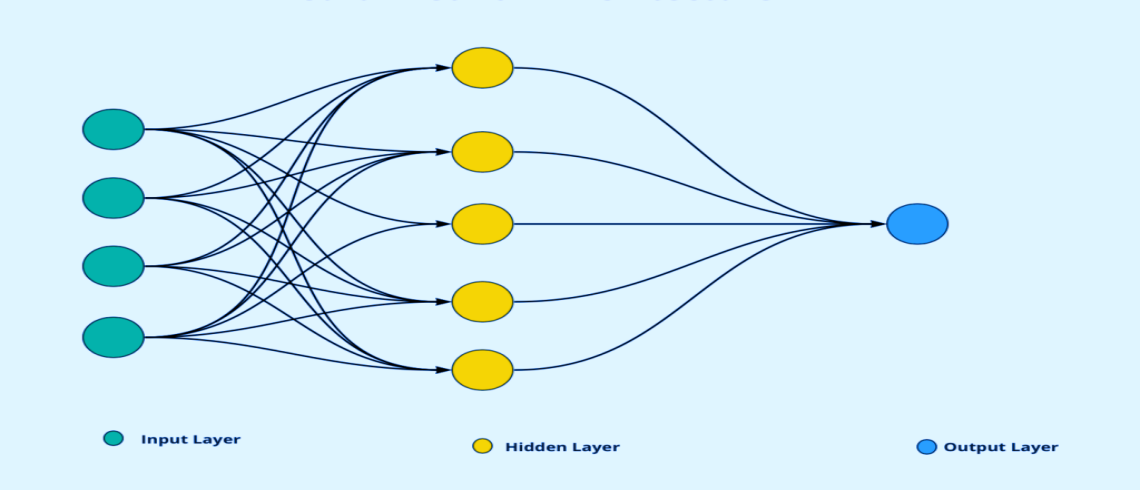


Figure 2.3 Neural Network Architecture Layer adapted from the web adapted from [30].

Why did the researcher decide to use a convolutional neural network?

Researchers have selected convolutional neural networks as a primary tool for tackling image processing tasks, such as image recognition, object detection, and classification. CNNs can be more successful in these endeavors because they can model by filtering the entire image on a pixel basis and combine this information with deep learning techniques.

CNNs are designed to automatically learn hierarchical representations of features directly from raw input data, like images. Each layer in a CNN learns increasingly abstract and complex features, capturing patterns at different levels of granularity. CNNs leverage convolutional layers that preserve the spatial structure of the input data. They are more computationally effective and scalable, especially for large-scale image datasets, as they use sparse connections between layers and share weights across spatial regions, reducing the number of parameters and calculations needed. Additionally, CNNs

inherently possess translation invariance properties, enabling them to recognize patterns regardless of their position or orientation within the image. This property is beneficial for tasks like object recognition, where objects may appear in different locations and orientations.

Transfer learning techniques have shown that pre-trained CNN models, such as those trained on large-scale image datasets like ImageNet, can be fine-tuned for specific tasks with relatively small datasets. CNNs have demonstrated state-of-the-art performance in various image processing tasks, including image classification, object detection, semantic segmentation, and image generation. Academics and practitioners in computer vision have come to rely on CNNs due to their ability to extract sophisticated and discriminative features directly from raw pixel data, making them a valuable tool for addressing image analysis and understanding challenges.

2.7.1 Convolutional neural network

Convolutional neural networks have emerged as a prominent technique in contemporary image processing and computer vision, enabling machines to interpret and understand visual information with exceptional accuracy and efficiency. Unlike previous computer vision methods, CNNs do not require extensive preprocessing and can operate directly on raw image data. CNNs are characterized by a feed-forward neural network architecture with multiple convolutional layers stacked on top of one another, each capable of recognizing increasingly complex shapes. This unique convolutional layer structure is the key to the power of CNNs. Researchers have leveraged the capabilities of CNNs to tackle wheat crop-related tasks, such as disease detection and yield prediction, by training the networks on labeled datasets of healthy and diseased wheat leaves or fields. The ability of CNNs to effectively process and analyze image data makes them a highly promising tool for advancing wheat crop research and management, enabling enhanced disease detection, yield forecasting, and other agricultural applications.

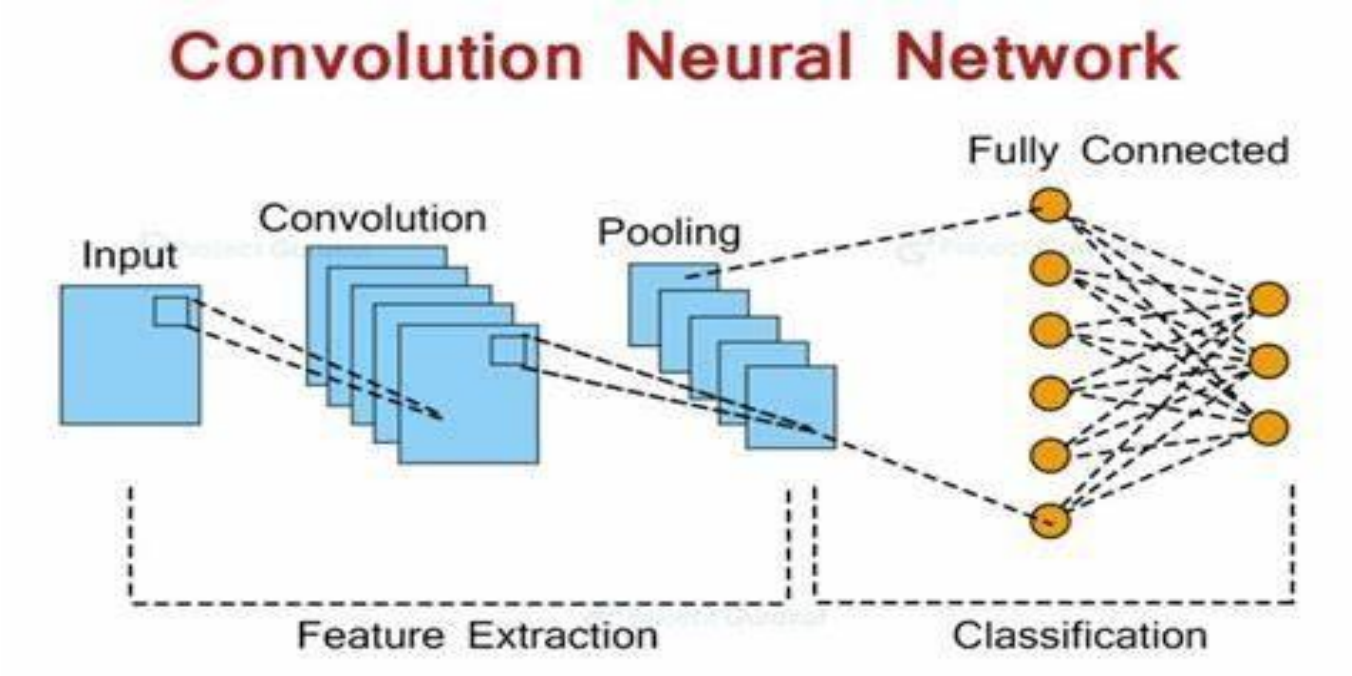


Figure 2.4 Convolution neural network architecture [31]

2.7.2 Convolution Layer

The convolutional layer represents the initial and most crucial component of a convolutional neural network architecture. This layer serves as the central element, where the majority of computational processes occur. It requires three key inputs: a feature map, a filter, and the input data. Assuming a color image composed of a 3D pixel matrix as the input, the input three dimensions - height, width, and depth - corresponding to the RGB color space. Another crucial tool is the feature detector, also known as a kernel or filter, which scans the image's receptive fields to identify the presence of specific features. These convolutional layers employ filters, or kernels, to apply convolutional operations on the input images, enabling the detection of various features such as textures, edges, and more complex patterns. The use of convolutional techniques preserves the spatial associations between pixels. During the training process, gradient descent and backpropagation are utilized to modify certain parameters, such as the weight values. However, the output volume size is dependent on three hyperparameters that must be established prior to the neural network training.

The depth of the output is determined by the number of filters applied. For example, using three distinct filters would result in three separate feature maps, thereby generating an output with a depth of three. The stride, or the number of pixels the kernel moves across the input matrix, is a known quantity. A

larger stride leads to a reduced output size, although stride values exceeding two are uncommon. In most scenarios, zero-padding is utilized when the filters are too large for the input image. This approach ensures that the output size is equal to or larger than the input by setting the values outside the input matrix to zero. Valid padding, also known as no padding, is when the dimensions do not align, and the final convolution is discarded[32]. Conversely, the same padding technique guarantees that the output layer size matches the input layer. Full padding involves filling the input border with zeros to increase the output size. After each convolution operation, a Convolutional Neural Network applies a Rectified.

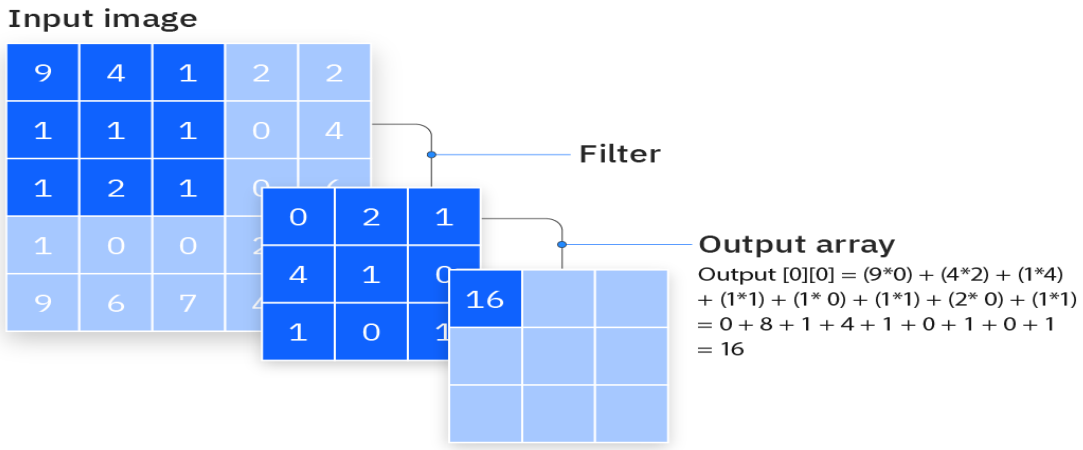


Figure 2.5 Convolutional layer[33]

2.7.3 Pooling Layers

Pooling layers in convolutional neural networks serve to downscale the spatial dimensions of the input, thereby reducing computational complexity and the number of network parameters. One common pooling technique is max pooling, which selects the largest value from a group of neighboring pixels. The pooling operation involves applying a filter across the entire input, akin to the convolutional layer, but without applying any weights to the filter. Instead, the output array is populated by the kernel, which applies an aggregation function to the values within the receptive field. In max pooling, the filter identifies and outputs the pixel with the highest value as it moves across the input. This method is often employed more frequently than standard pooling approaches. Another pooling technique is average pooling, where the filter calculates the average value to be sent to the output array as it traverses the input. The pooling layer offers several benefits to CNNs, despite the loss of some information. These advantages include reduced overfitting, increased efficiency, and decreased complexity.

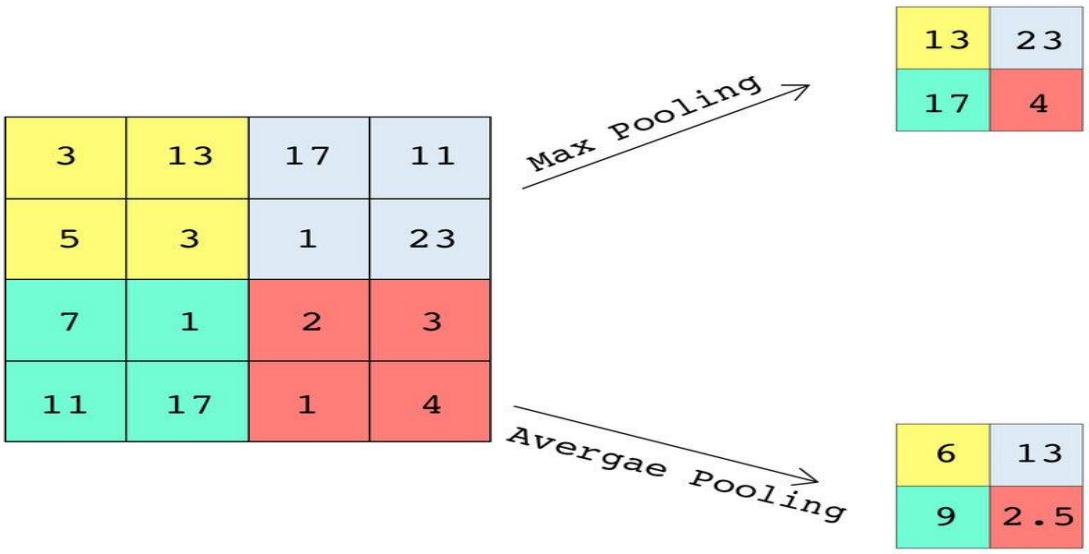


Figure 2.6 Max pooling vs Average pooling layers[34]

2.7.4 Fully Connected Layers

The final layers of the neural network architecture make predictions using the high-level features learned by the preceding layers. Each neuron in one layer is connected to every neuron in the next layer. The fully connected layer, named appropriately, directly connects the neurons in the previous layer to the output neurons. This layer utilizes the features extracted by the earlier convolutional and pooling layers, which typically employ ReLU activation, to classify the input data. In the fully connected layer, the output of the previous layer is multiplied by a weight vector, and the result is the final output. The number of output neurons in the fully connected layer matches the number of classes to be recognized. Each fully connected layer is followed by a non-linear activation function.

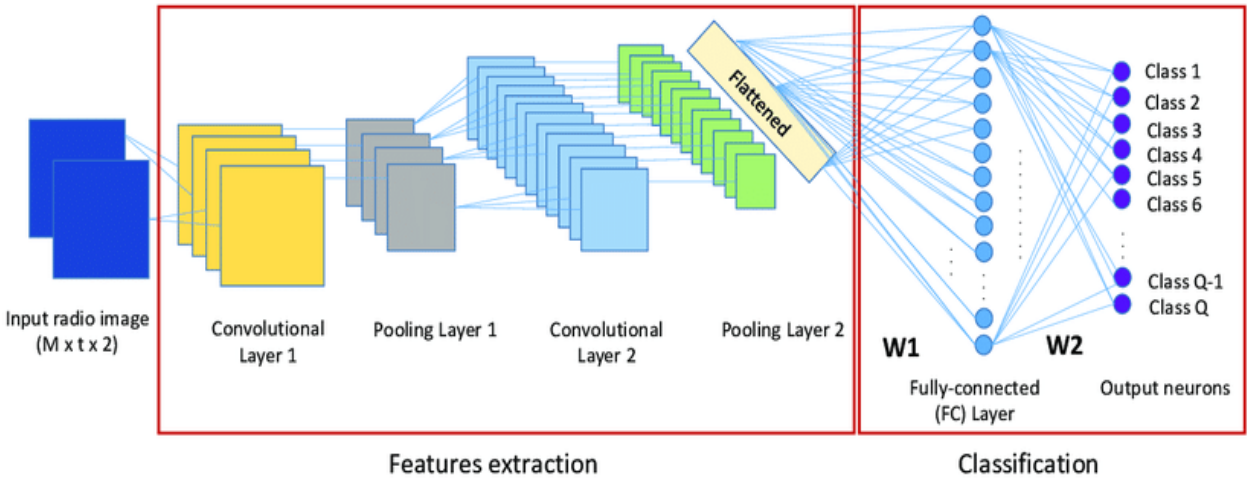


Figure 2.7 Fully Connected Layer is adapted from [35]

2.7.5 Activation Functions

Neural network layers require activation functions, which introduce non-linearities and enable the network to uncover complex relationships within the data. Activation functions are fundamental building blocks for neural networks, as they facilitate the identification of intricate patterns in the input data. In a neural network, these functions transform a node's input signal into an output signal passed on to the subsequent layer. Without activation functions, neural networks would be restricted to modeling only linear relationships between inputs and outputs. The introduction of non-linearity via activation functions allows neural networks to learn highly complex mappings between inputs and outputs. Selecting the appropriate activation function is crucial for developing neural networks with strong predictive performance and robust generalization capabilities. Neural networks would only be made up of linear operations like matrix multiplication if activation functions weren't there. The input would be transformed linearly by each layer, with no non-linearities added. By incorporating non-linear behaviors through activation functions, neural networks are able to learn these non-linear correlations. This significantly boosts neural networks' adaptability and capacity to represent intricate and nuanced input. Neural networks use several kinds of activation functions to introduce non-linearities and make it possible to learn intricate patterns. Every activation function has special qualities of its own and works well in specific scenarios. For instance, the sigmoid function works well in binary classification scenarios, SoftMax facilitates multi-class prediction, and ReLU aids in solving the vanishing gradient issue[36].

Rectified Linear Unit Activation Function

The Rectified Linear Unit is a commonly used activation function in convolutional neural networks within deep learning models. It is typically employed in the hidden layers of the network, as it can help mitigate overfitting and facilitate the learning of complex decision boundaries. ReLU's mechanism involves setting negative input values to zero, while passing positive values unchanged. This straightforward and non-saturating approach avoids the Vanishing Gradient Problem. However, ReLU is not without its limitations. The immediate zeroing of negative values can impair the model's ability to properly fit the training data, and the unbounded nature of the activation function can potentially lead to exploding activations, rendering some nodes unusable.

SoftMax activation

Artificial neural networks commonly employ the SoftMax activation function, which is particularly crucial for multi-class classification tasks. This function, also known as the normalized exponential function, is especially beneficial when applied to multi-class classification problems. The SoftMax function operates on a vector of raw predictions or scores for each class, calculated by the earlier layers of the neural network, which are sometimes referred to as logits. In terms of mathematics, the SoftMax activation function accepts as input a vector of real numbers and outputs another vector of the same dimension, with values in the range of 0 to 1. The sum of these values equals 1, which denotes real probabilities. The formula for the SoftMax activation function is as follows[37]:

$$P(y = i) = \frac{e^{zi}}{\sum_{j}^{k} i = \left(1^{e^{zj}}\right)}$$

- (P(y = i)) is the probability that the input belongs to class (i).
- (z _ i) is the raw score or logit for class (i).
- The denominator is the sum of exponentiated logits for all classes, ensuring a valid probability distribution.

The output of the SoftMax function is a probability distribution that sums up to one. Each element of the output represents the probability that the input belongs to a particular class. Because SoftMax amplifies differences, it can be sensitive to outliers or extreme values. SoftMax can "squash" the probabilities of other classes, leading to an overconfident model.

2.7.6 Loss Functions

One of the most crucial components of neural networks are loss functions, which work directly with optimization functions to fit the model to the provided training set. It is aware that the output layer, which performs the final classification, is the last layer in any CNN design that is based on classification. Using a loss function, we compute the prediction error that the CNN model produced throughout the training set in this output layer. As the CNN model learns, this prediction error—which tells the network how far its forecasts differ from the actual output get better.

The two factors that the loss function uses to calculate the error are the estimate output of the CNN model, also known as the prediction, and the actual output, also known as the label[37]. The performance of the model is assessed using the loss function. It determines how much the real data differs from the model's predictions. Your choice of loss function depend on the kind of problem you're trying to solve. Categorical cross entropy and sparse categorical cross entropy are the two most widely used methods for multi-class categorization.

2.7.7 dropout layers

With its ability to function well in fully connected layers, dropout has become more and more popular in deep learning. In the network, it brings regularization. Through the random skipping of some units or connections with a predetermined probability, this ultimately enhances generalization. With a low weight, it ultimately chose one symbolic network. Performance on the network is improved by the dropout layer. Considerably more popular in deep learning research and applications are its simplicity, effectiveness, and ease of implementation.

The purpose of the Dropout layer is to prevent overfitting by randomly setting the input units to 0 at a specific frequency at training time. The Dropout layer is only activated when the training is set to True in call (), which indicates that no values are dropped during inference. When using model, training set to True by default. Perfect. You can specifically specify the argument to True when calling the layer in other contexts. If you have a Dropout layer, an alternative is to set trainable=False. Trainable has no effect on the behavior of the layer because Dropout has no variables or weights that may be adjusted while training.

Rate: Float between 0 and 1. Fraction of the input units to drop.

Noise shape: 1D integer tensor representing the shape of the binary dropout mask that multiplied with the input. For instance, if your inputs have shape (batch size, time steps, features) and you want the dropout mask to be the same for all time steps, you can use noise shape= (batch size, 1, features).

Seed: A Python integer to use as a random seed.

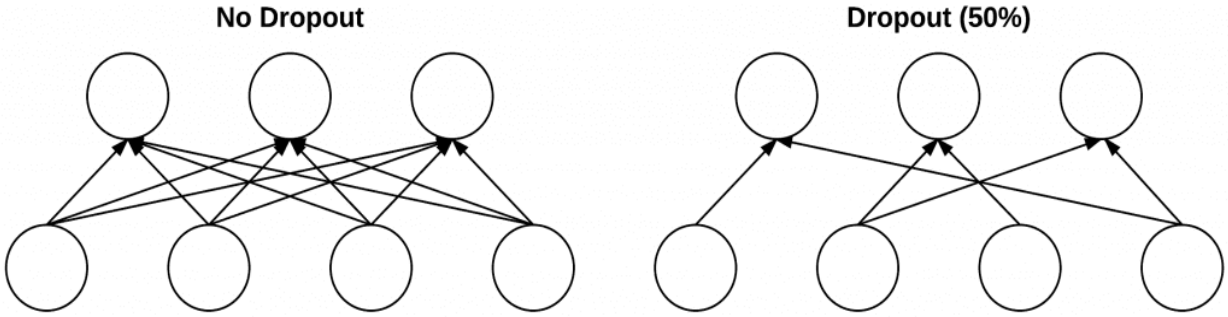


Figure 2.8 dropout layers[38].

2.7.8 Batch Normalization and Epochs

Batch normalization enables autonomous learning at each layer of the neural network by applying normalization across the layers rather than on the raw input. By applying a transformation, batch normalization keeps the output standard deviation close to one and the mean output close to zero. Rather of using the entire dataset, it is done along mini-batch. Through normalizing of the input layer by re-centering and re-scaling, artificial neural networks can be made faster and more stable. Moreover, by deducting the mini-batch mean and dividing it by the mini-batch standard deviation, it can be normalized to the input. Using the feature map, the internal covariance shift is addressed. By adjusting the feature map values to zero mean and unit variance, it unifies their distribution. By using a larger learning rate and simplifying the learning rate, batch normalization helps to accelerate training.

One full run through the entire training dataset during the training process is referred to as an epoch in the context of machine learning and neural networks. To put it another way, it's similar to going through every sample in your training data once. The model learns from the data at each epoch and modifies its parameters (weights and biases) in order to minimize the error or loss function. The model's performance is usually assessed on a different validation dataset at the end of each epoch to track how well it generalizes to new inputs. Until a predetermined number of epochs is reached or the model's performance on the validation dataset stops improving, signifying convergence, this process is repeated

several times. While a model's performance may occasionally be enhanced by increasing the number of epochs, doing so may also cause overfitting if the model begins to learn specific patterns from the training data instead of broad ones. One of the most crucial aspects of training neural networks is epoch balance.

2.8 Convolutional Neural Network Architectures

Convolutional neural networks represent a common deep learning architecture employed in the field of computer vision. This artificial intelligence discipline enables computers to comprehend and analyze visual data, such as images. CNNs, also known as ConvNets, are a specific type of deep neural network that excel at learning highly abstracted features from spatial data and identifying them effectively. These models consist of a limited number of processing layers that can learn input data features, like images, at various levels of abstraction. The initial layers focus on learning and extracting high-level features, while the deeper layers concentrate on learning and extracting low-level features. CNNs have been widely utilized for image identification and classification tasks, with numerous architectures developed over time, each offering unique advantages and applications. This section aims to discuss several successful instances of CNN architectures that demonstrate the recent significant advancements in the field of computer vision, addressing the three primary subdomains where these models play a crucial role in achieving outstanding results.

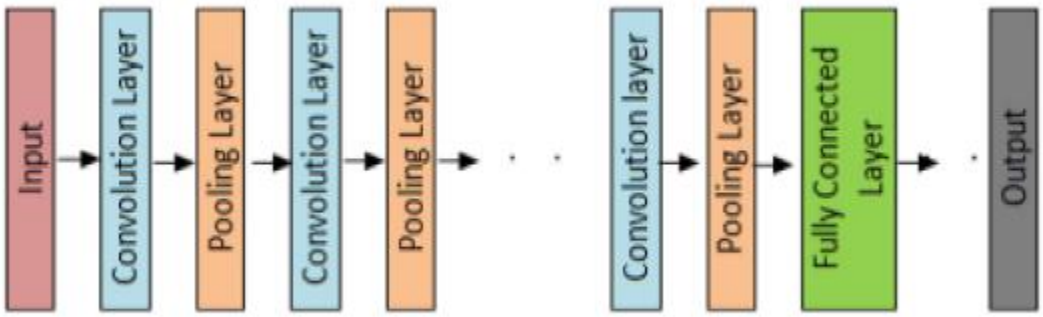


Figure 2.9 Diagram of the transfer learning architecture adapted from [39].

2.9 Image Processing

Image processing is a technique for manipulating photographs to improve them or extract data. It describes the processing and examination of digital photographs in order to retrieve data, improve their visual appeal, or get them ready for additional examination. It employs a technology to apply various operations to an image in order to extract important information or create an enhanced image[40].

2.9.1. Image Acquisition

This is the first step where an image is captured by a camera or sensor and then digitized if necessary for further processing. The process begins with acquiring digital images from various sources such as cameras, scanners, satellites, or other imaging devices. Images can be in different formats (e.g. JPEG and PNG) and resolutions, depending on the imaging device and application.

2.9.2. Image Pre-Processing

Before additional analysis or processing, a set of procedures known as "image pre-processing" are applied to raw images. Noise reduction, image normalization, scaling, color space conversion, and image registration are examples of common pre-processing processes. Pre-processing is done to improve the quality and usefulness of pictures so they may be used in subsequent procedures such as segmentation, feature extraction, or object recognition.

2.9.3. Image Segmentation

Image segmentation is the process of dividing a digital image into multiple distinct regions or segments based on specific characteristics or features. The goal of this technique is to partition the image into meaningful parts that correspond to objects or areas of interest. Various methods for image segmentation have been developed, including thresholding, edge detection, region growing, clustering, and watershed algorithms. Image segmentation is a crucial computer vision technique that partitions a digital image into discrete groups of pixels, enabling more efficient object detection and related tasks. By parsing an image's complex visual data into specifically shaped segments, image segmentation facilitates faster and more advanced image processing. Convolutional Neural Networks have also demonstrated exceptional performance in the segmentation task. Following the groundbreaking achievements of Alex Net in 2012, numerous state-of-the-art models for semantic segmentation and instance segmentation have been introduced, some of which are highlighted in the subsequent discussion.

2.9.4. Feature Extraction

Feature extraction involves identifying and describing relevant characteristics or patterns within images. Features include edges, corners, textures, shapes, colors, or other distinctive attributes that are useful for subsequent analysis or recognition tasks. Feature extraction methods vary based on the specific application and involve techniques such as convolutional filters, scale-invariant feature transform, histogram of oriented gradients, or deep learning-based feature learning. These methods are utilized to extract the most relevant features from large datasets by selecting and combining variables into features, thereby accurately and comprehensively describing the original data and effectively reducing the amount of data that requires processing. Wheat leaf diseases exhibit unique characteristics that distinguish them from other diseases. Consequently, identifying and understanding these distinguishing features is crucial for differentiating between various wheat leaf diseases. In this study, features are extracted using a convolutional neural network layer known as the convolution layer. By adjusting different parameters of the convolution layer, such as filter size, kernels, and padding, the system can automatically extract the features of an image, leveraging the capabilities of

2.9.5. Image Detection

Detection is the process of finding and recognizing particular items or areas of interest in an image. Finding and recognizing particular objects or areas of interest within photos is referred to as image detection. Localizing objects by detecting their presence and estimating their spatial extent, or bounding boxes, is a common task in detection tasks. Image detection techniques include object proposal methods, template matching, sliding window approaches, and deep learning-based object identification frameworks as Faster R-CNN and You Only Look Once (YOLO). Here, an input image may contain many objects, unlike in image categorization. Through the use of CNN models, we attempt to identify all objects present in the input image and their precise locations within it.

2.9.6. Image Classification

Assigning labels or categories to photos according to their visual content is known as image categorization. From conventional techniques like template matching and machine learning classifiers to deep learning-based strategies like convolutional neural networks (CNNs), object detection and classification techniques cover a wide spectrum of methodologies. Computers can analyze, interpret, and comprehend the visual information included in digital images thanks to these fundamental ideas of image processing and computer vision.

VGGNet

That Easy to use and with a consistent design, VGGNet was developed by the University of Oxford's Visual Geometry Group (VGG). Popular CNN architectures include VGGNet [41]. The researchers presented a total of 6 distinct CNN architectures, with the VGGNet-16 and VGGNet-19 being the most effective. These models consist of either 16 or 19 layers, utilizing small 3x3 convolutional filters and incorporating max-pooling layers after each convolutional block.

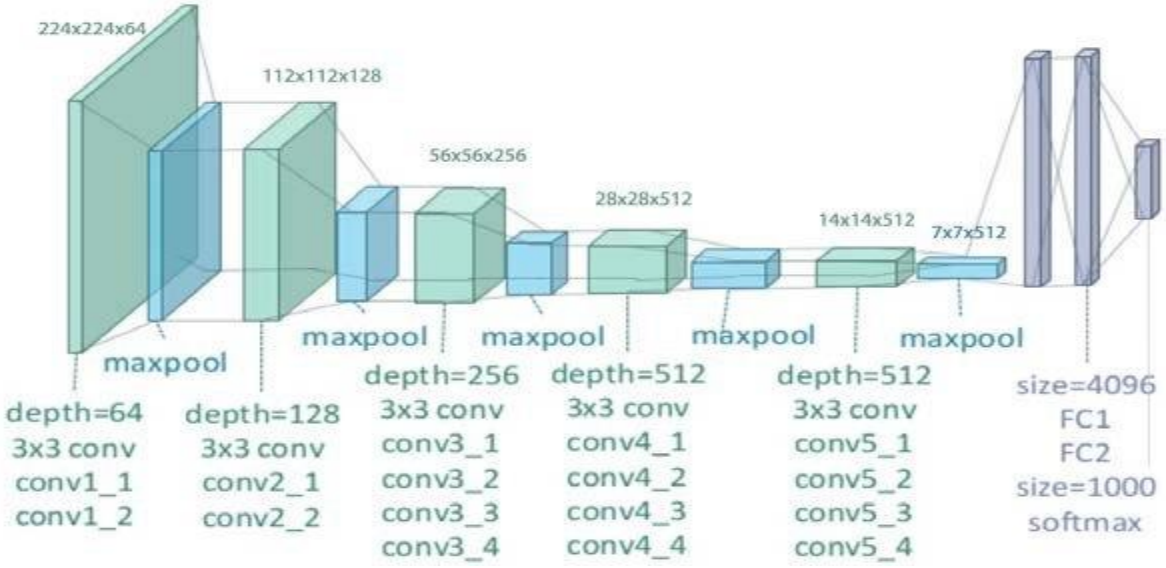


Figure 2.10 VGGNet Architecture adapted from [42]

GoogLeNet, developed by researchers at Google, introduced the concept of inception modules, which enable efficient utilization of computational resources by incorporating multiple filter sizes within the same convolutional layer. The GoogLeNet architecture departs from the conventional CNN models previously discussed, as it employs a network branching structure rather than a singular sequential architecture [43]. The Google Net architecture employs the "Inception Module" as its fundamental building block and consists of 22 weighted layers. This module processes data concurrently across the network. Each basic module contains three convolutional layers with filters applied sequentially - one filter, one filter, and five filters. The resulting feature maps are aggregated to generate a very high-dimensional feature representation.

ResNet

Residual Network, is a deep learning architecture introduced by[44]. It was developed to tackle the vanishing gradient problem that occurs when training very deep neural networks, which makes learning difficult as the network depth increases. ResNet's key innovation lies in the use of residual learning, where "shortcut connections" bypass certain layers, allowing the model to learn the residual (or difference) between the input and output of a layer, rather than the entire transformation. This design helps preserve gradient flow during backpropagation, enabling the training of much deeper networks than previously possible. Models like ResNet-50, ResNet-101, and ResNet-152, which have layers numbering into the hundreds, are notable examples. ResNet models have excelled in image classification tasks, particularly on benchmark datasets like ImageNet, achieving high levels of accuracy. The architecture's success also extends to transfer learning, where pretrained ResNet models are often fine-tuned for other tasks such as object detection, segmentation, and more, due to their powerful feature extraction capabilities. The ResNet architecture addressed the vanishing gradient problem in very deep neural networks by incorporating skip connections. These skip connections facilitate the flow of gradients during training, thereby enabling the effective training of extremely deep networks. Microsoft proposed the idea of "identity skip connections" as a solution to the vanishing gradient issue, which led to the development of the ResNet model[45]. The ResNet architecture utilizes residual mapping ($F(x) + x$) rather than learning a direct mapping ($F(x)$), and these building blocks are referred to as residual blocks. The complete ResNet architecture is composed of numerous residual blocks with 3x3 convolutional layers.

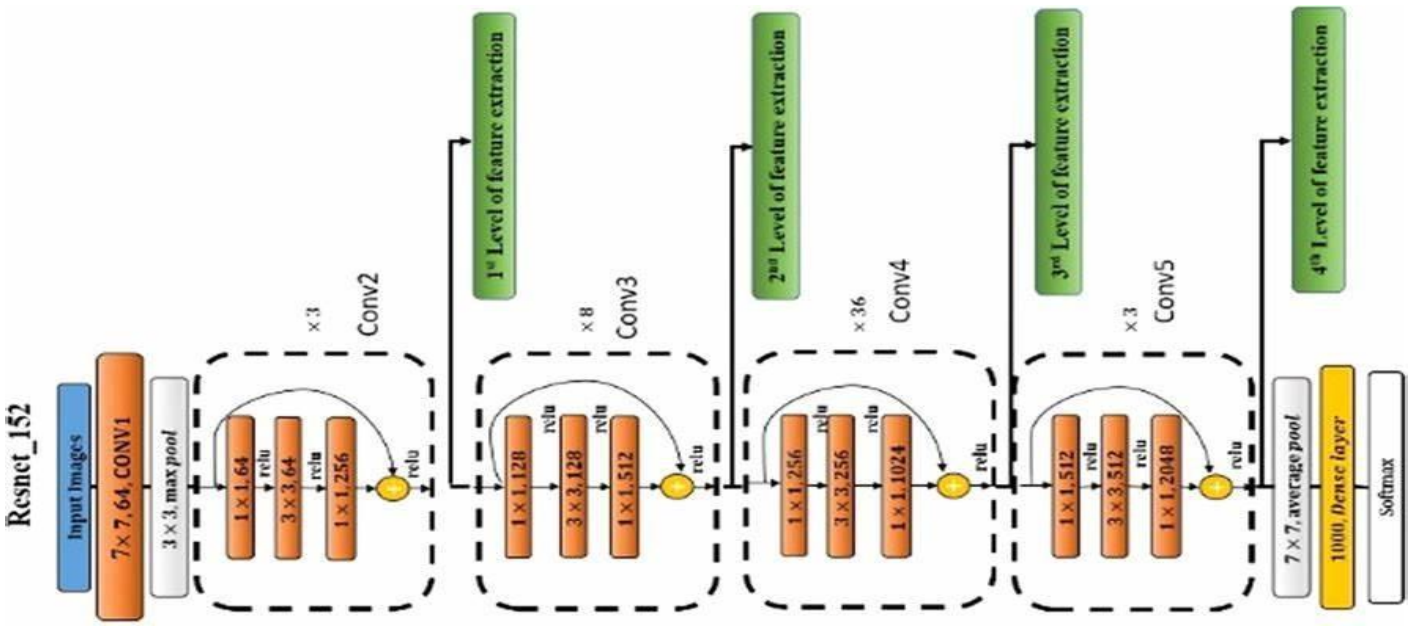


Figure 2.11. ResNet, Architecture adapted from [46]

DenseNet

Introduced Dense Convolutional Networks, an architecture that establishes direct connections between each layer and all other layers. This design promotes feature reuse, enhances feature diffusion, and alleviates the vanishing-gradient issue. The Dense Net model was proposed by Huang et al. [47]. Densely Connected Convolutional Network, is a deep learning architecture introduced in 2017 by Gao Huang and colleagues. Its unique design revolves around dense connectivity, where each layer in the network receives input from all preceding layers and passes its output to every subsequent layer. This structure ensures maximum feature reuse, leading to more efficient learning and reducing the number of parameters compared to traditional networks like ResNet. The dense connections also improve gradient flow during training, allowing for very deep networks without suffering from the vanishing gradient problem. Furthermore, Dense Net's ability to propagate features from earlier layers helps the network learn richer, more diverse features, enhancing its performance in tasks such as image classification, object detection, and segmentation. Models like DenseNet-121, DenseNet-169, and DenseNet-201 demonstrate its scalability and effectiveness, offering competitive accuracy while being more computationally efficient. This architecture's efficient parameter usage and robust learning capabilities make it especially suited for deep learning tasks on both large and smaller datasets.

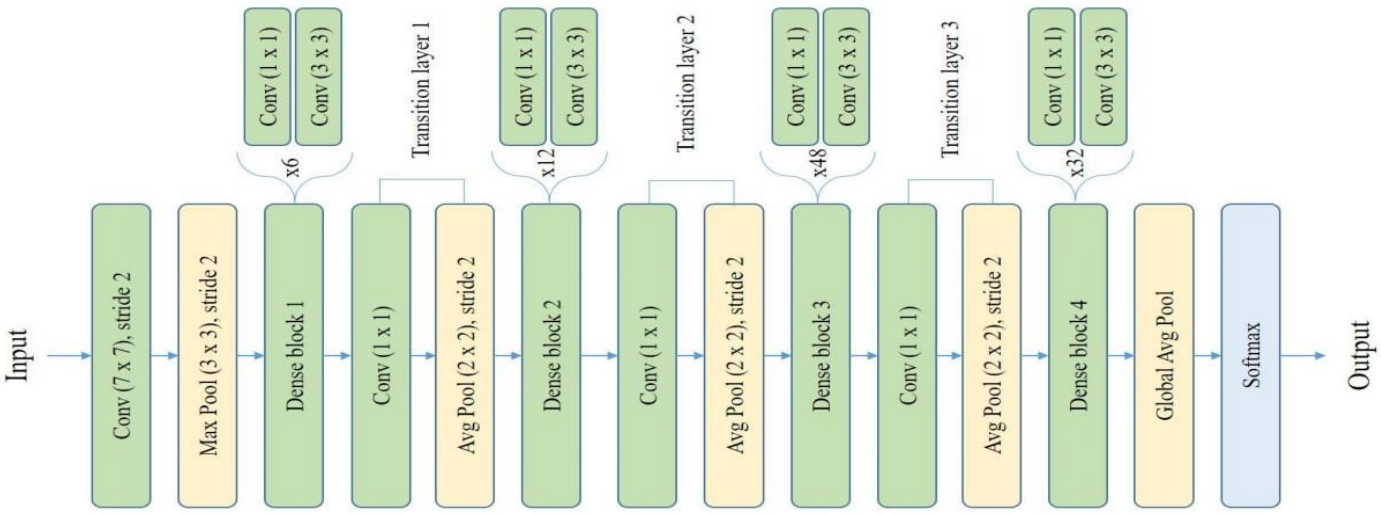


Figure 2.12. Dense Net Architecture is adapted from [46]

Inception

Inception, introduced as part of the Google Net architecture by Christian Szegedy and his team in 2014[43], is a deep learning model designed to achieve high accuracy in image classification tasks while being computationally efficient. Its key innovation is the Inception module, which applies multiple convolution operations with different filter sizes (e.g., 1x1, 3x3, 5x5) in parallel within the same layer. This approach allows the model to capture features at various scales and combine them effectively, enhancing its ability to extract both fine and coarse details from the input. Another important feature of Inception is the use of 1x1 convolutions, which are employed to reduce the dimensionality of feature maps before applying larger convolutions, making the model more efficient in terms of both computational cost and memory usage. The original version, Google Net (Inception-v1), won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2014 with 22 layers but far fewer parameters than previous architectures like VGGNet. Subsequent versions, such as Inception-v2 and Inception-v3, introduced further optimizations like batch normalization and factorized convolutions, improving both performance and efficiency. Inception has been widely used in computer vision applications due to its innovative approach to feature extraction and scalability.

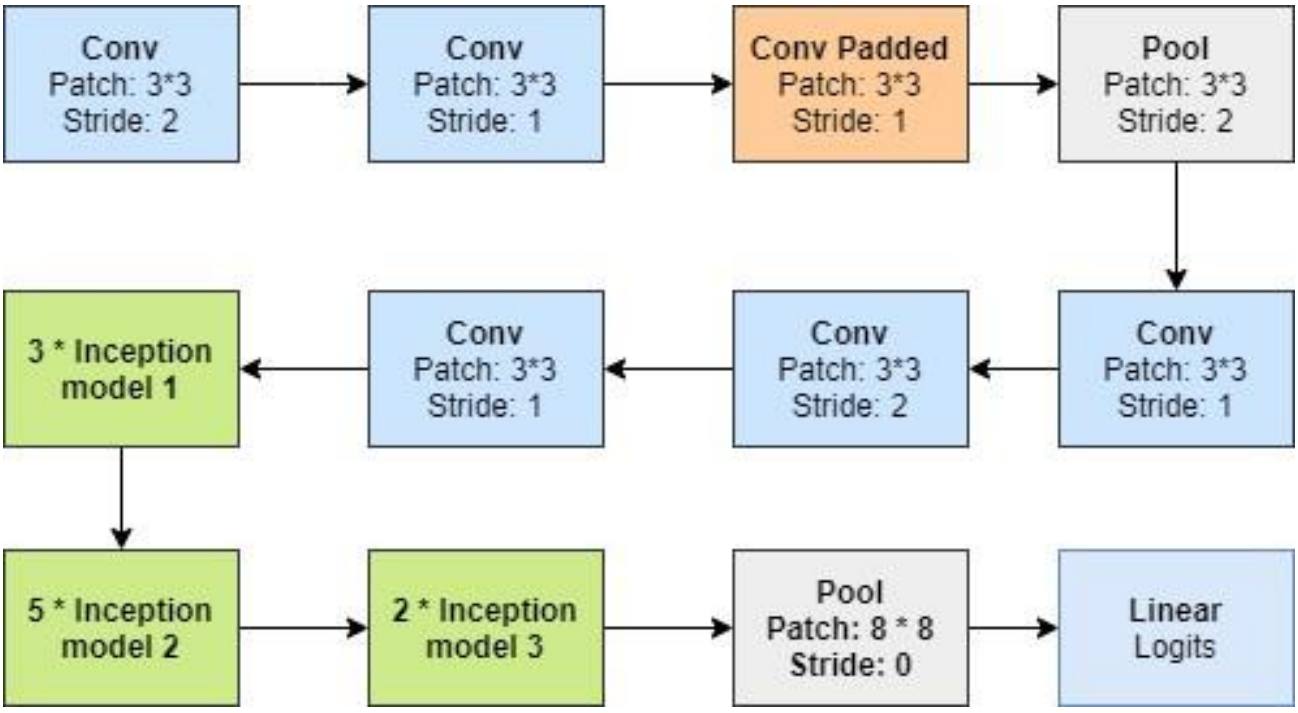


Figure 2.13. Inception Architecture is adapted from [42]

EfficientNet

EfficientNet is a family of convolutional neural network (CNN) models introduced by Google researchers in 2019[48]. It is designed to achieve state-of-the-art performance in image classification tasks while being significantly more efficient in terms of computational resources. The key innovation behind EfficientNet is its use of compound scaling, a method that systematically scales the network's depth, width, and resolution to improve both accuracy and efficiency. Instead of scaling just one dimension, such as making the model deeper or wider, EfficientNet scales all three in a balanced way, ensuring optimal performance with fewer parameters and lower computational cost.

The base model, EfficientNetB0, was developed using a neural architecture search (NAS) and serves as the foundation for the other models in the family, ranging from EfficientNetB0 to EfficientNetB7. These models progressively scale up in complexity, with EfficientNetB7 being the largest and most accurate but still more efficient than traditional architectures like ResNet or Inception. The EfficientNet family incorporates techniques like swish activation functions and squeeze-and-excitation blocks, which enhance the model's ability to capture complex features while maintaining computational efficiency.

Efficient Net has become widely used in image classification, object detection, and other computer vision tasks due to its balance of high accuracy and resource efficiency. It achieves better performance with fewer parameters and less computational power compared to many previous architectures.

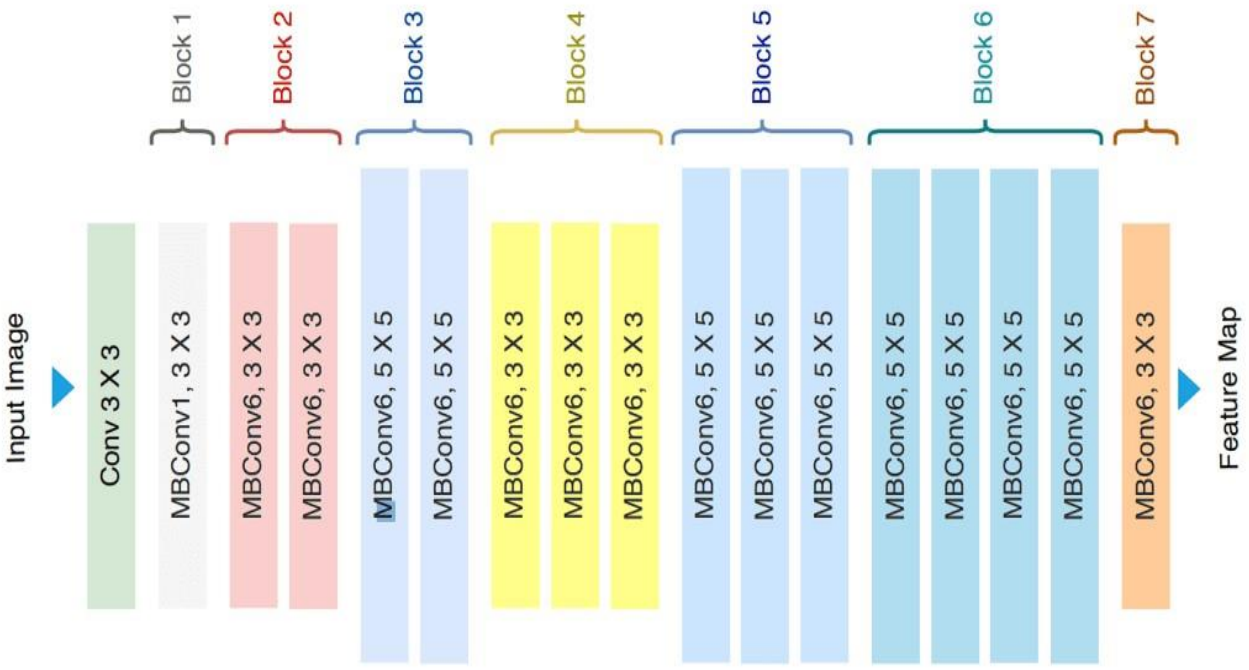


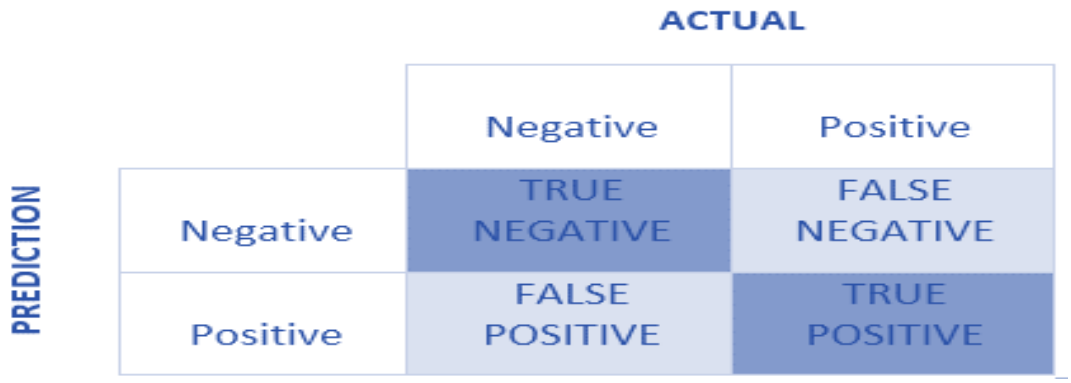
Figure 2.14. Inception Architecture adopted from [46]

2.10 Evaluation technique

A confusion matrix is a key evaluation method employed in machine learning to assess the performance of a classification model. This tabular representation allows data practitioners to visualize the model's performance and discern not only the errors it makes, but also the specific types of errors. Leveraging the insights gleaned from the confusion matrix, data practitioners can undertake the crucial task of ensuring rigorous and accurate evaluation processes to measure the performance of their machine learning models.

The confusion matrix is a visual tool that enables data practitioners to gain a deeper understanding of a model's shortcomings, flaws, and overall performance. By fine-tuning their model based on this information, data practitioners can then proceed to conduct further in-depth analysis.

Actual values are compared to expected values using the confusion matrix.



The terms employed in the confusion matrix are as follows:

True Positives: Correctly classified positive instances.

True Negatives: Correctly classified negative instances.

False Positives: Incorrectly classified positive instances.

False Negatives: Incorrectly classified negative instances.

From the confusion matrix, a variety of performance metrics can be derived[49].

Accuracy quantifies the proportion of correctly classified samples among the entire dataset. The formula for calculating accuracy is as follows:

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

The precision metric quantifies the positive predictive capability of the classifier model, as determined by the formula provided.

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

Recall: This measure quantifies the true positive instances, i.e., the positive samples that are correctly predicted as positive, and is defined as follows.

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

F-measure is a composite metric that jointly accounts for both recall and precision.

$$\mathrm{F1} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

2.11 Related work

Numerous research initiatives have been undertaken and proposed for the detection of plant and crop diseases utilizing a variety of machine learning techniques, including conventional learning and deep learning approaches.[50]. However, each job is designed to adjust different crop or parts of a crop. Therefore, in this section, I'll see the relevant works of researchers on the classification of plant diseases using a machine vision system.

Hasan .E, et al[51] focuses on using a combination of textural and deep features to classify wheat yellow rust disease. This approach could potentially enhance the ability to detect and diagnose wheat yellow rust disease, which is crucial for effective agricultural management. The proposed method likely involves preprocessing the images of wheat leaves to extract textural features such as texture descriptors or histograms. These features are then combined with features learned by a deep learning model, which may involve convolutional neural networks (CNNs) trained on a dataset of wheat leaf images.

The publication by Usha Ruby et al. [52], focuses on the application of deep learning, specifically a modified version of the ResNet50 model, to classify wheat leaf diseases. The research described addressed the challenge of classifying various wheat leaf diseases by employing a modified ResNet50 model. This approach likely aims to improve the accuracy and robustness of disease classification compared to traditional methods.

Tefera Bulo [53], discusses the increasing interest in plant identification based on leaves, highlighting the uniqueness of leaf characteristics for plant identification. It emphasizes the importance of preprocessing leaf images to extract critical features. The use of Convolutional Neural Networks (CNNs) is highlighted as a reliable method for classification and identification tasks due to their ability to learn patterns from images effectively. Supervised deep learning, particularly CNNs, is noted for its

accuracy in classifying leaf data. CNNs are capable of handling large amounts of image data, thus improving accuracy significantly. The passage outlines the three phases of the proposed approach: preprocessing, feature extraction, and classification.

Wan Quan Chen et al.[54] Stripe rust symptoms are characterized by yellow to orange, arranged in stripes on the leaves, leaf sheaths, glumes, and awns of infected plants. These pustules may cause significant damage to the plant if the cultivar is susceptible. In resistant wheat varieties, infection may result in minor symptoms, such as hypersensitive flecks or limited pustule formation.

Kahsay[55] The research relies on traditional machine learning algorithms (Naïve Bayes, K-Nearest Neighbor, Support Vector Machines, and Random Forest) rather than exploring deep learning methods such as convolutional neural networks (CNNs), which have shown superior performance in image-based classification tasks, particularly for plant disease detection. The research uses color-based segmentation, less effective in handling complex image datasets where lighting, shadow, or occlusion impacts feature extraction. Advanced segmentation techniques, like semantic segmentation using deep learning, could provide more accurate results. The research effectively applies machine learning methods to classify wheat leaf diseases, achieving a high accuracy (98.7%) with Random Forest. It uses the Gray Level Co-occurrence Matrix (GLCM) for feature extraction, which is well-established for texture analysis in image processing. The methodology is straightforward and easily reproducible using Python and its libraries. The lack of advanced techniques like deep learning and semantic segmentation restricts the scalability and robustness of the model.

T Nithya, A., and V. Sundaram[56] The study focuses on decision trees for wheat disease classification, which is interpretable but lacks generalization for complex datasets compared to advanced models like ensemble methods or CNNs. Manual feature selection in decision trees limits feature discovery, whereas deep learning can automate feature extraction for improved performance. The focus on binary classification ignores real-world scenarios involving multiple diseases and severity levels. Adopting multi-class classification and severity assessment would offer more practical and actionable insights for farmers.

Table 2.1. Summary of related works

No	Author(s)	Title	Model	Accuracy	Research gap
1	(Hossein et al., 2022) [57]	Wheat diseases detection and classification using convolutional neural network (CNN)	VGG16	98.84%	The creation and operational mechanisms of this model, in relation to traditional Convolutional Neural Networks, are not discussed in depth.
2	(Dunk, 2023) [23]	Wheat disease detection and classification using resnet152	ResNet152	97.81%	The study doesn't fully explore how different environmental factors (e.g., temperature, humidity, soil conditions) impact the model's ability to detect diseases.
3	(Goyal et al., 2021) [58]	Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture	VGG16	97.88%	The paper compares the proposed model with VGG16 and ResNet50 but doesn’t consider newer or more advanced models like InceptionV3 and others, which may offer even better performance or efficiency.
4	(Jouini et al., 2024) [50]	Wheat Leaf Disease Detection: A Lightweight Approach with Shallow CNN-Based Feature Refinement	EfficientNet B0	99.80%	The authors created a computationally efficient model for detecting wheat diseases using shallow convolutional neural networks; however, this model does not learn more complex patterns in the data. Heavyweight models typically achieve higher accuracy and performance metrics because their complex architectures and deeper

					layers allow them to learn more intricate patterns in the data. They developed a lightweight model, which has a size of 51 MB and consists of 6 million parameters
6	(Aboneh et al., 2021) [19]	Computer vision framework for wheat disease identification and classification using jetson GPU infrastructure	VGG19	99.38%	Small (1500) number of datasets is used it affects the performance
7	(Workneh Almarie, 2021) [59]	Convolutional neural network for wheat leaf disease detection	CNN	99%	Small (1570) number of datasets is used it affects the performance
8	Tefera [53]	Wheat leaf disease detection and classification model using transfer learning approach	VGG19	99.70%	The researcher used only 2,316 images representing five types of wheat leaf diseases.

This study investigated deep learning techniques, utilizing transfer learning with pre-trained models like VGG16, VGG19, InceptionV3, ResNet152, EfficientNetB7, and DenseNet201, alongside a custom CNN sequential model, achieving high classification accuracy and surpassing traditional image processing methods The study also highlights opportunities for future advancements, such as exploring advanced architectures like pre-trained models and CNN sequential mode to improve performance and efficiency and expanding datasets with diverse regional samples to enhance model generalizability.

CHAPTER THREE

METHODOLOGY

3.1 Overview

This chapter delineates the research methodology employed. It outlines the systematic process used to collect data, preprocess images, train the model, and evaluate its performance. This paper provides an overview of the methodological approaches employed in the research. The experimental research methodology was employed in this study. The methods used to complete this thesis are covered in this chapter, including the sources from which the image dataset was gathered, the methods used to prepare and analyze the image to develop a classification model, the tools used for implementation, and the metrics employed to measure the effectiveness of the developed model.

This investigation employs an experimental methodology to address the objectives of the study. Experimental research is an empirical inquiry undertaken according to a structured research plan. The purpose of experimental studies is to establish relationships between dependent and independent variables. Links between specific properties of the entity and the variables under study are confirmed or disproved upon completion of experimental research studies. Therefore, in this study, an investigation is conducted to detect and classify wheat leaf diseases based on the images provided.

Fundamental elements are used in the conduct of this thesis. Finding and choosing references that help you comprehend the subject matter better is the first step in the process. The study's overall goals and particular goals are then determined. the preparation of data is the first part of the thesis design, and the creation of a research prototype is the focus of phase 2. after being gathered from the Merti Agricultural Research Center, Debre Berhan Agricultural Research Center and Kaggle dataset, the data is expertly tagged and separated into subsets for testing, validation, and training. After the data is prepared, the model is constructed. The thesis's application is the final stage. Using the appropriate resources and protocols, the recommended model is implemented at this phase. The generated model is trained and evaluated using the necessary data. The performance of the model is assessed throughout training. Test data are used to determine and evaluate the optimal model during model evaluation. Subsequently, the model is compared with other models that were trained beforehand.

3.2 Data collection

The research is conducted around the Agricultural Research Center in the Oromia and Amhara regions. Both of Ethiopia's major wheat production areas suffer from the disease. To precisely assess the extent of crop damage caused by disease, pinpoint which areas of the crop are most affected, and identify the diseases impacting different crop parts, the researcher gathered sample data and supplemented it with a dataset downloaded from the Kaggle dataset. At the aforementioned farms and agricultural research centers, the dataset labels for various wheat crops in this investigation were refined with assistance from specialists.

The 80:10:10 ratio is a commonly utilized method in machine learning and deep learning for partitioning datasets into training, validation, and test sets. Researchers often adjust this ratio based on the size of the dataset. However, the optimal split may vary according to the unique characteristics of the dataset and the specific objectives of the study.

The dataset comprises 1103 images of healthy wheat leaves, 1097 images of yellow rust leaves, 1098 images of brown rust leaves, and 1099 images of orange rust leaves. As shown in Table 3.1, the original datasets were utilized for training, validation, and testing of the methodology. Additionally, the datasets have been divided for these purposes, with 80% of the total 4397 datasets used for training, and the remaining 20% comprising 441 datasets each for validation and testing.

The dataset comprises a diverse collection of images, encompassing a range of resolutions, as well as samples from various stages of the infection process, including early, intermediate, and late stages.

Table 3.1 distribution of the dataset across the various disease and non-disease categories

Category of Wheat Leaves	Initial images	Images Used for Training	Images Used for Validation	Images Used for Testing
Brown rust	1098	878	110	110
Healthy wheat	1103	881	111	111
Orange rust	1099	879	110	110
Yellow rust	1097	877	110	110
Total	4397	3515	441	441

3.3. Data augmentation

Image data augmentation is a technique used to artificially increase the size of a dataset by applying various transformations to the existing images. Increase the size of the training dataset: Having a larger dataset can help improve the performance of machine learning models by providing more diverse examples for training. Data augmentation techniques are essential for enhancing the diversity of images in a dataset without actually collecting new data. By applying transformations to the original images, we simulate variations that the model might encounter in real-world scenarios, improving its ability to generalize effectively. In this setup, we start by rescaling each image's pixel values to a range between 0 and 1, which ensures numerical stability during training. Next, shear and zoom transformations introduce slight skews and rescaling, respectively, mimicking shifts in perspective or distance that might occur in real images. Random horizontal and vertical flips allow the model to recognize the crop regardless of symmetry or orientation. Additionally, rotation (up to 20 degrees) and width and height shifts (each up to 20% of the image size) simulate minor misalignments and off-center placements, making the model more robust to images that aren't perfectly aligned or centered. If transformations create empty areas in the images, fill mode set to "nearest" uses nearby pixel values to fill in gaps, maintaining the visual consistency of the image. This process is commonly used in machine learning tasks, especially in computer vision, to increase the diversity of the dataset and improve the robustness of the trained model. After augmentation, typically end up with a larger dataset compared to the original dataset.

Table 3.2: Parameters for data augmentation

Data Augmentation Technique	Values
rescale	1.0/255
shear_range	0.2
zoom_range	0.2
horizontal_flip	True
rotation_range	20
width_shift_range	0.2
height_shift_range	0.2
vertical_flip	True
fill_mode	nearest

3.4.Image Preprocessing

Image preprocessing is a critical step in computer vision tasks, as it enhances the quality of images and extracts valuable features. After collecting the leaf images of wheat, the subsequent stage involves preprocessing these images. This image processing method allows the collected images from the crop farm to be resized into a standardized dimension and scaled to have consistent width and height before being fed into the learning algorithm. Image preprocessing is performed to maximize accuracy and efficiency, reduce noise in the images, and improve the overall quality of the original images. To minimize the computational burden and improve storage efficiency in subsequent processing, the original image size is reduced to 224 by 224 pixels. This specific size is widely used because many popular pre-trained models, such as VGGNet, ResNet, Dense Net, Inception and Efficient Net were trained on datasets like ImageNet that require inputs of 224x224. Resizing to this dimension ensures compatibility with these models, particularly when utilizing transfer learning, where a pre-trained model is fine-tuned for a specific task. Additionally, to minimize the computational burden and improve storage efficiency in subsequent processing, the image size is carefully chosen. The dimensions are selected through extensive testing to ensure that they provide the best balance between model performance and resource efficiency. This process helps maintain image quality while reducing the load on memory and computational power, allowing for faster training and inference without sacrificing accuracy. Furthermore, to ensure the autonomous system's efficacy in detecting and classifying crop diseases is not compromised, the ideal image size was meticulously selected through extensive testing at various sizes. Maintaining consistent image sizes also necessitates image scaling.

3.4.1. Image Resizing

Resizing images to a standard size can make them more manageable and ensure consistency in input dimensions for machine learning models. Image resizing is a core operation in image processing and computer vision. It encompasses altering the dimensions of an image while preserving its aspect ratio. This process is commonly employed for various purposes, such as visualization, data preprocessing for machine learning, and ensuring compatibility with specific input requirements of algorithms or models. Key factors to consider regarding image resizing include:

Aspect Ratio: Maintaining the aspect ratio of an image is crucial to prevent distortion of objects and features. The aspect ratio refers to the ratio of the image's width to its height. Resizing algorithms typically scale the image proportionally to preserve this ratio. Interpolation: When resizing an image

to a larger or smaller size, interpolation is utilized to estimate the values of the new pixels. Common interpolation methods include nearest-neighbor, bilinear, bicubic, and Lanczos interpolation, each with its own advantages and disadvantages in terms of computational complexity and the quality of the resized image. Batches and Streaming When working with large datasets or streaming applications, resizing images efficiently becomes crucial. Batch processing can be used to resize multiple images simultaneously, while streaming applications require resizing images on-the-fly as they are loaded into memory. Preservation of Information When resizing images for machine learning tasks, it's important to ensure that relevant information is preserved. Excessive down sampling can remove important details, while excessive up sampling can introduce noise or artificial features.

Bilinear interpolation is the method calculates the pixel value at a given position by taking the weighted average of the four nearest pixels. It results in a smoother image.

$$I(x', y') = \sum_{i,j} I(i, j).(1 - |x' - i|).(1 - |y' - j|)$$

Where I (x', y') is the intensity of the pixel in the resized image, and I (i, j) are the original pixel intensities.

3.4.2. Normalization

Normalization is a crucial preprocessing step in various machine learning tasks, including image processing and computer vision. It involves scaling the input data to a common range or distribution to make it more suitable for learning algorithms. When pixel values are scaled to a standard range, like [0, 1] or [-1, 1], it is referred to as normalization in visual processing contexts[60]. When training optimization algorithms, particularly in deep learning models, normalizing pixel values to a common range might facilitate faster convergence. To normalize feature maps at various network layers, CNNs frequently employ normalization layers like batch normalization or layer normalization. By lessening internal covariate fluctuations, these layers aid in training stabilization and speed. The normalization layers that pre-trained CNN models frequently feature are the results of extensive dataset training. Thorough evaluation of normalization techniques is necessary when fine-tuning these models to guarantee that they are compatible with the initial training configuration.

Normalization Formula[61]:

$$\boldsymbol{I}\ normalized = \frac{\boldsymbol{I}-\boldsymbol{I}\ min}{\boldsymbol{I}\ max-\boldsymbol{I}\ min}$$

Where: ***I****: is the original pixel intensity*

I_{min}*: is the minimum pixel intensity in the image or dataset*

I_{max}*: is the maximum pixel intensity*

3.4.3. Image histogram equalization

Histogram Equalization is a technique employed in the field of image processing to enhance the visual contrast of an image by redistributing the intensity values. This method effectively stretches the intensity range of an image's histogram, which represents the frequency distribution of intensity levels[62]. In an ideal scenario, the histogram would be uniformly distributed across the entire intensity range, but in real-world images, this is often not the case. in addition, Histogram equalization involves transforming the intensity values in such a way that the cumulative distribution function (CDF) of the histogram is stretched out to cover the entire intensity range. This effectively spreads out the intensity values, enhancing the contrast of the image.

$$Sk = (L-1).\sum_{i=0}^{k} pi$$

where Sk is the new intensity value, L is the number of intensity levels, and Pi is the probability of the intensity i. The CDF is used map the original intensities to new values that span the full intensity range, thereby enhancing contrast.

3.4.4. Noise Reduction

Noise reduction in image processing is a critical step aimed at improving image quality by eliminating unwanted distortions or artifacts while preserving essential features[63]. In computer vision tasks, especially when dealing with agricultural datasets like disease detection in wheat crops, noise can obscure important details, making it harder for machine learning models to identify patterns. Various algorithms can be applied to achieve effective noise reduction, Such as Gaussian Filter. One of the simplest and most commonly used methods is Gaussian blur, which smooths the image by averaging pixel values with their neighbors based on a Gaussian distribution. This technique effectively reduces Gaussian noise but may blur important details if overapplied. In cases where the noise is complex or spatially distributed, wavelet transform denoising is applied. It decomposes the image into different

frequency components, allowing for targeted noise removal at specific frequencies, thus preserving the essential structural elements of the image. Additionally, Total Variation Denoising (TVD) is used to reduce noise while maintaining edges by minimizing the total variation of the image. Given an image I(x,y)I(x, y)I(x,y), the Total Variation denoising problem can be formulated as an optimization problem[64]:

$$\min_{\mathrm{u}} \left\{ \frac{1}{2} \int_{\Omega} \big(I(x,y) - u(x,y)\big) 2dxdy + \lambda \int \Omega \parallel \nabla u(x,y) \parallel 2dxdy \; dxdy \right\}$$

Where:

- u(x,y)u(x, y)u(x,y) is the denoised image.
- I(x,y)I(x, y)I(x,y) is the noisy input image.
- ∇u(x,y)\nabla u(x, y)∇u(x,y) represents the gradient of the image u(x,y)u(x, y)u(x,y) (i.e., the first-order derivatives of the image in both directions: ∇u=(ux,uy)\nabla u = (u_x, u_y)∇u=(ux ,uy)).
- ∥∇u(x,y)∥2\|\nabla u(x, y)\|_2∥∇u(x,y)∥2 is the L2L_2L2-norm of the gradient of uuu, representing the Total Variation of the image.
- λ\lambdaλ is a regularization parameter that controls the balance between data fidelity (how well the denoised image matches the noisy input image) and the smoothness of the solution (how much variation is penalized).

3.5. Fully Connected Multilayer Perceptron

A fully connected multilayer perceptron is an artificial neural network in which each neuron in a layer is connected to every neuron in the layer below. This network architecture consists of an input layer, one or more hidden layers, and an output layer. While MLPs are powerful models capable of identifying complex patterns in data, they are also prone to overfitting, especially when dealing with high-dimensional data or having a large number of parameters. To mitigate overfitting, techniques such as

weight decay and dropout regularization are commonly employed in MLPs[65]. The network described here has two hidden layers, with 32 and 16 neurons, respectively.

The output of the network classifies the input image based on the type of disease, with a single neuron producing a float value as the result.

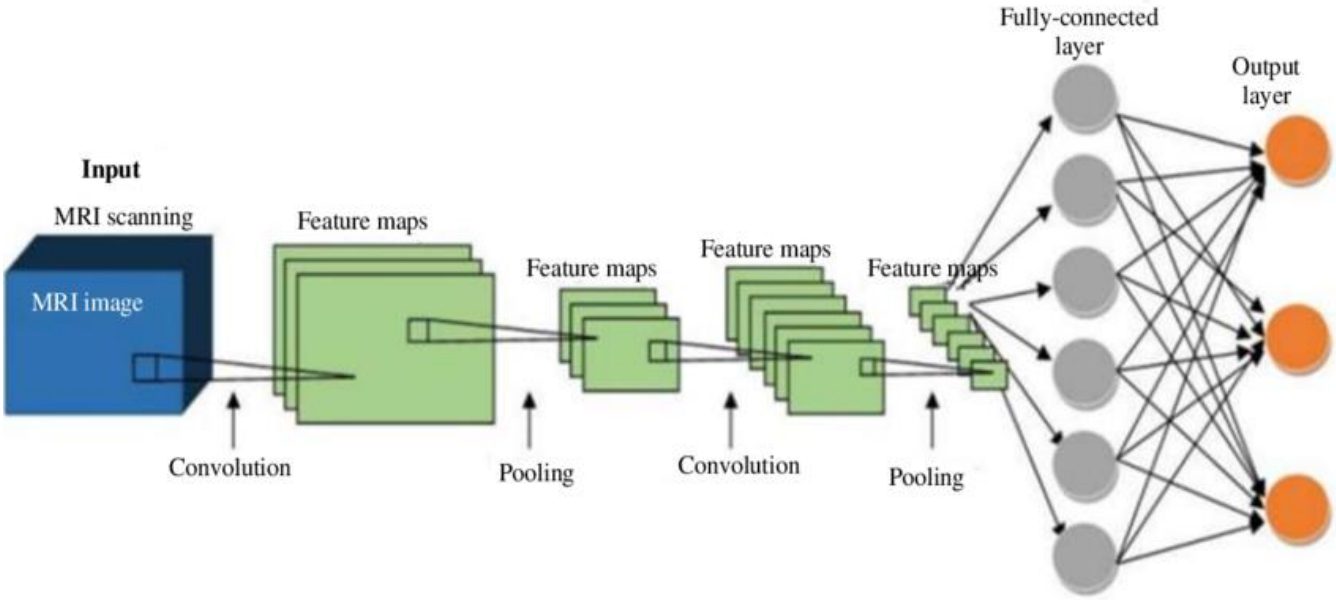


Figure 3.1. FC multi-layer perceptron NN for image classification tasks [66]

Using the OpenCV library, various preprocessing functions were built to extract and select features. The initial step is dubbed "image to feature vector," which involves resizing the picture to 128 by 128 pixels, converting RGB images to grayscale, and finally flattening the image into a list of row pixels. It was discovered through trial and error that while a larger input image with more input neurons doesn't improve accuracy, it does significantly drain the neural network's processing capacity.

The previously produced classification dataset is used to train the neural network through back-propagation training. Using a trained dataset as a guide, a back-propagation trainer backpropagates errors to adjust a module's parameters. Because they affect how well the neural network is trained, the input parameters to the trainer are crucial. The weight decay, learning rate, and momentum are inputs to the trainer. The parameters have the following effects on the trainer: The learning rate indicates the proportion of parameters that are altered in the gradient's direction. The accuracy and error rate of the neural network are measured after it has been trained using a dataset of 4397 image of both healthy and sick crops.

3.5.1. Optimization algorithms and Hyperparameters

Optimization algorithms are essential for training machine learning models, such as neural networks and multilayer perceptron’s. These methods fine-tune the model's parameters to minimize a predefined loss function, ultimately improving performance on unseen data. Hyperparameters, in contrast, are predetermined rather than learned during the process. Adjusting hyperparameters is crucial for achieving optimal performance in multilayer perceptron’s and other machine learning models.

3.5.2. Transfer learning

Transfer learning has emerged as a pivotal technique in the realms of machine learning and deep learning, enabling the development of more precise and efficient models, particularly in scenarios where labeled data is limited. This machine learning approach entails leveraging a model trained on one task and adapting it to initiate a new model on a related, yet distinct task. By harnessing the insights gained from solving one problem, transfer learning circumvents the need to start from scratch when training a new model. This method proves exceptionally beneficial when faced with limited labeled data for the target task. Its capacity to train deep neural networks with relatively minimal data has contributed to its widespread adoption in the deep learning domain. Given that most real-world problems do not generally have an abundance of labeled data points to train such complex models, transfer learning has become invaluable in the data science industry. Scenarios of Transfer Learning The process of feature extraction is Utilize the pre-trained model to extract features from the data by freezing its parameters. After that, a fresh classifier or model trained especially for the intended job is fed these features. Using the new dataset, the pre-trained model is fine-tuned in its entirety. This entails changing the model's overall parameters, frequently at a slower learning rate to avoid catastrophically losing the previously learned information.

3.5.3. Pre-Trained Base Model

CNNs (Convolutional Neural Networks) are specifically designed to process and analyze visual data by using convolutional layers to detect features in images. All the listed models primarily rely on these convolutional operations. DNNs (Deep Neural Networks), in a broader sense, simply refer to any network with multiple (deep) layers, and they may include various architectures like fully connected (dense) layers, recurrent layers, or convolutional layers. CNNs, a type of DNN, are specialized for spatial feature extraction, making them optimal for tasks like image classification.

Pre-trained convolutional neural networks are machine learning models that have been trained on large image datasets, such as ImageNet, for tasks like image classification. These models have learned to extract meaningful visual features from images and can effectively generalize to a wide range of image processing applications. The VGG network architecture, known for its simplicity and effectiveness, consists of several layers of convolution and pooling operations, and comes in different variants like VGG16 and VGG19[67]. InceptionV3, developed by researchers at Google, is another widely-used CNN architecture for image classification tasks, and it is the third version of the Inception architecture, following InceptionV1 and InceptionV2. ResNet, another influential CNN model, introduced the concept of residual connections, which help mitigate the vanishing gradient problem in deep neural networks, allowing for the training of networks with hundreds of layers. EfficientNetB7 and DenseNet201 are both advanced convolutional neural network architectures frequently used in image classification tasks, but they come with distinct design philosophies and computational trade-offs. EfficientNetB7, a model from the EfficientNet family, is designed with a compound scaling method that optimally adjusts depth, width, and input resolution. This approach allows EfficientNetB7 to achieve high accuracy while maintaining relatively lower computational costs compared to conventional models. However, as the largest model in the EfficientNet family, EfficientNetB7 typically requires a powerful GPU and longer training times, particularly when dealing with high-resolution images. DenseNet201, on the other hand, follows a different approach by connecting each layer to every other layer in a "dense" configuration. This design allows DenseNet201 to reuse features extensively, making it parameter-efficient despite its deep structure of 201 layers. While DenseNet201 often requires more memory due to its dense connections, it remains efficient in terms of parameter use and performs well even on smaller datasets. When selecting a pre-trained CNN for a specific task, factors to consider include the complexity of the task, the size of the available dataset, computational resources, and the trade-off between model size and accuracy[68]. Experimenting with different architectures and fine-tuning techniques can help identify the most suitable model for the given requirements.

3.5.4. CNN sequential model

One basic kind of convolutional neural network is a CNN sequential model, in which the layers are arranged in a consecutive, linear pattern, with each layer using the output of the one before it as input. Activation functions like ReLU are used to introduce non-linearity, pooling layers (MaxPooling2D) are used to minimize the spatial dimensions, convolutional layers (Conv2D) are used to extract features,

and fully connected (dense) layers are used to make predictions. The concept is referred to as sequential due to its linear form, which allows data to move seamlessly across layers without branching. Though it is simpler to create and less versatile than more intricate architectures like those with numerous inputs/outputs or skip connections, this architecture is especially helpful for jobs like image classification[69].

A CNN sequential model is constructed by stacking layers sequentially, where each layer serves a specific function in processing image data. The model begins with an input layer, where the shape of the images is defined, typically specifying the width, height, and number of color channels (e.g., 3 for RGB images). The first crucial step in the model is the convolutional layers (Conv2D), which apply filters to the input image to extract essential features such as edges, textures, and patterns. These layers help the network learn spatial hierarchies of features, which are crucial for understanding the content of images. After each convolutional layer, a batch normalization step is often added to stabilize and speed up training by normalizing the layer's inputs. Following convolutional layers, max-pooling layers (MaxPooling2D) are used to reduce the spatial dimensions of the feature maps, effectively down-sampling the data while preserving the most important features. Pooling also helps in reducing computational cost and preventing overfitting. To further combat overfitting, dropout layers are inserted after pooling or fully connected layers. Dropout randomly disables a fraction of neurons during training, ensuring the model does not rely too heavily on any single feature. Once the feature extraction is complete, the model uses a flatten layer to reshape the output into a 1D vector, which connects the convolutional part of the network to the fully connected (dense) layers. The last dense layer, typically with a SoftMax activation, is used to classify the image into one of the predefined categories (such as healthy or diseased plant classes in this study). The model is compiled with an optimizer like Adam, which adjusts the model's weights during training to minimize the loss function, often categorical cross-entropy for multi-class classification tasks. The performance is then evaluated using metrics such as accuracy. A typical CNN model is trained for several epochs, where each epoch involves a forward pass through the network, followed by the backpropagation process where the optimizer adjusts the weights based on the calculated loss. By carefully tuning the layers, such as the number of filters, the kernel size, dropout rates, and the number of epochs, a CNN sequential model can be built to effectively classify images while avoiding overfitting.

3.5.5. Model Selection

In this phase, we aimed to identify the best-performing model among pretrained network-based CNN, custom CNN, and pretrained network-based DNN models. After training and testing each model, we assessed their efficiency to determine which performed best. Some models consistently outperformed others, allowing us to clearly evaluate each model's strengths and ultimately select the top-performing deep learning model for our task. Selecting an appropriate model to address a given problem involves considering factors such as data type, dataset size, computational resources, and the desired outcome. This model selection process is essential in machine learning, particularly for image processing tasks, where additional factors like task complexity, available computational power, dataset size, and required inference speed must be carefully evaluated. Performance assessment should rely on relevant metrics, including computational efficiency, intersection over union (IoU), recall, accuracy, and precision. By experimenting with different architectures, methodologies, and hyperparameter tuning, the chosen model’s performance can be further optimized for specific applications.

3.5.6. Model Training

The data science team endeavors to configure the optimal weights and biases of an algorithm to minimize the loss function across the prediction range during the model training process, which is fundamental to the data science development lifecycle. The optimization of machine learning algorithms is governed by loss functions. Depending on the project objectives, the nature of the data utilized, and the type of algorithm employed, a data science team may leverage a variety of loss functions. An organization must employ a systematic and repeatable model training approach if it aims to develop deep learning at scale. A unified business platform that fosters collaboration rather than impeding it is essential for this purpose, and it should encompass all the necessary tools, libraries, and documentation. To enhance accuracy and mitigate false classification, the research center provides and the team downloads input images with diverse backdrops, light intensities, orientations, shapes, and sizes. The optimization of hyperparameters is crucial for efficient model training. Regularization techniques are employed to monitor and mitigate overfitting in machine learning models. The practice of training a model to recognize patterns and generate forecasts from incoming data is referred to as model training. It is akin to providing a learner with examples to follow and allowing them to take notes. To minimize the discrepancy between the model's predictions and the actual targets in the training data, the algorithm iteratively adjusts its internal parameters during the training process. To ensure the model's effective performance on the training dataset, it is typically necessary to provide it

with labeled data, comprising input-output pairs, and then utilize optimization methods to repetitively modify the model's parameters.

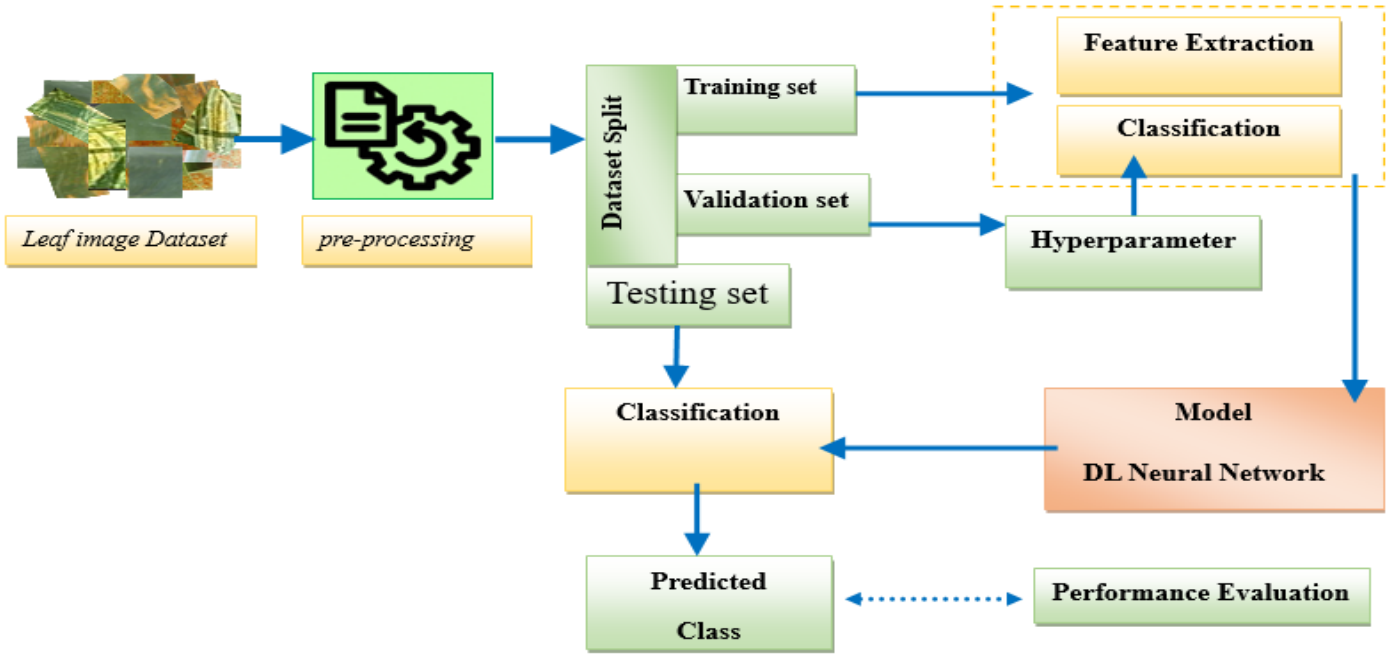


Figure 3.2. System Architecture for the proposed model

Image preprocessing is a crucial step in preparing the dataset for model training. The first task, resizing, involves adjusting images to a consistent size that matches the input dimensions expected by the model. This ensures that all images have the same shape, which is necessary for batch processing. Normalization or scaling follows, where pixel values are typically scaled to a consistent range, such as dividing by 255 for RGB images to bring values between 0 and 1. Alternatively, pixel values can be standardized by subtracting the mean and dividing by the standard deviation, helping the model to converge faster during training. Another important technique is histogram equalization, which enhances the contrast of the image, making it easier for the model to identify and distinguish different features. Lastly, image cropping may be applied to remove irrelevant portions of the image, focusing attention on the region of interest. These preprocessing tasks collectively ensure that the model receives high-quality and consistent input data, thereby improving its performance and accuracy during feature extraction and classification.

Feature extraction is typically performed after splitting the data to prevent data leakage and ensure unbiased model evaluation. Splitting the dataset into training, validation, and test sets first ensures that the model does not gain information from the validation or test sets during feature extraction.

feature extraction is done before splitting, the process could inadvertently include patterns or characteristics from the entire dataset, leading to overfitting and overly optimistic performance metrics. By splitting the data first, feature extraction can be applied independently to the training set, and the same process, with consistent parameters, can then be applied to the validation and test sets. This approach maintains the integrity of the evaluation process and ensures that the model generalizes well to unseen data.

During the training phase, individual images from the training dataset are converted into vector format for feature extraction. The extracted features undergo preprocessing, where data augmentation techniques such as flipping, zooming, rotating, and changing the range are applied. The reshaped images are then resized to 224x224 and fed into the system for further processing. Transfer learning is employed by utilizing a pre-trained neural network, specifically leveraging the weights from the ImageNet dataset to initialize the network. The system is trained using these pre-computed weights while simultaneously associating the labels of the leaves with these weights. This process helps the model learn to classify the input images based on the learned features. In the testing phase, a leaf image is provided to the system, which applies the same feature extraction procedure as in the training phase to evaluate the leaf's class. The processed image, in vector form, is then input into the trained model, where the pre-determined weights are used to calculate the class scores. The class corresponding to the highest score is selected as the predicted label for the leaf, assigning it to the correct class based on the learned features. This structured approach ensures accurate classification during testing.

3.6. Resources used for the research

The comprehensive list of all the hardware, software, frameworks, and libraries that were utilized in this research is in the documentation.

Table 3.3. hardware and Software Resources

Hardware	Software	Libraries	Data Preparation Tools	Documentation Tools
CPU, GPU, RAM, and storage.	OS Anaconda Colab	TensorFlow, PyTorch Libraries	annotation tools, augmentation, libraries	MS office Word, Microsoft Visio Mendeley

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1. Overview

Deep learning methods have gained significant attention in the classification and detection of wheat leaf diseases, as they hold potential for early disease diagnosis and control in agriculture. This study focuses on analyzing and assessing the results of the model's training and evaluation phases, specifically in terms of its performance in accurately recognizing and classifying diseases affecting wheat. The effectiveness of the model is discussed in comparison to alternative approaches, highlighting both strengths and limitations. Additionally, recommendations are provided for future research directions, identifying areas for improvement to further enhance the model's performance and robustness in agricultural applications.

We compared our CNN and DNN models with pretrained models used as feature extractors, evaluating their accuracy and losses to identify the most suitable model for our task. The results of this comparison are detailed in the following sections. To ensure a fair evaluation, all models were trained on the same collected dataset and with consistent parameters. We also tested data augmentation, which confirmed that it improved the performance of all proposed models.

4.2. Dataset Description

This chapter describe the experimental results, performance metrics, research resources required, and a discussion of the conclusions. Images of Healthy Wheat Leaf (1103), Brown Rust on Wheat Leaf (1098), Orange Rust on Wheat Leaf (1099), and Yellow Rust on Wheat Leaf (1097) are all used in the results and discussion part of the suggested solution using a different deep learning approach. Prior to testing, every image used in standard machine learning approaches goes through a phase of feature extraction and selection, and scaling. In contrast, every image used in deep learning techniques goes through these steps.

4.3. Model Implementation

This study employed a highly customized model implementation that was tailored to each specific design. This was achieved by conducting a series of tests on numerous convolutional neural network-based architectures. Each model was trained and evaluated independently to identify the optimal transfer learning model for the given dataset, and the results were subsequently compared.

4.4. Experiment Setup

This research is centered around three experimental models. Each model trained on the subset of the dataset mentioned earlier, which comprises four classes. All the experiments exhibit the same characteristics.

The experimental setup was implemented using the Python programming language, leveraging the Flask framework for model deployment and testing. All experiments were conducted on a laptop equipped with an Intel® Core TM i5 processor running at 2.40 GHz, 6GB of RAM, GPU NVIDIA GEFORCE, and a 64-bit Microsoft Windows 10 Home operating system. This configuration provided a stable and efficient environment for training and evaluating the models. Additionally, all the experiments employing the transfer learning approach utilize the same hyperparameters and pre-processing techniques. Several critical steps were executed to train the model using transfer learning of a pre-trained model and CNN Sequential architecture are as follows:

The experiment setup for evaluating machine learning models in wheat leaf disease classification involves a systematic approach to ensure reliable and comparable results. First, the objectives of the experiment are defined, setting clear benchmarks like accuracy, computational efficiency, precision, recall, and inference speed to guide model selection. This step is crucial in understanding what constitute success for the model and which criteria most relevant for evaluating performance. In addition, the data is prepared by collecting a labeled dataset representing various wheat diseases, followed by essential preprocessing steps such as resizing, normalization, and possibly data augmentation. The dataset is split into training, validation, and test sets, with each subset serving a specific role: training for learning, validation for tuning, and testing for final evaluation.

The model selection phase then involves choosing a range of models, including custom CNNs, pretrained CNN-based networks, and pretrained DNN-based networks, to cover a spectrum of architecture complexities. For pretrained models, transfer learning can be used to leverage them as feature extractors, capturing complex representations to aid in accurate classification. For consistency in evaluation, each model is assessed using the same metrics and conditions, which include accuracy, precision, recall, intersection over union (IoU), and computational efficiency. These metrics provide insights into overall correctness, balance between precision and recall, and performance in identifying relevant areas, especially in tasks requiring precise detection.

In the execution phase, each model is trained using the same parameters and conditions to ensure that differences in results are due to model performance alone. Hyperparameter tuning is conducted to

optimize each model further, employing techniques like grid or random search to determine the best parameter combinations. Data augmentation is also applied during this phase to observe its effect on each model's robustness and accuracy. Results are then analyzed comparatively, focusing on each model's strengths and weaknesses. Models excelling in multiple metrics are shortlisted, while trade-offs, like higher accuracy versus lower inference speed, are carefully considered.

The experiment concludes by selecting the model that best meets the objectives and balances performance with computational feasibility for real-world application. Observations on model behavior and potential areas for improvement are documented, along with recommendations for future research. This may include expanding the dataset, trying different model architectures, or employing advanced data augmentation methods to enhance model robustness. By following this structured setup, the experiment provides a comprehensive analysis that supports selecting the most effective model for wheat disease classification, paving the way for improvements and further research in agricultural applications.

4.5. Proposed for Wheat Disease Classification

The proposed for wheat disease Classification leverages a deep learning approach, focusing on accuracy and efficiency. It uses convolutional neural networks (CNN) to detect and categorize prevalent illnesses in wheat crops. Multiple convolutional layers are used in the design to extract disease-related information, such as discoloration and lesions, after an input layer processes photos of wheat leaves. Fully connected layers integrate the retrieved features for final classification, while max pooling layers lower computing costs and dimensionality. For multi-class classification, the system outputs the identified disease via a SoftMax layer.

To improve accuracy, preprocessing is applied to the raw data. Images are standardized for improved convergence, scaled for uniformity, and enhanced with flipping and rotation to boost data diversity. The model's capacity to capture intricate details is improved with more layers. To guarantee effective learning, the training procedure makes use of the Adam optimizer and categorical cross-entropy loss function. A small batch size is used to account for hardware limitations, and a learning rate scheduler is used to further improve the training processes.

Table 4.1. Summary of Hyperparameters Used during a model training

No of Epoch	Batch Size	Activation	Loss	Optimization	Dropout	l2	LR	Experiment
20	32	SoftMax	CCE	Adam	0.1	0.001	1e-5	1
40	64	SoftMax	CCE	Adam	0.1	0.001	1e-6	2

The code for the experiments is run in a Jupiter notebook.

The experiments were conducted using a Jupiter notebook with TensorFlow version 2.16.1 and Keras version 3.0. Essential Keras libraries such as MaxPooling2D, AveragePooling2D, Dense, Flatten, Dropout, and Batch Normalization were imported from keras. layers. The Data Generator for image preprocessing was sourced from keras. preprocessing. Image. For model evaluation, classification reports and confusion matrices were imported from sklearn. metrics. Additionally, models and optimizers from TensorFlow. keras, and plotting capabilities from Matplotlib's Pyplot were utilized. Importing specific model libraries from keras. applications are crucial, especially when employing pre-trained ImageNet weights for models like VGG16, VGG19, InceptionV3, ResNet152, EfficientNetB7 and DenseNet201which are used via keras. applications. vgg16. The input size of ImageNet images can be adjusted as required. The model configuration involves defining the input shape and setting the weight parameter to 'imagine'. Consistent hyperparameters were applied across all transfer learning experiments, including 20 and 40 epochs, a batch size of 32 and 64, the Adam optimizer, and a categorical cross-entropy loss function for multi-class classification. Data augmentation techniques, such as rotation, flipping, and zooming, were employed to expand the training dataset. The CNN model utilized "ReLU" and "SoftMax" activation functions, with Adam as the optimizer. The ReLU function, known for its non-saturating and rapid response, returns 0 for negative inputs and the input value for positive inputs, defined as f = max. SoftMax, used in the network's final layer for multi-class classification, provides a discrete probability distribution over all classes, resulting in a single class output.

Table 4.2. Outcomes of Models on training and validation data set (Experiment 1)

Model	Training		Validation	
	Accuracy	Loss	Accuracy	Loss
VGG16	0.9938	0.5146	0.9976	0.5098
VGG19	0.9914	0.5168	1.0	0.4869
InceptionV3	0.9949	0.8208	0.9928	0.8266
ResNet152	0.9787	0.8445	0.9976	0.8073
EfficientNetB7	0.9885	0.8766	0.9928	0.8607
DenseNet201	0.9995	0.6381	0.9870	0.6517
CNN_Sequential_32	0.9801	1.6400	0.9951	1.6035

Table 4.3. Outcomes of Models on training and validation data set (Experiment 2)

Model	Training		Validation	
	Accuracy	Loss	Accuracy	Loss
VGG16	0.9687	0.8781	1.0	0.8250
VGG19	0.9843	0.8698	1.0	08327
InceptionV3	0.8750	1.2288	0.8771	1.2053
ResNet152	0.9843	0.9229	1.0	0.8414
DenseNet201	0.9829	0.8480	0.9920	0.8387
CNN_Sequential_64	0.9921	1.5714	0.9946	1.5781

The main difference between Table 4.2 and Table 4.3 lies in the outcomes of the models on training and validation datasets for Experiment 1 and Experiment 2, respectively. In Table 4.2, the models generally demonstrate higher training and validation accuracy with lower loss values compared to those in Table 4.3. suggesting differences in experimental setups, datasets, or hyperparameters between the two experiments. The table 4.2 and 4.3 summarizes the outcomes for various fine-tuned deep network models, listing their anticipated performance metrics such as Training accuracy, training loss, validation accuracy and validation loss. Each model's remarks provide insights into their strengths and any trade-offs related to inference loss and accuracy.

4.5.1. Experiments of VGG16 using Transfer Learning

The VGG16 convolutional neural network architecture has been extensively employed in numerous computer vision tasks, including image classification, feature extraction. Its architectural simplicity and strong performance in image classification tasks have made VGG16 a widely-adopted choice for various computer vision applications, such as image classification, and feature extraction. Furthermore, the modular and flexible nature of VGG16 facilitates easy adaptation and seamless integration into diverse frameworks and pipelines. The initial experiment of this study involved a self-designed convolutional neural network using the Conv2D class in Keras, which creates 2D convolutional layers. This layer CNN was built independently from the ground up. The following steps were involved in constructing and training this model. The input images were first extracted from the local storage directory. As part of the pre-processing stage, the input images were converted to the RGB color space and resized from 224x224x3 to 224x224x1.

Table 4.4. VGG16 architecture for image classification and feature extraction

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
max_pooling2d (MaxPooling2D)	(None, 3, 3, 512)	0
average_pooling2d (AveragePooling2D)	(None, 1, 1, 512)	0
batch_normalization (BatchNormalization)	(None, 1, 1, 512)	2,048
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 4)	2,052

Descriptions of VGG16 layer

The provided model is a convolutional neural network designed for image classification tasks. It comprises multiple convolutional layers, pooling layers, normalization layers, and dense layers. The input layer accepts images of size 224x224 with 3 RGB color channels. The first block includes two convolutional layers with 64 filters of size 3x3, followed by a max pooling layer that reduces the spatial dimensions by half to 112x112 while maintaining 64 channels. The second block repeats this process with 128 filters, resulting in reduced dimensions of 56x56. The third block expands to three convolutional layers, each with 256 filters, and reduces the spatial dimensions to 28x28 using max pooling. The fourth and fifth blocks further increase the number of filters to 512 while progressively reducing the spatial dimensions to 14x14 and 7x7, respectively. Additional pooling layers further reduce the dimensions to 3x3 using max pooling and to 1x1 using average pooling. A batch normalization layer is included to stabilize and accelerate training. The model then flattens the 1x1x512 output into a 512-dimensional vector, which is passed through a fully connected dense layer with 512 units. A dropout layer is applied to prevent overfitting, and finally, a second dense layer with 4 units produces the output, likely corresponding to the classification of the input images into four classes.

Dropout is a regularization technique employed to mitigate overfitting. During the training process, it randomly sets a fraction of the input units to zero, which enhances the network's robustness and decreases the likelihood of overfitting. The non-trainable parameters originate from the pre-trained layers of the model, which are often frozen during training to preserve the learned features from the pre-trained weights. This approach is commonly used in transfer learning. The architecture is designed to progressively extract more intricate features from the input images through convolutional and pooling operations. These features are then utilized by the fully connected layers to perform the final classification into 4 categories. The inclusion of dropout improves generalization, and the use of pre-trained layers leverages previously learned features from a large dataset, thereby accelerating the training process and enhancing performance.

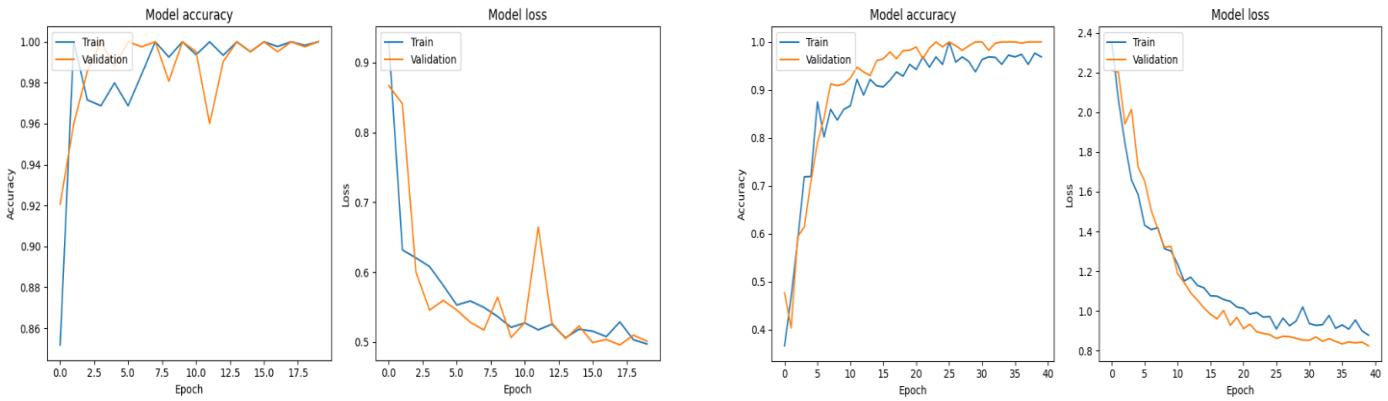


Figure. 4.1. VGG16 Model accuracy and model loss for Experiment 1(a) and 2(b)

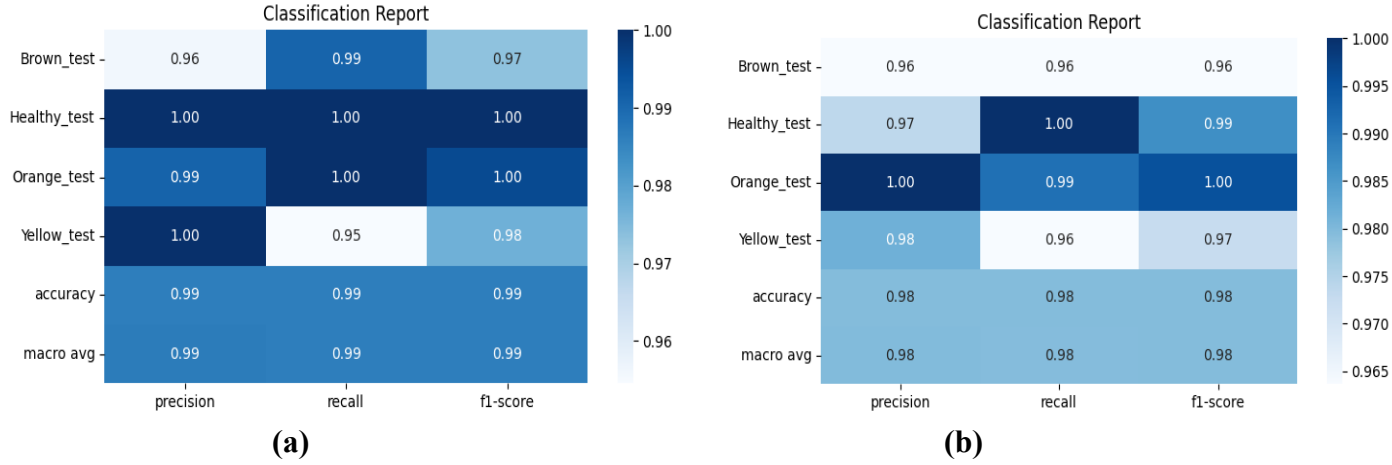
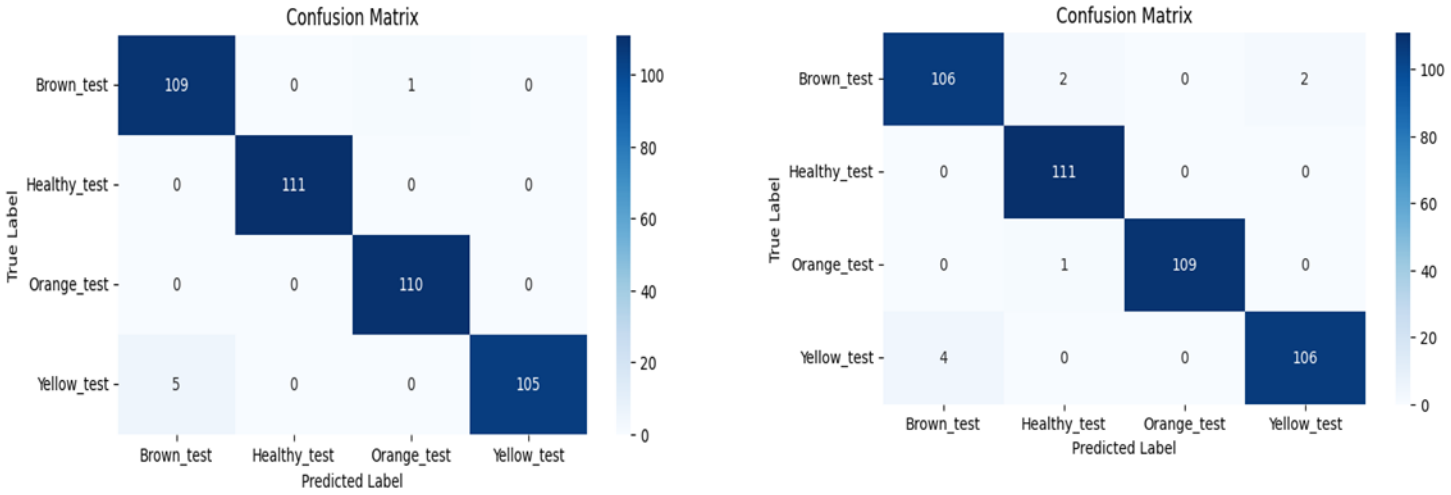


Figure 4.2. VGG16 Classification Reports for Experiment 1(a) and 2(b)

The reports show very high performance across all classes (brown, healthy, orange, and yellow). Precision, recall, and F1-score are generally above 0.95, indicating strong performance in identifying positive samples, capturing all positive samples, and a balance between precision and recall. The overall accuracy and macro average F1-score are also very high (around 0.98-0.99), suggesting excellent overall performance. While both reports demonstrate high performance, there are subtle differences based on Experiment 1(a): Which shows slightly higher performance overall. Most metrics are at or very close to 1.00, indicating near-perfect classification. The "Yellow" class has the lowest recall (0.95), suggesting a few instances of this class were misclassified. Experiment 2(b): Shows slightly lower overall performance compared to Report 1. While still very good, the metrics are generally a few points lower (e.g., recall for "Brown" is 0.96 compared to 0.99 in Report 1).



(a) **(b)**

Figure 4.3. VGG16 Confusion Matrix performance values for Experiment 1(a) and 2(b)

The Experiment 1(a) confusion matrix demonstrates slightly better overall performance, particularly for the Brown_test and Orange_test categories, with higher correct predictions (109 and 110, respectively) and fewer misclassifications than the Experiment 2(b) matrix. While both models classify Healthy_test perfectly (111 correct), the Experiment 2(b) matrix has a marginally better performance for the Yellow_test (106 correct vs. 105) but at the expense of increased misclassifications for the Brown_test, which is confused with Healthy_test and Yellow_test. The experiment 1(a) model appears more consistent overall, making fewer errors across all categories. To improve performance further, it would be beneficial to investigate why Brown_test and Yellow_test are occasionally confused and consider refining the model architecture, data balancing, or augmentation strategies.

4.5.2. Experiments of VGG19 using Transfer Learning

Leveraging transfer learning with the VGG19 model involves utilizing its pre-trained weights to fine-tune the network for a specific task, such as classifying a novel set of images. The VGG19 architecture is frequently trained on the expansive ImageNet dataset.

Table 4.5. VGG19 architecture for image classification and feature extraction

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
max_pooling2d (MaxPooling2D)	(None, 3, 3, 512)	0
average_pooling2d (AveragePooling2D)	(None, 1, 1, 512)	0
batch_normalization (BatchNormalization)	(None, 1, 1, 512)	2,048
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 4)	2,052

Model Architecture Layers Function

The neural network architecture in question is a deep Convolutional Neural Network, which is designed for image processing and classification tasks. The network begins with an Input Layer that accepts images of 224x224 pixels with 3 color channels. This layer has no learnable parameters. The first computational block, referred to as the First Convolutional Block, starts with the block1_conv1 layer. This layer applies 64 filters of size 3x3 to extract low-level features, such as edges, from the input image. This layer has 1,792 parameters. The subsequent layer, block1_conv2, also utilizes 64 filters to further process these extracted features, contributing 36,928 parameters to the network. The

block1_pool layer follows, which is a max-pooling layer that reduces the spatial dimensions by half, to 112x112, and keeps the most important features. The Second Convolutional Block starts with block2_conv1, a convolutional layer with 128 filters, doubling the feature maps and adding 73,856 parameters. This is followed by block2_conv2, another convolutional layer with 128 filters, refining these feature maps and adding 147,584 parameters. The block ends with block2_pool, a max-pooling layer that further reduces the spatial dimensions to 56x56, focusing on significant features. In the Third Convolutional Block, block3_conv1 begins with 256 filters, increasing the depth and complexity of the feature maps, with 295,168 parameters. This is followed by block3_conv2, block3_conv3, and block3_conv4, each with 256 filters, progressively refining the features, contributing 590,080 parameters each. The block concludes with block3_pool, which reduces the spatial dimensions to 28x28 through max-pooling. The Fourth Convolutional Block starts with block4_conv1, which has 512 filters and 1,180,160 parameters, capturing more complex features. It is followed by block4_conv2, block4_conv3, and block4_conv4, each adding 2,359,808 parameters, refining the features further. The block ends with block4_pool, a max-pooling layer that reduces the spatial dimensions to 14x14. The Fifth Convolutional Block mirrors the fourth, with block5_conv1 beginning with 512 filters and 2,359,808 parameters. It is followed by block5_conv2, block5_conv3, and block5_conv4, each also with 512 filters and contributing 2,359,808 parameters. The block concludes with the block5_pool layer, reducing the spatial dimensions to 7x7. Following the convolutional blocks, the network incorporates Final Layers for further processing. The max_pooling2d layer subsequently decreases the spatial dimensions to 3x3, succeeded by the average_pooling2d layer, which diminishes the dimensions to 1x1 by averaging the features within each channel. These pooling layers possess no trainable parameters. The batch normalization layer normalizes the output, stabilizing and accelerating the training process, with 2,048 parameters. The flattened layer converts the 2D feature maps into a 1D vector, preparing the data for the dense layers. The first dense layer, labeled 'dense', is fully connected with 512 neurons, combining the extracted features for classification, contributing 262,656 parameters. This is followed by a dropout layer, which randomly deactivates neurons during training to mitigate

overfitting. Lastly, the dense_1 layer is a fully connected layer with 4 neurons, corresponding to the final classification output, and has 2,052 parameters.

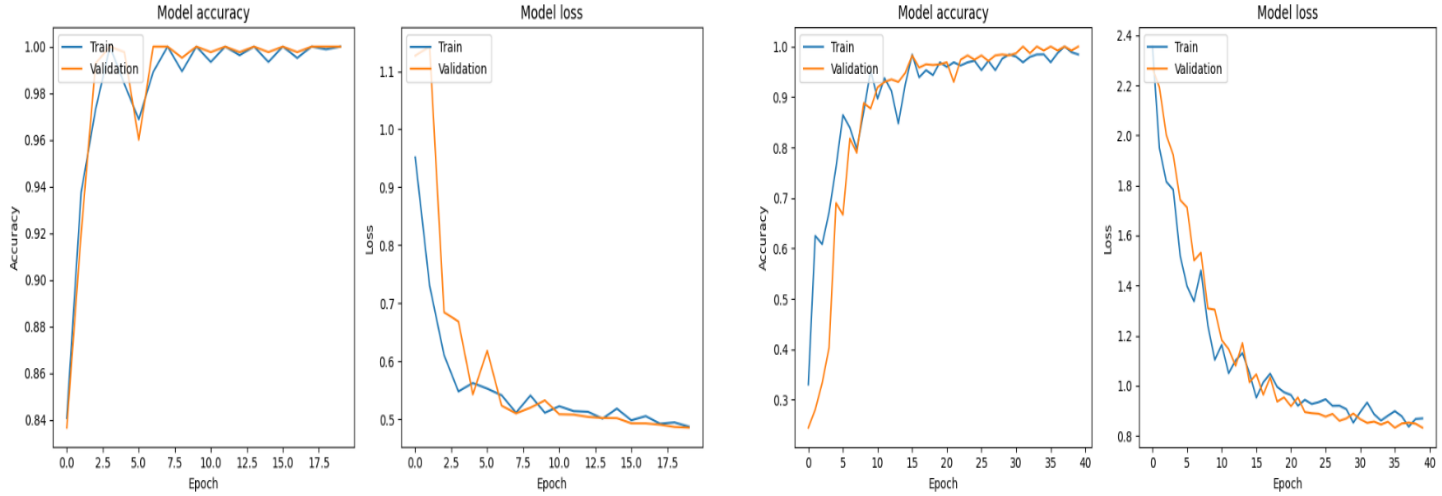


Figure. 4.4. VGG19 Model accuracy and model loss for Experiment 1(a) and 2(b)

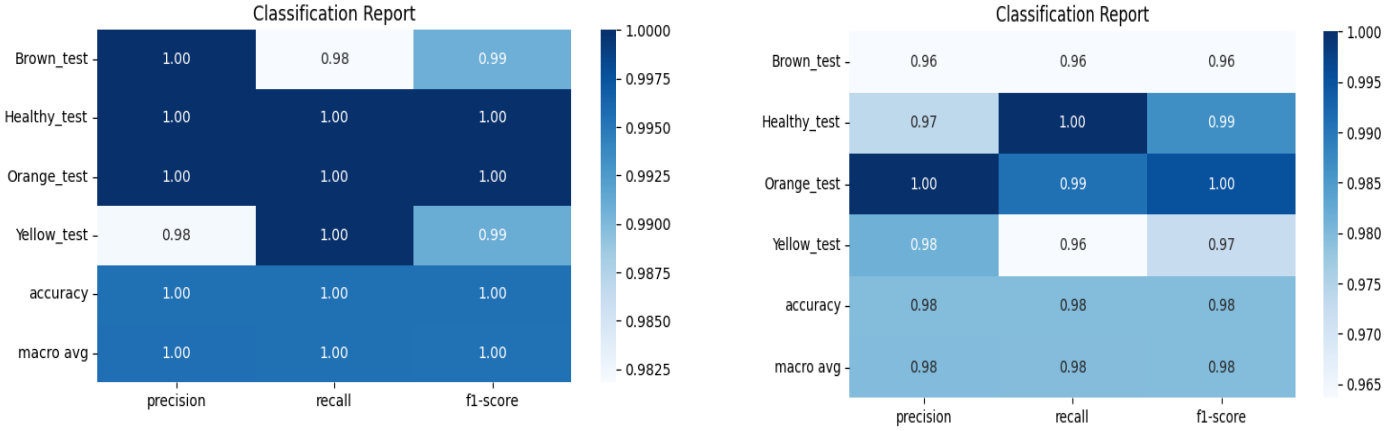


Figure 4.5. VGG19 Classification Reports for Experiment 1(a) and 2(b)

Both models demonstrate excellent performance, achieving high scores across precision, recall, and F1-score for most classes. This suggests that the models are effectively learning to distinguish between the different classes. Experiment 1(a): This model exhibits near-perfect performance, with most metrics at or very close to 1.00. Even the slightly lower scores (0.98 for "Brown test" recall and "Yellow test" precision) are still very high. This indicates exceptional accuracy in classifying instances and a strong ability to avoid both false positives and false negatives. Experiment 2(b): This model shows slightly

lower performance compared to Report 1, particularly for the "Healthy test" class, where the precision is notably lower (0.37). This suggests that the model may be misclassifying instances as "Healthy test" when they belong to other classes.

Figure 4.6. VGG19 Confusion Matrix performance values for Experiment 1(a) and 2(b)

The Experiment 1(a) confusion matrix shows better overall performance, particularly for Yellow_test, with all 110 samples correctly classified and no misclassifications, compared to the Experiment 2(b) matrix, where 5 Yellow_test samples are misclassified as Brown_test. Both models perform equally well for Healthy_test and Orange_test, with perfect classification of 111 and 110 samples, respectively. For Brown_test, the Experiment 2(b) matrix has a slight edge, correctly classifying 109 samples with only one misclassification to Orange_test, while the Experiment 1(a) matrix correctly classifies 108 samples with two misclassifications to Yellow_test. Overall, the Experiment 1(a) model demonstrates superior accuracy for Yellow_test, making it slightly more reliable, but both models show strong performance with minor areas for improvement in addressing specific misclassifications.

4.5.3. Experiment of InceptionV3 using Transfer Learning

The InceptionV3 model is used in this experiment to show how transfer learning may be used to classify images into four class Brown-test, Healthy-test, Orange-test, and Yellow-test. We may obtain good accuracy with little training data by utilizing the pretrained InceptionV3 model. The model's performance is thoroughly evaluated in the confusion matrix and classification report, which show that it performs exceptionally well for this classification assignment.

Table 4.6. InceptionV3 architecture for classification and feature extraction.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv2d (Conv2D)	(None, 111, 111, 32)	864	input_layer[0][0]
batch_normalization (BatchNormalization)	(None, 111, 111, 32)	96	conv2d[0][0]
activation (Activation)	(None, 111, 111, 32)	0	batch_normalization[0…
conv2d_1 (Conv2D)	(None, 109, 109, 32)	9,216	activation[0][0]
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 32)	96	conv2d_1[0][0]
activation_1 (Activation)	(None, 109, 109, 32)	0	batch_normalization_1…
conv2d_2 (Conv2D)	(None, 109, 109, 64)	18,432	activation_1[0][0]
batch_normalization_2 (BatchNormalization)	(None, 109, 109, 64)	192	conv2d_2[0][0]
activation_2 (Activation)	(None, 109, 109, 64)	0	batch_normalization_2…
max_pooling2d (MaxPooling2D)	(None, 54, 54, 64)	0	activation_2[0][0]
conv2d_3 (Conv2D)	(None, 54, 54, 80)	5,120	max_pooling2d[0][0]
batch_normalization_3 (BatchNormalization)	(None, 54, 54, 80)	240	conv2d_3[0][0]
activation_3 (Activation)	(None, 54, 54, 80)	0	batch_normalization_3…
batch_normalization_10 (BatchNormalization)	(None, 25, 25, 96)	288	conv2d_10[0][0]
batch_normalization_11 (BatchNormalization)	(None, 25, 25, 32)	96	conv2d_11[0][0]
activation_5 (Activation)	(None, 25, 25, 64)	0	batch_normalization_5…
activation_7 (Activation)	(None, 25, 25, 64)	0	batch_normalization_7…
activation_10 (Activation)	(None, 25, 25, 96)	0	batch_normalization_1…
activation_11 (Activation)	(None, 25, 25, 32)	0	batch_normalization_1…
conv2d_19 (Conv2D)	(None, 25, 25, 64)	18,432	mixed1[0][0]
conv2d_21 (Conv2D)	(None, 25, 25, 64)	76,800	activation_20[0][0]
conv2d_24 (Conv2D)	(None, 25, 25, 96)	82,944	activation_23[0][0]
conv2d_25 (Conv2D)	(None, 25, 25, 64)	18,432	average_pooling2d_2[0…
batch_normalization_19 (BatchNormalization)	(None, 25, 25, 64)	192	conv2d_19[0][0]
batch_normalization_21 (BatchNormalization)	(None, 25, 25, 64)	192	conv2d_21[0][0]
batch_normalization_24 (BatchNormalization)	(None, 25, 25, 96)	288	conv2d_24[0][0]

Model Architecture InceptionV3

The described model architecture consists of multiple interconnected layers forming a comprehensive image processing pipeline. It begins with an Input Layer that accepts images of dimensions 224x224 pixels and 3 RGB channels. This layer is non-trainable. The first convolutional block includes several Conv2D layers, starting with 32 filters of size 3x3, which generates an output shape and contains 864 trainable parameters. This is followed by Batch Normalization and Activation layers, which standardize the output and introduce non-linearity, respectively. Subsequent convolutional layers in this block employ 32, 64, and more filters, each with their own normalization and activation layers. Additionally, MaxPooling2D is used to reduce the spatial dimensions in this block. The second convolutional block starts with a Conv2D layer with 80 filters, producing an output shape of (54, 54, 80). This is succeeded by additional Conv2D layers with increasing filter counts, accompanied by Batch Normalization and Activation layers. MaxPooling2D further reduces the spatial dimensions. The third convolutional block

introduces a series of Conv2D layers with varying filter counts, each followed by their respective Batch Normalization and Activation layers.

Pooling and final convolutional layers include AveragePooling2D and additional Conv2D layers that refine and reduce the feature maps, preparing them for further processing. Concatenate layers, such as mixed0, mixed1, and mixed2, merge the outputs from different branches of the network, allowing the model to aggregate features from various convolutional paths. Convolutional layers applied to these concatenated outputs further process the features, supported by Batch Normalization and Activation layers. Finally, the architecture includes additional pooling and concatenation layers to compress and refine the feature maps. The output is a high-dimensional feature map of shape (None, 5, 5, 2048), indicating a deep feature extraction stage. This architecture is reminiscent of Inception-like networks, leveraging parallel convolutional paths with different kernel sizes to capture diverse feature abstractions. The use of concatenation layers facilitates the aggregation of features from these parallel paths, enhancing the model's ability to learn complex patterns.

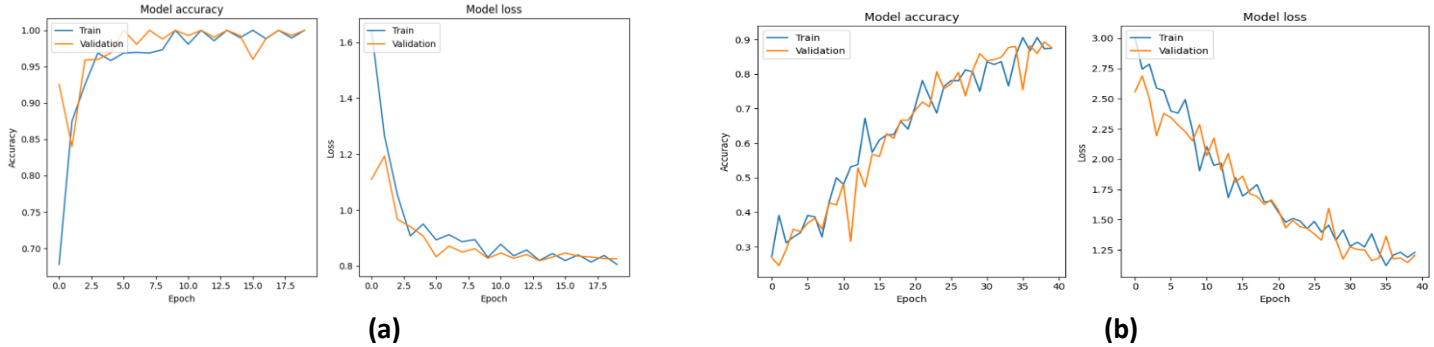


Figure. 4.7. InceptionV3 Model accuracy and model loss for Experiment 1

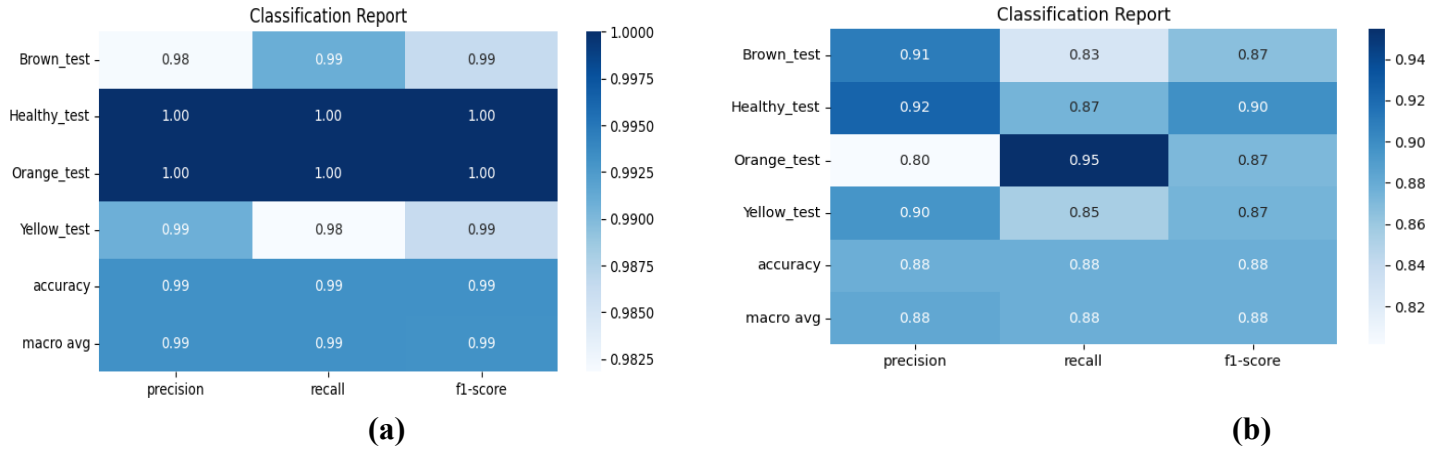
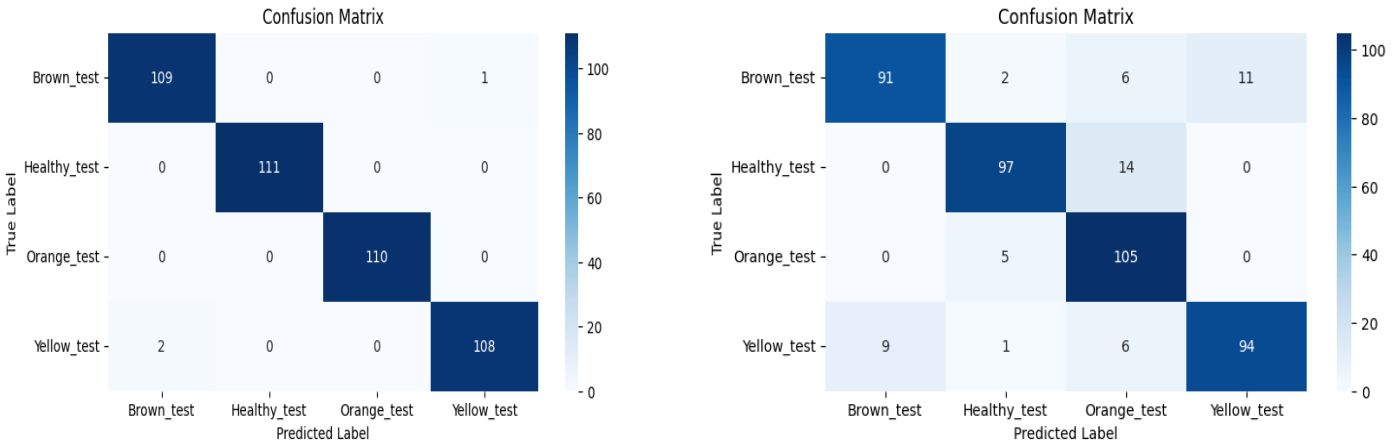


Figure 4.8. InceptionV3 Classification Reports for Experiment 1(a) and 2(b)

Both models generally perform well, with most metrics (precision, recall, F1-score) above 0.80, indicating good accuracy in classifying the four classes. Experiment 1(a): This model shows very high precision (0.99) and recall (0.99) for "Healthy test," "Orange test," and "Yellow test." "Brown test" has slightly lower performance, particularly in recall (0.98), suggesting some instances of this class were missed. Experiment 2(b): This model has more varied performance across the classes. "Orange test" stands out with the highest recall (0.95) but lower precision (0.80), indicating a tendency to over-classify instances as this class (potential false positives). "Yellow test" shows the lowest recall (0.85), meaning more instances of this class were missed compared to others.



(a) **(b)**

Figure 4.9. InceptionV3 Confusion Matrix values for Experiment 1(a) and 2(b)

The Experiment 1(a) confusion matrix indicates better overall performance than the Experiment 2(b) matrix, with higher accuracy across all classes. For the Brown_test, experiment 1(a) model correctly classifies 109 samples, with only 1 misclassification, while the experiment 2(b) model correctly classifies 91 samples, with notable misclassifications to Yellow_test (11) and Orange_test (6). For Healthy_test, the Experiment 1(a) model achieves perfect classification (111 correct), while the Experiment 2(b) model correctly classifies 97 samples, with 14 misclassified as Orange_test. For Orange_test, the Experiment 1(a) model perfectly classifies all 110 samples, whereas the Experiment 2(b) model misclassifies 5 samples as Healthy_test. Finally, for Yellow_test, the Experiment 1(a) model achieves high accuracy (108 correct, 2 misclassified as Brown_test), outperforming the Experiment 2(b) model, which correctly classifies 94 samples but misclassifies 9 as Brown_test and 6 as Orange_test.

4.5.4. Experiment of ResNet152 using Transfer Learning

This experiment demonstrates how to apply transfer learning using the ResNet152 model to classify images into four categories: Brown-test, Healthy-test, Orange-test, and Yellow-test. By leveraging the pretrained ResNet152 model, we can achieve high accuracy with minimal training data. The classification report and confusion matrix offer a comprehensive assessment of the model's performance, demonstrating its high effectiveness in this classification endeavor.

Table 4.7. ResNet152 architecture for image classification and feature extraction.

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
max_pooling2d (MaxPooling2D)	(None, 3, 3, 2048)	0	conv5_block3_out[0][0]
average_pooling2d (AveragePooling2D)	(None, 1, 1, 2048)	0	max_pooling2d[0][0]
batch_normalization (BatchNormalization)	(None, 1, 1, 2048)	8,192	average_pooling2d[0][…
flatten (Flatten)	(None, 2048)	0	batch_normalization[0…
dense (Dense)	(None, 512)	1,049,088	flatten[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 4)	2,052	dropout[0][0]
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out[0][0… conv2_block2_3_bn[0][…
conv2_block2_out (Activation)	(None, 56, 56, 256)	0	conv2_block2_add[0][0]
conv2_block3_1_conv (Conv2D)	(None, 56, 56, 64)	16,448	conv2_block2_out[0][0]
conv2_block3_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	conv2_block3_1_conv[0…
conv2_block3_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block3_1_bn[0][…
conv2_block3_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block3_1_relu[0…
(Activation)			
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_relu[0…
conv2_block1_2_bn (BatchNormalization)	(None, 56, 56, 64)	256	conv2_block1_2_conv[0…
conv2_block1_2_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][…
conv2_block1_0_conv (Conv2D)	(None, 56, 56, 256)	16,640	pool1_pool[0][0]
conv2_block1_3_conv (Conv2D)	(None, 56, 56, 256)	16,640	conv2_block1_2_relu[0…
conv2_block1_0_bn	(None, 56, 56, 256)	1,024	conv2_block1_0_conv[0…

Model Architecture ResNet152

The network architecture begins with an Input Layer that accepts images of size 224x224 pixels with 3 color channels, yielding an output shape with no associated parameters. Subsequently, a Zero Padding layer expands the image dimensions to 230x230, preserving the depth while introducing no additional parameters. The Convolutional Layer then applies 64 filters of size 7x7 to the padded input, reducing the spatial dimensions to 112x112 and incorporating 9,472 parameters. This is followed by Batch Normalization, which normalizes the convolutional layer's output to stabilize and accelerate the training process, encompassing 256 parameters. The Activation Function applies the ReLU non-linearity to the normalized output. Finally, another Zero Padding layer maintains the spatial dimensions before Max Pooling, which down samples the feature maps from 112x112 to 56x56 by extracting the maximum value within each 2x2 window, without any additional parameters.

The network then employs multiple Residual Blocks, starting with the first block that includes several convolutional layers with batch normalization and ReLU activations. This block, named conv2_block1, incorporates skip connections through the conv2_block1_add layer, which adds the input to the output of the convolutions, mitigating the vanishing gradient problem. The second, third, fourth, fifth, and sixth residual blocks (conv2_block2, conv2_block3, conv3_block1, conv3_block2, and conv3_block3) similarly build deeper features and refine representations through more convolutions and skip connections. In the fourth stage, the Residual Block (conv4_block1) includes convolutions with an increased number of filters and reduced spatial dimensions to 14x14, facilitating the capture of more abstract features. This stage progresses through additional blocks (conv4_block32, conv4_block33, conv4_block34) with residual connections, including layers like conv4_block32_1_conv to conv4_block36_add, applying a combination of convolutions, batch normalization, and ReLU activations to stabilize training and improve performance. Finally, in the fifth stage, the Residual Block (conv5_block1) reduces spatial dimensions to 7x7 while expanding depth to 2048 with convolutions and residual connections. This stage continues through further blocks (conv5_block2), involving convolutional layers with batch normalization and ReLU activations, effectively capturing high-level features before concluding with the final output layer.

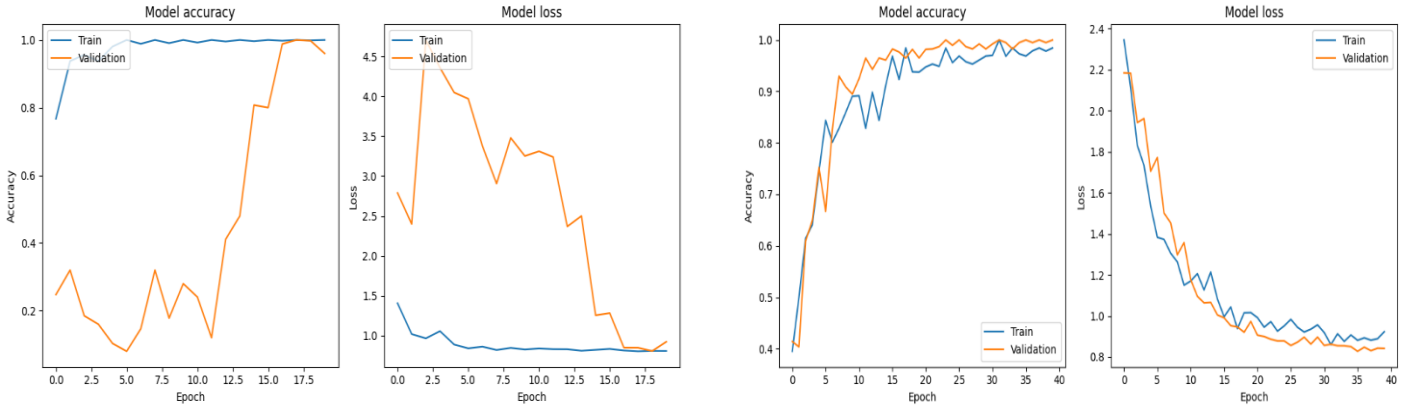


Figure. 4.10. ResNet152 Model accuracy and model loss for Experiment 1

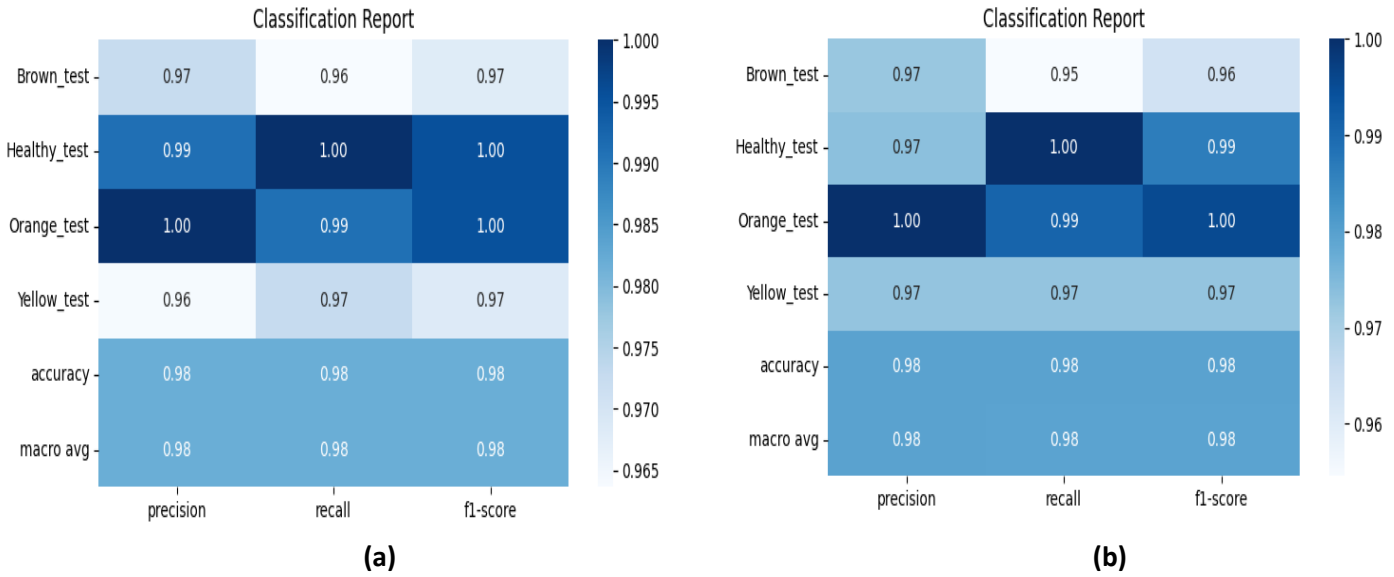
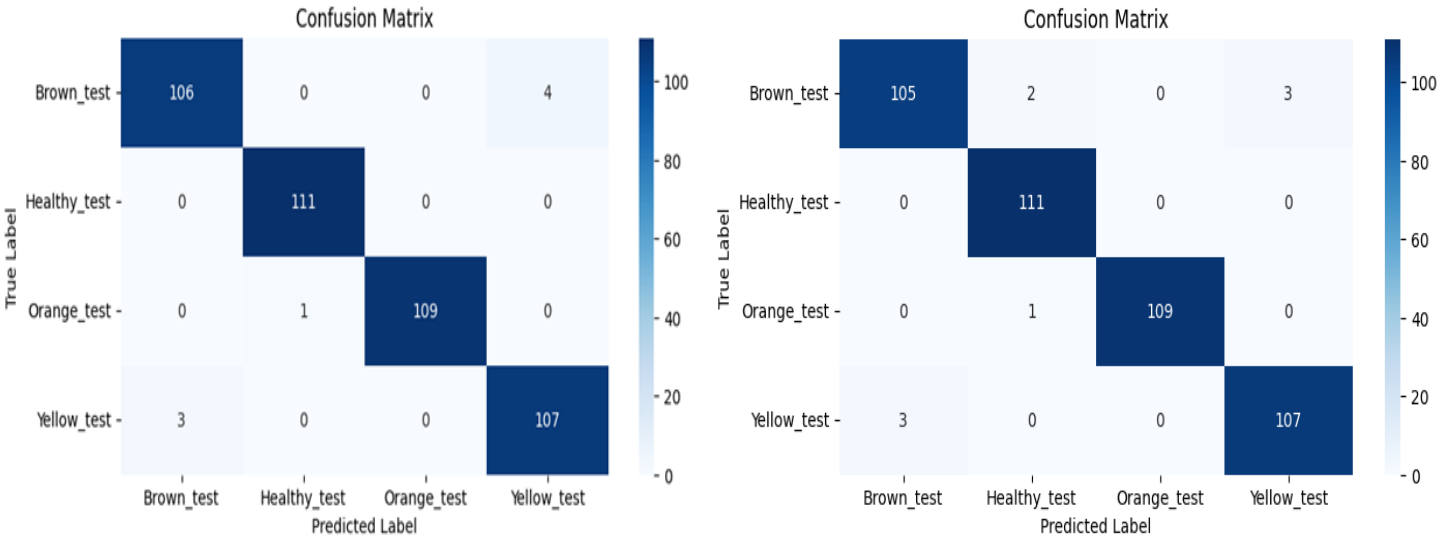


Figure 4.11. ResNet152 Classification Reports for Experiment 1(a) and 2(b)

Both models exhibit excellent performance with high precision, recall, and F1 scores across all classes. This indicates that both models are effectively learning to distinguish between the four classes with high accuracy. Experiment 1(a): This model shows near-perfect performance for "healthy test" and "orange test," with all metrics at 1.00. "Brown test" and "yellow test" have slightly lower, but still very good, performance, particularly in F1-score (0.97 for both). Experiment 1(a): This model shows slightly more balanced performance across all classes, with all F1 scores at 0.96 or higher. "Orange test" stands out with perfect precision, recall, and F1-score (1.00).



(a) (b)

Figure 4.12. ResNet152 Confusion Matrix performance values for Experiment 1(a) and 2(b)

The two confusion matrices depict the performance of a classification model on four classes: brown, healthy, orange, and yellow. Both matrices show strong diagonal dominance, indicating high accuracy in classifying instances correctly. Most classes exhibit near-perfect precision and recall, with "Healthy" achieving perfect classification in both matrices. Minor misclassifications occur: in both matrices, one "orange" sample is misclassified as "healthy," and three "yellow" samples are misclassified as "brown." The Experiment 1(a) matrix shows four "brown" samples misclassified as "yellow," while the Experiment 2(b) matrix shows two "brown" samples misclassified as "healthy" and three misclassified as "yellow." Overall, the model performs very well, with slight variations in the "Brown" class misclassifications between the two evaluations.

4.5.5. Experiment of EfficientNetB7 using Transfer Learning

The experiment using EfficientNetB7 with Transfer Learning begins by preparing the dataset, resizing images to 224x224, and applying data augmentation to enhance diversity. The pre-trained EfficientNetB7 model, excluding its top classification layer, is used as a feature extractor, leveraging its learned hierarchical features from ImageNet. A custom classification head is added, and the initial layers of EfficientNetB7 are frozen to preserve pre-trained weights.

Table 4.8. EfficientNetB7 architecture for image classification and feature extraction.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
rescaling (Rescaling)	(None, 224, 224, 3)	0	input_layer[0][0]
normalization (Normalization)	(None, 224, 224, 3)	7	rescaling[0][0]
rescaling_1 (Rescaling)	(None, 224, 224, 3)	0	normalization[0][0]
stem_conv_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	rescaling_1[0][0]
stem_conv (Conv2D)	(None, 112, 112, 64)	1,728	stem_conv_pad[0][0]
stem_bn (BatchNormalization)	(None, 112, 112, 64)	256	stem_conv[0][0]
stem_activation (Activation)	(None, 112, 112, 64)	0	stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D)	(None, 112, 112, 64)	576	stem_activation[0][0]
block1a_bn (BatchNormalization)	(None, 112, 112, 64)	256	block1a_dwconv[0][0]
block1a_activation (Activation)	(None, 112, 112, 64)	0	block1a_bn[0][0]
block6j_project_bn (BatchNormalization)	(None, 7, 7, 384)	1,536	block6j_project_conv[…
block6j_drop (Dropout)	(None, 7, 7, 384)	0	block6j_project_bn[0]…
block6j_add (Add)	(None, 7, 7, 384)	0	block6j_drop[0][0], block6i_add[0][0]
block6k_expand_conv (Conv2D)	(None, 7, 7, 2304)	884,736	block6j_add[0][0]
block6k_expand_bn (BatchNormalization)	(None, 7, 7, 2304)	9,216	block6k_expand_conv[0…
block6k_expand_activation (Activation)	(None, 7, 7, 2304)	0	block6k_expand_bn[0][…
block6k_dwconv (DepthwiseConv2D)	(None, 7, 7, 2304)	57,600	block6k_expand_activa…
block6k_bn (BatchNormalization)	(None, 7, 7, 2304)	9,216	block6k_dwconv[0][0]
block6k_activation (Activation)	(None, 7, 7, 2304)	0	block6k_bn[0][0]
block6k_se_squeeze (GlobalAveragePooling2D)	(None, 2304)	0	block6k_activation[0]…
block6k_se_reshape (Reshape)	(None, 1, 1, 2304)	0	block6k_se_squeeze[0]…
block7d_add (Add)	(None, 7, 7, 640)	0	block7d_drop[0][0], block7c_add[0][0]
top_conv (Conv2D)	(None, 7, 7, 2560)	1,638,400	block7d_add[0][0]
top_bn (BatchNormalization)	(None, 7, 7, 2560)	10,240	top_conv[0][0]
top_activation (Activation)	(None, 7, 7, 2560)	0	top_bn[0][0]
max_pooling2d (MaxPooling2D)	(None, 3, 3, 2560)	0	top_activation[0][0]
average_pooling2d (AveragePooling2D)	(None, 1, 1, 2560)	0	max_pooling2d[0][0]
batch_normalization (BatchNormalization)	(None, 1, 1, 2560)	10,240	average_pooling2d[0][…
flatten (Flatten)	(None, 2560)	0	batch_normalization[0…
dense (Dense)	(None, 512)	1,311,232	flatten[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 4)	2,052	dropout[0][0]

Total params: 65,421,211 (249.56 MB)
Trainable params: 65,105,364 (248.36 MB)
Non-trainable params: 315,847 (1.20 MB)

The model processes images of shape (224, 224, 3) through initial rescaling and normalization layers to standardize inputs, followed by a series of efficient depth wise separable convolutions that reduce computational overhead while maintaining accuracy. With over 258 million parameters, EfficientNetB7 is designed to balance performance and efficiency, making it suitable for tasks like image classification or disease Classification in crops. For optimal performance on smaller datasets, techniques like fine-tuning, data augmentation, and regularization can be applied to enhance generalization and mitigate overfitting. This architecture provides a robust starting point for addressing complex image-based problems with limited training data.

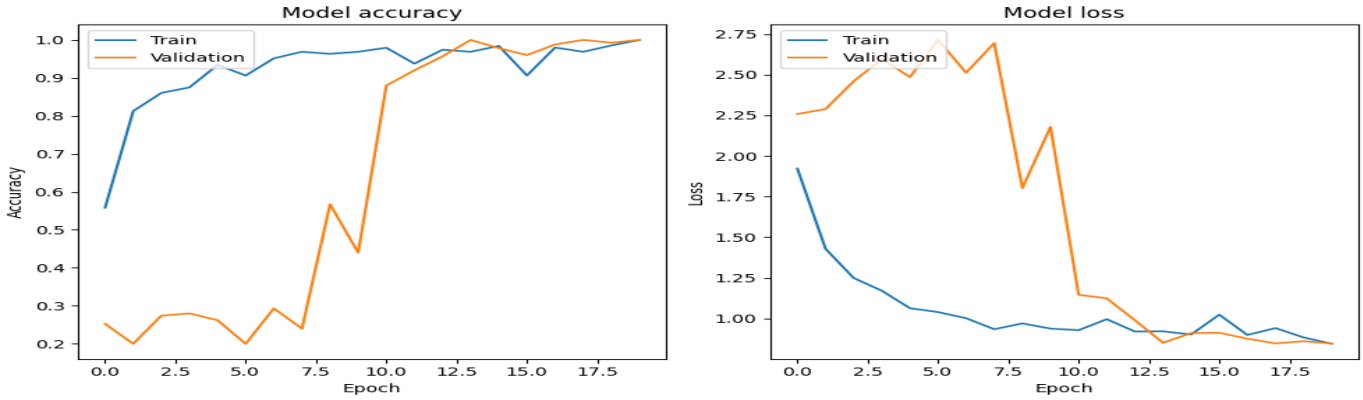


Figure. 4.13. EfficientNetB7Model accuracy and model loss for Experiment 1

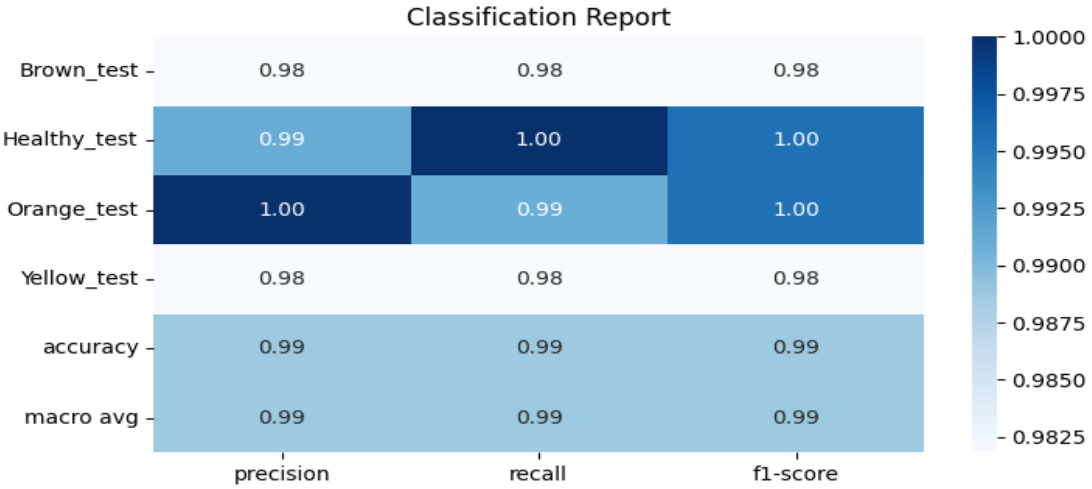


Figure 4.14. EfficientNetB7 Classification Reports for Experiment 1

The model demonstrates excellent performance across all metrics (precision, recall, and F1-score) and for all classes. The accuracy and macro average F1-score are also very high (0.99), indicating strong overall performance. healthy test: This class achieves perfect recall (1.00) and very high precision (0.99), indicating that the model correctly identifies all instances of this class with very few false positives. orange test: This class also performs exceptionally well, with perfect precision (1.00) and

very high recall (0.99), showing that the model rarely misclassifies instances as this class and correctly identifies almost all instances. brown test and yellow test: Both these classes achieve very high precision and recall (0.98 for both), indicating a strong ability to correctly classify instances of these classes.

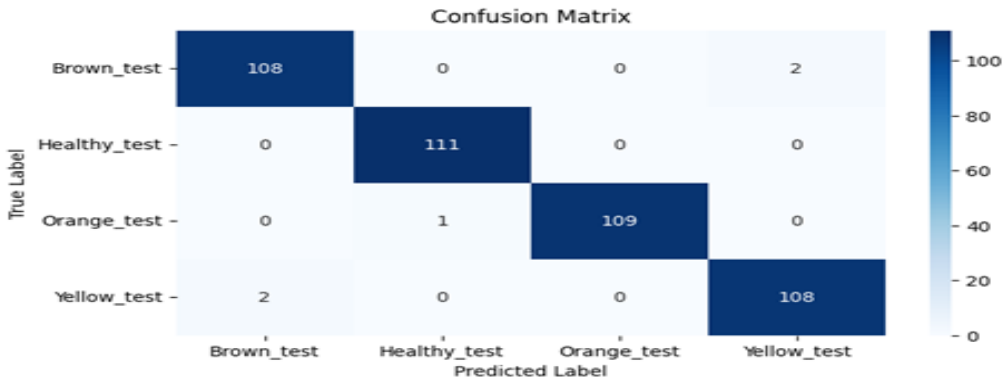


Figure 4.15. EfficientNetB7 Confusion Matrix performance values for Experiment 1

The confusion matrix evaluates the model's classification performance across four categories: Brown_test, Healthy_test, Orange_test, and Yellow_test, demonstrating high accuracy overall. Brown_test has 108 correct classifications, with 2 misclassified as Yellow_test, while Healthy_test achieves perfect classification with all 111 samples correctly identified. Similarly, Orange_test records 109 correct classifications, with just 1 misclassified as Healthy_test, and Yellow_test shows 108 correct predictions, with 2 misclassified as Brown_test. Despite the model's strong performance, minor misclassifications in Brown_test, Orange_test, and Yellow_test highlight areas where further improvements could enhance its accuracy.

4.5.6. Experiment of DenseNet201 using Transfer Learning

DenseNet201 is a deep convolutional neural network architecture that utilizes a unique connectivity pattern, where each layer is connected to every subsequent layer within a dense block. This design enhances feature reuse and significantly reduces the number of parameters compared to traditional architectures. With 201 layers, DenseNet201 is a highly efficient model, balancing depth and computational efficiency. It employs batch normalization, ReLU activation, and convolutional and pooling layers to extract features effectively. Its compactness and efficiency make DenseNet201 a popular choice for tasks requiring high accuracy while maintaining reasonable resource requirements.

Table 4.9. DenseNet201 architecture for image classification and feature extraction.

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
zero_padding2d (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,408	zero_padding2d[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
zero_padding2d_1 (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1 (MaxPooling2D)	(None, 56, 56, 64)	0	zero_padding2d_1[0][0]
conv2_block1_0_bn (BatchNormalization)	(None, 56, 56, 64)	256	pool1[0][0]
conv5_block32_concat (Concatenate)	(None, 7, 7, 1920)	0	conv5_block31_concat[… conv5_block32_2_conv[…
bn (BatchNormalization)	(None, 7, 7, 1920)	7,680	conv5_block32_concat[…
relu (Activation)	(None, 7, 7, 1920)	0	bn[0][0]
max_pooling2d (MaxPooling2D)	(None, 3, 3, 1920)	0	relu[0][0]
average_pooling2d (AveragePooling2D)	(None, 1, 1, 1920)	0	max_pooling2d[0][0]
batch_normalization (BatchNormalization)	(None, 1, 1, 1920)	7,680	average_pooling2d[0][…
flatten (Flatten)	(None, 1920)	0	batch_normalization[0…
dense (Dense)	(None, 512)	983,552	flatten[0][0]
dropout (Dropout)	(None, 512)	0	dense[0][0]
dense_1 (Dense)	(None, 4)	2,052	dropout[0][0]

The model is a functional neural network architecture with an input shape of (224,224,3)(224, 224, 3)(224,224,3), designed to process image data. It starts with an **InputLayer**, followed by **ZeroPadding2D** and a convolutional layer (Conv2D) with 64 filters, progressively extracting features. Subsequent layers include **BatchNormalization** and **Activation** (ReLU) for stabilizing and enhancing learning. The architecture employs **Concatenate** operations, combining features across blocks, and DenseNet-inspired patterns of connecting convolutional layers in blocks. These blocks gradually increase the number of channels while maintaining the spatial resolution using pooling and down-sampling layers like **MaxPooling2D** and **AveragePooling2D**. Each block includes Batch Normalization, ReLU activation, 2D convolution layers, and a concatenation step to merge previous feature maps, thereby ensuring efficient feature reuse. Starting from the conv3_block1 layer, feature dimensions gradually increase through successive operations, while the network's depth expands. Each layer performs specific operations like batch normalization, activation (ReLU), convolution, and

concatenation. In each block, feature maps from all preceding layers are concatenated to form the input for subsequent layers, enhancing feature propagation and reducing the risk of vanishing gradients. The convolutions progressively reduce the feature dimensions while increasing the channel depth. This structure exemplifies the key DenseNet principle of reusing features across layers to achieve computational efficiency and enhanced representational capacity.

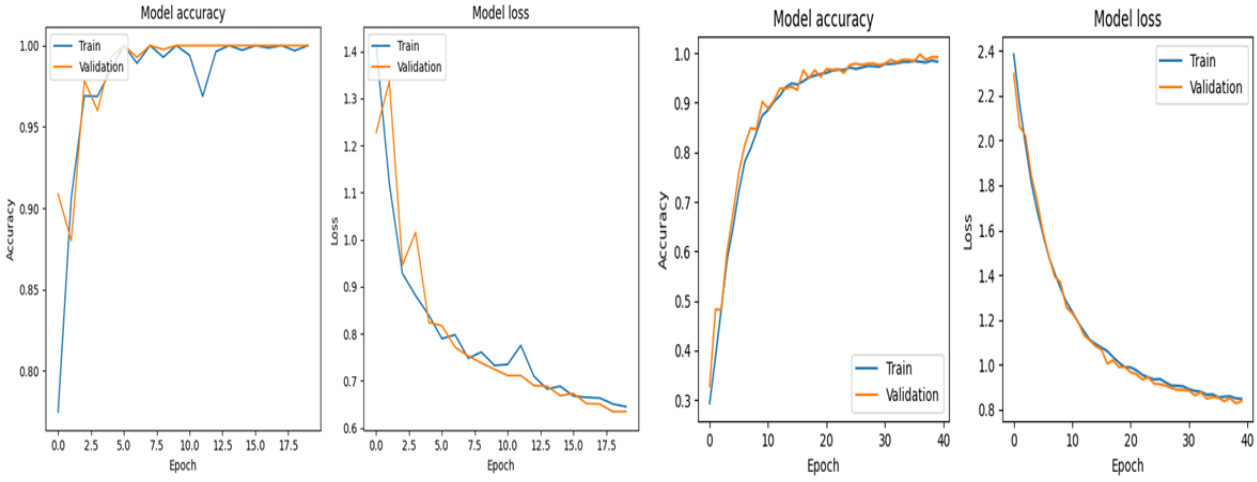


Figure. 4.16. DenseNet201 Model accuracy and model loss for Experiment 1(a) and 2(b)

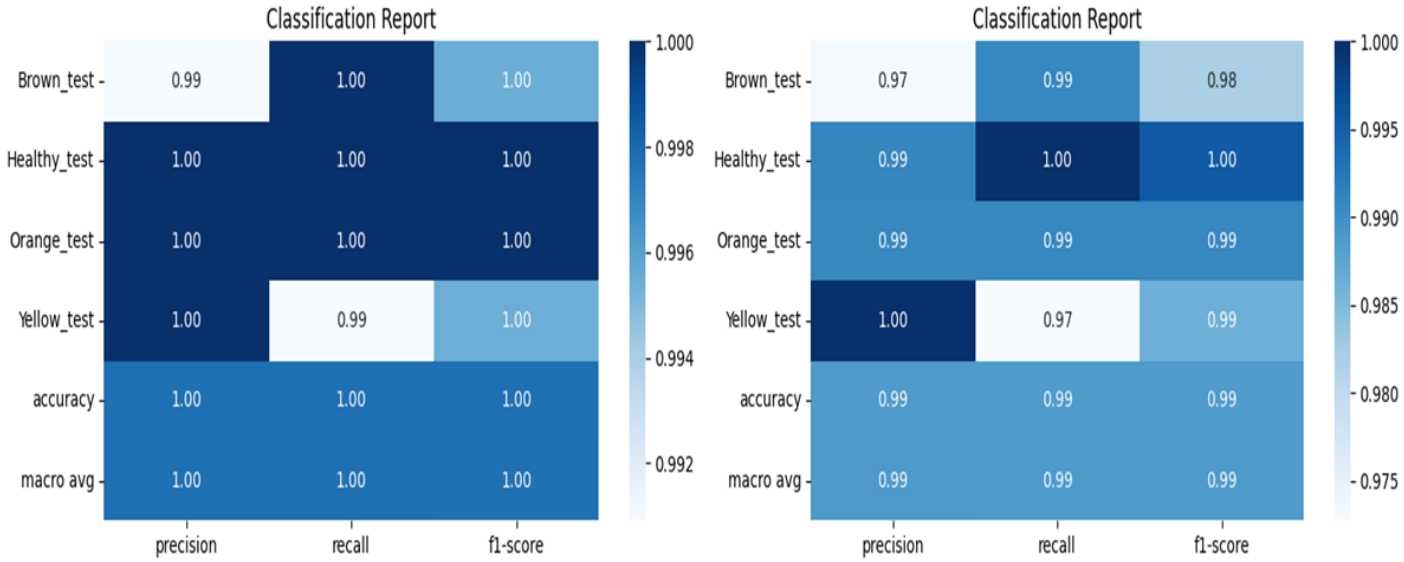
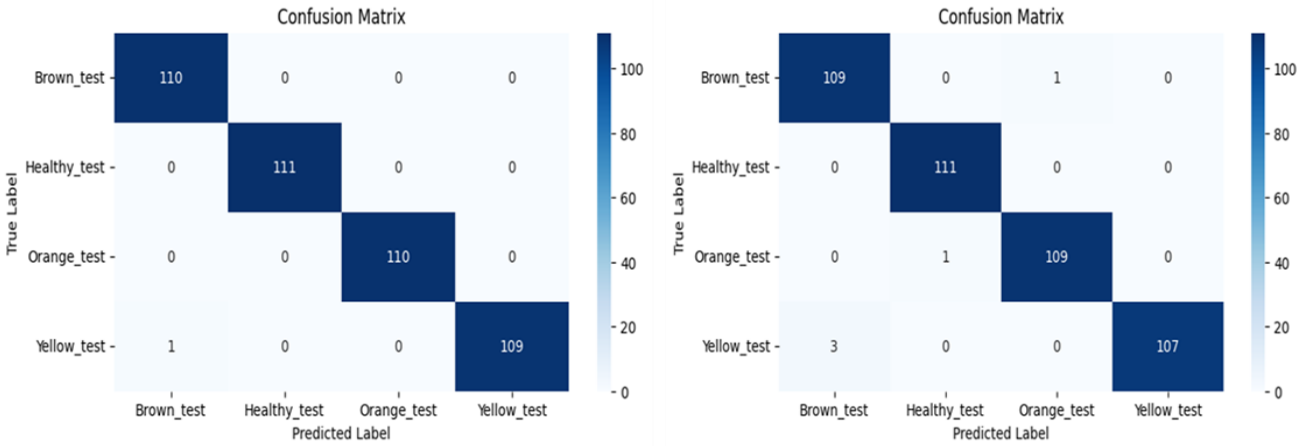


Figure 4.17. DenseNet201 Classification Reports for Experiment 1(a) and 2(b)

Both reports indicate excellent performance for DenseNet201. Most metrics (precision, recall, F1-score) are very high, close to or at 1.00, suggesting robust classification capabilities. Experiment 1(a):

This report shows near-perfect performance. All metrics for "brown test," "healthy test," and "orange test" are at 1.00, indicating perfect classification for these classes. "Yellow test" has a slightly lower recall (0.99), but still very high, suggesting minimal misclassifications. The overall accuracy and macro average F1-score are also 1.00, signifying excellent overall performance. Experiment 2(b): This report also demonstrates excellent performance, though slightly less perfect than Report 1. "Healthy test" achieves perfect scores across all metrics. "Brown test" and "orange test" have very high precision and recall (0.97-0.99), indicating very few misclassifications. "Yellow test" has a slightly lower recall (0.97), but still very good. The overall accuracy and macro average F1-score are 0.99, indicating very strong overall performance.



(a) **(b)**

Figure. 4.18. DenseNet201 Confusion Matrix values for Experiment 1(a) and 2(b)

The confusion matrices depict the performance of two different models or variations of a single model in classifying four categories: Brown_test, Healthy_test, Orange_test, and Yellow_test. In the first matrix (a), the model achieves near-perfect accuracy, with only one misclassification where a Yellow_test sample is classified as Brown_test. In the second matrix (b), the model shows slightly degraded performance, with misclassifications spread across more categories. Specifically, one Brown_test is classified as Orange_test, one Orange_test as Yellow_test, and three Yellow_test samples as Brown_test. Overall, while both models perform well, the first model demonstrates better precision and fewer errors.

4.5.7. Experiment of CNN Sequential Learning

CNN Sequential Learning is a structured approach for designing convolutional neural networks (CNNs) layer-by-layer using a sequential model. This method allows for the straightforward stacking of layers, such as convolutional, batch normalization, pooling, and dense layers, in a defined order. Each layer performs a specific role, with convolutional layers extracting features, pooling layers reducing spatial dimensions, and dense layers enabling final classification. For example, in this model, a series of convolutional layers with increasing filters (32, 64, 128, 256, and 512) are interspersed with max-pooling and batch normalization to enhance stability and reduce overfitting. A fully connected dense layer, followed by dropout, ensures robust learning before the final classification layer.

Table 4.10. Experiment of CNN Sequential Learning

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
batch_normalization (BatchNormalization)	(None, 222, 222, 32)	128
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 52, 52, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_3 (Conv2D)	(None, 24, 24, 256)	295,168
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_3 (Conv2D)	(None, 24, 24, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 24, 24, 256)	1,024
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 256)	0
conv2d_4 (Conv2D)	(None, 10, 10, 512)	1,180,160
batch_normalization_4 (BatchNormalization)	(None, 10, 10, 512)	2,048
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 512)	0
flatten (Flatten)	(None, 12800)	0
dense (Dense)	(None, 512)	6,554,112
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 4)	2,052

Total params: 8,128,708 (31.01 MB)

Trainable params: 8,126,724 (31.00 MB)

Non-trainable params: 1,984 (7.75 KB)

Model Architecture CNN Sequential

The model is a sequential convolutional neural network (CNN) designed for a classification task involving four output classes. It begins with a 3×33 \times 33×3 convolutional layer (conv2d) that

extracts features using 32 filters, resulting in an output shape of (None,222,222,32) (None, 222, 222, 32) (None,222,222,32), where the batch size is unspecified (None), and the spatial dimensions are 222 by 222 with 32 feature maps. This layer has 896 parameters, calculated based on the filter size, input channels, and bias terms. Following this, a batch normalization layer (batch normalization) stabilizes and accelerates training by normalizing activations, adding 128 trainable parameters for scaling and shifting. A max pooling layer (max_pooling2d) then reduces the spatial dimensions by half, with no additional parameters. The network deepens through additional convolutional blocks, each consisting of a convolutional layer, a batch normalization layer, and a max pooling layer. The number of filters doubles at each stage, increasing from 32 to 64, 128, 256, and finally 512, enabling the model to extract progressively more abstract features. Correspondingly, the spatial dimensions reduce after pooling operations, transitioning from (222,222) (222, 222) (222,222) to (111,111) (111, 111) (111,111), then (54,54) (54, 54) (54,54), and eventually to (5,5) (5, 5) (5,5). Each convolutional layer significantly increases the number of trainable parameters due to the growing number of filters and input channels. After the convolutional layers, the output is flattened into a vector of size 12,800 and fed into a dense layer with 512 neurons. This layer, with 6,554,112 trainable parameters, serves as a fully connected layer for feature interpretation. A dropout layer follows, reducing overfitting by randomly disabling neurons during training, though it does not add any parameters. The final dense layer contains four neurons corresponding to the output classes, with 2,052 parameters. In total, the model has 8,128,708 parameters, of which 8,126,724 are trainable and 1,984 are non-trainable. With its deep architecture, batch normalization, and dropout, this model is well-suited for complex image classification tasks.

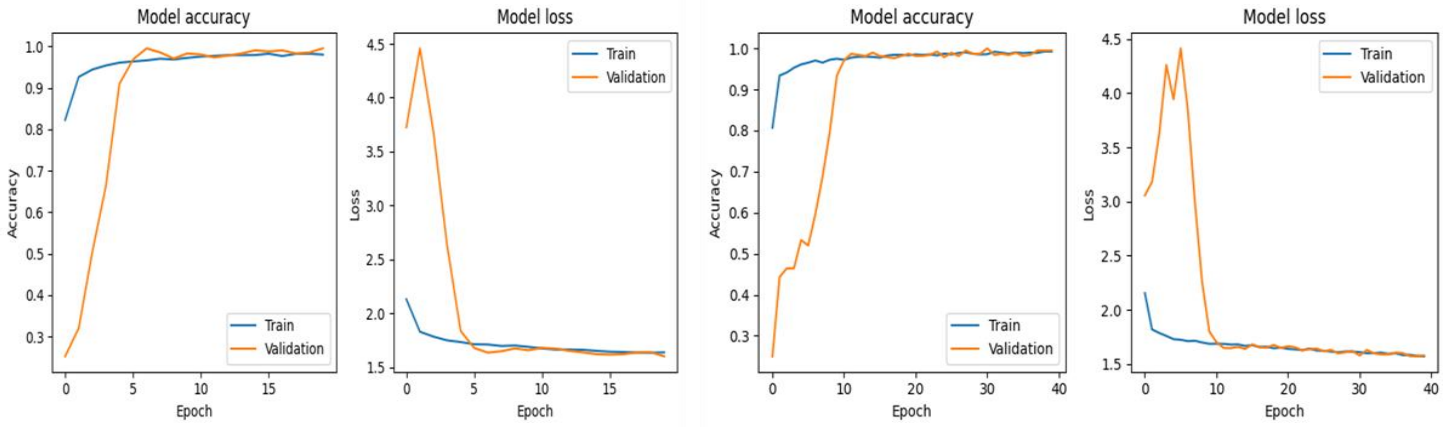
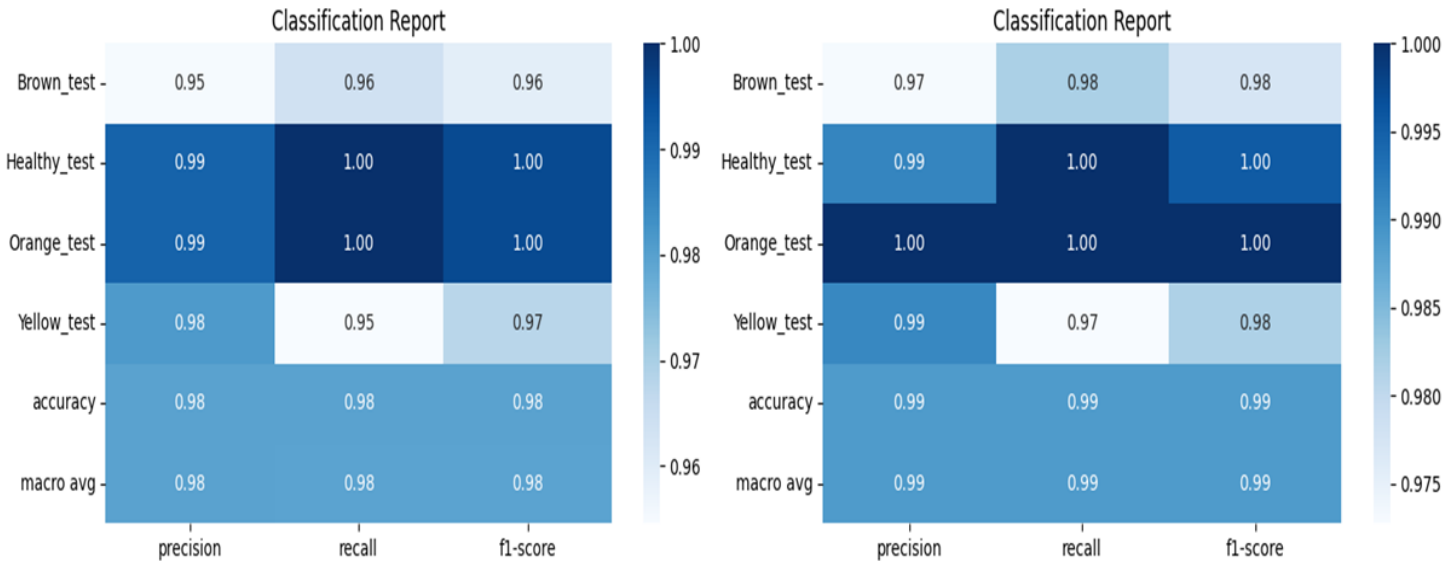


Figure.4.19. CNN Model accuracy and model loss for Experiment 1(a) and 2(b)



(a) (b)

Figure 4.20. CNN_Sequential Classification Reports for Experiment 1(a) and 2(b)

Both models show good performance, with most metrics (precision, recall, F1-score) above 0.90, indicating a strong ability to classify the four classes. However, the performance is not as consistently high as seen in some of the other models (like DenseNet201, EfficientNetB7). Experiment 1(a): "brown test" has lower recall (0.95), suggesting some instances of this class were missed. "Healthy test" and "orange test" have very high performance, with all metrics at or very close to 1.00. "Yellow test" has a lower recall (0.95) and F1-score (0.97), indicating some instances were misclassified. Experiment 2(b): All metrics for "healthy test" and "orange test" are at 1.00, indicating perfect classification for these classes. "Brown test" and "yellow test" have very high precision and recall (0.97-0.99), indicating very few misclassifications.

(a) (b)

Figure. 4.21. CNN_Sequential Confusion Matrix values for Experiment 1(a) and 2(b)

The confusion matrices compare the performance of two models in classifying four categories: Brown_test, Healthy_test, Orange_test, and Yellow_test. In the first (a) matrix, the model shows slight misclassifications, with 106 Brown_test samples correctly classified and minor errors spread across other categories, while Healthy_test and Orange_test are perfectly classified (111 and 110, respectively). For Yellow_test, 105 samples are correctly classified, with 5 misclassified as Brown_test. In the second (b) matrix, performance improves, with 108 Brown_test samples correctly classified and fewer misclassifications overall. Healthy_test and Orange_test maintain perfect classification, while Yellow_test accuracy improves to 107 correct classifications, with 3 misclassified as Brown_test.

As shown in Table 4.1, the difference between Experiments 1 and 2, the evaluation of model accuracy and loss focused on various networks tested in Experiments 1(a) and 2(b). Results demonstrated that the models began converging at lower epochs, reflecting an effective balance between model fitting and complexity. Both training and validation accuracy showed steady improvement, culminating in full convergence. The best-performing model achieved a remarkable validation accuracy of 99.80% and a validation loss of 0.46, surpassing all other models. Training accuracy and loss closely aligned with validation metrics, indicating a well-optimized model with minimal overfitting. As accuracy increased, loss decreased, and both metrics stabilized, resulting in an optimal fit. The performance of models using architectures such as VGG16, VGG19, InceptionV3, ResNet152, EfficientNetB7, DenseNet201, and a custom CNN_Sequential was also evaluated. These models exhibited strong convergence behavior, reaching full convergence between epochs 20 and 40, respectively. Loss values for both training and

validation datasets steadily decreased, although DenseNet201 outperformed the other networks in overall metrics. To prevent overtraining, an early stopping mechanism was implemented, halting training after two consecutive epochs without improvement in the monitored metric. This strategy ensured efficient training while avoiding overfitting.

Table 4.11. Performance on Testing Set (Experiment 1)

Model Name	Accuracy	Loss	Precision	Recall	F1
VGG16	0.993	0.540	0.986	0.986	0.986
VGG19	0.991	0.503	0.995	0.995	0.995
InceptionV3	0.994	0.825	0.993	0.993	0.993
ResNet152	0.978	0.839	0.981	0.981	0.981
EfficientNetB7	0.989	0.882	0.989	0.989	0.989
DenseNet201	0.998	0.639	0.998	0.998	0.998
CNN_Sequential_32	0.979	1.638	0.979	0.979	0.979

Table 4.12. Performance on Testing Set (Experiment 2)

Model Name	Accuracy	Loss	Precision	Recall	F1
VGG16	0.979	0.885	0.979	0.979	0.979
VGG19	0.984	0.872	0.984	0.984	0.984
InceptionV3	0. 877	1.193	0. 882	0. 877	0. 877
ResNet152	0.979	0.892	0.979	0.979	0.979
DenseNet201	0.988	0.841	0.988	0.988	0.988
CNN_Sequential_64	0.988	1.581	0.988	0.988	0.988

The table 4.11 and 4.12 shows the performance of various fine-tuned models on the testing set, providing metrics such as accuracy, precision, recall, F1 score. Each model's remarks highlight notable

attributes, including trade-offs between accuracy and inference speed. The best overall model is DenseNet201 from Experiment 1. It achieves outstanding metrics, including an accuracy of 0.998, a loss of 0.639, and precision, recall, and F1 scores of 0.998. With its superior accuracy, minimal loss, and excellent balance across all metrics, this model is the strongest selected for deployment.

A popular metric in image classification, the confusion matrix helps evaluate a model's performance by highlighting how well it distinguishes between correctly and incorrectly classified images across different classes. It provides a comprehensive overview by organizing the predictions into a table, where the rows represent actual classes and the columns represent predicted classes. This allows for the calculation of key performance metrics such as accuracy, precision, recall, and F1 score, offering valuable insights into the model's ability to identify and differentiate between classes, as well as its tendencies to make specific types of errors.

The performance of the proposed CNN and DNN models for wheat disease Classification is compared using confusion matrices, as illustrated in the figures during the testing phase. In a test dataset consisting of 441 images, an analysis revealed that VGG16, VGG19, InceptionV3, ResNet152, EfficientNetB7, DenseNet201 and CNN Sequential model provide detailed counts of correctly and incorrectly classified predictions for each class. On the other hand, DenseNet201 presented exceptional results, as it outperformed the other models. Because of its remarkable efficacy in comparison to previous research, our improved DenseNet201 model is a good fit for analyzing wheat leaf health.

The confusion between the yellow test and brown test classes in the confusion matrix could stem from a lack of training data, overlapping data, or dataset imbalance. Other contributing factors may include model limitations such as inadequate feature extraction, overfitting, or underfitting, as well as labeling errors or hyperparameter issues. To reduce this confusion and improve classification performance, strategies such as data augmentation, feature embedding visualization, enhancing dataset quality with more representative samples, and fine-tuning the model architecture can be employed.

The confusion matrices, generated using a smaller dataset of 4,397 images, compare the performance of the model in classifying four categories of wheat leaf diseases. The percentages of correct and incorrect predictions for each category reveal that the transfer learning model outperformed the Sequential model by 1%, achieving an overall accuracy of 99.80% on this dataset. [The color figure is displayed above.]

4.6. Answering Research Questions

Three research questions that are thought to be important for assessing the study's success were developed in the first chapter of this research. The following analysis is done in response to these study questions:

Q1. The most suitable pre-trained models for detecting and classifying wheat leaf diseases through transfer learning depend on dataset size and computational resources. For smaller datasets, VGG16, VGG19, and DenseNet201 are effective due to their simpler architectures and lower risk of overfitting. Medium-sized datasets benefit from InceptionV3 or DenseNet201, which offer a good balance of performance and efficiency. For larger datasets, ResNet152 and EfficientNetB7 excel with advanced feature extraction capabilities and scalability but require more computational power. A lightweight custom CNN may also be suitable for specific dataset characteristics or resource constraints. based on Keras applications, the accuracy achieved by these models is as follows: VGG16 (90.1%), VGG19 (90.0%), InceptionV3 (93.7%), ResNet152 (93.1%), EfficientNetB7 (97.0%), and DenseNet201 (93.6%). These results demonstrate the high effectiveness of these models in images classification.

Q2. The model's performance metrics show near-perfect results on both the training and test datasets. After developing the classification model, it is crucial to assess its effectiveness using common performance metrics such as precision, recall, and F1-score. The model achieved an accuracy of 99.80% on the test set, indicating that nearly all test samples were correctly classified. However, the test loss was 0.634, which is relatively high compared to the accuracy, suggesting potential overfitting or the need for further fine-tuning. The precision is 99.8%, meaning that when the model predicted a positive class, it was correct 99.8% of the time. The recall is 99.8%, indicating that the model correctly identified 99.8% of all actual positive instances. The F1-score, which balances precision and recall, is 99.80%, suggesting consistent performance in both areas. These results indicate that the model is highly effective. Notably, the DenseNet201 model achieved an impressive accuracy of 99.80%, outperforming the pre-trained model, which had already shown promising results in detecting and classifying wheat leaf diseases.

CHAPTER FIVE

SUMMARY, CONCLUSION, CONTRIBUTION AND RECOMMENDATION

5.1 Introduction

The objective of this study is to construct a model using convolutional neural networks to effectively detect and classify wheat crop diseases on leaves. This section outlines the findings and conclusions drawn by the investigator. The researcher's recommendations and insights based on the research results are presented here to assist other scholars interested in this subject area. These suggestions may help future researchers further enhance the study's capabilities and explore its specifics in greater depth.

5.2 Summary

This research paper is structured into five main chapters covering various subtopics. The first chapter provides a concise introduction and background to the study, further outlining its purpose the classification of wheat crop diseases on leaves using a convolutional neural network. Additionally, it addresses the significance of the study, research questions, objectives, hypotheses, scope and limitations, methodology, research contribution, and the organization of the thesis. The second chapter reviews related studies, analyzing and discussing the general structure and experimental evidence presented by other researchers.

The definition of AI, ML, DL, and neural networks, machine learning approaches in relation to crop disease diagnosis, and neural network classification were just a few of the themes that were covered in detail. A review of the body of knowledge regarding the topic was provided in this chapter. The chapter's primary topics are image processing, its applications, image processing methods, CNN, and other machine learning approaches. The research's third chapter covers the supplies and techniques employed to produce the intended study outcome. This section outlines the methodological components utilized to achieve the study's objectives. It encompasses the processes of data collection, image preprocessing, data augmentation techniques, and the exploration of algorithms pertaining to the management of the classified disease.

Chapter four of the research provides an overview of the results, analysis, and explanation of the collected data and findings. This section outlines the methodology employed in conducting the study. The results are presented in straightforward formats, including graphs, tables, charts, and concise explanations to facilitate easy reading and analysis. The final chapter of the research summarizes the

entire thesis by delving into the details of the study. It also offers an overview of the research by highlighting the main topics explored. Key topics addressed in this chapter encompass the summary, conclusion, and recommendations.

5.3 Conclusion

The research aimed is Construct a model using a convolutional neural network to detect and classify wheat crop diseases visible on wheat leaves. In this study, the researcher proposed a deep learning-based model capable of effectively recognizing wheat diseases. The model leverages transfer learning and fine-tuning to enhance its performance, classifying wheat disease images into four categories: three for different wheat diseases and one for healthy wheat. Detailed information is provided about the dataset images used, along with the methods employed for processing and preparing these images for training. The model is evaluated using various metrics, including accuracy, precision, recall, F1-score, and a confusion matrix. The experiment incorporates freezing the pre-trained model for feature extraction while adding a trainable layer to construct an optimal model.

By applying image processing and machine learning techniques, the model effectively identifies and classifies diseases affecting crop foliage, enabling targeted chemical treatments and reducing agricultural production losses. Timely and accurate diagnosis is essential to minimize production losses and control disease spread. Advances in deep learning have opened up new possibilities for precise and effective crop disease Classification. In the image pre-processing stage, features with identical values were removed, especially for classical machine learning techniques, and a Python library was used to obtain the dataset. Traditional methods for detecting and classifying wheat leaf diseases have proven inefficient, underscoring the need for more reliable solutions. This study explored deep learning techniques, particularly transfer learning with pre-trained models such as VGG16, VGG19, InceptionV3, ResNet152, EfficientNetB7, and DenseNet201, along with constructing a custom CNN sequential model. it is important to note that the dataset used may not fully capture the diversity of disease presentations in wheat, which could limit the model's applicability to real-world conditions. Nonetheless, our proposed deep learning model achieves high accuracy in classifying three wheat leaf diseases and healthy leaves, even with a limited dataset. Among the tested models, the DenseNet201 model demonstrated the best performance, achieving an impressive 99.80% accuracy in identifying and classifying wheat leaf diseases

5.4 Contribution of the Study

The constructed model significantly advances wheat leaf disease classification by offering a reliable, efficient tool for identifying and categorizing diseases based on leaf images. By automating disease Classification, it reduces the need for manual inspections, saving time and labor, especially beneficial for large-scale wheat farms. With its precise disease classification, the model enables timely, targeted interventions that help minimize crop losses by effectively addressing specific diseases. Designed to be scalable across different regions enhancing its utility for farmers.

This approach fosters data-driven decision-making, helping farmers optimize resources, reduce chemical usage, and improve overall yield. Beyond practical applications, the model serves as a baseline for future research on crop disease Classification, potentially inspiring similar advancements for other crops or multi-crop systems. By supporting resilient crop management, this model ultimately contributes to increased agricultural productivity, sustainability, and food security.

Leveraging deep-learning models, particularly CNNs, this study achieves high classification accuracy, outperforming traditional image processing methods. The research contributes to integrating artificial intelligence into agriculture to boost productivity, reduce resource waste, and enhance precision farming. It underscores the importance of early detection and classification of wheat diseases to reduce crop losses and maximize yields. By comparing the proposed model with existing ones, the study highlights its superior accuracy, offering researchers a valuable resource for studying, managing, and preventing wheat infections. Furthermore, this model is practical for extensive agricultural applications as it can be scaled to cover large fields and various crop types, reducing the need for costly laboratory testing and expert consultations. Accurate disease classification enables targeted treatments, which minimizes chemical use and promotes sustainable farming practices. This model strengthens disease management techniques and supports food security in wheat-producing regions.

5.5 Recommendations

To mitigate production losses in agriculture, it is essential to focus on addressing crop diseases. Dedicating substantial attention to this sector is crucial for ensuring optimal productivity. More training on disease classification should be given to farmers and agricultural extension staff.

Future improvements to the proposed model require testing and training on a high-dimensional dataset to significantly enhance its accuracy. This study focused on three diseases and one healthy wheat crop, but numerous other disease types and varieties of wheat should be addressed in future research. The emphasis on leaf-based disease Classification is primarily due to its practicality and diagnostic value, as leaves are the most exposed part of the plant and the first to show visible symptoms such as discoloration, spots, or lesions, which are critical indicators of plant health. Leaves are also easier to image and analyze compared to stems, roots, or seeds, which often involve invasive or complex data collection techniques. While stems, roots, and seeds hold diagnostic potential, challenges such as soil interference, seed coating variations, and hidden infections make their study more necessary. Future research should address the lack of justification for focusing on leaves by exploring the diagnostic potential of other plant parts, such as roots and spikes, and validating their effectiveness. Additionally, future studies should adapt the model to diverse environmental settings, including different climate zones, soil types, and weather conditions, by incorporating data sources like soil composition and meteorological information. To further enhance its utility, the model's performance could be integrated with agricultural technologies such as precision irrigation and smart farming systems, ultimately improving productivity and supporting sustainable farming practices.

References

[1] C. A. Meyer, M.; Bacha, N.; Tesfaye, T.; Alemayehu, Y.; Abera, E.; Hundie, B.; Woldeab, G.; Girma, B.; Gemechu, A.; Negash, T.; Mideksa, T.; Smith, J.; Jaleta, M.; Hodson, D.; Gilligan, "Wheat rust epidemics damage Ethiopian wheat production: A decade of field disease surveillance reveals national-scale trends in past outbreaks," *PLoS One*, vol. 16, no. 2 February, 2021, doi: 10.1371/journal.pone.0245697.

[2] D. Dutta, S. Karmakar, A. Hossain, R. Sadhukhan, K. Atta, and S. Pramanick, "Wheat and abiotic stress challenges: An overview," *Abiotic Stress. Wheat Unfolding Challenges*, no. January, pp. 1–13, 2023, doi: 10.1016/B978-0-323-95368-9.00006-0.

[3] A. A. Alatawi, S. M. Alomani, N. I. Alhawiti, and M. Ayaz, "Plant Disease Detection using AI based VGG-16 Model," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, pp. 718–727, 2022, doi: 10.14569/IJACSA.2022.0130484.

[4] E. Byamukama, C. Strunk, C. Tande, and S. Ali, "Wheat Diseases Identification - Pocket Guide," p. 22, 2018.

[5] Y. R. Mehta, "Wheat diseases and their management," *Wheat Dis. Their Manag.*, no. August, pp. 1–256, 2014, doi: 10.1007/978-3-319-06465-9.

[6] S. Patil, "Plant Disease Detection on Wheat Plant using Deep Learning Sayali Patil National College of Ireland Supervisor : Prashanth Nayak".

[7] M. Nigus, H. Shimelis, I. K. Mathew, and S. Abady, "Wheat production in the highlands of Eastern Ethiopia: opportunities, challenges and coping strategies of rust diseases," *Acta Agric. Scand. Sect. B Soil Plant Sci.*, vol. 72, no. 1, pp. 563–575, 2022, doi: 10.1080/09064710.2021.2022186.

[8] N. Shelar, S. Shinde, S. Sawant, S. Dhumal, and K. Fakir, "Plant Disease Detection Using Cnn," *ITM Web Conf.*, vol. 44, p. 03049, 2022, doi: 10.1051/itmconf/20224403049.

[9] K. A. Mottaleb, P. K. Singh, K. Sonder, and G. Kruseman, "Threat of Wheat Blast to South Asia ' s Food Security : An Ex ante Analysis Threat of Wheat Blast to South Asia ' s Food Security : An Ex ante Analysis," *Plosone*, pp. 1–22, 2004.

[10] A. Anteneh and D. Asrat, "Wheat production and marketing in Ethiopia: Review study," *Cogent Food Agric.*, vol. 6, no. 1, 2020, doi: 10.1080/23311932.2020.1778893.

[11] A. Dadrasi *et al.*, "Global insight into understanding wheat yield and production through Agro-Ecological Zoning," *Sci. Rep.*, vol. 13, no. 1, pp. 1–17, 2023, doi: 10.1038/s41598-023-43191-x.

[12] A. Atinafu, M. Lejebo, and A. Alemu, "Adoption of improved wheat production technology in Gorche district, Ethiopia," *Agric. Food Secur.*, vol. 11, no. 1, pp. 1–8, 2022, doi: 10.1186/s40066-021-00343-4.

[13] M. L. Mann and J. M. Warner, "Ethiopian wheat yield and yield gap estimation: A spatially explicit small area integrated data approach," *F. Crop. Res.*, vol. 201, pp. 60–74, 2017, doi: 10.1016/j.fcr.2016.10.014.

[14] W. Abebe, "Wheat Leaf Rust Disease Management: A Review," *Int. J. Nov. Res. Interdiscip. Stud.*, vol. 8, no. 5, pp. 1–14, 1992, [Online]. Available: www.noveltyjournals.com

[15] R. Malladi and R. C. Gogineni, "Identification of Rust Diseases in Wheat Crop Using Image Analysis," *Adv. Appl. Math. Sci.*, vol. 20, no. 12, pp. 3379–3387, 2021.

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, Global Edition, 4ed*.

[17] I. G. and Y. B. and A. Courville, “Deep learning 简介 一 、 什么是 Deep Learning ？,” *Nature*, vol. 29, no. 7553, pp. 1–73, 2016, [Online]. Available: http://deeplearning.net/

[18] A. Dixit and S. Nema, “International Journal of Computer Science and Mobile Computing Wheat Leaf Disease Detection Using Machine Learning Method-A Review,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 7, no. 5, pp. 124–129, 2018, [Online]. Available: www.ijcsmc.com

[19] T. Aboneh, A. Rorissa, R. Srinivasagan, and A. Gemechu, “Computer Vision Framework for Wheat Disease Identification and Classification Using Jetson GPU Infrastructure,” *Technologies*, vol. 9, no. 3, pp. 1–18, 2021, doi: 10.3390/technologies9030047.

[20] E. Haile, “Plant Disease Detection and Classification Using Artificial Neural Network,” *Addis Ababa Univ. Repos.*, vol. 01, p. 83, 2019.

[21] S. Nigam *et al.*, “Deep transfer learning model for disease identification in wheat crop,” *Ecol. Inform.*, vol. 75, no. November 2022, p. 102068, 2023, doi: 10.1016/j.ecoinf.2023.102068.

[22] H. Orchi, M. Sadik, and M. Khaldoun, “On using artificial intelligence and the internet of things for crop disease detection: A contemporary survey,” *Agric.*, vol. 12, no. 1, 2022, doi: 10.3390/agriculture12010009.

[23] S. Dunk, “Wheat Disease Detection,” *Kaggle*, vol. 11, no. 5, pp. 483–491, 2023, [Online]. Available: https://www.kaggle.com/datasets/sinadunk23/behzad-safari-jalal

[24] S. K. Singh, “Artificial Intelligence Techniques for Plant Disease Detection : A Review,” vol. 6, no. 6, pp. 117–121, 2019.

[25] R. El-Sayed, A. Darwish, and A. E. Hassanien, “Wheat Leaf-Disease Detection Using Machine Learning Techniques for Sustainable Food Quality,” *Stud. Comput. Intell.*, vol. 1000, pp. 17–28, 2023, doi: 10.1007/978-3-031-13702-0_2.

[26] M. A. Genaev, E. S. Skolotneva, E. I. Gultyaeva, E. A. Orlova, N. P. Bechtold, and D. A. Afonnikov, “Image-based wheat fungi diseases identification by deep learning,” *Plants*, vol. 10, no. 8, 2021, doi: 10.3390/plants10081500.

[27] C. Amira, G.-K. Dhliwayo, and F.-J. Dube, “Deep Learning Models for Wheat Diseases Detection and Recognition,” *Proc. 9th World Congr. Electr. Eng. Comput. Syst. Sci.*, no. MVML, pp. 1–9, 2023, doi: 10.11159/mvml23.112.

[28] L. Goyal, C. M. Sharma, A. Singh, and P. K. Singh, “Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture,” *Informatics Med. Unlocked*, vol. 25, no. April, p. 100642, 2021, doi: 10.1016/j.imu.2021.100642.

[29] P. Sindhu and G. Indirani, “IoT based Wheat Leaf Disease Classification using Hybridization of Optimized Deep Neural Network and Grey Wolf Optimization Algorithm,” vol. 24, no. 2, pp. 35–53, 2020, [Online]. Available: http://annalsofrscb.ro

[30] N. Networks, S. Nns, and M. Nns, “17.Week7B-Neuralnetwork（2.3）.Pdf”.

[31] B. Mills and J. A. Grant-Jacob, “Lasers that learn: The interface of laser machining and machine learning,” *IET Optoelectron.*, vol. 15, no. 5, pp. 207–224, 2021, doi: 10.1049/ote2.12039.

[32] “CNN | Introduction to Padding - GeeksforGeeks.” Accessed: Dec. 31, 2024. [Online]. Available:

https://www.geeksforgeeks.org/cnn-introduction-to-padding/

[33] "Understanding Convolutional Neural Networks | Machine Learning Archive." Accessed: Dec. 31, 2024. [Online]. Available: https://mlarchive.com/deep-learning/understanding-convolutional-neural-networks/

[34] "5: Example of max pooling and average pooling operations. In this... | Download Scientific Diagram." Accessed: Dec. 31, 2024. [Online]. Available: https://www.researchgate.net/figure/Example-of-max-pooling-and-average-pooling-operations-In-this-example-a-4x4-image-is_fig4_332092821

[35] W. Njima, W. Njima, R. Zayani, M. Terre, and R. Bouallegue, "Deep cnn for indoor localization in iot-sensor systems," *Sensors (Switzerland)*, vol. 19, no. 14, pp. 1–20, 2019, doi: 10.3390/s19143127.

[36] I. Goodfellow, Y. Begio, and A. Courville, "Deep learning," *Nature*, vol. 29, no. 7553, pp. 1–73, 2019.

[37] F. Chollet, "Machine Learning Lecture 06: Deep Feedforward Network," *Mach. Learn.*, vol. 45, no. 13, pp. 40–48, 2017, [Online]. Available: https://books.google.ca/books?id=EoYBngEACAAJ&dq=mitchell+machine+learning+1997&hl=en&sa=X&ved=0ahUKEwiomdqfj8TkAhWGslkKHRCbAtoQ6AEIKjAA

[38] "Convolutional Neural Networks (CNNs) and Layer Types - PyImageSearch." Accessed: Dec. 31, 2024. [Online]. Available: https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/

[39] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, and D. De, *Fundamental concepts of convolutional neural network*, vol. 172, no. January. 2019. doi: 10.1007/978-3-030-32644-9_36.

[40] "Image Processing Techniques, Types, & Applications - GeeksforGeeks." Accessed: Dec. 31, 2024. [Online]. Available: https://www.geeksforgeeks.org/image-processing-techniques-types-applications/

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

[42] M. Sarrouti, A. Ben Abacha, and D. Demner-Fushmen, "Visual question generation from radiology images," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, no. January, pp. 12–18, 2020, doi: 10.18653/v1/2020.alvr-1.3.

[43] S. R. Christian Szegedy1, Wei Liu2, Yangqing Jia1, Pierre Sermanet1 and A. A. 4Magic L. I. Dragomir Anguelov1, Dumitru Erhan1, Vincent Vanhoucke1, Andrew Rabinovich4 1Google Inc. 2University of North Carolina, Chapel Hill 3University of Michigan, "Going Deeper with Convolutions Christian," *J. Chem. Technol. Biotechnol.*, vol. 91, no. 8, pp. 2322–2330, 2016, doi: 10.1002/jctb.4820.

[44] X. Z. S. R. J. ng He Sun, "Deep Residual Learning for Image Recognition," *Eng. Access*, vol. 7, no. 1, pp. 50–60, 2021, doi: 10.14456/mijet.2021.8.

[45] N. L. W. Keijsers, "Neural Networks," *Encycl. Mov. Disord. Three-Volume Set*, pp. V2-257-V2-259, 2010, doi: 10.1016/B978-0-12-374105-9.00493-7.

[46] G. D. Deepak and S. Krishna Bhat, "Optimization of deep neural networks for multiclassification of dental X-rays using transfer learning," *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 12, no. 1, 2024, doi: 10.1080/21681163.2023.2272976.

[47] E. Mocsari and S. S. Stone, "Colostral IgA, IgG, and IgM-IgA fractions as fluorescent antibody for the detection of the coronavirus of transmissible gastroenteritis.," *Am. J. Vet. Res.*, vol. 39, no. 9, pp. 1442–

1446, 1978.

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

[49] “Understanding the Significance of a Confusion Matrix.” Accessed: Dec. 31, 2024. [Online]. Available: https://futuremachinelearning.org/understanding-the-significance-of-a-confusion-matrix/

[50] O. Jouini, M. O.-E. Aoueileyine, K. Sethom, and A. Yazidi, “Wheat Leaf Disease Detection: A Lightweight Approach with Shallow CNN Based Feature Refinement,” *AgriEngineering*, vol. 6, no. 3, pp. 2001–2022, 2024, doi: 10.3390/agriengineering6030117.

[51] T. Hayıt, H. Erbay, F. Varçın, F. Hayıt, and N. Akci, “The classification of wheat yellow rust disease based on a combination of textural and deep features,” *Multimed. Tools Appl.*, vol. 82, no. 30, pp. 47405–47423, 2023, doi: 10.1007/s11042-023-15199-y.

[52] U. R. A, “Wheat Leaf Disease classification using modified ResNet50 Convolutional Neural Network model,” pp. 1–19, 2022.

[53] C. Science, “WHEAT LEAF DISEASE DETECTION AND CLASSIFICATION MODEL USING TRANSFER LEARNING APPROACH,” no. June, 2023.

[54] W. Chen, C. Wellings, X. Chen, Z. Kang, and T. Liu, “Wheat stripe (yellow) rust caused by Puccinia striiformis f. sp. tritici,” *Mol. Plant Pathol.*, vol. 15, no. 5, pp. 433–446, 2014, doi: 10.1111/mpp.12116.

[55] M. Kahsay, “Classification of Wheat Leaf Septoria Disease Using Image Processing and Machine Learning Techniques,” *Addis Ababa Univ. Repos.*, vol. 01, no. June, p. 80, 2019.

[56] and V. S. Nithya, A., “Wheat disease identification using Classification,” *Int. J. Sci. Eng. Res.*, vol. 2, no. 9, pp. 1–5, 2011.

[57] M. H. Hossen, M. Mohibullah, C. S. Muzammel, T. Ahmed, S. Acharjee, and M. B. Panna, “Wheat Diseases Detection and Classification using Convolutional Neural Network (CNN),” *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 11, pp. 719–726, 2022, doi: 10.14569/IJACSA.2022.0131183.

[58] L. Goyal, C. M. Sharma, A. Singh, and P. K. Singh, “Leaf and spike wheat disease detection & classification using an improved deep convolutional architecture,” *Informatics Med. Unlocked*, vol. 25, no. June, p. 100642, 2021, doi: 10.1016/j.imu.2021.100642.

[59] E. Workneh Alamrie, “Convolutional Neural Network for Wheat Leaf Disease Detection,” *BDU institutional Repos.*, vol. Thesis, pp. 1–68, 2021, [Online]. Available: http://ir.bdu.edu.et/handle/123456789/13186

[60] “Normalization vs Standardization - GeeksforGeeks.” Accessed: Jan. 03, 2025. [Online]. Available: https://www.geeksforgeeks.org/normalization-vs-standardization/

[61] S. Jaiswal, “What is Normalization in Machine Learning? A Comprehensive Guide to Data Rescaling | DataCamp,” Software Developer, Programmer, Data Scientist, Business Intelligence Developer. Accessed: Jan. 03, 2025. [Online]. Available: https://www.datacamp.com/tutorial/normalization-in-machine-learning

[62] “Histogram Equalization in Digital Image Processing - GeeksforGeeks.” Accessed: Jan. 10, 2025. [Online]. Available: https://www.geeksforgeeks.org/histogram-equalization-in-digital-image-processing/

[63] “Practical Image and Video Processing Using MATLAB - Oge Marques - Google Books.” Accessed: Jan. 05, 2025. [Online]. Available: https://books.google.com.et/books?hl=en&lr=&id=xzD25QEo8qYC&oi=fnd&pg=PR21&dq=Matlab+Documentation+-+Image+Processing+Toolbox&ots=46Nf6JWbz8&sig=Z4I-aNczu-yU32HBgjHRwMktbNI&redir_esc=y#v=onepage&q&f=false

[64] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Phys. D Nonlinear Phenom.*, vol. 60, no. 1–4, pp. 259–268, Nov. 1992, doi: 10.1016/0167-2789(92)90242-F.

[65] H. Wang, R. Czerminski, and A. C. Jamieson, “Neural Networks and Deep Learning,” *Mach. Age Cust. Insight*, pp. 91–101, 2021, doi: 10.1108/978-1-83909-694-520211010.

[66] A. Parai, S. Deshpande, A. Iyer, A. Kumbhare, and S. Bendale, “Predictive Cancer Detection and Treatment Using Machine Learning and Artificial Intelligence,” *Int. J. Innov. Res. Eng. Manag.*, no. November, pp. 1–10, 2022, doi: 10.55524/ijirem.2022.9.6.1.

[67] “The Best Guide to VGG: Understanding Architecture and Implementation.” Accessed: Jan. 05, 2025. [Online]. Available: https://www.thinkingstack.ai/blog/vision-ai-9/understanding-vgg-models-vgg16-vgg19-and-their-role-in-computer-vision-42

[68] “Top Pre-Trained Models for Image Classification - GeeksforGeeks.” Accessed: Jan. 06, 2025. [Online]. Available: https://www.geeksforgeeks.org/top-pre-trained-models-for-image-classification/?ref=gcse_outind#benefits-of-pretrained-models-for-image-classification

[69] N. A. G. Organization, “Delving into Convolutional Neural Networks (CNNs): Structure, Application, Limitations | by North American Geoscientists Organization | Medium,” North American Geoscientists Organization. Accessed: Jan. 06, 2025. [Online]. Available: https://medium.com/@northamericangeoscientistsorg/delving-into-convolutional-neural-networks-cnns-structure-application-limitations-49d3d95035ce

APPENDIX

Appendix a: source code for model construction using transfer learning

```
import tensorflow as tf
import cv2
import numpy as np
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, VGG19, ResNet152, InceptionV3, EfficientNetB7, DenseNet201
from tensorflow.keras.models import Model
from tensorflow.keras.layers import MaxPooling2D, AveragePooling2D, GlobalAveragePooling2D, Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from tensorflow.keras.regularizers import l2
import pandas as pd
import seaborn as sns
import time
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report, confusion_matrix


# Load the DenseNet201 model, excluding the top (final) layers
base_model = DenseNet201(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

# Unfreeze the last  layers
for layer in base_model.layers[]:
    layer.trainable = True

x = base_model.output
x = MaxPooling2D(pool_size=(2, 2))(x)
x = AveragePooling2D(pool_size=(2, 2))(x)
x = BatchNormalization()(x)
x = Flatten()(x)
x = Dense(512, activation='relu', kernel_regularizer=l2(0.001))(x)
x = Dropout(0.1)(x)  # Adjusted dropout rate
predictions = Dense(train_generator.num_classes, activation='softmax')(x)

# Define the model
model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model
model.compile(optimizer=Adam(learning_rate=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Printing the model summary
model.summary()
```

```
# Calculate training accuracy and loss
training_accuracy = history.history['accuracy'][-1]
training_loss = history.history['loss'][-1]

# Calculate validation accuracy and loss
validation_accuracy = history.history['val_accuracy'][-1]
validation_loss = history.history['val_loss'][-1]

# Print training and validation accuracy and loss
print(f"Training Accuracy: {training_accuracy}")
print(f"Training Loss: {training_loss}")
print(f"Validation Accuracy: {validation_accuracy}")
print(f"Validation Loss: {validation_loss}")

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

# Predict labels for the test data
y_pred = model.predict(test_generator)
y_pred_classes = y_pred.argmax(axis=1)
y_true = test_generator.classes

# Calculate precision, recall, and F1-score
test_precision = precision_score(y_true, y_pred_classes, average='weighted')
test_recall = recall_score(y_true, y_pred_classes, average='weighted')
test_f1 = f1_score(y_true, y_pred_classes, average='weighted')

# Print test metrics
print(f"Test Precision: {test_precision}")
print(f"Test Recall: {test_recall}")
print(f"Test F1-Score: {test_f1}")
print(f"Training Time: {training_time} seconds")

# Plot training & validation accuracy values
plt.figure(figsize=(8, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='lower right')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper right')
```

```
plt.tight_layout()
plt.show()
```

Appendix b: source code for model construction a custom CNN sequential

```
import tensorflow as tf
import cv2
import numpy as np
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, AveragePooling2D,
GlobalAveragePooling2D, Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.layers import GaussianNoise
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
from tensorflow.keras.regularizers import l2
import pandas as pd
import seaborn as sns
import time
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers.schedules import ExponentialDecay
from sklearn.metrics import precision_score, recall_score, f1_score,
classification_report, confusion_matrix

#CONSTRUCTING MODEL

model = Sequential()
# Convolutional Layer 1
model.add(Conv2D(32, (3, 3), activation='relu', kernel_regularizer=l2(0.001),
input_shape=(img_width, img_height, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional Layer 2
model.add(Conv2D(64, (3, 3), activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional Layer 3
model.add(Conv2D(128, (3, 3), activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional Layer 4
model.add(Conv2D(256, (3, 3), activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional Layer 5
model.add(Conv2D(512, (3, 3), activation='relu', kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the output from the convolutional layers
model.add(Flatten())

# Fully Connected Dense Layer
model.add(Dense(512, activation='relu', kernel_regularizer=l2(0.001)))
model.add(Dropout(0.01))

# Output Layer (assuming you have 'num_classes' as the number of output classes)
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()


# Calculate training accuracy and loss
training_accuracy = history.history['accuracy'][-1]
training_loss = history.history['loss'][-1]

# Calculate validation accuracy and loss
validation_accuracy = history.history['val_accuracy'][-1]
validation_loss = history.history['val_loss'][-1]

# Print training and validation accuracy and loss
print(f"Training Accuracy: {training_accuracy}")
print(f"Training Loss: {training_loss}")
print(f"Validation Accuracy: {validation_accuracy}")
print(f"Validation Loss: {validation_loss}")

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")

# Predict labels for the test data
y_pred = model.predict(test_generator)
y_pred_classes = y_pred.argmax(axis=1)
y_true = test_generator.classes


# Calculate precision, recall, and F1-score
test_precision = precision_score(y_true, y_pred_classes, average='weighted')
test_recall = recall_score(y_true, y_pred_classes, average='weighted')
test_f1 = f1_score(y_true, y_pred_classes, average='weighted')

# Print test metrics
print(f"Test Precision: {test_precision}")
print(f"Test Recall: {test_recall}")
```

```
print(f"Test F1-Score: {test_f1}")
print(f"Training Time: {training_time} seconds")
# Plot training & validation accuracy values
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')

plt.tight_layout()
plt.show()
```