DEBRE BERHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT INFORMATION SYSTEMS

IMAGE BASED SORGHUM LEAF DISEASE CLASSIFICATION USING DEEP LEARNING APPROACH

By

DAGIM FIRIDE YIMENU

DEBRE BERHAN, ETHIOPIA

September 3, 2021

DEBRE BERHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF INFORMATION SYSTEMS

IMAGE BASED SORGHUM LEAF DISEASE CLASSIFICATION USING DEEP LEARNING APPROACH

A THESIS SUBMITTED TO THE DEPARTMENT OF INFORMATION SYSTEMS OF DEBRE BERHAN UNIVERSITY IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SYSTEMS

By

DAGIM FIRIDE YIMENU

DEBRE BERHAN ETHIOPIA

July 2020

DEBRE BERHAN UNIVERSITY
COLLEGE OF COMPUTING
DEPARTMENT OF INFORMATION SYSTEMS

IMAGE BASED SORGHUM LEAF DISEASE CLASSIFICATION USING DEEP LEARNING APPROACH

By
DAGIM FIRIDE YIMENU

DECLARATION

I declare that this thesis entitled "**IMAGE BASED SORGHUM LEAF DISEASE CLASSIFICATION USING DEEP LEARNING APPROACH"** was prepared by me, with the guidance of my advisor. The work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted, in whole or in part, for any other degree or professional qualification.

Dagim Firide Yimenu

July 2021

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

Micheal Melese (Ph.D.)

July 2021

DEDICATION

This research work is dedicated to my father Mr. Firide Yimenu, and My mother W/ro Belynesh Bekele (Mom) for their never-ending love, support, motivation, and guidance for me from the very beginning of my birth till now. Dear Father May your soul rest in heaven. You are my hero.

ACKNOWLEDGMENTS

First, I would like to express my profound gratitude to Almighty GOD and HOLY VIRGIN MARY for giving me the moral, psychological, and spiritual strength to accomplish this research work. Next to Him, I would like to express my sincere gratitude to my mom for all the indescribable sacrifices she made so that I have become who I am today. Next, my best gratitude goes to my advisor **Dr. Michael Melese,** for his, invaluable support and guidance starting from the beginning to the completion of this research work. His suggestions, ideas, and comments were invaluable.

I would like to extend my special gratitude to **Dr. Negash Hailu** (Lecture and plant pathologist at, DBU), and also thanks to Deber Berhan Agriculture Researcher center workers for their help in obtaining samples and provide me with information about the leaf diseases.

Finally, I am deeply grateful to my friends, families, and classmates for their endless support and encouragement throughout this research. In addition, I would like to give my gratitude to people who are not mentioned by name but whose effort helped me much all along.

Dagim Firide

____

August 2021

ABSTRACT

Sorghum is a grain crop that is used for human and animal consumption. In areas that are too hot, sorghum is grown and a minimum average temperature of 25 ° C is required to ensure maximum grain production. There are many factors in sorghum production and productivity enhancement, among them crop diseases are the major ones. The early detection of sorghum diseases is one of the main reasons that can reduce the yield production loss, and this requires a huge amount of effort, money, and time. To address these problems, the researcher proposed a deep learning approach for the classification of sorghum diseases based on their leaves. To do so, the design science research methodology was followed. To conduct this study, a total of 4000 images were collected from shewarobit werda kobo villages, North Shewa zone, and prepared. After collecting the necessary images, the researcher applies image preprocessing techniques such as image resizing, normalizing images, and noise removing were performed. And also, data augmentation techniques were performed. In feature extraction, the researcher applies Gabor filter on the raw image for texture feature extraction. It is used for detecting and selecting important features that account for the symptom of the disease. This research work focuses on classifying three types of sorghum leaf diseases: Anthracnose, leaf blight, and rust. Based on this, two Convolutional Neural Network frameworks were proposed namely: train the deep neural network model from the scratch and transfer learning a pre-trained network model. Finally, the developed classifier model has been through accuracy, precision, recall, and F-measure. Experimental result shows that the accuracy obtained from transfer learning model VGG19 and VGG16 achieves an accuracy of 91.5%, and 87.75% respectively. Conversely, the proposed model achieves an accuracy of 94.91%, while after applying Gabor filter the proposed model achieves an accuracy of 96.75%. As a result, training from the scratch model with Gabor was selected for developing an effective and robust model for classifying sorghum leaf disease.

Keywords: Sorghum crop, Deep-learning, Convolutional Neural Network, Transfer learning, training from the scratch, Gabor filter

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ACRONYMS AND ABBREVIATIONS

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CSA	Central Statistical Authority
DSR	Design science research
GDP	Gross domestic product
GLCM	Grey Level Co-occurrence Matrix
HIS	Hue Saturation Intensity Color Space
HSV	Hue, Saturation, Value Color Space
KNN	K-Nearest Neighbors
LUV	Luminance chromaticity values of color images
RGB	Red Green Blue
WEKA	Waikato Environment for Knowledge Analysis
YCbCr	Green (Y), Blue (Cb), Red (Cr) (digial video color space)
DNN	Deep neural networks
ML	Machine learning
DL	Deep learning
SVM	Support vector machine
ILSVRC	ImageNet large-scale visual recognition challenge
VGG	Visual Geometry Group
UCI	University of California, Irvine

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Over the 10,000 years since agriculture began to be developed, people everywhere have discovered the food value of wild plants and animals, and domesticated and bred them. The word Agriculture is the most comprehensive used to indicate the many ways in which crop plants and domestic animals sustain the global human population by providing food and other products [1]. The English word agriculture derives from the Latin Ager (field) and Culture (cultivation) when they combined, Agriculture is the science and art of farming including the work of cultivating the soil, producing crops, planting forest plants (trees), raising livestock, and rearing fisher [2].

Agriculture is diverse and the backbone of the economy for developing countries ranging from relatively small hunting/fishing to forestry and beyond [3]. Nowadays, in the middle- and low-income countries, where the world's farmers are found, agriculture accounts for a much greater share of national income and employment for income and 54% of employment [4]. Besides, this sector is also the backbone of the Ethiopian economy, and it determines the growth of all other sectors and, eventually, of the completely national economy. It accounts for more than 50% of the gross domestic product (GDP), more than 85% of the labor force, and more than 90% of the foreign currency [5], from these crop productions, accounts for 60% of the sector's output on average, while livestock accounts for 27% and other areas account for 13% of the overall agricultural benefit.

Over the past decades, Ethiopian agriculture has achieved remarkable growth.. The use of newer inputs, such as chemical fertilizers and improved seeds, has more than doubled significantly, accounting for part of that growth. However, significant land expansion, increase in labor use and growth in total factor productivity (TFP) are also estimated at 2.3% per year [6].

The crops of Ethiopian are complex, involving significant variation in crops grown across the various regions and agro-ecologies of the country. Smallholders account for 96% of the total area (central statistics agency of Ethiopia 2004-2005 Ec.) cultivated and, for the main crops, produce the key share of total production. The Meher Season, with harvests between September and February, is the key crop season. The center of Ethiopia's agriculture and food economy are five major bowls of cereal (teff, wheat, maize, sorghum, and barley) accounting for around three-quarters of the total cultivated area and 29% of agricultural GDP in 2005/06 (14 % of total GDP) [7].

Sorghum [Sorghum bicolor (L.) Moench] is classified under the family Poaceae (grass family), tribe and ropogoneae, genus bicolor, species bicolor [8]. Sorghum is a grain crop that is used for human and animal consumption. In areas that are too hot, sorghum is grown and a minimum average temperature of 25 ° C is required to ensure maximum grain production. It is one of the cereals currently grown with the highest drought tolerance due to the morphological features of the community. It rolls its leaves during the drought to decrease the water loss due to perspiration. If the drought goes on, instead of dying, it becomes dormant [9].

Sorghum is the third-largest cereal in Ethiopia, after teff and maize in terms of area coverage and total production. According to, (CSA (Central Statistical Authority, 2018), accounts for 18.53% of the total area allocated to cereals, and 19.3% of the area covered by cereals. After Sudan, Ethiopia is also the second-largest sorghum producer in eastern and southern Africa [10]. However, sorghum crop production is affected by various biotic and abiotic constraints. The biotic stresses, diseases caused by different fungal pathogens play a significant role in restricting its development. Leaf blight, downy mildew, rust, anthracnose, sorghum smuts, loose smut, and long smuts are the diseases that affect sorghum crop production [11]. These diseases are major in Ethiopia that is now considered one of the most damaging sorghum crops in most of the country's where sorghum is grown. These diseases affect the various sections and stages of the crop, decreasing its production significantly [11]. Therefore, Identifying and recognizing these diseases through different methods including manual and computer-based systems are critical.

Author in [12] stated that in most cases, plant diseases are seen as spots on their leaves that are more visible to the human eye. However, some diseases do not appear on the leaves and others are appearing in later stages after they already caused severe effects on the plants. In such a case [13] [14] suggested that use a computerized system that would able to detect the diseases timely and accurately through computer algorithms and analytical tools.

With the evolution of soft computing and the advancement of image processing techniques, the very way we are living today is radically changed. Soft computing is a collection of artificial intelligence-based computational techniques including the fundamentals of the neural network, fuzzy logic, and genetic algorithms. Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to consider like humans and imitate their actions from the simplest to those that are even more complex. The goals of artificial intelligence include learning, reasoning, and perception. Image processing is the method of manipulating an image to either enhance the quality or extract relevant information from it [15].

AI Image Processing Services combine advanced algorithmic technology with machine learning and computer vision to process large volumes of pictures easily and quickly. When applied to image processing, artificial intelligence (AI) can power recognition and authentication functionality for ensuring security in public places, detecting and recognizing objects and patterns in images and videos, and so on [16].

AI is making a huge impact in all domains of the industry. Every industry looking to automate certain jobs through the use of intelligent machinery. Agriculture and farming are one of the oldest and most important professions in the world. It plays an important role in the economic sector. Nowadays, this industry is turning to Artificial Intelligence technologies to help healthier crops, control pests, monitor soil, and growing conditions, organize data for farmers, help with the workload, and improve a wide range of agriculture-related tasks in the entire food supply chain [17].

Modernizing conventional practices is an urgent need for developing countries where a large percentage of the population still relies on agriculture. Furthermore, the manual surveillance of the disease does not produce a satisfactory result. Therefore, many new techniques for early detection of plant diseases, image processing has now become a possible method by which not only the disease can be identified early and accurately, but it can also be quantified successfully [18].

Therefore, Artificial Intelligence in agriculture is not only helping the farmers to identify the diseases at early stages but also automate their farming and shifts to precise cultivation for higher and better-quality crop yield. Companies involved in improving Artificial Intelligence-based products or services like training data for agriculture, drone, and automated machine making will get technological advancement in the future will provide more useful applications to this sector helping the world deal with food production issues for the growing population. The integration of different technology from AI enables to design of a system with high performance [15].

1.2 Motivation

Why the researcher was inspired to conduct this study is, agriculture sector contributes to the daily needs of the Ethiopian population and is also the backbone of the country's economy. However, this sector suffers poor cultivation practices and frequent drought. Among this sector sorghum is one of the most cereal crops used for human and animal consumption. In this sector, farmers are the main elements, but they are facing complications occurring due to plant pests and diseases. Most farmers in Ethiopia do not have access to information about pests and disease types and also their identification mechanism [19]. They have to depend on plant pathologists to resolve these problems. Examining the plant affected by pests and disease through a plant pathologist manually is an expensive and time-consuming process. Plants affected by a disease if not diagnosed within time will result in poor quality and low quantity of production. On the other hand, one way to reduce this issue is early detection of the diseases through the current technology, deep convolutional neural networks, which achieve better results in detecting diseases that consist of related or closed features.

Deep learning simplifies the complex feature extraction problem and hence will be used for image recognition. Nowadays, by using deep convolutional neural networks, we can diagnose and detect different leaf diseases which have very similar features. Thus, the researcher motivated to develop a model that uses a computer vision to recognize and classifies sorghum leaf disease in its early stage will help the sorghum growers to use the biological pest control mechanism effectively.

1.3 Statement of the Problem

Agriculture is a key for the development of the country while it leads to growth and played an important role in reducing poverty and transforming the economies of many countries [3]. Hence this sector has recorded remarkable rapid growth in the last decade in Ethiopia. The sector has five major kinds of cereals namely: teff, wheat, maize, sorghum, and barley are the core agriculture and food economy of Ethiopia. These cereals comprise 78.23% (8.8 million ha) of the field crops of which sorghum accounts for 14.41%, this crop is grown in almost all regions occupying an estimated total land area of 1.6 million ha [20].

Currently, Sorghum production has significantly increased in recent years, from 1.7 million tons/ha in 2004/05 to nearly 4.0 million in 2010/11. Whereas the national average sorghum productivity in Ethiopia is 2.1 tons/ha (CSA, 2012) which is far below the global average of 3.2 tons/ha (FAO, 2005), this is because due to the challenge of different factors. These factors are a hindrance to sorghum production and productivity enhancement. Among these factors, crop diseases are the major ones. These diseases can affect the quality and quantity of sorghum production and economic losses in agriculture [20].

Researchers in [20] [21], stated that these diseases are caused by pests, insects, and pathogens, there are several sorghum leaf diseases, among them, Anthracnose, Leaf blight, and rust are the major ones and mostly occurred in a too hot area which is sorghum grown. Sorghum anthracnose, caused by *Colletotrichum sublineolum* is one of the major widespread, mostly the disease observed when the weather is hot and humid. In Ethiopia, 93.7% of sorghum fields are infected by this disease, and 26% of crop yield losses in the susceptible cultivar. In addition, sorghum leaf blight disease also affects 84.8% of sorghum fields and 35% of sorghum crop yield losses [22] [23].

Therefore, having an early detection model will help the farmers and extension workers to have timely treatment and most importantly to prevent the diseases from spreading into the whole farm which helps to alleviate major damages in crop production losses. Accordingly, developing a classifier model that can be able to accept a given sorghum leaf and classify the disease based on the characteristics of the leaves through a system becomes very important. Towards, solving the stated problem there some works previously have been done.

Soni et al., [24], developed a classifier model for detecting and classifying sorghum leaves diseases. The classifier model was developed through deep learning approaches with a pre-trained model know as Alex-Net. Thus, the developed classifier model is able to recognize leaf diseases as leaf blight, Sooty Stripe, and Rough Leaf Spot. However, the developed model didn’t include one of the serious sorghum leaf diseases such as rust and anthracnose. Additionally, the study has used only one pre-trained model that means others currently developed pre-trained models may achieve the best accuracy for classifying sorghum leaf diseases.

Rahman et al., [25] have proposed a comparative analysis of the machine learning approach for plant disease identification and classification. To accomplish the study, sorghum, cabbage, and citrus crops are selected. The main objective of the study was comparative analysis for identifying sufficient features to classify the selected crop leaf diseases as healthy and non-healthy. Towards identifying the best feature, the researcher has selected color and statical features, and test them through traditional machine learning algorithms such as RF, SVM, and ANN. Conversely, the study was not developed a classifier model rather than identifying the best features.

Ahmed et al., [26], the researchers have proposed a model that detects and classifies rice leaf diseases. To develop the model WEKA tool and KNN, decision tree, Naïve Bayes, and logistic regression classification algorithms are implemented. Thus, the developed classifier model can recognize leaf diseases as Bacterial leaf blight, Brown spot, and Leaf smut. However, the researcher didn’t include major sorghum leaf diseases such as anthracnose and rust. In addition, which diseases are correctly detected and classified by the proposed model is not mentioned.

As stated in [6], the traditional approach (i.e., visual examination) for the classification of these diseases is through eye observation by experts, this examination method is prone to error as the diagnosis is based on their experience of the farmer and extension worker. The method also takes a great deal of effort and time, is tedious, susceptible to error, and also it is subjective to identify sorghum crop disease. The decision-making capability of a human inspector also depends on physical conditions such as workload, pressure, tiredness, and eyesight besides the level of expertise.

Studies in [24] [25] [26], tried to attempt the problem of plant disease detection and classification using machine learning and datamining. However, the first study didn't include major sorghum leaf diseases such as anthracnose and rust, and the researcher used a single classifier namely Alex-Net. According to [27] this classifier is very less and hence it struggles to learn features from images and also it takes more time to achieve higher accuracy results compared to future models. In the second studies the researchers didn't develop a classifier model rather they tried to identify which are the best features for recognizing the diseases. In addition, the researchers extract features manually, whereas the Author in [17], stated that extracting features manually can affect the performance of the classifier model because the generated features are not high-level features that describe the disease. Moreover, this learning algorithm (i.e., machine learning) has suffered some limitations like occlusion and deformation limitation, which means that if the targeted images were covered by another object and/or the targeted images captured in different positions this learning algorithm didn't recognize it as well.

In this study, the deep-learning approach has been chosen, [28] this approach is the sub-field of machine learning and has the advantage of the ability to generate new features from the limited available training datasets and it reduces the time required for feature engineering. Besides, the approach has several algorithms, the one and the most popular is CNN, [29] this algorithm can detect important features without stating them explicitly, it has computationally efficient, and also the algorithm has captures the three-dimensional features from an image. The algorithm presenting an operative class of models for a better understanding of contents present in an image, so resulting in better image detection and recognition. CNN's are efficiently and effectively used in many pattern and image recognition applications [30]. Besides, the researcher extracts the texture features from an image using GLCM and Gabor filtering rather than color or shape feature because for this study texture feature can differentiate the different categories of the diseases whereas there are disease that change the color of one sorghum leaf into other color sorghum and also the diseases can't affect the size of the leaf. Since, the researcher used two CNN architectures namely: training from scratch and transfer learning. In training from the scratch model the researcher setting different hyper-parameter values and trained the network from scratch. In transfer learning the research selects two pre-trained models namely VGG19 and VGG16. These pre-trained networks are more effective for image classification and have high classification accuracy than other pre-trained models [31].

Therefore, accurate detection and classification of sorghum leaf disease using deep learning are very crucial to control the diseases as well as to increasing agricultural productivity. The latest improvements in deep learning have increased its capability to solve complex visual recognition tasks. Thus, there is a need to detect and classify sorghum leaf diseases using deep learning techniques. Therefore, the purpose of this research work is to build a model for sorghum leaf disease using a deep learning approach which will be crucial for the sorghum producer and agricultural extension worker to draw up plans and policies for enhancing sorghum crop production. Besides, this study attempts to obtain answers for the following main research questions:

- Which training methods are more appropriate for classifying sorghum leaf diseases?
- Which features extraction techniques is the best for classifying sorghum leaf diseases?
- To what extent does the developed model correctly classify sorghum leaf diseases?

1.4 Objectives

1.4.1 General Objective

The general objective of this research is to design and develop a model that detects and classifies sorghum leaf diseases using a deep convolutional neural network.

1.4.2 Specific Objectives

To achieve the general objectives, the following specific objectives are formulated.

- To conduct a literature review of previous studies to understand the domain.
- To identify and collect the required datasets for the Sorghum disease.
- To make the data cleaned and smooth using pre-processing techniques.
- To apply an appropriate features technique.
- To identify the right training.

- To design the architecture of sorghum leaf diseases classifier.
- To develop the classifier model.
- To evaluate the performance of the developed model.
- To develop a prototype of the proposed system.
- To forward appropriate recommendations for future research based on the finding.

1.5 Scope and Limitation of the Study

As detection and classification of leaf disease is a very vital area of study to deal with, it is important to make some sort of boundary of task coverage for having better outcomes. The study is conducted based on the data obtained from Shewa Robit worda kobo village, North Shewa zone. Therefore, pre-processing techniques were implemented to make the data smooth and cleaned. The main objective of this study is to develop sorghum leaves disease classifier model. To do so, a deep learning-based approach with training from scratch (i.e., setting different parameters and hyper-parameter from the beginning of building the model), and transfer learning (i.e., reusing previously trained model). Hence, these methods automatically extracting the representative features from the images and identifying the features of image elements that are given to a convolutional neural network classifier. In addition, the researcher extracts texture features using Gabor filtering. Finally, the researcher has developed a graphical user interface for the developed model to be suitable and applicable for the end-users. Besides, the following are the major limitation of this study.

- As a result, this research work here is limited to only classifying an input sorghum leaf image into either of four different classes i.e., Healthy (uninfected leaf), Rust, Anthracnose, and leaf blight, neither stem nor head and other parts of the crop.
- This study was limited to developing a classifier model for sorghum leaf disease using two pre-trained models, namely VGG-16 and VGG19. However, other pre-trained models might show better accuracy. This was happened due to time and resource constraints.
- After the identification and detection of the disease, recommending the appropriate treatment for the identified disease, classifying the sorghum leaves damaged due to

nutrient deficiencies, and estimating the severity of the detected disease is beyond the scope of this research work.

1.6 Significance of The Study

It is a viable fact that artificial intelligence has come to make the life of a human being easier and there is no way to eliminate its use. It is therefore left to the users to utilize it appropriately to improve their living conditions. At the end of the study, this study will help pathologists, agricultural extension workers, farmers, and the government to understand the significance of artificial intelligence in general. In addition, the developed classifier model will have the following significance.

- It will help pathologists and agricultural workers to easily classify the disease at the beginning stage. And also, the developed model provides reliable, less time-consuming, coast effective, and effective way of classifying sorghum diseases from their leaves, and also it avoids subjectivity. In addition, Speed up the detection and diagnosis process and reduce the time and effort of farmers and pathologist. Hence, it improves the effectiveness of disease control mechanisms.
- For the government, the annual production of sorghum crops will increase, hence the crop will not only be for local consumption rather the opportunities for exporting the crop this can increase foreign currency.
- The study also will help the farmers to increase the sorghum yield production/productivity of the crops, which means that the income also will be increased. In addition, it will reduce the cost of production that brings huge losses due to the excessive use of fungicides on their plants.
- Finally, this study would serve as an input for other researchers who interesting to conduct further studies into the problems related to image processing and plant disease.

1.7 Methodology

The following methods and procedures are incorporated in the advancement of the study to achieve the objective of this research work.

1.7.1 Research Design

Design science is seen as a research activity that builds new or invents, innovates artifacts for problem-solving or improvement attainment such new innovative artifact creates a new reality, rather than the existing reality been explain [32]. For the completion of this study, design science research methodology has been followed.

1.7.2 Literature Review

To accomplish the objective of the study and to gain a deeper understanding of the problem, different works of literature such as thesis, articles and conference papers, and books that are related to the research topic have been reviewed and communicated with domain experts.

1.7.3 Data Collection

The required data is collected and prepared by the researcher from the Amhara region with the help of the domain expert. The data is from the Shewarobit Woreda Kobo village, North Shewa zone. The digital camera has used to collect the required images, and 4000 images were collected.

1.7.4 Tools and Implementation

For this study, the experiments and related analysis processes are done on a computer with Intel® Core^{TM} i5-5200U CPU @ 2.70 GHz 2.20GHz Speed, 8.00 GB RAM, and 1 TB hard disk space with 64-bit Microsoft Windows 10 operating system. For software, python 3.9 programming language, the main reason for selection of this language is mainly, it’s simple to use, efficient, code readability and flexible [33]. Python utilizes different libraries like, OpenCV (Open-Source Computer Vision) is a library of python programming functions mainly designed to solve computer vision problems, NumPy is the fundamental package for scientific computing with Python that contains a powerful N-dimensional array object this library is needed to treat images as matrices, and Scikit-learn is a free machine learning library for Python. It features various classification, regression and clustering algorithms and is designed to interoperate with NumPy and SciPy.

Keras is a modular neural network library written in Python capable of running on top of either TensorFlow (TF) or Theano. The library was conceived to let the users start experimenting as fast as they could, being able to go from idea to result with the least possible delay. The reason TF was selected as a backend is that both TF and Keras were optimized to perform deep learning tasks. Both systems are implemented in Python which allows the user to work with them compactly without having to use multiple files. With Keras, the model has to be first defined, either a Sequential model or a Graph model. In a Sequential model, the layers are stacked and the output from a layer feeds the input of the next layer until it reaches the output layer. On the other hand, the Graph model allows the users to get the output from the desired layer and feed that output to the desired layer, permitting the generation of multiple output networks or getting the output in an intermediate layer of the model [34].

For this researcher work, the Sequential model is selected. Finally, the researcher has used the flask micro-framework to develop a final prototype. Flask is a micro web framework written in Python. It does not require particular tools or libraries and it has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

1.7.5 Evaluation method

The researcher has used performance evaluation metrics to measure the developed classifier model performance for sorghum leaf diseases before applying final deployment in real-world use, it has to be tested because it is important to evaluate the model, and review the steps executed to construct the model and to be certain it properly achieves the business objectives [35]. Therefore, for this study, the developed classifier model is evaluated through different performance evaluation metrics such as accuracy, precision, recall, and F-measure.

1.8 Organization of the Thesis

This research work is organized into five chapters. The first chapter mainly Dealt with background of the study, motivation, statement of the problem objective of the study, significance of the study, scope, and limitation, and research methodology of the study.

Chapter two deals with the literature review and related work. The literature review section tries to give an introduction to sorghum production in Ethiopia, sorghum leaf disease, computer vision, digital image processing, machine learning, and deep learning concepts in general.

Under the related work section, studies previously done by different authors which is relevant to this study and more related to leaf disease detection using image processing and machine learning algorithms are included, and finally an executive summary of the related work.

Chapter three deals with the selected methodology for conducting this study and the process model of the methodology. Chapter four discuss about the proposed framework development, and the process flow in detail of this study. It discloses the technique and the algorithms that the researcher used to accomplish this study.

Chapter five provides research findings, analysis, and discussion obtained through the proposed and pre-trained model. The final chapter, deals with conclusions, the contribution of the study, and recommendations for future work that will need to be done.

CHAPTER TWO

LITERATURE REVIEW AND RELATED WORK

2.1. Introduction

This chapter is concerned with the overview of sorghum in section 2.2. Next to this, total sorghum production and major sorghum leaf disease are covered in sections 2.3 and 2.4 respectively. Then section 2.5 tries to give details about what a computer vision is all about. Section 2.6 raises a detailed discussion of what digital image processing is, following this, what are the basic digital image processing steps are raised in section 2.7 in line with this section 2.8 discusses machine learning. Then Section 2.9 tries to list out some deep learning approaches. In addition, in this chapter, the different hyper-parameters, the architecture of CNN, evaluation techniques, related works, and executive summaries are discussed.

2.2. Overview of Sorghum

Sorghum is also known as *Sorghum bicolor* is one of the most cultivated tropical cereal grass. Originated from North Africa, possibly in the Nile or Ethiopian regions as recently as 1000 BC [36]. The cultivation of sorghum played a crucial role in the spread of the Bantu (black) group of people across Sub-saharan Africa. Nowadays, sorghum is cultivated across the world in warmer climatic areas. It is recognized as a significant crop throughout the arid tropical and sub-tropical regions of Africa, Asia, and Central America. This crop is the world's fifth major important cereal grain, after wheat, maize, rice, and barley [21].

Sorghum is still mainly a survival food crop and also the main food for millions of people in the semiarid regions of Africa and Asia where it is used to make food products such as 'tortillas', 'bread', 'cakes', 'noodles', 'couscous', 'beer', and 'porridgee' [37]. In Ethiopia, this crop is used for making 'injera', 'kitta', 'kollo', and locally made beverages (local drinks such as Tela and Areke) [20]. Hence, the crop is mostly cultivated in drier areas, particularly on shallow and heavy clay soils. It is a broadly adapted species capable of growing in semiarid, subtropical, tropical, and temperate climates. It has a widespread root system and the ability to become dormant during water stress make refined sorghum drought-resistant typically requiring only one-half to two-thirds the amount of rainfall [38]. Figure 2.1 shows that sorghum crop varieties.

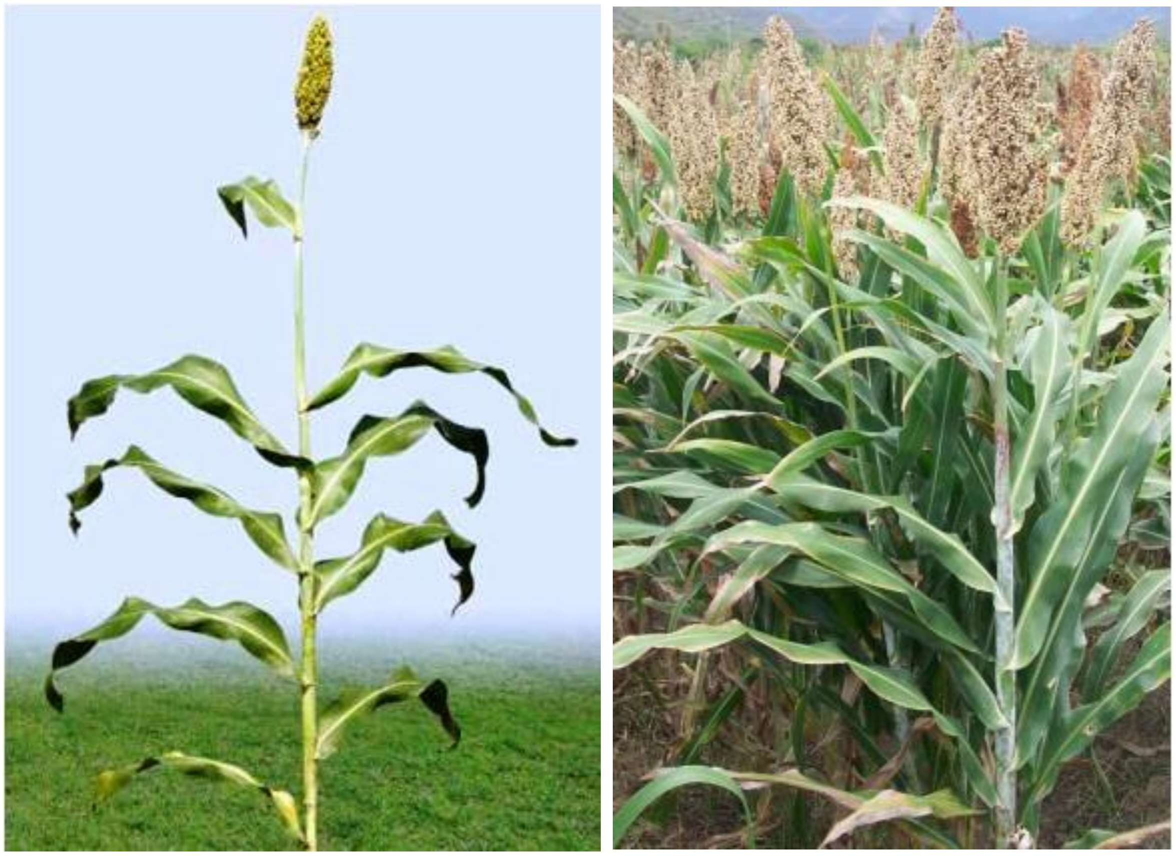


Figure 2. 1 Sample image of sorghum crop adopted from [38]

As shown in Fig 2.1, Food-grade sorghum varieties are defined by several key traits including white pericarp color, thin mesocarp, normal endosperm type, low tannin content, and tan-plant necrotic lesion color.

2.3. Sorghum in Ethiopia

Sorghum is cultivated on 44.4 million hectares worldwide with an average of 1314 kg/ha while the average from developed and developing countries is 1127 kg/ha. Ethiopia is the third-largest producer of sorghum in Africa after Nigeria and Sudan with a contribution of about 12% of yearly production [20] [39]. According to [20] sorghum production in Ethiopia is improved by both land expansion and yield improvement, which means that the yield increased from an average of 1.4 tonnes/ha to 2.1 in 2004/05 and 2010/11 respectively, meanwhile, land expansion increased by 50 percent, while the area under sorghum production increased by 51 percent (from 1.2 million ha in 2004/05 to 1.9 million ha in 2011. The crop is grown in almost all Ethiopian regions occupying an estimated total land area of 1.6 million (ha).

Sorghum is produced in different parts of Ethiopia including, East and West Hararge, Amhara, Tigray, Southern Nations Nationalities and Peoples' Region (SNNPR), and others, are contributing 39.9%, 38.7%, 11.8%, 4.4%, and 5.2% of the national production, respectively [40]. From the total production of sorghum crop, 11.5% of the crop is wholesaled, 74.4% being consumed at the local level, 9.2% retained as seed, 1.2% of the crop used as payment of wages in kind, and 0.9% used as animal feed [41]. With the diverse nature of the farming systems and climatic conditions under which sorghum is grown, the production of sorghum in Ethiopia is adversely affected by different biotic and abiotic diseases. The most constraint sorghum diseases are such as (anthracnose, leaf blight, gray leaf spot, Rust, and downy mildew). These diseases have become increasingly constrained to sorghum production [42] [43].

2.4. Sorghum diseases

Sorghum leaf disease reduces crop production and affects the agricultural economy, the diseases are caused by three types of attacks including viral, bacterial, or fungal. During the past decade, sorghum improvement has been important in eastern Africa [44]. Several diseases affect grain sorghum grown such as Anthracnose, Rust, Leaf blight, sooty stripe, sooty stripe, Downy Mildew, and Smut are the diseases with the greatest potential to yield losses, widespread and economically impactful [39]. These diseases are primarily caused by fungi and are incline to be problematic just before or during the reproductive stages of development.

2.4.1 Anthracnose sorghum leaf disease

Sorghum anthracnose, caused by *Colletotrichum sublineolum* is one of the major diseases throughout the world wherever sorghum is grown and is considered one of the most damaging leaf diseases of grain sorghum production. Mostly observed when the weather is hot and humid and on susceptible hybrids can be severe and cause tremendous yield losses [45] [46]. This occurs in all sorghum-growing regions of the globe. Sorghum anthracnose happens worldwide but is more typically observed in tropical and subtropical regions where frequent rainfalls and high humidity contribute to the development and spread of the disease. This disease can also cause of reduction in kernel number and size. In Ethiopia, this disease can cause yield loss that ranges from 26-35% in vulnerable cultivars [47] [43].

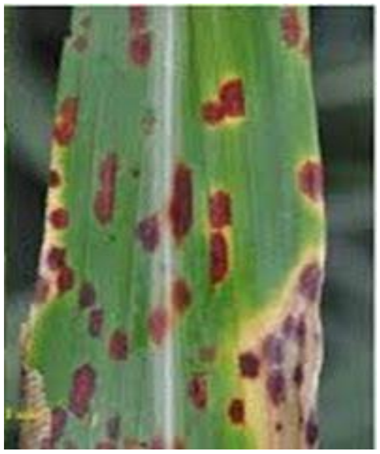


Figure 2. 2 Anthracnose Sorghum Leaf diseases

As shown in Fig 2.2, Fungus is the cause of Anthracnose (*Colletotrichum graminicolum*) leaf diseases. This disease appears as small red-colored spots on both surfaces of the leaf. The center of the spot is white encircled by a red, purple, or brown margin.

2.4.2 Leaf blight sorghum leaf disease

Leaf blight disease has been found or observed in all of the major sorghum-growing areas of the world this disease is caused by the fungus called Exserohilum turcicum [48]. In our content, leaf blight is considered a widespread and major disease of productive cultivars. this disease reduces or delays plant growth and development and as a consequence reduces the yield of both grain and fodder. Leaf blight occurs before flowering, leading to yield loss reaching up to 50% [46] [47].

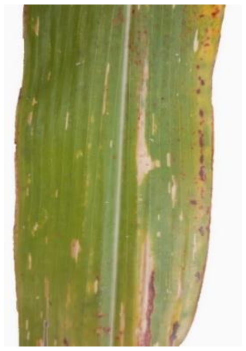


Figure 2. 3 Sorghum leaf blight disease

As illustrated in figure 2.3, sorghum Leaf blight is caused by the fungus. The disease is most readily identified by large cigar-shaped lesions on the leaf with purple margins. The straw-colored center becomes darker during sporulation, and the lesions can be several centimeters long and wide. Many lesions may develop and combine on the leaves, destroying large areas of leaf tissue, giving the crop a burnt appearance [49].

2.4.3 Rust sorghum leaf disease

Sorghum rust is caused by the obligate fungal pathogen, Puccinia purpurea Cooke. This disease is widely distributed and occurs in almost all sorghum-growing areas of the world, particularly East Africa, India, and South and Central America [46]. Loss of grain yield from leaf rust can be up to 50 % in susceptible varieties, mostly due to reduced seed weight. In addition, severe rust infection also reduces the sugar content of the juice in sweet sorghum. It has also been reported that rust is conducive to the occurrence of other diseases, such as anthracnose [49] [50]. This fungus affects the crop at all stages of growth. The first symptoms are small flecks on the lower leaves (purple, tan, or red depending upon the cultivar). Pustules (uredosori) appear on both surfaces of the leaf as purplish spots which rupture to release reddish powdery masses of uredospores. An example of sorghum rust leaf diseases is shown in Figure 2.4.

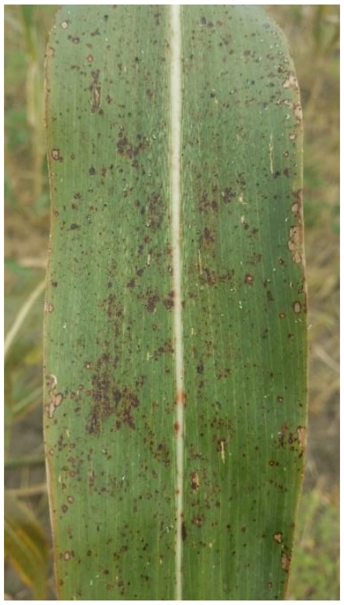


Figure 2. 4 Sorghum rust leaf disease

As shown in Figure 2.4, Rust is caused by the obligate fungal pathogen. The fungus affects the crop at all stages of growth. The first symptoms are small flecks on the lower leaves (purple, tan, or red depending upon the cultivar). Pustules appear on both surfaces of the leaf as purplish spots which rupture to release reddish powdery masses of pustules [42].

Present agriculture technologies are mostly diverted to machine learning (ML) algorithms, the algorithm enables the farmer to enhance crop selection and crop yield prediction, crop disease prediction, weather forecasting, minimum support price, and smart irrigation system [51]. Likewise, computer vision is applicable in agricultural sectors to recognize and visualize a leaf image.

2.5. Computer vision

In the past few decades, computer vision inspection systems have become important tools in agricultural operations, and their use has significantly increased. The application of computer vision technology in different areas can improve the efficiency of production to a certain extent [52]. This technology can also continuously improve the efficiency and accuracy of production in the image processing process, achieve non-destructive construction, and promote the continuous development of various fields such as industrial, agricultural, medical, electric power automation systems [53].

Expert and intelligent systems based on computer vision algorithms are becoming a common part of agricultural production management, and computer vision-based agricultural automation technology is increasingly used in agriculture to increase productivity and efficiency [54]. The ability of computer vision technology has been greatly improved, and the improvements in resource efficiency have provided many suggestions and insights for decision support and practices for farmers, ensuring the efficiency of agricultural production [55] [51]. It facilitates the development of precision agriculture by observing, measuring, and responding to inter and intra-field variability in crops. There are numerous applications of computer vision technology in agricultural automation, such as yield estimation, disease detection, weeds identification, and quality control [56].

Compared with manual operations, the real-time monitoring of crop growth by applying computer vision technology can detect the subtle changes in crops due to malnutrition much earlier than human monitoring and can provide a reliable and accurate basis for timely regulation [56]. Therefore, computer vision technology will be increasingly applied to the field of agricultural automation and will steadily promote the development of agriculture to the era of intelligent agriculture [52]. It has a combination of concepts, techniques, and ideas from digital image processing, pattern recognition, artificial intelligence, and computer graphics. Nowadays, most of the tasks in computer vision are related to the process of obtaining information on events or descriptions, from digital images to automate tasks that the human visual system can do [57].

2.6. Digital image processing

For mathematical analysis, an image may be defined as a two dimensional function $f(x, y)$ where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or gray level of the image at that point [58]. Whenx, y, and the amplitude values of f are all finite, discrete quantities, we call the image a digital image. Digital image is a matrix representation of a two-dimensional image using a finite number of point cell elements, usually referred to as pixels (picture elements), each pixel is represented by numerical values [59].

In imaging science, Image Processing is the processing of images using mathematical operations by using any form of signal processing for which the input is an image, a series of images, or a video, such as a photograph or a video frame [60]. Accordingly, image processing is categorized into two types namely Analog and Digital Image Processing. Analog image processing can be used for hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques [61]. Digital image processing refers to processing digital images utilizing a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, and pixels [58]. However, one useful model is to consider three types of computerized processes, these are: low-level, intermediate-level (middle-level), and high-level processing. The Basic Digital Image Processing Steps will discuss in the following subsection.

There is a different step in digital image processing. For the complete implementation of an application using digital image processing, various image processing, and neural network approaches can be applied for the identification of the leaf diseases on the plant. Fig 2.5 shows the fundamental steps involved in the plant leaf disease identification process.

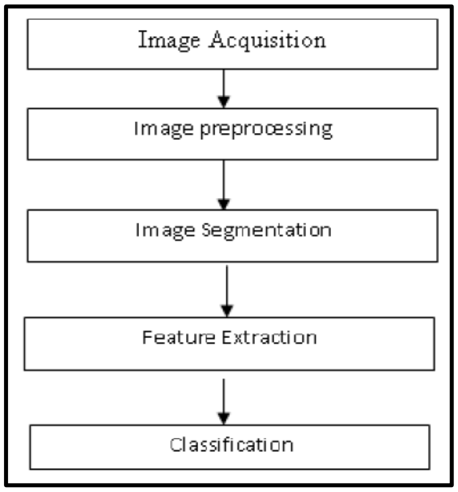


Figure 2. 5 Fundamental steps of digital image processing adopted from *[62]*

As we have seen in figure 2.5, in digital image processing collecting the necessary images is the first step which is called image acquisition. Then the next step preprocesses the collected images so this phase is smoothing the collected images to get high-quality images through various methods like resizing, color conversion, noise removing, and so on. Then image segmentation comes, in this phase, the images divide into several regions or objects. The segmentation is used to increase the chance of representation of an image. Following this, the next step is extracting the relevant information from an image, different types of image features are used to represent an image for object recognition and/or identification like color, shape, and texture of an image. Finally, the last step is to predict the class labels for given images.

A. Image acquisition

Image acquisition is the first step in any image processing work, in which an original input image is acquired from the initial source. Image can be acquired differently, either capturing an image from the actual environment or can browsing already existed image files from an electronic source [60] Whereas, the acquired images were unprocessed.

B. Image preprocessing

Pre-processing of an image involves the removal of distortion, which has a major effect on accuracy for identification and classification. The major task of this phase is to improve the image quality by removing unnecessary distortions or enhancing the image quality for future processing. The techniques have various methods like image resizing, image enhancement, image restoration, removing noise, and soon [63].

C. Image segmentation

After preprocessing an image the next step is image segmentation. In this phase, an image divides into several objects or regions. The segmented image features are robust for the identification and classification of diseases. Segmentation is used to improve the chance of representation of an image, by analyzing the image data and extract useful information for further processing [62]. The image segmentation can be done in two ways based on similarities and discontinuities. In similarities, the images are partitioned based on some specific predefined criteria. But in discontinuities, the images are partitioned based on the sudden changes in the intensity of values [48]. The popular techniques used for image segmentation are shown in Figure 2.6, the Threshold method, edge detection-based techniques, region-based techniques, clustering-based techniques, watershed-based techniques, partial differential equation-based, and artificial neural network-based techniques, etc.

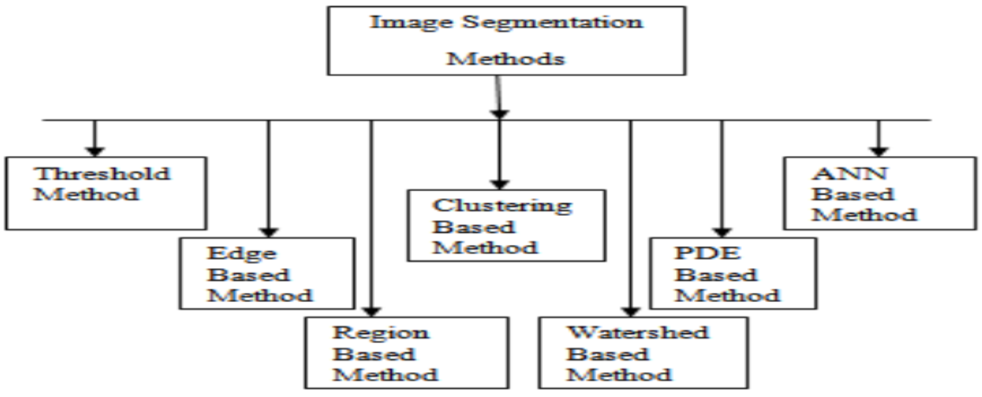


Figure 2. 6 Image segmentation techniques

Region-based method

In this technique pixels that are related to an object are assembled for segmentation. Also, this technique is termed similarity-based segmentation, partitioning an image into regions that are similar according to a set of predefined criteria. There won't be any gap due to missing edge pixels in this region-based segmentation [64]. The different types of region-based segmentation are discussed below:

A. Region growing

The region growing based segmentation methods are the methods that segment the image into various regions based on the growing of seeds or initial pixels. These seeds can be selected manually (based on prior knowledge) or automatically (based on a particular application) [65].

B. Region split

This segmentation method uses two basic techniques splitting and merging for segmenting an image into various regions. Splitting stands for iteratively dividing an image into regions having similar characteristics and merging contributes to combining the adjacent similar regions.

Edge-based method

This segmentation locates the edges, in either the first derivative of intensity is greater than a particular threshold or the second derivative has zero crossings. In edge-based segmentation methods, first of all, the edges are detected and then are connected to form the object boundaries to segment the required regions [66]. Edges are detected to identify the discontinuities in the image. Edges on the region are traced by identifying the pixel value and it is compared with the neighboring pixels. The most well-known edge base detection techniques are discussed below:

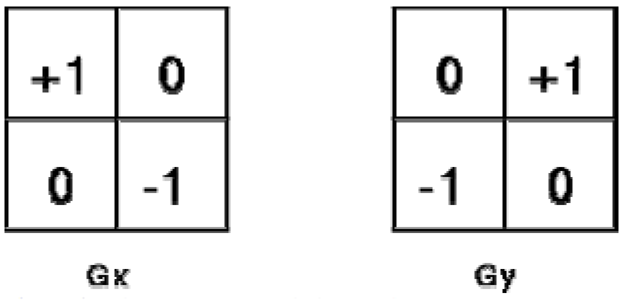
A. Roberts Edge Detection

This technique performs a simple, quick to compute, 2-D spatial gradient measurement on an image. It is a simple approximation of the first derivative. It marks edge points only; it does not return any information about the edge orientation. The simplest way of edge detection operator and works well in binary images. Equation 2.1 shows that the Robert operators

$$\sqrt{[I(r,c) - I(r-1,c-1)]2 + [I(r,c-1) - I(r-1,c)]2} \qquad 2.1$$

Where I(r,c) is image pixel at r (row) and c (column).

It performs a simple, quick to compute, 2-D spatial gradient measurement on an image. This method emphasizes regions of high spatial frequency which often correspond to edges. The operator consists of a pair of 2×2 convolution masks as shown in Figure 2.7 One mask is simply the other rotated by 90°.



$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Figure 2. 7 Roberts Cross convolution masks

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \qquad 2.\,2$$

Although typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y| \qquad 2.\,3$$

B. Sobel Edge Detection

The Sobel operator is mainly used for edge detection, and it is technically a discrete differential operator used to calculate the estimation of the gradient of the image luminance function. The Sobel operator is a typical edge detection operator based on the first derivative [65]. As a result of the operator in the introduction of a similar local average operation, so the noise has a smooth effect, and can effectively eliminate the impact of noise. The operator consists of a pair of 3×3 convolution masks.

It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction. The Sobel operator uses two 3 x 3 kernels. One for changes in the horizontal direction, and one for changes in the vertical direction. The two kernels are convolved with the original image to calculate the approximations of the derivatives. If we define Gx and Gy as two images that contain the horizontal and vertical derivative approximations respectively, the computations are:

$$G_X = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A \qquad and\ G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \qquad 2.\,4$$

Where A is the original image.

C. Prewitt Edge Detection

It will be estimated in the 3x3 neighborhood for eight directions. We have to calculate all eight convolution masks. One complication mask is then selected, namely with the purpose of the largest module [66]. It is slightly simpler to implement computationally than the Sobel detection, but it tends to produce somewhat noisier results. This operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If A define as the source image, and $G_x$and G_y are two images which at each point contain the horizontal and vertical derivative approximations, the latter are computed as:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * A \text{ and } G_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * A \qquad 2.5$$

Where * here denotes the 2-dimensional convolution operation.

Threshold-based method

These methods divide the image pixels concerning their intensity level. These methods are used over images having illumination objects than the background. The selection of these methods can be manual or automatic (i.e., can be based on prior knowledge or information of image features). The method split an image into smaller segments, by using one color or grayscale value to define their boundary in image processing. Besides, this segmentation method is generating only two classes, based on this, this technique is not applicable for complex images.

A. Global thresholding

This technique is used when the pixel values of the components and that of the background are fairly consistent in their respective values over the entire image. This is done by using any appropriate threshold value/T. The value of T will be constant for the whole image. Based on T the output image $g(\mathrm{x,y})$can be obtained from $f(\mathrm{x,y})$original image as:

$$g(x,y) = \begin{cases} 1, if\ f(x,y) \geq T \\ 0, otherwise \end{cases} \qquad 2.6$$

There are different global thresholding techniques some of them are: iterative thresholding, maximum correlation thresholding, Otsu thresholding, optimal thresholding, histogram analysis thresholding, and clustering thresholding.

B. **Local thresholding**

This method divides the input image into several subregions and each subregion selects a different Threshold value. When we have illumination that comes from shadows or the direction of illumination, a single threshold will not work well. Threshold function T(x, y) is given by

$$g(x,y) = \begin{cases} = 0, iff(x,y) < T(x,y) \\ = 1, iff(x,y) \geq T(x,y) \end{cases} \quad 2.7$$

Where $T(x,y) = f(x,y) + T$

C. **Adaptive Thresholding**

This technique will take a grayscale or color image as input and, outputs a binary image representing the segmentation. The drawback of this method is that it is computationally expensive and, therefore, is not appropriate for real-time applications.

Watershed based method

The watershed segmentation algorithms are based on the representation of an image in the form of a topographic relief, where the value of each image element characterizes its height at this point. These algorithms can process not only 2D images but also 3D images, so the term element is used to combine the terms pixel and voxel [67]. For a better outcome, watershed segmentation is often applied to the result of the distance transform of the image rather than to the original one. Accordingly, there are two main approaches of watersheds: by flooding and by rain falling [64]. Segmentation using the watershed transform works well if we can identify, or "mark," foreground objects and background locations. Marker-controlled watershed segmentation follows this basic procedure [68]:

1. Compute a segmentation function. This is an image whose dark regions are the objects you are trying to segment.
2. Compute foreground markers. These are connected blobs of pixels within each of the objects.
3. Compute background markers. These are pixels that are not part of any object.

4. Modify the segmentation function so that it only has minima at the foreground and background marker locations.
5. Compute the watershed transform of the modified segmentation function.

D. Feature Extraction

After segmenting the images the next phase is extracting the necessary features. Feature extraction is a special form of dimensionality reduction, extract the most relevant information from the original data and represent that information in a lower dimensionality space. It is concerned about the extraction of various attributes of an object and thus associates that object with a feature vector that characterizes it [62] [63]. Feature extraction is most important because the specific features made available for discrimination directly influence the efficiency of the classification task. Therefore, features can be classified into two categories: local features which are typically geometric (such as number of endpoints, joint, branches, etc), and global features which are typically for topological or statistical (e.g., number of holes, connectivity) [69]. Figure 2.8 illustrates different types of image features are used to represent an image for object recognition and/or identification.

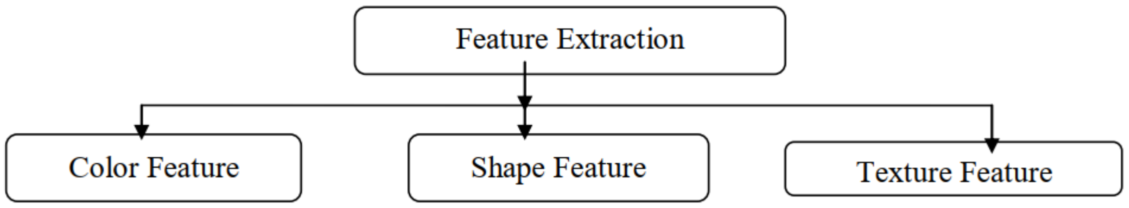


Figure 2. 8 Classification feature extraction method

A. Color features

Color is one of the most important features of an image and it is defined as a particular color space or model. Several color spaces have been used such as RGB, LUV, HSI, YCbCr, HSV, and HMMD. Once the color space is identified, the color feature can be extracted from the images or regions [70]. Color features are represented using Color Histogram, Color Correlogram, Dominant Color Descriptor, and Color Co-occurrence Matrix.

B. Texture features

The texture is an important element of human visual perception and is used in many computer vision systems. There is no precise definition of texture has been adopted yet, in addition, different authors define it as a measure of roughness, contrast, directionality, line-likeness, regularity, and roughness. Texture describes how the patterns of color are scattered in the images. This feature also describes the physical composition of the surface [69]. Texture features can be extracted by using various methods. Grey Level Co-occurrence Matrix (GLCM), Gabor filter, Histogram of Oriented Gradients, and Haar Wavelet Decomposition are examples of popular methods to extract texture features.

Gabor filter

Gabor filter is one of the most established texture descriptors introduced by Gabor in 1946 [71]. It extracts texture features by analyzing the frequency domain of the image. It is a Gaussian function modulated by complex sinusoidal frequency and orientation. It can perform both in the spatial and frequency domain and can be in any number of dimensions [69]. This filtering technique is broadly used to extract features for image classification because of the nature of the spatial locality, orientation selectivity, and frequency characteristic. Gabor filter or wavelets characterize an image by obtaining the center frequency and orientation parameter. The frequency and orientation representation, are useful for texture representation and discrimination and the same concept is used in the human visual system. The Gabor features are never changing to illumination, rotation, scale, and translation [72, 73]. Gabor filter is defined as in equation 2.4.

$$g(x,y,\lambda,\theta\psi,\sigma,\gamma) = exp\left(-\frac{x'^2+\gamma^2+\gamma'^2}{2\sigma^2}\right) exp\left(i\left(2\pi\frac{x'}{\lambda}+\psi\right)\right) \qquad 2.8$$

Where, x, = x cos θ + y sin θ

y, = -x sin θ + y cos θ

λ: represents the wavelength of the sinusoidal factor,

θ: represents the orientation of the normal to the parallel stripes of a Gabor function,

ψ: phase offset,

σ: Sigma or standard deviation of the Gaussian envelope,

γ: spatial aspect ratio

C. Shape features

The shape is one of the most important features in feature extraction. They are usually described when the image has been segmented into different regions or objects. Shape description can be categorized into either region-based or boundary-based. A good shape representation feature for an object should be invariant to translation, rotation, and scaling [74]. The most well-known shape techniques are the Shape moment invariant and binary image algorithm.

E. Classification

Classification of the image consists of a database that holds re-defined patterns that are compared with detected objects to classify them in the right category. Classification will be executed based on spectral-defined features such as density, texture, etc. Classification has a two-step process training and testing. In the training steps, a classification algorithm builds the classifier by analyzing and learning from a training dataset and its associated class labels. In the testing step, the model is used to predict class labels for given data. Machine learning and deep learning techniques are uses for classification algorithms.

2.7. Machine learning

Machine learning (ML) is the subfield of artificial intelligence that is dedicated to the design and development of algorithms and techniques that allow computers to learn. It provides techniques that can automatically build a computational model for complex relationships by processing the available data and maximizing a problem-dependent performance criterion [17]. According to [75] ML can determine patterns and correlations and discover knowledge from datasets. The models need to be trained using datasets, where the outcomes are represented based on experience. The predictive model is built using several features, and as such, parameters of the models are determined using historical data during the training phase. For the testing phase, part of the historical data that has not been used for training is used for performance evaluation purposes.

Machine learning applies in more and more scientific fields including, bioinformatics, biochemistry, medicine, meteorology, economic sciences, robotics, aquaculture, and food security, and climatology, chemical informatics, social network analysis, stock market analysis, robotics, fraud detection, credit analysis, fault prediction models, image recognition patterns, intelligent spam filters and product quality analysis [15].

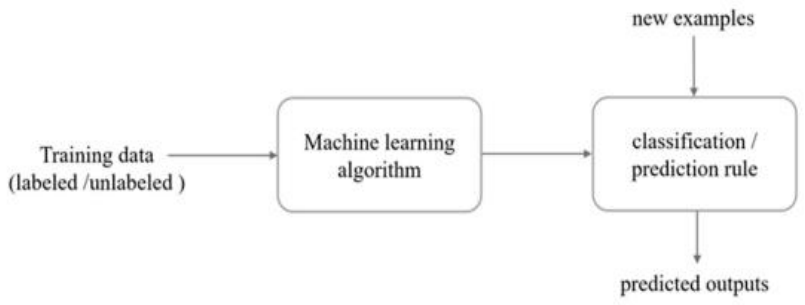

new examples
Training data (labeled /unlabeled)
Machine learning algorithm
classification / prediction rule
predicted outputs

Figure 2. 9 Typical machine learning approach adopted from *[76]*

Figure 2.9 shows that Data in machine learning consists of a set of examples. Usually, an individual example is described by a set of attributes, also known as features or variables. After the end of the learning process, the trained model can be used to classify, predict, or cluster new examples (testing data) using the experience obtained during the training process. The performance of machine learning models and algorithms, various statistical and mathematical models are used [76]. Machine learning techniques can be broadly classified into four main categories that are supervised, unsupervised, semi-supervised and reinforcement learning depending on the learning signal of the learning system.

Supervised learning algorithms are trained using labeled examples, such as an input where the desired output is known. As a result, each training sample comes in the form of a pair of input and output values. The algorithm then trains a model that predicts the value of the output variables from the input variables using the defined features in the process. If the output variables take a discrete set of values, then the predictive model is called a classifier. For Classification and regression algorithms, common algorithms are used including random forests, decision trees, and support vector machines, logistic regression, neural networks, and so on [17].

Unsupervised learning techniques require only the input feature values in the training data and the learning algorithm discovers hidden structures in the training data based on them. There is no distinction between training and test sets with data being unlabeled [77]. The learner processes input data to discover hidden patterns. In bioinformatics, these techniques are used for problems such as microarray and gene expression analysis Clustering algorithms, such as K-means, nearest-neighbor are often used in unsupervised machine learning [17] [76].

Reinforcement learning uses a system of reward and punishment to train the algorithm. The algorithm or an agent learns from its environment, the agent gets rewards for correct performance and penalties for incorrect performance. In reinforcement learning, the algorithm is not told how to perform the learning; however, it works through the problem on its own [16]. In semi-supervised learning, it uses both labeled and unlabeled data for training typically a small amount of labeled data with a large amount of unlabeled data (because unlabeled data is less expensive and takes less effort to acquire). This type of learning can be used with methods such as classification, regression, and prediction. Common algorithms include graph theory inference algorithms, Laplacian support vector machines, and so on [17].

2.8. Deep Learning Approach

Deep learning is a subfield of machine learning which attempts to learn high-level abstractions in data by utilizing hierarchical architectures. Deep learning takes a new on learning representations from data that emphasizes learning successive layers of increasingly meaningful representations. The deep in deep learning isn’t a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations [16] [78].

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering complicated structures in high-dimensional data and is therefore applicable to many domains of science, business, and government intricate structures in high-dimensional data and is therefore applicable to many domains of science, business, and government [79].

It contains a quite large number of processing layers. These models have revolutionized sectors such as image recognition, voice recognition, semantic parsing, transfer learning, natural language processing, computer vision, and other complex processes such as analysis of large volumes of data, self-driving vehicles, machine translation, interpretation, sentiment analysis, question answering, language translation and many more [16] [80].

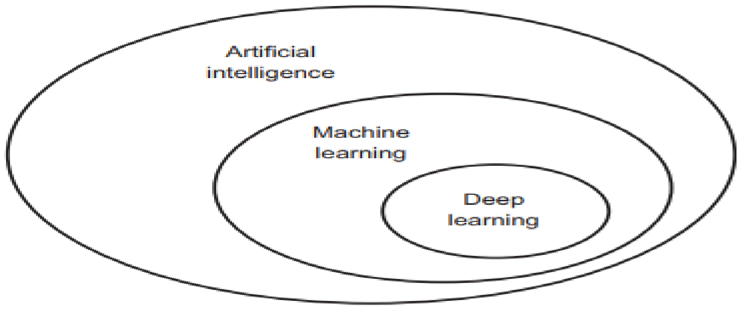


Figure 2. 10 Venn diagram for deep learning

Modern deep learning often involves tens or even hundreds of successive layers of representations and they have all learned automatically from exposure to training data. Meanwhile, other approaches to machine learning tend to focus on learning only one or two layers of representations of the data. These layered representations are (almost always) learned through models called neural networks, the term neural network is a reference to neurobiology. but some concepts in deep learning were developed in part by drawing inspiration from our understanding of the brain deep-learning models are not models of the brain, there's no evidence that the brain implements anything like the learning mechanisms used in modern deep-learning models. [78].

Authors in [16] stated that the most popular deep learning methods that have been widely used are Convolutional Neural Networks (CNNs), Recurrent Neural Network (RNN), Restricted Boltzmann Machines (RBMs), Autoencoder, and Sparse Coding. Among them, Convolutional Neural Network is the most popular emerging technology, as many scholars in [29] [30] [81] prove that convolutional neural networks (CNNs) have achieved impressive results in the field of image classification, hence from this point of view the researcher tried to implement this architecture for sorghum leaf diseases classification. The next subtopics will discuss the most widely used deep learning architecture.

2.8.1 Autoencoder

The deep autoencoder is a special form of the DNN (with no class labels), whose output vectors have the same dimensionality as the input vectors. It is mainly used to process complex high dimensional data, and it is often used for learning a representation or effective encoding of the original data, in the form of input vectors, at hidden layers. Note that the autoencoder is a nonlinear feature extraction method without using class labels. As such, the features extracted aim at conserving and better-representing information instead of performing classification tasks, although sometimes these two goals are correlated [82].

2.8.2 Restricted Boltzmann Machines (RBMs)

A Restricted Boltzmann Machine (RBM) is a generative stochastic neural network that comes from the Boltzmann machine and was proposed in 1986. RBM is different from the Boltzmann Machine, with the restriction that the visible units and hidden units must form a divided graph. Even though RBM has inherited the two-layer neuron structure of the Boltzmann machine, there is no connection between neurons in the same layer with only the whole connection between the visual layer and the hidden layer. Restricted Boltzmann Machines (RBM) is used to train layers of the Deep Belief Networks (DBNs) [83].

2.8.3 Recurrent Neural Network (RNN)

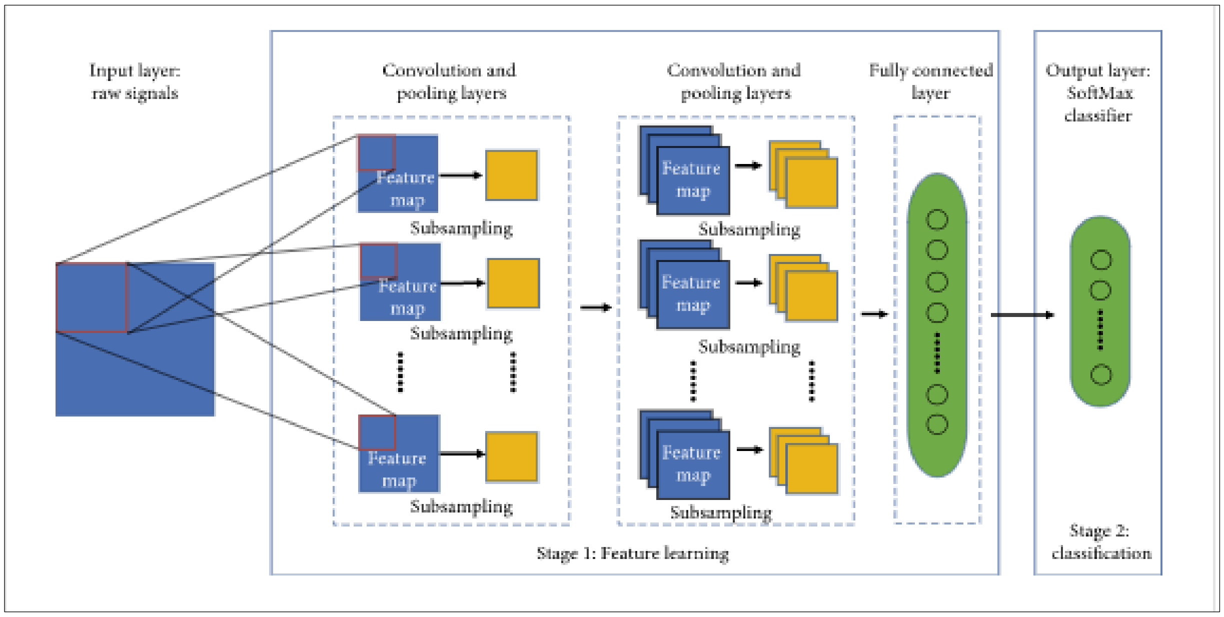
Recurrent neural networks (RNNs) are a type of artificial neural network. This structure allows the information to be circulated in the network. RNN is the ability to recognize and predict sequences of data such as text, genomes, handwriting, spoken word, or numerical time series data. They have loops that allow a constant flow of information and can work on sequences of random lengths. The RNNs attempts to address the necessity of understanding data in sequences. RNN can be used for time series prediction because it can remember previous inputs also, which is called Long-Short Term Memory (LSTM).

2.8.4 Convolutional Neural Network

Convolutional neural network (CNN) is one of the main learning mechanisms to do images recognition, images classifications, objects detections etc. It is based on learning levels of representations. The higher lever concepts are defined from lower-lever ones, and the same lower lever concepts can help to define many higher lever concepts. It learn multiple levels of representation and abstraction which helps to understand dataset such as images, audio and text. It is advantageous of simple structure, less training parameters because of shared weights and adaptability [84].

Their multistage architectures are inspired by the science of biology, through these models, invariant features are learned hierarchically and automatically. They first identify low-level features and then learn to recognize and combine these features to learn more complicated patterns [31]. CNN's constitute one of the most powerful techniques for modeling complex processes and performing pattern recognition in applications with large amounts of data [80].

According to [75] machine learning classifiers algorithms, such as random forests, Naïve Bayes, support vector machines, and decision tree uses hand-crafted feature extraction techniques to extract color, size, and texture. Whereas CNN does not require hand-crafted feature extraction, it has better accuracy, it does not require image segmentation by experts and it has millions of learnable parameters to estimate [85]. Figure 2.11 shows the architecture of the convolutional neural network.



Input layer: raw signals
Convolution and pooling layers
Feature map
Subsampling
Convolution and pooling layers
Feature map
Subsampling
Fully connected layer
Output layer: SoftMax classifier
Stage 1: Feature learning
Stage 2: classification

Figure 2. 11 Overview of CNN architecture adopted from [86]

Figure 2.11, shown that the overview of convolutional neural network architecture and the way of its process to train the network with the help of input data. This architecture has two building blocks feature extraction learning block and classification block. The first stage (i.e., feature extraction block) consists of convolutional layer, pooling layer, and fully connected layer, the first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final stage (i.e., classifier block) this stage contains classifier algorithms such as soft mask [86] [85].

There are two stages for training the network: forward propagation and backward propagation. The forward propagation is to represent the input image with the current parameters (weights and bias) in each layer. Then the prediction output is used to compute the loss cost with the ground truth labels. Based on the loss cost, the backward propagation computes the gradients of each parameter with chain rules. All the parameters are updated based on the gradients and are prepared for the next forward computation. After sufficient iterations of the forward and backward stages, the network learning can be stopped [16]. As described in [86] [85], deep learning can extract features from the images automatically, and classify them into a certain class label. There are three feature extraction blocks: convolution layer, pooling layer, and a fully connected layer including activation function.

Convolution layer

Convolution is a special type of linear operation which is used for extracting features from an image, with a small array of numbers called a kernel. The kernel is applied to the image, which is an array of numbers called a tensor. This plays a significant role in how CNN operates it forms the fundamental unit of a ConvNet wherever most of the computation is concerned [87]. The layer served as a feature extractor, and thus they learn the feature representation of their input images.

The neurons in the convolutional layers are arranged into feature maps. Each neuron in a feature map has a receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights, sometimes referred to as a filter bank [81]. According to [16], There are three main advantages of the convolution layer over just fully connected layers: the weight sharing mechanism in the same feature map reduces the number of parameters, local connectivity learns correlations among neighboring pixels, and invariance to the location of the object. In the convolutional layers, a CNN utilizes various kernels to convolve the whole image as well as the intermediate feature maps, generating various feature maps, as shown in figure 2.12.

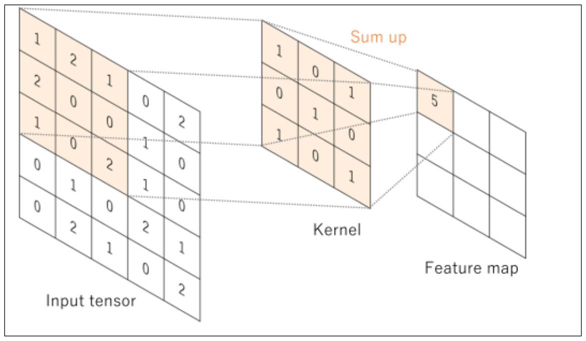


Figure 2. 12 The operation of the convolutional layer adopted from *[16]*

Figure 2.12, shows an example of a convolution operation with a kernel size of 3*3, no padding, and a stride of 1. A kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (Activation map).

Hyper-Parameters of Convolution Layer

Different setups of hyper-parameters such as learning rate, learning max-iterations, and mini-batch for CNN would lead to different performances. Convolutional layers are also able to significantly reduce the complexity of the model through the optimization of its output. These are optimized through hyper-parameters, depth, filter size, stride, and setting zero-padding [88].

A. Filter size

The size of the filter concerning the size of the image (or activation layer), determines what features can be detected by the filters. Each filter length, on the side, comprises only 4.3% of the (square) image-side length. These filters in the first layers cannot extract features that span more than 0.24% of the input image area [89].

B. Padding

The convolution operation illustrated in Figure 2.13, one of the drawbacks of the convolution step is the loss of information that might exist on the border of the image. Padding is a technique that addresses this issue, where rows and columns of zeroes are added on each side of the input tensor, to fit the center of a kernel on the outermost element and keep the same in-plane dimension through the convolution operation. Zero paddings, as shown in Figure 2.13, can have the effect of canceling dimensional reduction, and maintaining the input dimension at the output. This hyper-parameter is the straightforward method of padding the border of the input and an efficient technique to provide additional management on the dimensionality of the output volumes [89] [87].

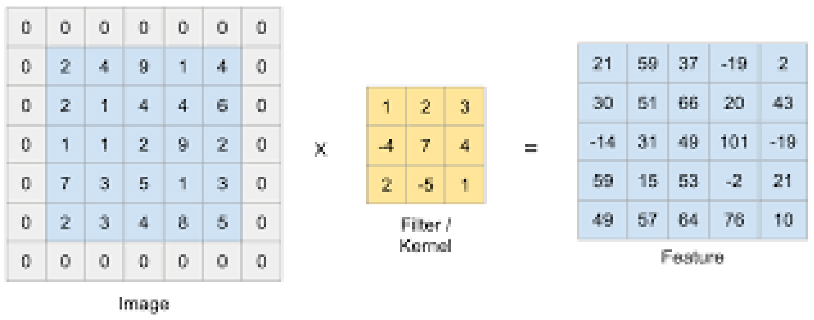


Figure 2. 13 Convolution operation with zero paddings Adopted from [87]

As illustrated in figure 2.13 Let with N=5, F=3, and stride 1, the output will be 3×3 (which shrinks from a 5×5 input). However, by adding one zero paddings, the output will be 5×5, which is the same as the original input.

C. Stride

CNN has more options which provide a lot of opportunities to even decrease the parameters more and more, and at the same time reduce some of the side effects. One of these options is stride. Stride is the distance between two successive kernel positions [90].

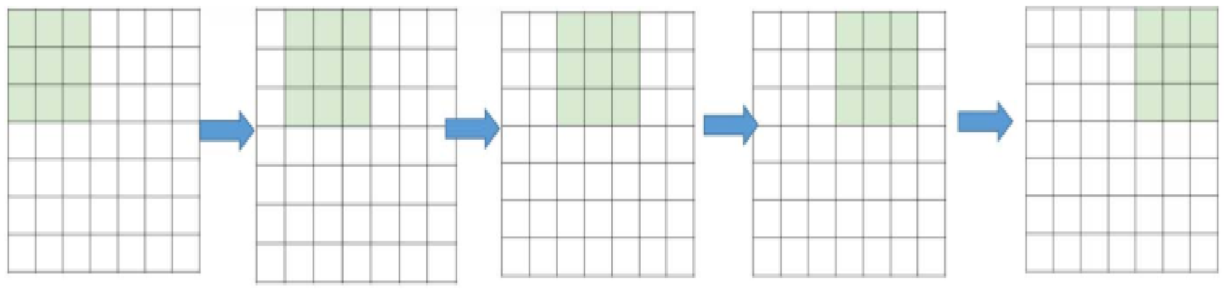


Figure 2. 14 Applying stride to convolve the image adopted from *[90]*

Figure 2.14, shows a given 7×7 image. If we move the filter to one node every time, we can have a 5x5 output only. However, if we move and make every stride 2, then the output will be 3x3. Put simply, not only overlap but also the size of the output will be reduced.

D. Activation Function

The relation between image class and image data is non-linear. For a neural network to construct the non-linear relation between the data and image class, the activation function must have a nonlinearity. Without non-linearities, a neural network would be capable of only linear classification [89]. The activation function is applied to the last fully connected layer which can activate the feature of neurons to solve nonlinear problems [91].

The activation function is used to increase the expression ability of the neural network model, which can be either linear or non-linear depending on the function it represents and is used to control the outputs of neural networks. Moreover, using activation functions seems essential while dealing with complex problems like face recognition, image processing, or computer vision [92]. A few commonly nonlinear activation functions are discussed below.

1. Sigmoid function

The sigmoid function is shown in Figure 2.17, which is a common non-linear activation function. The output of this function is bounded, and it was widely used as the activation function in deep neural networks during the early age of deep learning. That is, the slope of the graph tends to be zero when the input is very large or very small. When the slope of the function is close to zero, the gradient that passed to the underlying network becomes very small, which will make network parameters difficult to be trained effectively. Meanwhile, the direction of weight update only to one direction due to the output of this function is always positive, which will affect the convergence rate [92]. The formula of the sigmoid function is presented in Eq 2.9.

$$f(x) = \frac{1}{1+e^{-x}} \qquad 2.9$$

Sigmoid is still very popular in classification problems, especially in the output layer of binary classification, where the result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, the result can be predicted easily to be 1 if the value is greater than 0.5 and 0 otherwise.

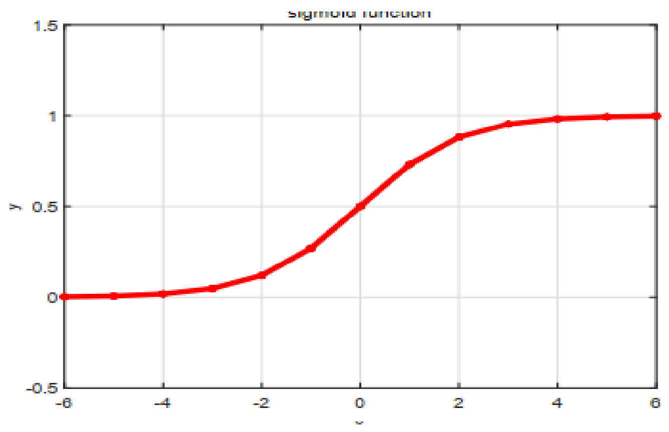


Figure 2. 15 Sigmoid functions adopted from *[92]*

2. Hyperbolic Tangent Function (Tanh)

Like Sigmoid, this function is also continuous as well as differentiable on all the points. The range of tanh is -1 to 1 [91], thus makes the mean of the outputs come to be 0 or very close to 0, furthermore, makes the data more concentrated, and makes the learning much easier, thus it is usually used in hidden layers of ANNs [93]. The tanh function is the updated version of the sigmoid function on the range, which is an asymmetric function centered on zero. Its output is bounded, and it brings nonlinearity to the neural network. The curve of this function can be seen in Figure 2.16. The convergence rate is higher than the sigmoid function. The formula of tanh function is presented in Eq 2.16.

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \qquad 2.10$$

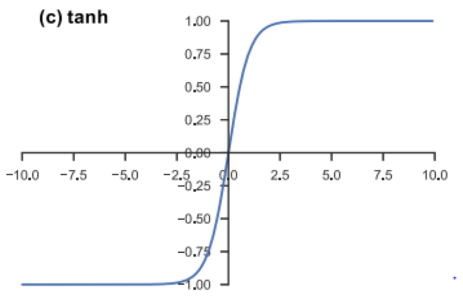


Figure 2. 16 Tanh function adopted from *[85]*

The tanh function became the preferred function compared to the sigmoid function in that it gives better training performance for multi-layer neural networks. However, the tanh function could not solve the disappear gradient problem suffered by the sigmoid functions as well. The main advantage provided by the function is that it produces zero centered output thereby aiding the back-propagation process [94]. Hard hyperbolic or Hardthanh is an extension of tanh activation function, this activation is a cheaper and more computational efficient version of tanh.

3. Rectified Linear Unit (ReLU)

ReLU is commonly used as an activation function in convolutional networks due to the advantage of the speed of unsaturated nonlinear functions when the training gradient descends. the function form of ReLU is shown in equation 2.11, where x is the input of the activation function [86].

$$f(x) = ma\,x(0, x) \qquad 2.\ 11$$

Where 'X' is the input data to a neuron. ReLu returns the positive value calculated by a neuron and ignores the negative values. So, if the calculated value at any node is negative; ReLu will return 0 otherwise it will forward the calculated value [91].

The disadvantage of this function is if the value of the input is positive, it will not change anything and will simply forward the value to the next layer's node. However, the major advantage considered is that if the ReLu function receives a negative value, it stops the neuron to stimuli by making it 0 and hence preventing the neuron to fire [91]. The shape of the Relu function is shown in figure 2.17.

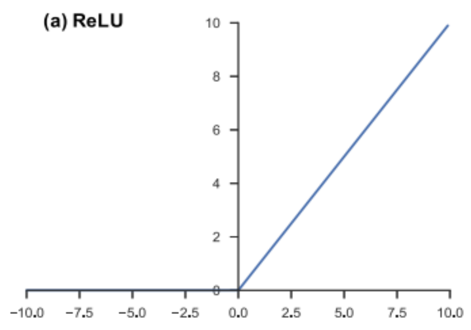


Figure 2. 17 Rectified linear unit (ReLU) activation function

Leaky ReLU

When a large number of neurons have a negative value and are restricted to fire (dead neurons) by the ReLu function, the contribution to the overall model is dependent on a smaller number of neurons and may not be accurate. Leaky ReLu is an extension of the ReLU activation function. If any neuron's value is negative in the ReLu function, it is simply converted to 0 regardless of how much negative value the neuron has. However, Leaky ReLu argues that even if a neuron has a negative value, it should still play a role in action prediction. As a result, each input value must be multiplied by a constant term [91].

$$y = ax \qquad 2.\ 12$$

if and only if $X < 0$. The constant value 'a' could be a small value like .01 as per the requirements. Y= x, if x>0, where x contains the input value to Leaky ReLu function.

4. SoftMax

The SoftMax function is another type of activation function used in neural computing. It is used to compute probability distribution from a vector of real numbers. The SoftMax function produces an output which is a range of values between 0 and 1, with the sum of the probabilities been equal to 1 [94]. In [93] states that, SoftMax function is used in multi-class models where it returns probabilities of each class, with the target class having the highest probability, mostly appears in almost all the output layers of the deep learning architectures. This function is differing from Sigmoid in that Sigmoid is used in binary classification while SoftMax is used for multivariate classification tasks.

Pooling Layer

A Pooling layer is frequently inserted between successive Convolution layers in a ConvNet architecture. Its purpose is to gradually shrink the spatial dimension of the representation to reduce the number of parameters and computations in the network, as well as to prevent overfitting. A pooling layer with 2x2 filters applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, removing 75% of the activations, and is the most frequent variant [90].

Pooling operators consist of a fixed-shape window that is slid over all regions in the input according to its stride, computing a single output for each location traversed by the fixed-shape window (also known as the pooling window). However, the pooling layer contains no parameters (there is no kernel), instead, its operations are typically calculating either the maximum or the average value of the elements in the pooling window. These operations are called maximum pooling (max pooling for short) and average pooling [28].

A. Maximum-Pooling

Max-pooling is one of the most common types of pooling methods. It partitions the image into sub-region rectangles, and it only returns the maximum value of the inside of that sub-region. One of the most common sizes used in max-pooling is 2×2 [90]. Figure 2.18 shows the operation of max pooling.

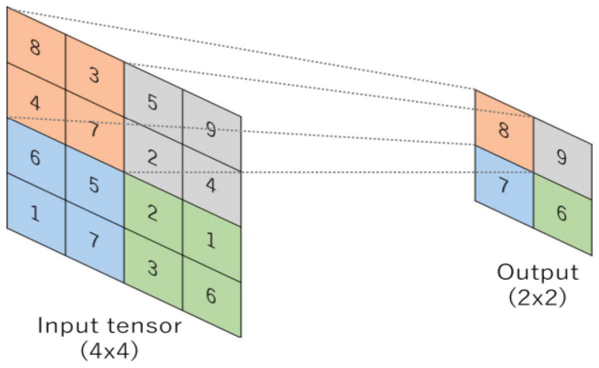


Figure 2. 18 Max-pooling operations adopted from *[85]*.

As illustrated in Figure 2.18, an example of max pooling operation with a filter size of 2 × 2, no padding, and a stride of 2, which extracts 2 × 2 patches from the input tensors, outputs the maximum value in each patch, and discards all the other values, resulting in downsampling the in-plane dimension of an input tensor by a factor of 2 [85]. The intuition of what max pooling is doing is that the large number means that there may be detected a feature [95].

B. Average Pooling

The idea of average or mean for pooling and extracting the features that are the first convolution-based deep neural network. As shown in Figure 2.19, the average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region [96].

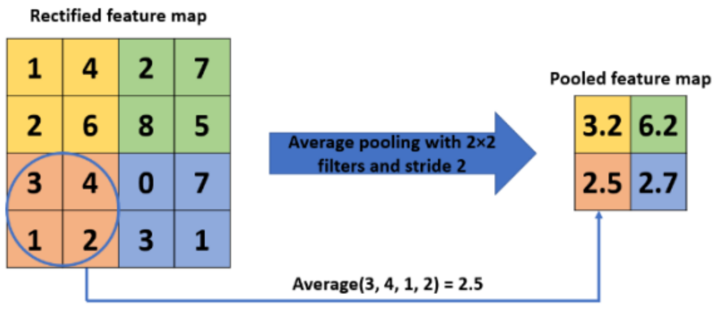


Figure 2. 19 Average pooling operation adopted from *[96]*.

Fully Connected Layer

After several convolution and pooling layers, the CNN generally ends with several fully connected layers, also known as dense layers, in which every input is connected to every output by a learnable weight. Once the features extracted by the convolution layers and down sampled by the pooling layers are created, they are mapped by a subset of fully connected layers to the final outputs of the network, such as the probabilities for each class in classification tasks [95] [85].

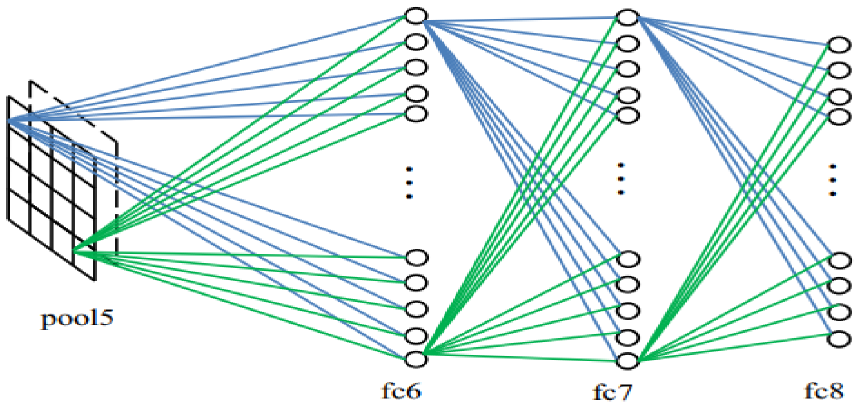


Figure 2. 20 The operation of the fully-connected layer [adopted from *[16]*]

As shown in figure 2.20, fully connected layers possess a large number of parameters and so require powerful computational resources.

Epochs

An epoch is a term used in machine learning that indicates the number of passes of the entire training dataset the machine learning algorithm has completed. Datasets are usually grouped into batches (especially when the amount of data is very large).

2.9. Loss functions

A loss function, also referred to as a cost function, measures the compatibility between output predictions of the network through forwarding propagation and given ground truth labels. Depending on the study and CNN model there are several loss calculation functions. Some of them are the Binary Cross-entropy class, Categorical Cross-entropy class, Sparse Categorical Cross-entropy class, Poisson class binary cross-entropy function. Based on this, for this study, categorical cross-entropy was used as calculating the loss function.

Categorical cross-entropy

Also called **logarithmic loss**, **log loss**, or **logistic loss.** This loss calculation function is widely used when there are two or more label classes (i.e., in this study there are four label classes namely: Anthracnose, healthy, leaf bight, and rust), and also if the activation function is Softmax. The expected label is provided in a one-hot representation [85]. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem. The score is minimized and a perfect cross-entropy value is 0 [97].

2.10. Optimization Techniques

During the training process, we change the parameters (weights, learning rate) of the model to try and minimize the loss function, and make our predictions as correct and optimized as possible. To do this, optimizations come in, this method ties together the loss function and model parameters by updating the model in response to the output of the loss function [97].

All the optimizers have parameters that can also be modified, each optimizer has its parameters, but there is one that is shared between all of them, the Learning Rate. This parameter will define how much the weights are updated after each epoch. For a high Learning Rate, the weight change will be higher than for a small Learning Rate, after each epoch. Also, another important parameter is the weight decay which is an additional term in the weight update rule that causes the weights to exponentially decay to zero if no other update is scheduled. Some of the most well-known optimization techniques are Nadam, Gradient Descent (GD), Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), RMSprop, Adagrad, and Adadelta.

A. Nadam

Nadam (Nesterov-accelerated Adaptive Moment Estimation) is an extension of the Adam algorithm by combining RMSprop and momentum: RMSprop contributes the exponentially decaying average of past squared gradients, while momentum accounts for the exponentially decaying average of past gradients. Namam update rule are illustrated in the following equation.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\nu_t + \epsilon}} (\beta_1 m_t + \frac{(1 - \beta_1) gt}{1 - \beta_1^t})$$

B. Gradient Descent

Gradient descent is an optimization algorithm that is used when training a deep learning model. This optimization algorithm finding a local minimum of a differentiable function. Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible it’s fast, robust, and flexible. This algorithm is iterative, which starts from a random point on the function and traverses down its slope in steps until the loss function gets as low as possible. The weight is initialized using some initialization strategies and is updated with each epoch according to the update equation.

The magnitude and direction of the weight update is computed by taking a step in the opssosite direction of the cost gradient.

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

Where η is the learning rate. The weights are then update after each epoch via the following rule:

$$w \coloneqq w + \Delta w,$$

Where Δwa vector that contains the weight is updates of each weight coefficient $w,$

C. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an extension of Gradient Descent, where it overcomes some of the disadvantages of the Gradient Descent algorithm. SGD tries to overcome the disadvantage of computationally intensive by computing the derivative of one point at a time. The design of stochastic gradient descent is using one sample randomly to update the gradient per iteration, instead of directly calculating the exact value of the gradient. The cost of the stochastic gradient descent algorithm is independent of sample numbers and can achieve sublinear convergence speed. SGD reduces the update time for dealing with large numbers of samples and removes a certain amount of computational redundancy, which significantly accelerates the calculation. SGD in contrast performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$

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

D. Adaptive Moment Estimation (Adam)

Adaptive Moment Estimation is the combination of RMSprop and Stochastic Gradient Descent with momentum. Adam computes adaptive learning rates for each parameter. In addition to storing the previous decaying average of squared gradients, it also holds the average of past gradients similar to Momentum.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$m_t$and $v_t$are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. As mtmt and vtvt are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. β1β1 and β2β2 are close to 1).

2.11. Regularization of Convolutional Neural Network

Deep neural networks are complex learning models that are exposed to overfitting. Overfitting refers to a situation where a model learns statistical regularities specific to the training set, i.e., ends up memorizing the irrelevant noise instead of learning the signal, and, therefore, it performs less well on a subsequent new dataset. To overcome this issue, the regularization technique is used. Regularization refers to a set of different techniques that lower the complexity of a neural network model during training and thus prevent overfitting. Studies in [85] [16], suggest that the better solution for decreasing overfitting is obtaining more training data, hence if the model trained through large data typically generalizes better. and also, other overfitting solutions are batch normalization, data augmentation, early stopping, dropout, and soon.

A. Early stopping

The idea behind early stopping is that when we're fitting a neural network on the training data and model is evaluated on the unseen data after each iteration. When we see that the performance on the validation set is getting worse, the network immediately stops the training on the model. This is known as early stopping. If the performance of the model on the validation data is not improving i.e., a validation error is increasing or remaining the same for certain iterations, then there is no point in training the model further. This process of stopping model training before it reaches the lowest training error.

B. Dropout

This is one of the most particular types of regularization techniques. This technique produces very good results and is consequently the most frequently used regularization technique in the field of image classification.

C. L1 **and** L2

This regularization technique makes the Weight Penalty that is quite commonly used to train models. It works on an assumption that makes models with larger weights more complex than those with smaller weights. The role of the penalties in all of this is to ensure that the weights are either zero or very small. The only exception is when big gradients are present to counteract. Weight Penalty is also referred to as Weight Decay, which signifies the decay of weights to a smaller unit or zero.

D. Data augmentation

Augmentation is a process of generating data artificially from the existing training data by doing slight changes such as rotation, flips, adding blur to some pixels in the original image, or translations. Augmenting with more data will make it harder for the neural network to drive the training error to zero. By generating more data, the network will have a better chance of performing better on the test data. Depending on the task, we might use all the augmentation techniques and generate more training data.

2.12. Convolutional Neural Network Architectures

With the recent developments of CNN schemes in the computer vision domain, some well-known CNN models have emerged. This section presents the most efficient and widely used architectures of convolutional neural networks for classifying [98].

2.12.1 AlexNet

The first famous CNN architecture is AlexNet, which popularizes the convolutional neural network in Computer vision, developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. As shown in figure 2.21, this architecture has five convolutional layers starting from an 11 * 11 kernel. It was the first architecture that deploys max-pooling layers, ReLu activation function, and dropout for the huge linear layers. After inputting one fixed-size image, the network would repeatedly convolve and pool the activations, then forward the results into the fully-connected layers. The network was trained on ImageNet and integrated various regularization techniques, such as data augmentation, dropoutNetwork had similar architecture to LeNet; however, it was the most profound, most significant architecture with all convolution layers stacked together rather than the altering convolution and pooling layers as it was in LeNet [87].

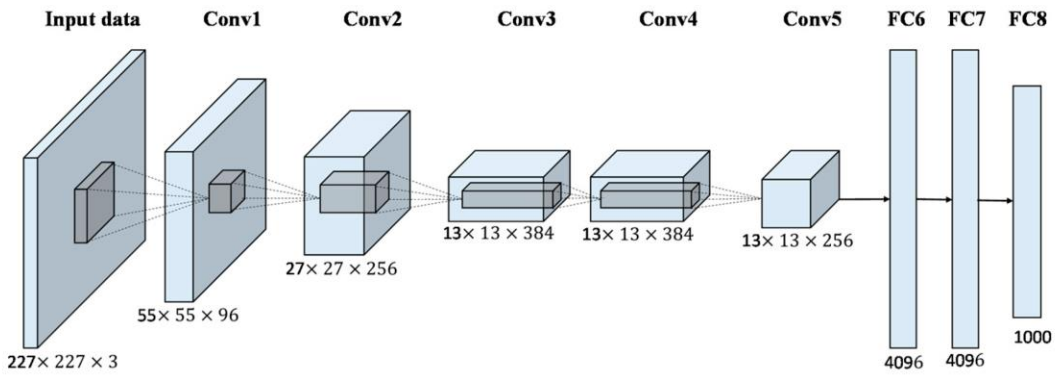


Figure 2. 21 AlexNet Architecture

2.12.2 GoogLeNet(Inception)

This architecture was developed by Google; it was the winner of the ImageNet Large Scale Visual Recognition Challenge ILSVRC in the 2014 competition. Its main contribution to this architecture was the development of an inception module that dramatically reduced the number of parameters in the network in its onset module known as inception-v1. It had only 4 million parameters compared to AlexNet with 60 million [98]. GoogLeNet has twenty-two layers of the Inception module; however, it has fewer parameters as compared to AlexNet. Later several enhancements had been done on Inception-v1, the key being an introduction of batch normalization and RMSprop that led to Inception-v2. There are several subsequent versions of

GoogleNet with added refinements, such as Inception-v3, and most presently Inception-v4 [87].

2.12.3 VGG-Net

Visual Geometry Group Network (VGGNet) was the runner-up in the ILSVRC in 2014. It improves AlexNet and has 19 layers in total. Its main contribution was in showing that the depth of the network or the number of layers is a critical component for good performance. Although VGGNet achieves a phenomenal accuracy on the ImageNet dataset, its deployment on even the most modest-sized Graphics Processing Units (GPUs) is a problem because of huge computational requirements, both in terms of memory and time. It becomes inefficient due to the large width of convolutional layers [98]. It is currently the most preferred choice in the community for extracting features from an image. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor. This architecture has two different versions namely: VGG16 and VGG19 (the number represents the number of layers in the network). VGG19 managed to win the challenge. This was a new deeper model than what had previously been used, which led to deeper networks becoming prominent within the field.

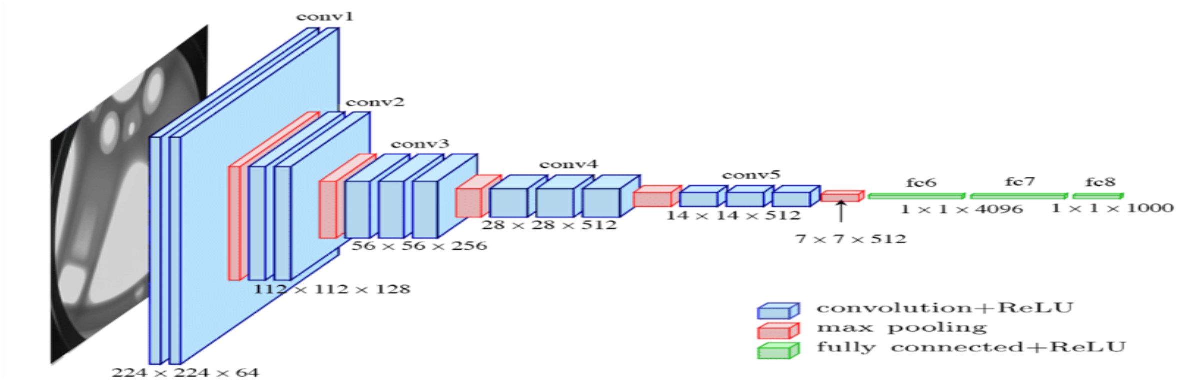


Figure 2. 22 VGG-Net Architecture

2.12.4 ResNet

The winner of the ImageNet large scale visual recognition challenge (ILSVRC) 2015 competition with a top-5 error of 3.57% was an ensemble of six networks of the ResNet (Residual Network) type, which was developed by Microsoft. ResNet is a 152-layer network, which was ten times deeper than what was usually seen when it was invented It features special skip connections and heavy use of batch normalization [98] [99]. It uses a global average pooling followed by the classification layer. ResNets are currently by the far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice [98].

2.12.5 DenseNet

DenseNet was published by Gao Huang and his teammate in 2017. It has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet) [98].

2.13. Evaluation technique

Confusion Matrix provides a complete overview by summarizing the classification results. It shows the individual results for each of the categories by tabulating the predicted and actual categories. In addition, in this study, the performance of the classifier model is evaluated through accuracy, true positive rate, false-positive rate, precision, recall, and f-measure. Table 2.1 shows the confusion matrix.

Table 2. 1 Confusion matrix

Actual class	Predicted class	
	Positive	Negative
Positive	True positive	False-negative
Negative	False positive	True negative

Where:

True positive (TP): Represents the number of images that are correctly classified as positive by the developed classifier model.

False-positive (FP): Represents the number of images that are classified as positive in the predicted class but they are negative.

False-negative (FN): Represents the number of images that are classified as negative in the predicted class but they are positive.

True negative (TN): Represents the number of images that are correctly classified as negative in both actual and predicted classes.

Accuracy: is calculating the samples of which are classified correctly among the whole samples of the dataset. It is calculated using the formula as follow:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100 \qquad 3.1$$

Precision: It measures the positive predictive value of the classifier model and is given by the formula as follows:

$$Precision = \frac{TP}{TP+FP} \qquad 3.2$$

Recall: It measures the positive instance which is predicted as positive and is defined as follows:

$$Recall = \frac{TP}{TP+FN} \qquad 3.3$$

F-measure: is a combination mean for both recall and precision as in the following:

$$F-measure = 2 \times \frac{Precision \times Recall}{Presision + Recall} \qquad 3.4$$

False-positive rate (FPR): The false-positive rate (FPR) measures the rate of the falsely recognized samples.

$$\boldsymbol{FPR = \frac{FP}{FP+TN}} \qquad 3.5$$

2.14. Related work

Several research efforts have been conducted and proposed for automatic plant disease identification. However, each work is designed to address different plants or parts of a plant. So, in this section, related works of different researchers in the area of automatic plant disease identification using machine vision systems are reviewed.

In paper [24], the researchers proposed a machine learning approach for crop yield improvement using plant leaf disease detection. Detecting and classifying sorghum leaf diseases as Leaf blight, Sooty Stripe, Zonate Leaf Spot, and Rough Leaf Spot are the main objectives of the proposed model. The necessary data were collected by the researcher. Hence, to get the quality data different preprocessing techniques were applied. Towards creating a classifier model, the researchers have used the Alex-Net classification algorithm. The developed classifier model, Alex-Net consists of the first 5 Convolutional layers and the last 3 fully connected layers, in between, there is a pooling and activation layer. However, in this study, the researcher concludes that the selected algorithm is the better classifier without any justification may other classifiers will achieve better accuracy. Nothing was reported on the accuracy of their classifier model. In addition, what kind of test options are implemented, and the number of images is not stated.

In [25], machine learning approach for plant disease identification. The proposed model is detecting and classifying plant leaf diseases as sorghum, cabbage, citrus, damaged sorghum, damaged cabbage, and damaged citrus. In this study, color-based information, statistical information was extracted from colors such as mean, median, max, and HOG. From the total of 1183 images, 60% were used for training and the remaining 40% was used for testing. Based on this, the developed classifier model accuracy of random forest 95.4%, ANN 94.9, and SVM 80.5% were achieved. Towards identifying the best feature, the researcher has selected color and statically features, and test them through traditional machine learning algorithms such as RF, SVM, and ANN. Conversely, the main aim of this study was not developing a classifier model rather than identifying which is the best features for representing the images.

In [26], the researchers proposed a model that classifies rice leaf diseases into Bacterial leaf blight, Brown spot, and Leaf smut using machine learning. For their experimentation, 480 images were collected from the University of California, Irvine (UCI) machine learning repository. Different preprocessing techniques were used using the WEKA tool. In addition, for attribute selection, CorrelationAttributeEval technique and classification algorithms KNN, decision tree, Naïve Bayes, and Logistic regression are implemented. In this study, color features are used for identifying the infected leaves. From the collected instance 90% are assigned for training and the remaining 10% for testing. Accordingly, the developed classifier model achieves an accuracy of 97.91% with the decision tree classifier algorithm. However, which diseases are correctly detected and classified by the proposed model is not mentioned. In addition to that, the researchers used only 480 image datasets for conducting their study which is a small amount for building a machine learning model.

Detecting and classifying maize leaf as gray leaf spot, common rust, northern leaf blight, and healthy using machine learning is proposed by [62]. Based on this experimentation total number of 3823 instances were collected from the public machine learning repository. Based on this, Python tools were for preprocessing the instance, and different classification algorithms such as SVM, Naïve Bayes, KNN, Decision tree, and RF are implemented. From the collected data 90% were assigned for training and the remaining 10% for testing. Accordingly, the developed classifier model achieves an accuracy of SVM 77.56%, Naïve Bayes 77.46%, KNN 76.16%, Decision tree 74.35%, and RF79.23%. However, in this study, there were imbalanced datasets between classes.

In [100], the researchers proposed detection of the three common rice leaf diseases namely: Bacterial blight, Brown spot, Rice blast by using image processing. A total number of 60 images were collected from the public image database. In this study, different processing techniques such as remove background and irrelevant portion, cropping the small region of leaf were done using the MATLAB tool. In addition, Naïve Bayes classification algorithms are implemented. The researchers have used the percentage split test option 75% of the data are assigned for training and the remaining 25% for testing. Therefore, for each rice leaf disease the developed classifier model achieves an accuracy of 89%, 90%, and 90% for Rice blast, Bacterial blight, and Rice brown spot respectively. But the researchers used only 60 image dataset for conducting their study which is too small for building a machine learning model.

In paper [101], the authors develop a deep learning model for the detection of leaf disease for rice crops. The model classifies the diseases into Rice stacburn, Leaf Scald, Leaf Smut, Rice Blast, Rice Kernel Smut, Rice Stem Rot, and so on. The required data were collected from two different sources; 600 images are collected from the public UCI machine learning repository and 500 images were captured from the agricultural field. Hence, the MATLAB tool used for preprocessing the collected image and also different pre-trained classification algorithms such as VGGNET-19, ResNet50, DenseNet-201, Inception-V3, and VGG19-SVM are implemented. The proposed architecture has a fully connected dense layer and 3 convolution layers. The first convolution layer had 28 filters, the second layer had 56 filters and the third layer had 112 filters. Rectified Linear unit (ReLu) activation function is used for all layers as well as researchers used filter size 3×3 for all layers. They used only one Maxpooling layer and had a 1×1 pool size. The experimental findings on the proposed model are an average accuracy of 98.63% was achieved.

Detecting and classifying maize leaf disease using CNN along with machine learning proposed by [102]. The model classifies the leaf diseases as Cercospora, common rust, northern leaf blight as well as healthy. For their experimentation, the required data was collected from a public image database, a total number of 200 images were collected. Towards creating the classifier model python was used as a tool, and KNN, SVM, and decision tree classification algorithms were implemented. The 10-fold cross-validation was used for training and testing data. In this study, they used seven CNN architectures (AlexNet, VGG16, VGG19, ResNet50, ResNet110, GoogleNet, and Inception-V3). While AlexNet architecture has twenty-five layers: Input layer, 5 convolutional layers, the first convolutional layer has an 11×11 filter, the second layer has a 5×5 filter, and the third, to fifth layer has 3×3 filters. 7 ReLU activation function layer layers, 2 normalization layers, 3 max-pooling layers, 3 fully connected layers, 2 dropouts 0.5, Softmax, and output layer. Finally, the developed classifier model achieves an accuracy of SVM 82.7%, KNN 88.33%, and decision tree 74.51%. However, they only used 200 image datasets for conducting their study which is too small for building a machine learning model.

In paper [103], the proposed model identifies corn leaves diseases using a convolutional neural network. The researchers focused on three types of diseases namely: Common Rust, Late Blight, and Leaf Spot. The necessary data were collected from UCI public machine learning repository; a total of 2000 images are collected. Different processing techniques were applied using the python tool, and Softmax, Random Forest, and SVM classification algorithms are implemented. The researchers used four pre-trained convolutional neural network architectures, VGG-16 with 8 layers, ResNet-50 with 50 layers, Inception-v2 and MobileNet-v1. For the experimentation 10-fold cross-validation testing option was implemented. Finally, the proposed model accuracy of 98% with precision 97%, recall 98% and f1-score 97% achieved.

In paper [104], the researchers developed a model for the detection and classification of maize leaf diseases. The model classifies the leaf disease as Common rust, Gray leaf spot, Northern leaf blight, and Healthy leaves. For the experimentation, 3852 images were collected from the public image database. Hence different processing techniques were performed using the Python tool. A deep convolutional neural network (CNN)-based architecture (Modified LeNet) has been used for classifying the leaf diseases. CNN LeNet architecture has a five-layer network consists of 2 convolutional layers, 3 fully connected layers, 2 * 2 Max pooling, and ReLU activation function to backpropagate the errors. The learned way of this model is through a back-propagation algorithm. The researchers were used the percentage split test option 80% of the data are assigned for training and the remaining 20% for testing. After all, the result shows LeNet performs an accuracy of 97.89% achieved.

In paper [105] the researchers proposed an automatic identification of plant diseases by using CNN. The necessary data have been collected from a public machine learning repository and captured using different mobile phones. In this study, different classification algorithms such as Baseline Triple network and Deep Adversarial Metric Learning (DAML) are implemented. Accordingly, three CNN models were used namely, ResNet18, ResNet34, and ResNet50. All residual architectures have the same structure but have a different number of blocks and layers, they used a fully connected layer with ReLU activation function. All the models were trained using the entire dataset resource each model was fine-tuned small set of target images ranging from 5 of 50 increments by 5. The evaluation demonstrates shows that transfer learning using the baseline model achieves an accuracy of 99% for the data that are collected from a public machine learning repository and 81% of accuracy for the dataset that is captured under different conditions. However, there is a limitation in the collected datasets such as illumines light, shadow, and background image.

Recognizing plant disease using multichannel CNN is proposed by [106]. For the experimentation purpose, the necessary data were collected from UCI. Towards creating a classifier model different classic CNNs architecture are used namely AlexNet and GoogleNet are implemented. The designed Multichannel CNN (M-CNN) architecture has two input channels, each channel receives one different type of dataset, and late-fusion techniques were used to merge separated channel networks in the fully connected layer. To handling, overfitting in neural networks was drop-out layer was inserted between the pooling layer. The researchers used the percentage split test option 80% of the data are assigned for training and the remaining 20% for testing. After all, M-AlexNet V1 has achieved an accuracy of 99.59% and M-GoogleNet V1 achieved an accuracy of 99.55%. Even this study tries to develop a model for 14 plant species. However, they didn't incorporate sorghum leaf diseases.

Leaf disease detection and classification using neural networks is proposed by [107]. The proposed model is classifying the images as affected and unaffected leaves. For this, the necessary 169 images were collected by the researchers. Towards creating a classifier model Backpropagation Neural Network classification algorithm is implemented. Hence to handling the data different processing techniques were used. After all, the developed classifier model achieves an accuracy of ranges between 88% and 92%. However, the researchers didn’t specify which plant leaf is used for conducting this study. In addition, the researchers used a small number of datasets.

Recognition and classification of maize leaf diseases using convolutional neural networks are proposed by [108]. The proposed model recognizing three different types of diseases namely: Northern Corn Leaf Blight, Common Rust, and Gray Leaf Spot, out of healthy leaves. The necessary data collected by researchers using Samsung smartphone camera in the maize field, the total number of 300 images are collected. Towards creating a classifier model, Python 3.3 was used as a tool, and CNN for classification algorithms was implemented. From those images, 70% of assigned for training, and the remaining 30% were for testing. Finally, the developed classifier model achieves an accuracy of 92.85%. However, the researchers used a small number of datasets.

2.14.1 Executive summary

As observed in the above studies in [24] [25], were carried on sorghum leaf disease classification and studies in [26], [62], [100], [102], [103], [104], [106], [107], [108], were carried on rice leaf disease using machine learning. However, the researchers didn’t include major sorghum leaf diseases such as anthracnose and rust, for their extermination they test the model with a single classifier, and also, they used a small amount of data.

Using image processing and machine learning techniques selecting the best minimum set of features is a challenge especially when more variations exist in the data [109]. According to [74] using manual feature extraction can reduce the classification tasks because the researcher specifies restrictions on what features represent the input data. In addition, the author in [17] stated that this learning approach (i.e., machine learning) has some limitations like the machine wants to see the object, but can’t see the object due to occludes by other objects, in this case, this technique can’t properly extract features, mostly called occlusion. And also, deformation limitation, when the images are captured in under various positions and lastly lighting condition. Because of this, the detection of these features is really difficult.

Therefore, in this study, the researcher collects a greater number of datasets collects to develop an effective model, this study also includes basic sorghum leaf diseases such as anthracnose and rust. In addition, to remove the stated limitation related to machine learning techniques the researcher used a convolutional neural network (CNN). Because this technique avoids hand-crafted feature extraction rather automatically extracts the necessary features from the given input images. Also, the presence of a large amount of dataset suitable candidates for the current application.

The work was done in [101], [103], [105], a deep learning-based via CNN, however, the researchers used only transfer learning (pre-trained model) classifiers, the other classifier will obtain better performance. Whereas, in this study, the researcher uses two CNN algorithms namely: training from the scratch and transfer learning.

CHAPTER THREE
METHODOLOGY

3.1. Introduction

These sections give a detailed discussion about the selected research methodology, the proposed framework and its process, the developed classifier model with transfer learning and training from the scratch, and finally the performance evaluation metrics are discussed.

3.2. Research methodology

In this research work, the researcher follows design science research methodology. This approach aims to create and evaluate the artifacts to solve the identified problems by enabling the transformation of the current state into the desired state. It is increasingly recognized as an equal companion to Information System (IS) behavioral science research and the novel paradigm that can tie all the other Information System paradigms together [110].

Hence, design science research is so far another lens or set of analytical techniques and perspectives for performing research. The design-science paradigm has its roots in engineering and the sciences of the artificial. It seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [111].

Based on this the researcher has some reasons why DSR is chosen to conduct this study: firstly, this methodology mainly emphasizes developing knowledge and/or solution, in the building and application of the designed artifact that supports practically the specialist can use to solve the real-world problem [32]. Secondly, DSR centered on practical problem solving, includes prescriptive or solution-oriented knowledge where the outcome from scientific justification (such as predicting) can be used in designing solutions to complex and relevant field problems [110] [112]. Finally, DSR targets attaining knowledge and understanding of a problem and develop a solution for the needs of the business and its environment in the field of information systems (IS) [32].

This study follows DSRM. DSRM has its process model that we follow in conducting this study. There are many DSR process models recommended by different researchers. Among them for this study, Peffers DSR process model is chosen, because other process models such as Archer, Takeda, Nunamaker, and others miss the important phase like objectives of a solution, demonstration, evaluation, and communication. However, Peffers process model is complete and it improves the suggestion of different process models by different scholars. Figure 3.1 shows that the conceptual framework of the Peffers design science research methodology.

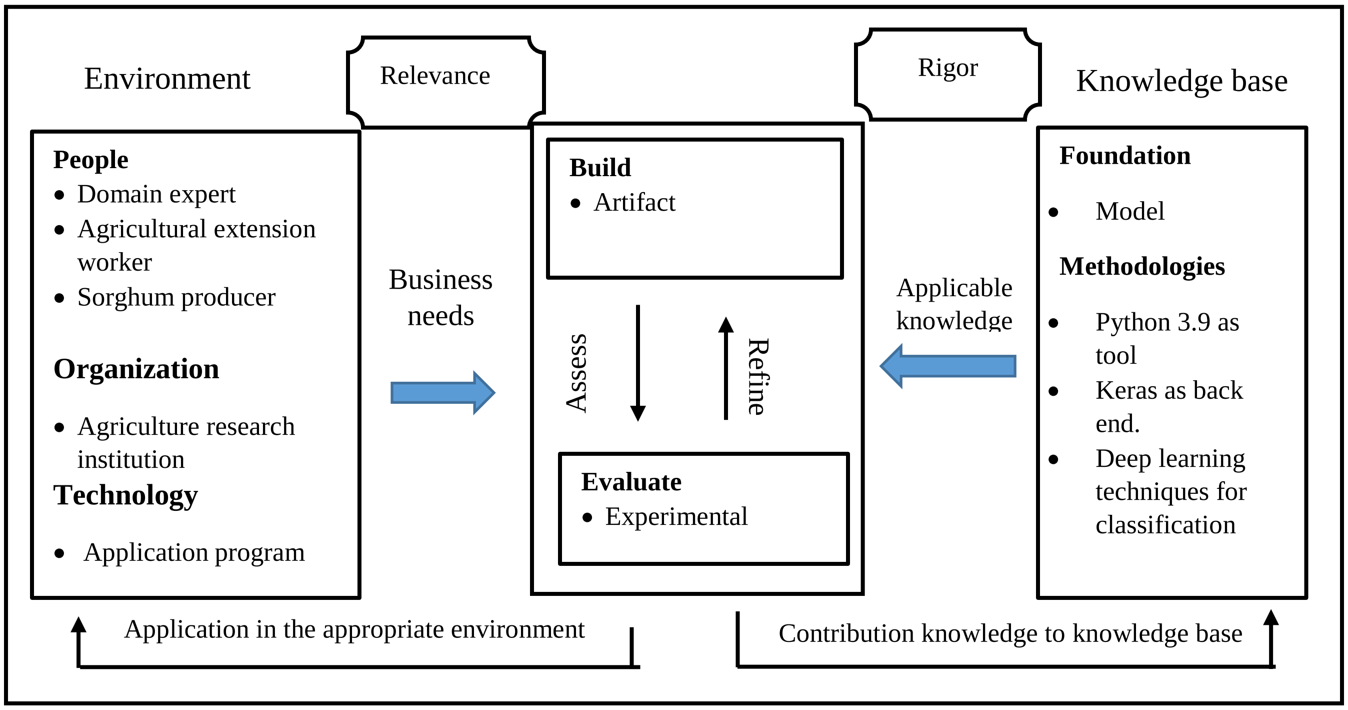


Figure 3. 1 Design Science Research Methodology adopted from [112]

As shown in Figure 3.1, this study is grown from relevance by identifying the problems that exist in the environment which is consists of people, organization, and technology. On the other hand, rigor provides appropriate applicable knowledge. The provided applicable knowledge allows to build and evaluate the artifact activities of the design itself. Therefore, the above framework is iterative until the developed artifact are added to the knowledge base thus provide the basis for future research and the designed artifact satisfies the business needs in terms of ability. This framework has six steps namely: problem identification and motivation, the objective of the solution, design and development, demonstration, evaluation, and communication. Each design step will discuss in the next section.

3.2.1. Problem identification and motivation

In this phase, the researcher has been reviewed different literature and related works previously done related to this research work, to identify practical problems in an actual application environment and also the researcher informally interviewed farmers and agricultural extension workers to know the gap between existing disease identification and the state-of-the-art. After identifying and formulated the gaps, the researcher was inspired to fulfill the gap by designing an artifact.

3.2.2. The objective of the solution

The objectives of a solution are inferred from the problem definition. The objectives of the study that are inferred from the problem specification are explained. Various resources have been reviewed to know the state of the problem, the state of current solutions, and their efficacy. Besides the developed artifact is mainly used by a domain expert. Based on this, the study is iterative until it satisfies the users in terms of ability.

3.2.3. Design and development

In this section, designing the actual artifact using a deep-learning approach, among deep-learning approaches CNN has been chosen for different reasons. Keras (using TensorFlow as backend) is used for designing the CNN model. Python is used for writing the required source codes. Keras is a modular neural network library written in Python capable of running on top of TensorFlow (TF). In addition, the researcher develops a web-based graphical user interface using Flask. Flask which is a python framework for web application development, HTML, CSS, and JavaScript were used as backend (Flask) and frontend (HTML, CSS, and JavaScript).

3.2.4. Demonstration

The developed system is demonstrated by simulating how the developed system detects and classifies sorghum leaf disease. Window 10pro 64-bit Operating system, Python development environment, is used to implement the model of the system. The window is an open-source cross-platform IDE for scientific programming in Python.

3.2.5. Evaluation

The developed system is evaluated to measure how well it supports a solution to the problem. To evaluate the system in a rational method, testing datasets were fed into the developed model. Subsequently, the developed artifact evaluate through different performance evaluation metrics such as precision, recall, F-score, and accuracy.

3.2.6. Communication

In this section, the researcher presents a research work report for the defense to get feedback and to evaluate the importance and effectiveness of the designed artifact.

CHAPTER FOUR
SYSTEM DESIGN

4.1 Introduction

In this chapter, detailed description of the proposed system or model for the classification of sorghum leaf disease is discussed. It requires to pass via a series of steps starting from preprocessing of images, feature extraction and learning to classification into predefine classes. Classification mainly encompasses two major phases; these are training phase and testing phase. In section 4.2, general description about the proposed system architecture is presented.

4.2 Proposed Framework

The proposed system architecture for sorghum leaf disease consists of four components, namely: image preprocessing, feature extraction, loading the model, and data augmentation. The system architecture for the proposed model is diagrammatically depicted in Figure 3.2. The first component of the proposed model is image preprocessing. This component performs tasks including image resizing and removing noise. In addition, this component does the initial task of making the input image ready for the feature extraction component. In feature extraction, the researcher used Gabor filtering to extract texture features. The third component of the proposed architecture is loading the CNN pre-trained model. In this study, two CNN models were used namely: training from scratch and transfer learning. Following this, the next component is data augmentation, this technique applies during training to make the system see a given image in a different dimension and to overcome model over-fitting. Finally, the convolutional neural network classifier models are trained and create a model. Therefore, the developed model can classify the leaves as anthracnose, healthy, leaf blight, and rust.

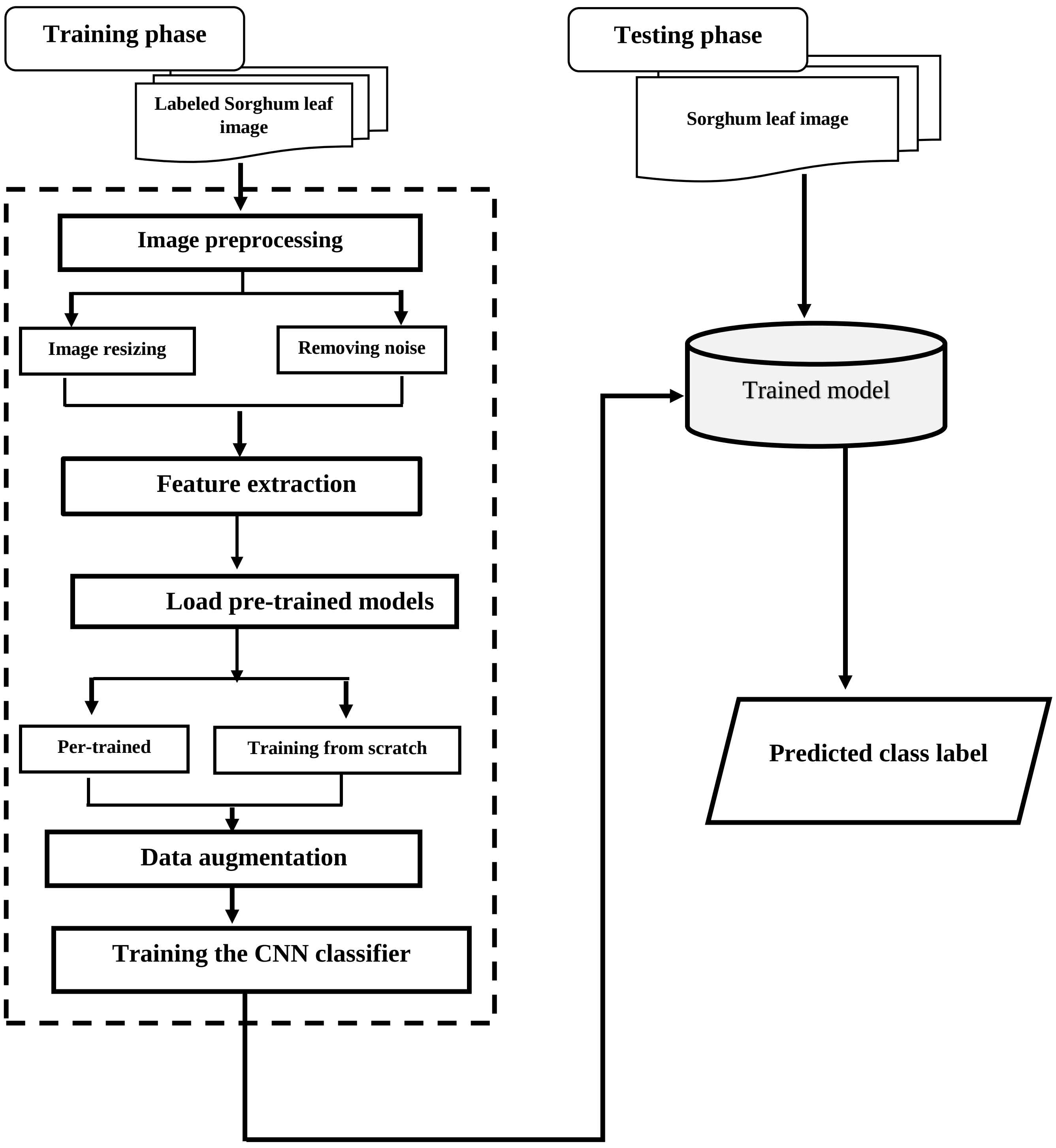


Figure 4. 1 Proposed architecture

Generally, this architecture shows the overall process followed to classify a particular input image in either of four classes. As shown in figure 3.1, there are two separate phases in the proposed architecture. The training phase and the testing phase with a slight difference in appearance. The training phase starts from importing a lot of input images, which means several images are started to be processed at a given time in one after the other manner. In the testing phase, a single image is imported for the process. After image is imported the training phases pass through a different process like preprocessing, feature extraction, loading the pre-trained, and data augmentation techniques then it outputs the trained model. Besides, The testing phase provides a SoftMax classifier for labeled returns from a model that is trained previously.

4.3 Image Acquisition

In this study, the image data samples were taken from Shewarobit wereda kobo village, North Shewa zone and from each class of sorghum leaf images, 1000 samples were taken. Accordingly, the mobile phone Samsung J8 camera was used in automatic mode with autofocus and the camera was approximated to 40 cm above the surface of the leaf. Besides, All the taken images are in JPEG (Joint Photographer Expert Group) file format. During the image acquisition, both the front and back sides of the leaf images are taken. The images were manually labeled to the Anthacranose, Healthy, Leaf blight, and Rust with the help of domain experts. To avoid the external effects of sunlight and other environmental conditions as much as possible the researcher was taken the images under the same controlled environment. The total number of samples collected for this study is presented in the following Table 3.1.

Table 4. 1 Number of collected datasets

Image type	Number of datasets
Anthracnose	1000
Healthy	1000
Leaf blight	1000
Rust	1000

Table 3.1 shows that the total number of samples that were collected for this study experiment. Therefore, 1000 images were collected for Anthracnose leaf disease, 1000 leaves for healthy,

1000 leaves for leaf blight disease and the remaining 1000 has assigned for the rust. Therefore, a total of 4000 images were collected and utilized for this study.

4.4 Image Preprocessing

In the preprocessing task, the input image passes through different filtering techniques to get a quality image. The preprocessing task is the basic step in image processing application that can help to get a more meaningful interpretation of an image. In addition, the preprocessing tasks can affect the overall performance of a classifier model. Consequently, in this classifier model the following sub-tasks are can be used one or more times during the process of implementation.

4.1.1 Image resizing

Image resizing is the first image pre-processing task. Image resizing defines preserving an important region of an image, minimizing distortions, and improving efficiency. Hence the state-of-the-art models take as input an image size of 224*224 [113]. Therefore, image resizing helps to reduce the training of time a neural network as more is the number of pixels in an image more is the number of input nodes that in turn increases the complexity of the model. This fixed size is used for all imported images because the accuracy of the feature extraction process is affected if the images are in different sizes. Algorithm 4.1 shows an algorithm that is implemented in this study to resizing an image.

```
Input: RGB image
    Begin
      For each Image in Dataset
            resized_image=resize (image, target_size= (height, width)
      return image
     End
Output: resized image
```

Algorithm 4. 1 Image resizing

4.1.2 Removing noise

During the acquisition of an image, using capturing media, noise is expected. The noise comes from the camera flash, environmental conditions. Noise refers to variation in an image [101]. Accordingly, for this study Gaussian noise is occurred. Hence, to remove this noise, the researcher has used the Gaussian filtering technique because for most noise that is appearing on the images, there are recommended filtering techniques that work better that is why the Gaussian method has been selected from the other filtering algorithms. So removing this noise from an image is an advantage of enhancing the quality of the input image, getting a maximum accurate result and good efficiency. The implemented algorithm for smoothing the images is shown in algorithm 4.2.

```
Input: RGB image
Begin
    Image= original image I
    For I in rage (0,3999)
        Read the RGB image
        Convert the RGB image to image in the gray color model
        Gaussian filter the components gray of the converted image
        Convert the image to RGB image
    return image
End
Output: Noise removed and enhanced image
```

Algorithm 4. 2 Algorithm for removing noise

The Gaussian filter works by using the 2D distribution as a point-spread function. This is achieved by convolving the 2D Gaussian distribution function with the image. We need to produce a discrete approximation to the Gaussian function. This theoretically requires an infinitely large convolution kernel, as the Gaussian distribution is non-zero everywhere. Fortunately the distribution has approached very close to zero at about three standard deviations from the mean. Gaussian kernel coefficients are sampled from the 2D Gaussian function

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where σ is the standard deviation of the distribution. The distribution is assumed to have a mean of zero. We need to discretize the continuous Gaussian functions to store it as discrete pixels.

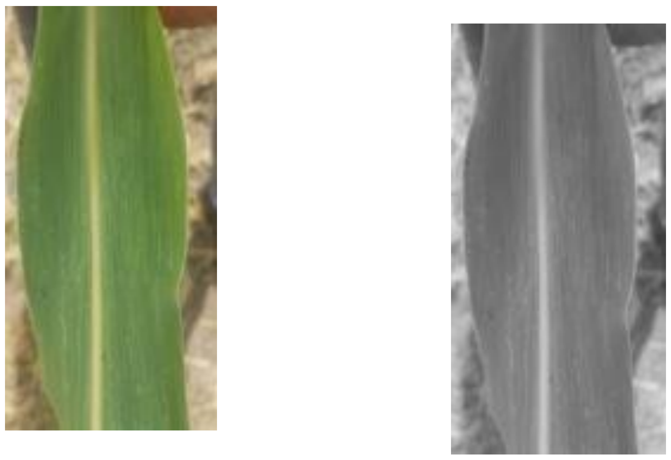


Figure 4. 2 Images (left) before Gaussian noise (right) after applying Gaussian filtering

4.1.3 Label Encoding

In this study, the images have collected four class labels namely: Healthy, anthracnose, rust, and leaf blight. However, the collected images are in the form of string. Based on this, for machine-readable and understandable these string values should convert in the form of numeric values. Table 4.2 shows the corresponding values of each class.

Table 4. 2 Label Encoding

Class	Numeric value
Anthracnose	0
Healthy	1
Leaf blight	2
Rust	3

4.1.4 One hot Encoding

After the labels are encoded, the next step is One-Hot Encoding. The One-Hot Encoding is used to avoid confusion of the model, that a model thinking a column has data with some sort of order. This (i.e., One-Hot Encoding) takes a column with categorical data and then splits the column into multiple values. Therefore, the numbers were replaced by 1s and 0s, depending on which column has which value, a particular class has a value of one and the remaining have zero. Table 4.3 describes the encoded image concerning their values.

Table 4. 3 One-hot encoding

Anthracnose	Healthy	Leaf blight	Rust
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

4.5 Feature extraction

As is described in Chapter 2 Section 2.7.4, images have different features like shape, texture, and color features. These features can be used to represent an image. For this study, texture features are used to represent an image for the identification of the normal and diseased sorghum leaf. The researcher used these features because the texture feature of an image can differentiate the different categories of the disease. To extract the texture feature of an image the researcher used GLCM and Gabor filtering. These extraction techniques are one of the most popular techniques for texture feature extraction or analysis [69]. Textural features within images can be used for the accurate detection and classification of images. For extracting textural features (orientation and frequency) from the image, a set of Gabor filters with different frequencies and orientations are used.

```
Input: RGB image
Begin
    // Initialize an array, which holds the generated filters. e.g: filters = []
    // initialize the kernel size so that one filter can cover. e.g: ksize = 31
        for θ in range of (0, pi, pi / 16) // pi = 3.14
            // initialize or assign a value to the parameter sigma; θ; lambda; gamma; and psi
            // example: sigma = 4.0; lambda = 10.0; gamma = 0.5; psi = 0;
            // generate list of Gabor kernel
              kern = getGaborKernel(params)
              filters = kern //Assign or append kern to filters array
          Apply the filters list on image I
        Return image filtered by the filter list, filters
End
Output: image filtered by the generated filter list
```

Algorithm 4. 3 Generating and applying filters on image

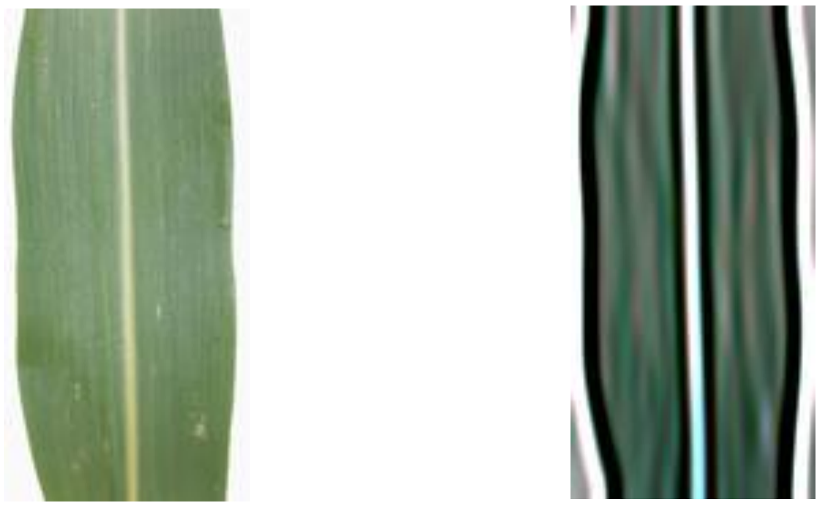


Figure 4. 3 Left: original image Right: image on the left applied with Gabor filter

4.6 Data augmentation

For this study, the data augmentation technique has been applied to the collected images, to generate a larger dataset and more varied for training. To do this, Keras deep learning neural network library has been utilized to transform, randomly shift, randomly rotate, and randomly scale and it provides the capability to fit models via the ImageDataGenerator class The ImageDataGenerator built-in function in the python library is used for experimenting with the data augmentation technique. To augment the input image the following algorithm is applied:

```
Input: preprocessed image
Start
      Initialize rotation range,
      Width shift range,
      Height shift range,
      Shear range,
      Zoom range,
     Horizontal flip,
     Vertical flip
Return aug; //augmented images
End
```

Algorithm 4. 4 Data augmentation

During augmentation, images are not stored in a disk and don’t require storage memory rather the transformed images are generated at run-time. This technique will overcome the over-fitting problem and also increase training performance. In addition, Data augmentation is a regularization method used to generate additional training data and improve the model’s ability to generalize, using this the model to see new, slightly and modified forms of the input image and the network can learn more robust features.

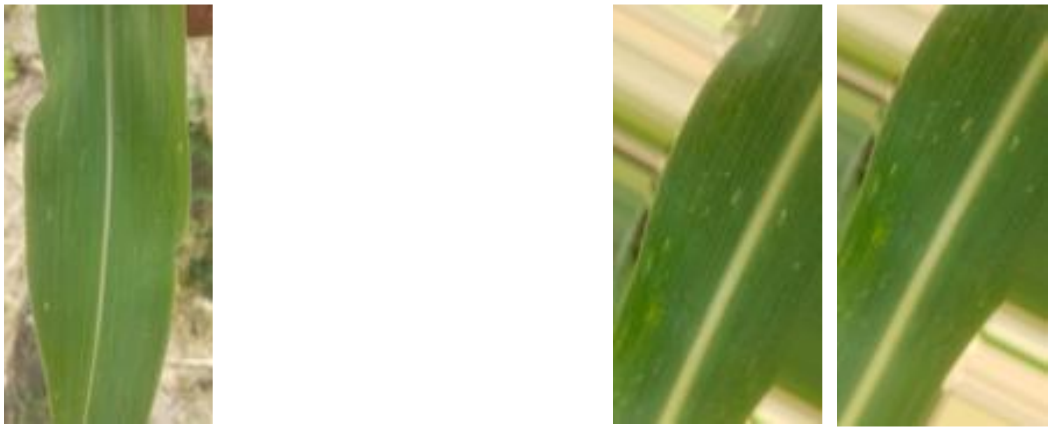


Figure 4. 4 Original images (Left) augmented image (right)

4.7 Normalization

Normalizing is also used to pre-process an input image before further processing. Normalization is used to scale down the feature values in the range between 0 and 1. Image pixel values are an integer between the ranges of 0 to 255. Although these pixel values can be presented directly to the model, they can result in slower training time and overflow. Overflow happens when numbers get too big and the machine fails to compute correctly. Using algorithm 4.5 the researcher normalizes the data values down to a decimal between 0 and 1 by dividing the pixel values by 255.

```
Input: resized image
Start
    Divide_By=255.0
    For each data in dataset
    Image=float32(image)
    Image=image/Divide_By
End
Output: normalized image
```

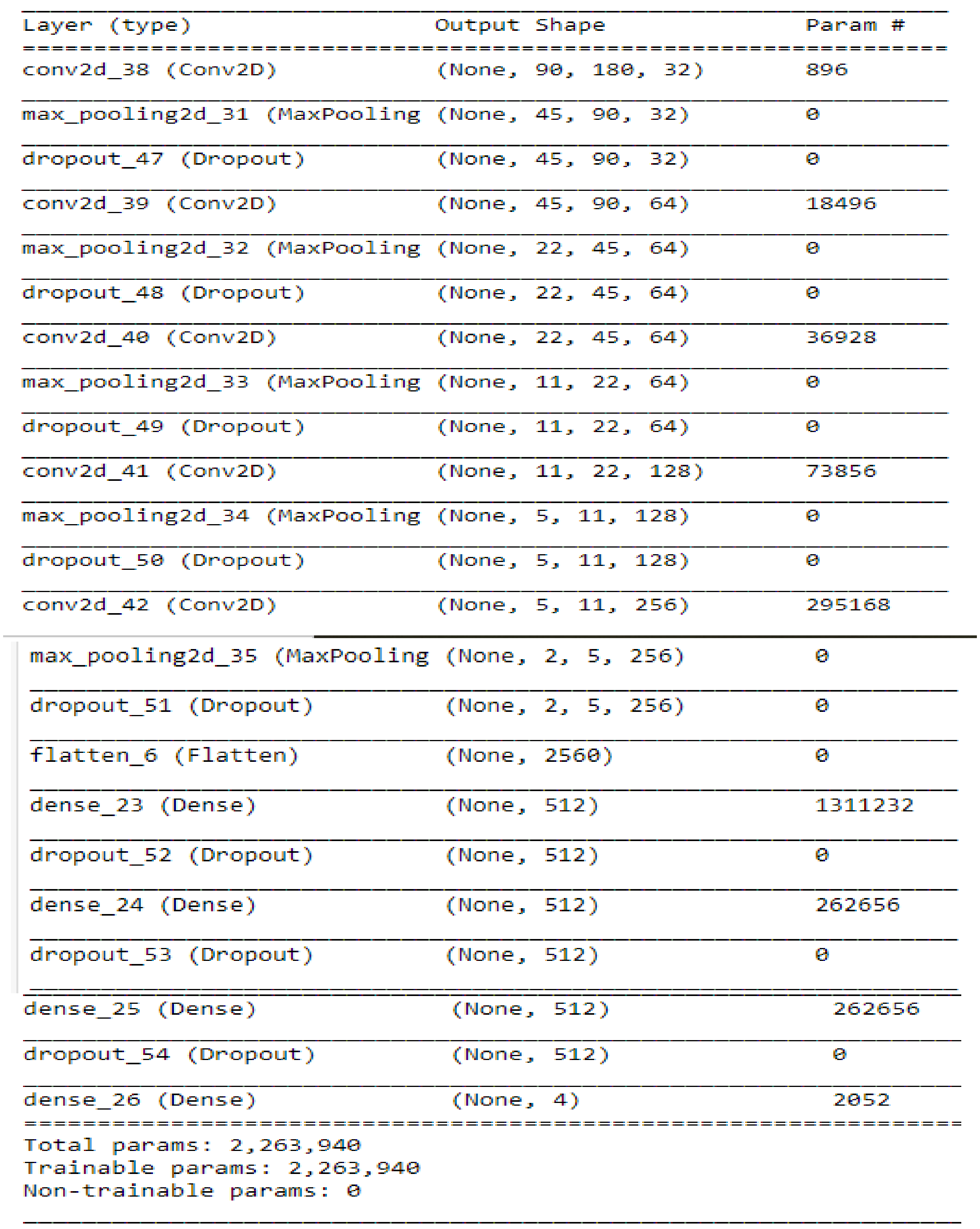
Algorithm 4. 5 Algorithm for normalizing an image

4.8 Loading the classifier model

As stated in chapter one section 1.6.5, in this study two methods are developed training from the scratch and transfer learning. In the next sub-section discuss in detail.

4.8.1 Training the model from the scratch

As stated in chapter one section 1.6.5, in this study two methods are developed training from the scratch and transfer learning. The proposed model (i.e., training from the scratch) consists of three layers namely: convolutional layers, max-pooling layer, and activation function and fully connected layers. The input to the first convolution layer is 224 x 224 x 3 images. The convolution operation requires four parameters. The first parameter is the number of filters that are used to control the depth of the output volume. In this model, the researcher has used 32, 64, 64, 128, and 256 filters. Different numbers of convolution layers and different numbers of filters are tested and those that achieve higher accuracy are selected. The second parameter is the receptive field size, which determines the size of each filter (kernel) and is nearly always square. The researcher has used 3 x 3, and 5 x 5 filter sizes at a single layer. A characteristic feature that cannot be detected by a 3 x 3 filter size can be detected by a filter size of 5 x 5. The third parameter is stride size, which determines the number of pixels skipped (horizontally and vertically) each time we make a convolution operation. So, the researcher has used a stride size of one (1, 1). The last (fourth) parameter is the amount of zero padding, which is used to control the size of the output. The researcher has used “same” padding, which means the size of the output is equal to the size of the input if the stride size is one.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_38 (Conv2D)           (None, 90, 180, 32)       896
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 45, 90, 32)        0
_________________________________________________________________
dropout_47 (Dropout)         (None, 45, 90, 32)        0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 45, 90, 64)        18496
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 22, 45, 64)        0
_________________________________________________________________
dropout_48 (Dropout)         (None, 22, 45, 64)        0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 22, 45, 64)        36928
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 11, 22, 64)        0
_________________________________________________________________
dropout_49 (Dropout)         (None, 11, 22, 64)        0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 11, 22, 128)       73856
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 5, 11, 128)        0
_________________________________________________________________
dropout_50 (Dropout)         (None, 5, 11, 128)        0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 5, 11, 256)        295168
_________________________________________________________________
max_pooling2d_35 (MaxPooling (None, 2, 5, 256)         0
_________________________________________________________________
dropout_51 (Dropout)         (None, 2, 5, 256)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 2560)              0
_________________________________________________________________
dense_23 (Dense)             (None, 512)               1311232
_________________________________________________________________
dropout_52 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_24 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_53 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_25 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_54 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_26 (Dense)             (None, 4)                 2052
=================================================================
Total params: 2,263,940
Trainable params: 2,263,940
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. 5 Proposed CNN classifier model

Figure 4.4, shows that the architecture of the proposed network consists of convolution, pooling, and activation functions. Accordingly, In the proposed method the researcher has used the LeakyReLU activation function as described in chapter two section 2.9.1.1, it is an extension of ReLU activation function, in ReLu activation function if the neuron value is negative it changes into 0 irrespectively, whereas in LeakyReLU if the neuron has negative it believes that neuron contributes the prediction of some action.

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate to reduce the losses during the training phase. Based on this, for this study, Adam optimizers with a 0.0001 learning rate value were implemented in the proposed method. As discussed in the previous chapter, Adam combines the advantages of two other extensions of stochastic gradient descent and solving vanishing gradient descent problems, this optimization is better computational than stochastic gradient descent and has higher convergence speed by the adaptive learning rate. The learning rate parameter defines how much the weights are updated after each epoch. To reduce overfitting, dropout regularization method is used at fully each convolution layer and fully connected layer. It enables multiple, redundant nodes to activate when given with similar patterns (inputs), which also helps our model to generalize. The value of the dropout layer is determined using experimentation hence researcher has started with the value of p (dropping probability) = 0.25.

To stable the network during the training phase of the model, there need to be normalized or standardize the input and adjusting the scale since it used to speed up the training process and allows each layer of the network to learn by itself, hence the researcher used a batch size of 64 and the network will train in 100 epochs. The method trained with a back-propagation, Back-propagation trainer trains the parameters of a module according to a supervised dataset by back-propagating the errors. SoftMax is often used as the activation for the last layer of a classification network because the result could be interpreted as a probability distribution. And finally, the researcher was used categorical cross-entropy to calculate the loss. Categorical cross-entropy calculates a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem. This loss function appropriates when there are two or more label classes, the labels to be provided in a one-hot representation.

4.8.2 Designing CNN models using Transfer Learning

In this study, VGG19, and VGG16 architectures are chosen from others pre-trained networks. These pre-trained networks are more effective for image classification and have high classification accuracy than other pre-trained models. These are the way to retrain the CNN networks, which are previously trained on more than a million images and able to classify images into 1000 object categories then classify new images. Whereas for this study the last layer and weights are updated to obtain the desired response. The different hyper-parameters and their values used for the pre-trained model are well shown in the following table.

Table 4. 4 Hyper-parameter description

Hyper-parameters	Values	Description
Optimizer	LeakyReLU	These are applied during training the model for the fully connected layer.
Batch size	64	
Epoch	100	
Loss function	Categorical cross-entropy	

4.9 Classification

Classification performs classifying the sample images of sorghum leaf into their types of disease (i.e. normal, anthracnose, leaf blight, and rust) based on the features of the image. It is done after the distinctive features are extracted and learned. Based on their origin using SoftMax classifier into four classes.

4.10 Performance evaluation

After building the classifier model the upcoming process is to evaluate the effectiveness of the models. As described in chapter 2 section 2.13, the performance of the models is evaluated by accuracy, precision, recall, and FP rate.

CHAPTER FIVE
EXPERIMENT AND RESULT DISCUSSION

5.1 Introduction

In this chapter, the experimental evaluation of the proposed model for automatic classification of sorghum leaf disease is described in detail. The dataset used and the implantation of the proposed model are thoroughly described. The effect of the Gabor filter is evaluated and compared with CNN. In addition, the test results are presented and compared with other researchers' work.

5.2 Description and Preparation of Dataset

The process used for getting data ready for the classification model can be summarized in: collect dataset, preprocess data, extracting features and transform data. This process is iterative with many loops to prepare the dataset required. The datasets were partitioned randomly into training and testing sets. In the training, the dataset is repeatedly presented to the pattern recognizer, while weights are updated to obtain the desired response, and the testing dataset is then provided a standard used to evaluate the model performance. For this experiment, the researcher used a percentage split test option such that splitting the dataset into training and testing. Therefore, 70% of the data is assigned for training the model, and the remaining 30% is assigned for testing. Assigning 2/3rd of the dataset for more than 100 images to training is close to optimal for reasonably sized datasets [114].

5.3 Building the Model

For this study, the researcher used the Jupyter IDE (Integrated Development Environment) coding tool for writing, testing, and debugging our code more easily. The program is done using Python 3.9 language with OpenCV and deep learning framework, Keras. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. On the other hand, Keras is a built-in python, it provides high-level user-friendly APIs for building and training the models. And finally, the research design graphical user interface using flask. In this study, the classification of a sorghum image based on its leaf has developed using one of the deep learning-based approaches with convolutional neural networks. Besides, the researcher applies two distinct convolutional neural network methods were applied to develop classification models namely training from scratch and transfer learning. The experimentation is done on Intel Core ™ i5-5200 CPU, 8 GB of RAM, and a 64-bit window operating system.

5.4 Experimental Analysis and Result

In this study, the researcher has conducted four experiments. The model is trained for 100 epochs, a batch size of 64, and a starting or initial learning rate of 0.001 (1e-3).

5.4.1 Experiments on transfer learning

In the process of model building, the first CNN architecture was using training from scratch (i.e. the researcher setting different hyper-parameters into the network from scratch as discussed in chapter three section 3.9.1). Transfer learning technique (i.e. through-loading VGG16 and VGG19 pre-trained model), then fine-tuning these networks, finally build a model.

5.4.1.1 Experiment I using VGG19 (Visual Geometry Group19)

The first selected transfer learning model is VGG19. VGG19 is a variant of the VGG model which in short consists of 19 layers (16 convolution layers, 3 fully connected layers, 5 MaxPool layers, and 1 SoftMax layer). The model uses 3*3 kernel size with stride size of 1, spatial padding was used to preserve the spatial resolution of the image, 2*2 max-pooling with the stride of 2 followed by a Rectified linear unit (LeakyReLu). Implemented three fully connected layers from which the first two were of size 4096 and after that, a layer with 1000 channels and the final layer is a softmax function. However, for this study, the last layer was changed into 4 classes. This model has around 57 million trainable parameters. When training the model the datasets split into 70% for training the model and 30% for testing the model. So, the researcher trained the model with 2800 sample images and 1200 sample images for testing the trained model. Whereas in this experiment the misclassification of the model between each class label has also a little different. As the researcher observed in this experiment also model overfitting occurred and the researcher tried changing different overfitting methods and their values and gotten a little change. The following tables show that the confusion and matrix and classification report of the VGG19 model.

Table 5. 1 Confusion matrix of VGG19

		Predicted class				
Actual class	Diseases type	Anthracnose	Healthy	Leaf blight	Rust	Total
	Anthracnose	**270**	2	8	20	300
	Healthy	3	**280**	13	4	300
	Leaf blight	5	0	**265**	30	300
	Rust	10	1	4	**285**	300
	Total	288	283	290	339	**1200**

As it is shown in table 5.1, 270 images are correctly classified as anthracnose disease, while 30 images were incorrectly classified as healthy, leaf blight, and rust disease. Besides, 280 images were correctly classified as healthy, and the remaining 20 images were incorrectly classified as Anthracnose, leaf blight, and rust disease. Also, the model 265 images were correctly classified as Leaf blight disease and 35 images are incorrectly classified as Anthracnose and rust disease. Finally, 285 images are correctly classified as Rust disease while the remaining 15 images are incorrectly classified as Anthracnose, healthy, and late blight disease.

Table 5. 2 Classification report of VGG19

Class	**Performance metrics**			
	Precision	**Recall**	**F1-score**	**False Positive Rate**
Anthracnose	0.93	0.90	0.91	0.1
Healthy	0.98	0.93	0.95	0.07
Leaf blight	0.91	0.88	0.89	0.12
Rust	0.84	0.95	0.89	0.05
Weighted avg	0.91	0.91	0.91	0.08

Table 5.2 depicted that the developed classifier model was able to correctly classify 90% of anthracnose disease the remaining 10% were incorrectly classified as healthy leaf blight and

rust disease. Besides the model correctly classified 93% as Healthy but the remaining 7% were incorrectly classified as anthracnose, leaf blight, and rust disease. Similarly, in the model, 88% were correctly classified as leaf blight disease, and 12% were incorrectly classified as anthracnose and rust disease. Finally, in the model, 95% were correctly classified as rust disease but 16% were incorrectly classified as healthy, leaf blight, and anthracnose disease. Therefore, the classifier model can correctly classify an accuracy of **91.5%.**

5.4.1.2 Experiment II using VGG16 (Visual Geometry Group16)

The first experiment, conducting with the VGG16 network. This model has 16 layers deep, the model trained on more than a million images from ImageNet datasets can classify images into 1000 objects. VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used the same padding and a max-pool layer of 2x2 filter of stride 2. In the end, it has 2 FC (fully connected layers) with the LeakyReLU activation function. In the last layer, a Softmax activation function is used for output. This model is pretty large and it has about 17 million trainable parameters. An experiment is conducting by splitting total datasets 70% of images are assigned for training, and the remaining 30% is for testing. As we can see in the following Table 4.1, in this experiment the misclassification of the model between each class label has a difference. As the researcher observed this indicates that the model well not recognize the leaf diseases image and also model overfitting occurred but the researcher tried changing different hyper-parameter to mitigate overfitting whereas nothing changed. In addition, the researcher also changes the percentage split test options to 80% for training and 20% for testing but nothing is changed. Table 5.3 shows the confusion matrix of the VGG16 model.

Table 5. 3 Confusion matrix of VGG16

		Predicted class				
Actual class	Diseases type	Anthracnose	Healthy	Leaf blight	Rust	Total
	Anthracnose	**249**	0	26	25	300
	Healthy	0	**256**	28	16	300
	Leaf blight	6	27	**261**	6	300
	Rust	6	0	5	**289**	300
	Total	261	283	320	336	**1200**

As it is shown in table 5.3, 249 images are correctly classified as anthracnose disease, while 51 images were incorrectly classified as leaf blight and rust disease. Besides, 256 images were correctly classified as healthy, and the remaining 44 images were incorrectly classified as leaf blight and rust disease. Also, the model correctly classifies 261 images as Leaf blight disease, and the remaining 39 images were incorrectly classified as Anthracnose, Healthy, and rust disease. Lastly, 289 images are correctly classified as Rust disease while 11 images are classified incorrectly as Anthracnose and leaf blight disease.

Table 5. 4 Classification report of VGG16

Class	**Performance metrics**			
	Precision	**Recall**	**F1-score**	**False Positive Rate**
Anthracnose	0.95	0.83	0.89	0.17
Healthy	0.90	0.85	0.88	0.15
Leaf blight	0.82	0.87	0.84	0.13
Rust	0.86	0.96	0.91	0.04
weighted avg	0.88	0.88	0.88	0.12

Table 5.4 depicted that the developed classifier model was able to correctly classify 83% of anthracnose disease the remaining 17% were incorrectly classified as leaf blight and rust disease. Besides the model correctly classified 85% as Healthy leaf and the remaining 15% were incorrectly classified as leaf blight and rust disease. Similarly, in the model, 87% were correctly classified as leaf blight, and 13% were incorrectly classified as anthracnose, healthy, and rust disease. Finally, the model 96% were correctly classified as rust, and the remaining 4% were incorrectly classified as leaf blight and anthracnose disease. In addition, the classifier model can correctly classify with an accuracy of **87.75%.**

Finally, in this experiment, the researcher applies different percentage split test options such as 80:20 and 70:30 to get the desired result. However, the better results were gotten in the 70:30 percentage split. Based on this the experimental result shows that VGG19 has gotten better accuracy than VGG16. This is because of different reasons. The first one is the number of their trainable parameters VGG19 has 57 million and VGG16 has 16 million parameters which have a big difference and this might affect the performance of their model. The second one, there was model overfitting in both models. Nevertheless, in VGG19 after changing the value of the parameter (i.e. the default value was 0.5 whereas the researcher changes the value into 0.25 up to 0.5) the model overfitting has decreased.

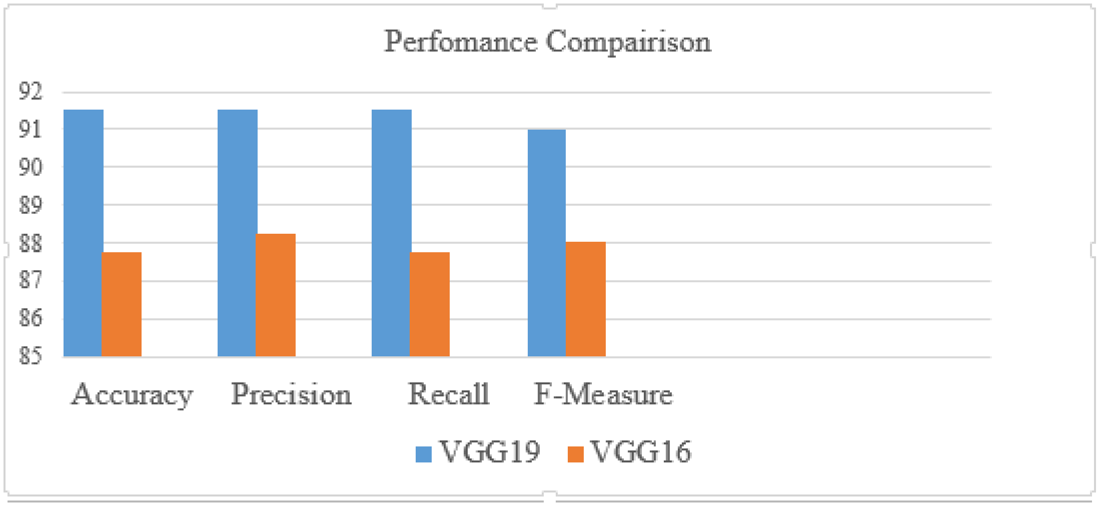


Figure 5. 1 Performance comparison of transfer learning

5.4.2 Experiments on training from scratch

The second method was training from scratch (i.e. the researcher setting different hyper-parameters into the network from scratch as discussed in chapter 3 section 3.3.9).

5.4.2.1 Experiment III before applying Gabor filtering

For this experiment, the researcher setting different parameters, and hyper-parameters were considered and applied as mentioned in chapter 3 section 3.9.2. The proposed convolutional neural network model experiment has been tested using percentage split, the whole dataset split into 70% of training and 30% testing. In this experiment, the misclassification of the model between each class label has also a little different. But as compared to transfer learning models the misclassification between each class is few. In addition to this, the researcher observed that in this experiment the occurrence of model overfitting is low. The following tables show that the confusion matrix and classification report of the proposed model.

Table 5. 5 Confusion matrix of the proposed model before applying Gabor

		Predicted class				
	Diseases type	Anthracnose	Healthy	Leaf blight	Rust	Total
Actual class	Anthracnose	**280**	0	13	7	300
	Healthy	2	**290**	6	2	300
	Leaf blight	5	0	**285**	10	300
	Rust	6	0	10	**284**	**300**
	Total	293	290	314	303	**1200**

As shown in Table 5.5, **280** images are correctly classified as anthracnose disease, while the remaining 20 images were incorrectly classified as leaf blight and rust disease. Besides, **290** images were correctly classified as healthy, and the remaining 10 images were incorrectly classified as Anthracnose, leaf blight, and rust disease. Also, the model correctly classified **285** images as Leaf blight disease, and the remaining 15 images are incorrectly classified as Anthracnose and rust disease. Finally, **284** images are correctly classified as Rust while 16 of the images are classified incorrectly as Anthracnose and late blight disease.

Table 5. 6 Classification report of the proposed model before applying Gabor

Class	**Performance metrics**			
	Precision	**Recall**	**F1-score**	**False Positive Rate**
Anthracnose	0.95	0.93	0.93	0.07
Healthy	1.00	0.96	0.97	0.04
Leaf blight	0.90	0.95	0.92	0.05
Rust	0.93	0.94	0.93	0.06
Weighted avg	0.94	0.94	0.93	0.05

Table 5.6 depicted that the developed classifier model was able to correctly classify **93%** of anthracnose disease the remaining **7%** were classified incorrectly as leaf blight and rust disease, besides the model correctly classified **96%** as Healthy leaf the remaining **4%** were classified as anthracnose, leaf blight and rust disease. Similarly, the model **95%** were correctly

classified as leaf blight, and **5%** were incorrectly classified as anthracnose and rust disease. Finally, the model **94%** were correctly classified as rust disease, and **6%** were incorrectly classified as anthracnose and leaf blight disease. Therefore, the developed classifier model was able to correctly classify an accuracy of **94.5%**.

5.4.2.2 Experiment IV after applying Gabor filtering

In the previous chapter, the researcher explained how the texture features were computed. After extraction of these features, the features were used to identify the different types of disease in the sorghum image. The researcher setting Gabor filter parameters such as lambda, gamma, psi, orientation, frequency, kernel size. Poorly set parameters impact the result of feature extraction. The more we have features that represent the leaf image, the more accurate our classification result will be. As clearly shown in the following tables the gap between the training accuracy and testing accuracy is much lower as compared to the previous experiments. This is due to the application of the Gabor filter, the model performs very well as compared to in the training and testing phase. Training accuracy is increased nearly linearly and passes over above 95 after epoch 60.

Table 5. 7 Confusion matrix of the proposed model after applying Gabor filtering

		Predicted class				
Actual class	Diseases type	Anthracnose	Healthy	Leaf blight	Rust	Total
	Anthracnose	**290**	0	2	8	300
	Healthy	0	**295**	5	0	300
	Leaf blight	5	0	**287**	8	300
	Rust	2	0	3	**295**	300
	Total	297	295	297	311	**1200**

As shown in Table 5.7, **290** images are correctly classified as anthracnose disease, while 10 images were incorrectly classified as leaf blight and rust disease. Besides, **295** images were correctly classified as healthy, and the remaining 5 images were incorrectly classified as leaf blight disease. Also, the proposed model **287** images were correctly classified as Leaf blight and 13 images are incorrectly classified as Anthracnose and rust disease. Finally, **295** images are correctly classified as Rust disease while 5 of images are classified incorrectly as Anthracnose and late blight disease.

Table 5. 8 Classification report of the proposed model after applying Gabor

Class	**Performance metrics**			
	Precision	**Recall**	**F1-score**	**False Positive Rate**
Anthracnose	0.97	0.96	0.96	0.04
Healthy	1.00	0.98	0.98	0.02
Leaf blight	0.96	0.95	0.95	0.05
Rust	0.94	0.98	0.95	0.02
Weighted avg	0.96	0.96	0.96	0.03

As shown in table 5.8 the developed classifier model was able to correctly classify 96% of anthracnose disease the remaining 4% were classified incorrectly as leaf blight and rust disease. Besides the model correctly classified 98% as Healthy leaves were the remaining 2% classified as leaf blight. Similarly, in the model, 95% were correctly classified as leaf blight, and 5% were incorrectly classified as anthracnose, and rust disease. Finally, in the proposed model 98% were correctly classified as rust, and 2% were incorrectly classified as leaf blight and anthracnose disease. Finally, after applying Gabor filtering the developed classifier model was able to correctly classify an accuracy of 96.75%.

Finally, in this experiment, the researcher applies different percentage split test options such as 80:20 and 70:30 to get the desired result. However, the better results were gotten in the 70:30 percentage split. Based on this the experimental result shows that the proposed model with Gabor filtering has gotten better performance accuracy than the proposed model without Gabor filtering. This is due to applying Gabor filtering application on the CNN classifier model might has a great effect on model performance.

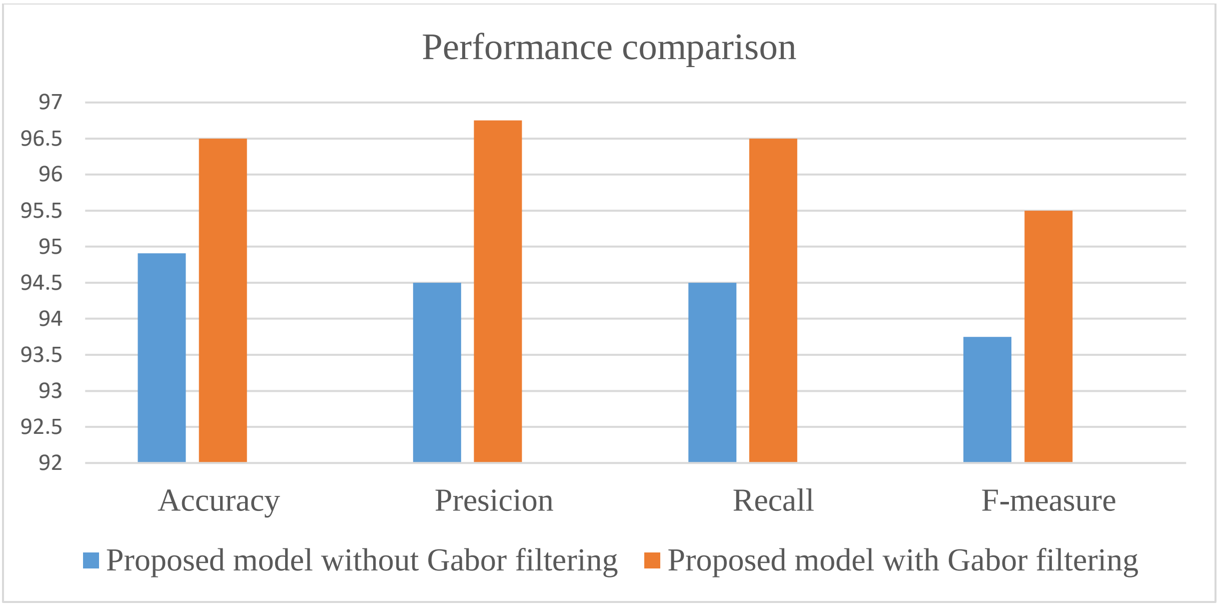


Figure 5. 2 Performance comparison of training from scratch

5.5 Summary of experimental result

As illustrated in table 5.9 this study has been done using both training from scratch and transfer learning. The table summarizes the result of the training and testing accuracy of the study. The model is trained for 100 epochs for each experiment with a 70:30 test option. The proposed model architecture was achieved better accuracy after applying Gabor filtering.

Table 5. 9 Comparison of all experiment

Performance metrics	Training from scratch		Transfer learning	
	Proposed CNN model before applying Gabor filtering	Proposed CNN model after applying Gabor filtering	VGG19	VGG16
Tested images	1200	1200	1200	1200
Accuracy (%)	94.5	**96.75**	91.5	87.75
Precision (%)	94.5	96.75	91.5	88.25
Recall (%)	94.5	96.5	91.5	87.75
F-measure (%)	93.75	95.5	91	88
False-positive rate (%)	5.5	3.5	8.5	12.25

As shown in table 5.9, the proposed model has 3.9 million parameters, which is much smaller than the VGG16 model, which has 17.3 million parameters, and the VGG19 model, which has 57.1 million parameters. The proposed model after applying the Gabor filtering algorithm achieves a training accuracy of 96.5%. In addition, the proposed model has no model overfitting compared to the pre-trained model and proposed model without Gabor filtering. On the other hand, in terms of test accuracy VGG16 and VGG19 achieve an accuracy of 91.5% and 87.75% respectively.

5.6 Comparing this Study with Related Works

The performance of the developed model of this study result has compared with the previously published method and results. Therefore, an attempt has been made to discuss the results of this study and comparison with previous work. As the researcher has discussed in the literature review in chapter two, there are two published studies were conducted on sorghum leaf disease classification. The discussion has focused on the major findings of previous works to compare the findings of this study. However, Study in the first row, the researchers used AlextNet with the size of 224*224 sorghum leaf images. The total number of datasets and the performance of the classifier model isn't mentioned. In the second row study, the researchers used machine learning approaches such as ANN, RF, and SVM. For their experiment, a total number of 1183 images were used. Whereas in this study the researcher used two CNN models namely transfer learning and training from the scratch. A total number of 4000 sorghum leaf images were used to conduct the study. Besides, the researcher resize the images into the input size of the CNN model which is 224*224. The model is trained with the LeakyReLU activation function with 64 batch sizes and 100 epochs. When training the model it takes two and half days with Dell Intel core i5 -5200U CPU. Table 5.10 shows the detailed results of the comparison.

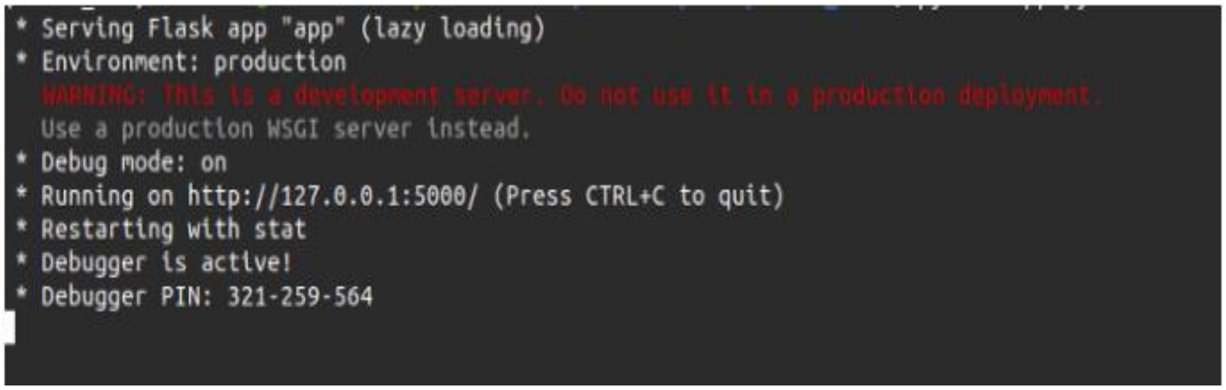
Table 5. 10 Comparison of the Proposed System with other researchers' work

No.	Author	Title	Proposed method	Accuracy	Remark
1.	Soni et al. 2020	Crop Yield Improvement Using Plant Leaf Disease Detection	AlexNet	-	
2.	Rahman et al. 2017	A comparative analysis of machine learning approaches for plant disease identification	Random forest ANN SVM	95.4%, 94.9, 80.5%	*
			Proposed method	**96.75%**	

Note the symbol (*) in Table 5.10 indicate the study in the second row the main aim of the researchers was trying to identify which features are better for classifying the diseases rather not for developing a classifier model. The performance of the proposed system for sorghum leaf disease recognition and classification showed a substantial result, which is **96.75%** of detection accuracy.

5.7 Development the Prototype

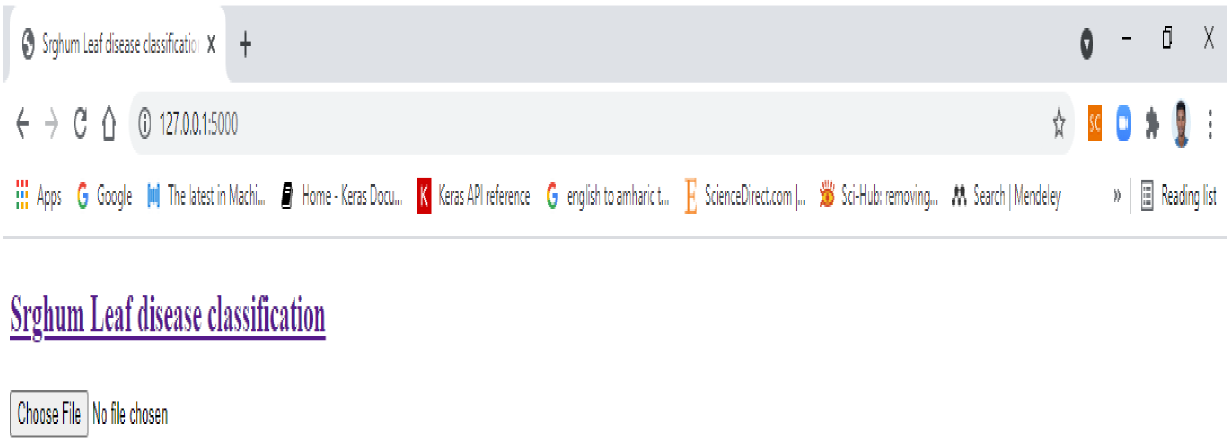
Finally, based on the results of this study the researcher developed a user interface for the proposed research work. This user interface was mainly used for domain experts or end-users to classify sorghum leaf diseases. The user interface is designed with python using Flask. As already discussed in chapter one Flask is used for developing web applications in python. It does not require particular tools or libraries and it has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. The end-user of the system can lunch a system by writing the URL of the system on any web browser, after that the web browser sends HTTP request to the webserver and then the webserver gives back an HTTP response to the browser. As shown in figure 5.3 the user enters the http://127.0.0.1:5000/ to get the home pages.



```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 321-259-564
```

Figure 5. 3 Launch the Flask server and get an IP address

The following figure 5.4, shows the Illustration Design of the Classification model after the user enters the URL of the sorghum leaf disease detection and classification system.



Srghum Leaf disease classification

Choose File No file chosen

Figure 5. 4 User interface

As shown in figure 4.4, after the end-user entering the URL the requested home page is displayed. Then the end-user must upload a sorghum leaf image to know which disease affects the leaf. Once the user clicks the choose file button the system displays the file upload window, and the users can upload the images from local disk or external disk. Figure 5.5 illustrates uploading the sorghum leaf images.

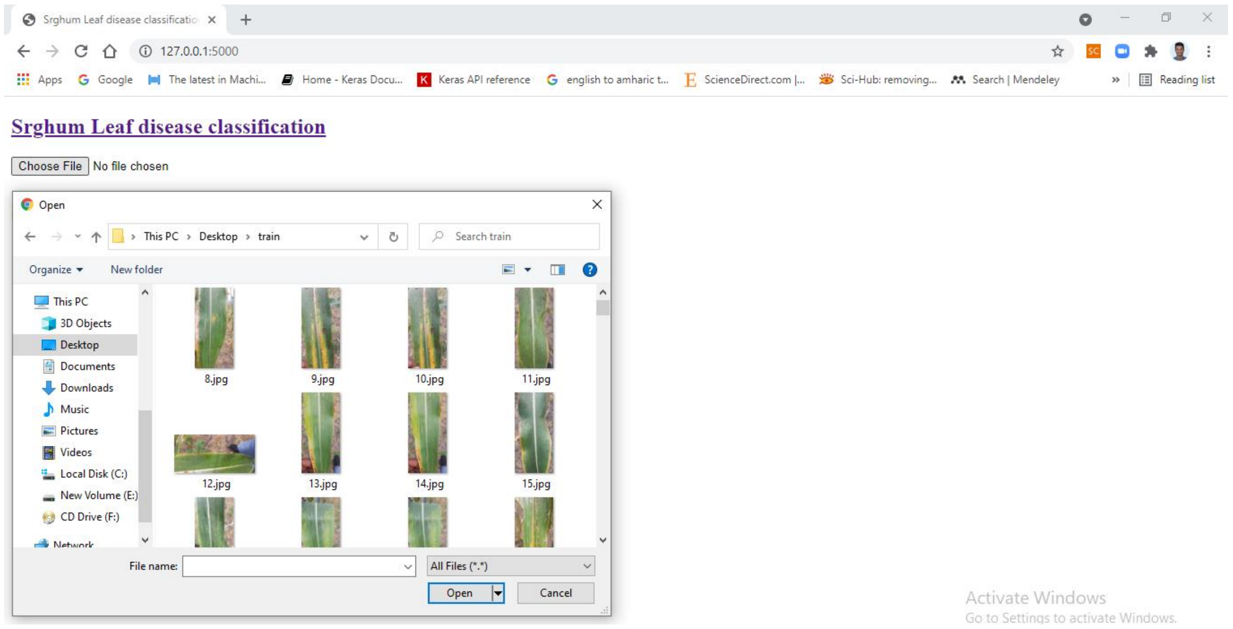

Srghum Leaf disease classification
Choose File No file chosen
Open
This PC > Desktop > train
Organize New folder
8.jpg 9.jpg 10.jpg 11.jpg 12.jpg 13.jpg 14.jpg 15.jpg
File name: All Files (*.*)
Open Cancel
Activate Windows
Go to Settings to activate Windows.

Figure 5. 5 Uploading sorghum leaf image from disk

As shown in figure 5.5, the user uploads the images from local or external disk by clicking the choose file button then after uploading the required image the users click the predict button as shown in figure 5.6.

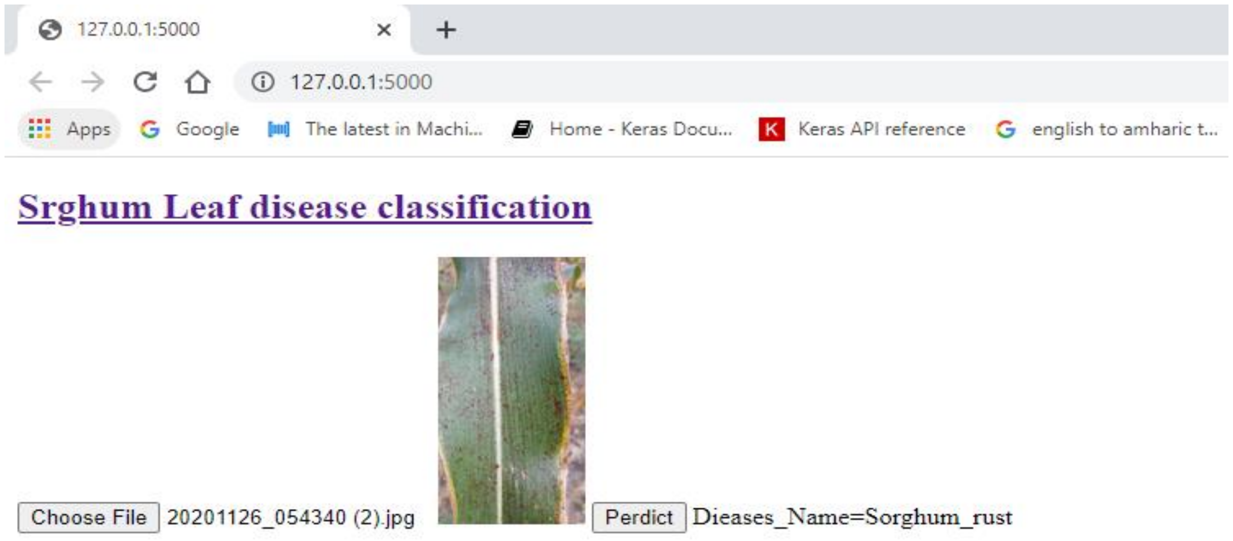

127.0.0.1:5000
Srghum Leaf disease classification
Choose File 20201126_054340 (2).jpg
Perdict
Dieases_Name=Sorghum_rust

Figure 5. 6 Sample result of the model

5.8 User Acceptance Test

The user acceptance testing ensures how the users or domain experts view the system on the bases of the rules and performance to classify sorghum leaf disease.

The entire proposed system is evaluated by 5 people, the researcher select randomly from different profession 3 of them are form Debre Berhan University plant science department and the remaining 2 from Deber Berhan Agricultural research center. The selection is based on the assumption that those with plant science background can see and evaluate the applicability, accuracy and importance of the system. Before starting the evaluation process, the system was explained in detail to the evaluators. Therefore, for this study, the evaluation criteria has been used such as Excellent = 5, Very Good =4, Good =3, Fair =2 and Poor =1. Therefore, evaluators were allowed to the following closed ended questions. Table 5.11 shows the feedbacks obtained from the evaluators on system interactions.

Table 5. 11 User acceptance evaluation

No	Criteria evaluation	Poor	Fair	Good	Very Good	Excellent	Average
1	The prototype system user friendly			1	1	3	4.4
2	The attractiveness of the system				3	2	4.4
3	Response time of the system			1	3	1	4
4	The correctness of the prototype in the classification of sorghum leaf			1	2	2	4.2
5	Applicability of the system				3	2	4.4
6	How do you rate the significance of the system in the domain area?				1	4	4.8
	Average						4.36

As illustrated in Table 5.11, 20% of evaluators replied Good for user friendly of the prototype and 20% of respondents, replied Very good and the remaining 60% are replied Excellent. Concerning the second question, that is Attractiveness of the system 60%, 40% of evaluators replied Very Good and Excellent respectively. The third question is about the response time of the system 20% of evaluators scored Good, 60% of evaluators scored Very Good and the remaining 20% of evaluators scored Excellent. The fourth evaluation criteria are about the correctness of the prototype 20% of evaluators scored Good, 40% of evaluators scored Very

Good and 40% of them as excellent. The fifth criteria are the applicability of the system among the evaluators 60% of evaluators scored Very Good and 40% of evaluators scored Excellent. Finally the last criteria is how do you rate the significance of the system. Among the evaluators, 20% replied very Good and 80% of them replied excellent.

5.9 Discussion

Based on the conducted research, in this section, the researcher will discuss insight into the result concerning the research objectives stated in chapter one. As described, in chapter one section 1.3 the major objective of this proposed work was to develop a model for sorghum leaf disease recognition and classification this study work attempts to answers the below-listed questions:

- ➢ Which training methods are more appropriate for classifying sorghum leaf diseases?

In this study, the researcher applies two convolutional neural network training methods namely, training from scratch and transfer learning models (i.e., reusing the previously trained model). Comparatively, training from the scratch with Gabor filtering algorithm achieves an accuracy of 96.75% whereas training from the scratch without Gabor filtering, VGG19, and VGG16 achieve an accuracy of 94.5, 91.5%, and 87.75% respectively. Therefore, the proposed model with Gabor filtering is the most suitable classifier for sorghum leaf diseases.

- ➢ Which features extraction techniques are the best for classifying sorghum leaf diseases?

In this study, the researcher applied two different image feature extraction algorithms to make the convolutional neural networks easily learn the important features from the given images. Currently, different researchers applied Gabor filtering and GLCM feature extraction algorithm to extract the texture feature and to minimize computation time and space in deep learning. Based on this experimental result shows that Gabor filtering achieves an accuracy of 96.75% whereas GLCM achieves an accuracy of 87.5%. For this study, Gabor achieves better results than GLCM. Therefore, the researcher recommends that applying the Gabor filtering algorithm on a convolutional neural network will help the model to easily identify the high-level features from the given images.

- To what extent does the developed model correctly classify sorghum leaf diseases?

After developing a classifier model, always requires to determine whether the developed model is a correct predictor or not. In this study, the performance of the developed classifier model was evaluated using a confusion matrix test. Besides, the most widely used classifier model performance evaluation metrics of precision, recall, and F-measure were also applied. Somewhat, training from the scratch with Gabor filtering achieves an accuracy of 96.75% has given promising results for the detection and classification of the sorghum leaf disease than transfer learning.

5.10 Summary of the experiment

In this chapter, we have seen the experimental results of this research work from dataset collection, dataset preparation, building the model, and the performance of each classifier model. However, the challenge in the experiments is overfitting and oscillation in the training and validation loss or accuracy. It is due to a random sample from our dataset: the dataset at each evaluation step is different, so is the validation loss. In addition, at each epoch, there are 18 (70% of the training dataset, 1200 divided by the batch size, 64) iterations. At each iteration, different samples are taken, trained, and tested, thereby oscillation or overfitting occurred. The other challenge in this experiment is a computational resource as stated in comparison to the study each experiment has to take two and half days. So, when the research changes each hyper-parameter value and re-runs it takes a lot of time to execute.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1 Introduction

In this chapter, the study highlights conclusions based on the findings on the experimentation of the thesis work is carried out and reported. Similarly, a suggestion for future research and contribution of the study is forwarded.

6.2 Conclusion

Sorghum is one of the heavily cultivated grains in the world. Governments and farm owners invest their arable land, energy, time, and money to cultivate this crop. Countries, including Ethiopia, produce sorghum for domestic consumptions. The grain is used as a major food item around the world and especially in sub-Saharan Africa. This crop production is damaged due to different factors, among them, diseases are the major ones. These diseases reduce the quality and quantity of sorghum crops. Diseases such as anthracnose, rust, and leaf blight are among the fungal groups, which are now considered as one of the most destructive diseases of sorghum in most of the major growing regions of the country. These diseases affect different parts and stages of the crop that significantly reduce its productivity. Hence, the identification of these diseases through a system is critical. Therefore, artificial intelligence with Image processing techniques are plays important role in the field of agriculture research.

This study aimed to develop a classifier model for sorghum leaf disease using a deep-learning approach. To do so, the researcher followed a design science research methodology. In this research work, the required 4000 sorghum leaf images were collected from Shewarot woreda kobo village North Shewa zone. After collecting the required images the researcher applies different image pre-processing techniques such as image resizing, noise removing, and normalization. Then the researcher extracts texture features from images using Gabor filtering.

Accordingly, two deep CNN architectures were used to conduct this study. Specifically, training from scratch (i.e. the researcher settings different hyper-parameters and trained the model from the scratch) and transfer learning convolutional neural network methods were used to build a model that can classify the sorghum leaf disease. During training the model the researcher applies image augmentation techniques in both CNN architecture to mitigate the overfitting. Additionally, to develop a classification, model 70%:30% percentage split test options have been utilized and a confusion matrix and classification report to visualize the model performance.

Experimental results show that after applying Gabor filtering the overall success rate for the classification of sorghum leaf is 96.5%. The success rates for classifying anthracnose, healthy, leaf blight, and rust are 97%, 100%, 97%, and 92% respectively. Moreover, these results show that the proposed CNN model with Gabor filtering is effective in classifying sorghum leaf diseases. From the result achieved, the researcher can conclude that the use of texture features is an excellent choice for automatic plant leaf classification. Also, the proposed model can be used to help monitor the changes in sorghum leaf disease over time, to ensure healthy growth. Overall, the system devised is very user-friendly and can successfully classify detect and measure disease of sorghum leaf while saving time and effort.

6.3 Contribution of the Study

As a contribution to the new knowledge, this research work has contributed the following.

- The researcher designed a classifier using CNN that can successfully classify sorghum leaf diseases.
- Gabor filter application: CNNs were trained with an input of raw (image) data. The researcher has changed this trend by the application of the Gabor filter. This makes the model train on the already distinguished texture (orientation and frequency) features. This enables the proposed model to have higher training and testing accuracy than CNN applied on the raw image data.
- Performance improvement: the proposed model achieves better results in terms of accuracy, the classifier has gotten 96.5% testing accuracy that are far above state-of-the-art models. The proposed model weighs very little and trains faster as compared to pre-trained models. The application of Gabor filter on the raw image was also used to improve the performance of state-of-the-art models as well by 5% (VGG19) and 8% (VGG16).

- This research work can be used as a reference for the recognition of objects from images as the basic underlying principle is the same concerning disease recognition and classification.

6.4 Recommendation

As we know Agriculture is the backbone of the Ethiopian economy no matter how well it is developed. Therefore, there is the need to pay much attention to it so that the right output will come from it. In our part of the globe where there is limited application of technology in Agriculture, farmers find it very difficult to produce up to their maximum strength due to factors like plant diseases. It is therefore recommended that much attention will be given to the treatment of plant diseases to avoid much loss in agriculture. In addition, the government should have a focus on training agricultural extension worker, users, and pathologists to use and apply the non-destructive way of detecting and classifying plant disease early. Moreover, the government also thinks about it implementing this kind of research work for early detection and classification of plant leaf disease and reducing the production loss from plant leaf diseases.

6.5 Future work

The primary objective of this study has been achieved and the research questions have been answered. Although, the researcher has tried the best to get the desired output and the accuracy value is quite good for this method, yet there is still room for improvement as long as the accuracy is not exactly 100% few works remain unsolved. The following are the possible future works.

- In this work, the researcher has built a system that identifies which type of disease attacks sorghum. Nevertheless, the system does not estimate the severity of the identified disease and does not recommend the appropriate treatments. Thus, another research direction.
- In this study, only texture features especially for the CNN training from scratch have been extracted. In future work, other researchers should try to use Meta features so that they can be optimized to achieve a higher recognition accuracy.

- This research work addresses three diseases for sorghum crops, however, due to lack of dataset (images), the remaining Downy Mildew disease. Therefore, future studies can extend this research work to include other diseases as a fifth class.

References

1. Harris, David R., and Dorian Q. Fuller. "Agriculture: definition and overview." New York: Springer, 2014.
2. Price, T. Douglas, and Ofer Bar-Yosef. "The origins of agriculture: new data, new ideas: an introduction to supplement 4." *Current Anthropology* , Vol. 52, No. 4, Pp. 163-174, 2011.
3. Diao, Xinshen, et al. The role of agriculture in development: Implications for Sub-Saharan Africa. Vol. 153, Intl Food Policy Res Inst, 2007.
4. Alston, J.M. and Pardey, P.G., 2014. Agriculture in the global economy. Journal of Economic Perspectives, 28(1), pp.121-46, 2014.
5. Ministry of Economic, Development and Cooperation, survey of the ethiopian economy (1992/93 - 1997/ 98), Addis Ababa, Ethiopia, September 1999.
6. Bachewe, Fantu Nisrane, Guush Berhane, Bart Minten, and Alemayehu Seyoum Taffesse. Agricultural growth in Ethiopia (2004-2014): Evidence and drivers. No. 81. International Food Policy Research Institute (IFPRI), 2015.
7. Taffesse, Alemayehu Seyoum, Paul Dorosh, and Sinafikeh Asrat Gemessa. "3 Crop production in Ethiopia: regional patterns and trends." In Food and agriculture in Ethiopia, Pp. 53-83. University of Pennsylvania Press, 2013.
8. Åkesson, Mrs Tanja, Codex Contact Point, and Viale delle Terme di Caracalla. "joint fao/who food standards programme codex committee on contaminants in foods." 2015.
9. Ramatoulaye, F., C. Mady, and S. Fallou. "Production and use sorghum: a literature review." Journal of Nutritional Health & Food Science, Vol. 4, No. 1, Pp. 1-4, 2016.
10. B. Assefa and B. Lanos, "Analysis of price incentives for Sorghum in Ethiopia," FAO, Rome, 2015.
11. Teferi, Teklay Abebe, and Muruts Legesse Wubshet. "Prevalence and intensity of economically important fungal diseases of sorghum in South Tigray, Ethiopia." Journal of Plant Sciences, Vol. 3, No. 2, Pp 92-98, 2015.
12. Ray, Monalisa, Asit Ray, Swagatika Dash, Abtar Mishra, K. Gopinath Achary, Sanghamitra Nayak, and Shikha Singh. "Fungal disease detection in plants: Traditional assays, novel diagnostic techniques, and biosensors." Biosensors and Bioelectronics 87 pp.708-723, 2017.

13. Zechmann, Bernd, and Günther Zellnig. "Rapid diagnosis of plant virus diseases by transmission electron microscopy." Journal of virological methods, Vol. 162, No. 2, Pp.163-169, 2009.
14. Zhang, M., X. Liu, and M. O'neill. "Spectral discrimination of Phytophthora infestans infection on tomatoes based on principal component and cluster analyses." International Journal of Remote Sensing, Vol. 23, No. 6, Pp. 1095-1107, 2002.
15. Patrício, Diego Inácio, and Rafael Rieder. "Computer vision and artificial intelligence in precision agriculture for grain crops." Computers and electronics in agriculture, Vol. 153, Pp. 69-81, 2018
16. Guo, Yanming, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. "Deep learning for visual understanding." Neurocomputing, Vol. 187, Pp. 27-48, 2016.
17. Wehle, Hans-Dieter. "Machine learning, deep learning, and AI: What's the difference?" In International Conference on Data scientist innovation day, Bruxelles, Belgium. 2017.
18. Rath, A. K., and J. K. Meher. "Disease detection in infected plant leaf by computational method." Archives of phytopathology and plant protection, Vol. 52, No. 20, Pp. 1348-1358, 2019.
19. B. Haile, T. Babege and A. Hailu, "Diseases and Insect Pests of Onion (Allium cepa L.) in Masha District of Sheka Zone, South-West Ethiopia," Journal of Agricultural Research, Vol. 4, No. 10, Pp. 629-632, 2016.
20. Kinfe, Hailegebrial, and Adane Tesfaye. "Yield performance and adoption of released Sorghum varieties in Ethiopia." Edelweiss Applied Science and Technology, Vol. 2, No. 1 Pp. 46-55, 2018.
21. Amelework, Beyene A., Hussein A. Shimelis, Mark D. Laing, Dawit Getnet Ayele, Pangirayi Tongoona, and Fentahun Mengistu. "Sorghum production systems and constraints, and coping strategies under drought-prone agro-ecologies of Ethiopia." South African Journal of Plant and Soil, Vol. 33, No. 3, Pp. 207-217, 2017.
22. Halder, Monishanker, Ananya Sarkar, and Habibullah Bahar. "Plant disease detection by image processing: a literature review." Journal of Food Science & Technology, Vol. 3, No. 6, 2019.

23. Sujatha, R., Y. Sravan Kumar, and Garine Uma Akhil. "Leaf disease detection using image processing." Journal of Chemical and Pharmaceutical Sciences, Vol. 10, No. 1, Pp.670-672, 2017.
24. Tejashree Soni, Prajakta Joshi, Pranali Chavan, Ajjay Gaadhe. "Machine Learning Approach for Crop Yield Improvement Using Plant Leaf Disease Detection", International Research Journal of Engineering and Technology, Vol. 7, No. 4, Pp. 4920-4923, 2020.
25. Rahman, Hidayat, Nadeem Jabbar Ch, SanaUllah Manzoor, Fahad Najeeb, Muhammad Yasir Siddique, and Rafaqat Alam Khan. "A comparative analysis of machine learning approaches for plant disease identification." Advancements in Life Sciences, Vol. 4, No. 4, Pp. 120-125, 2017.
26. Ahmed, Kawcher, Tasmia Rahman Shahidi, Syed Md Irfanul Alam, and Sifat Momen. "Rice leaf disease detection using machine learning techniques." In 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, 2019.
27. Tejas Mohan Ayyar, "Medium," [Online]. Available: https://tejasmohanayyar.medium.com/a-practical-experiment-for-comparing-lenet-lexnet-vgg-and-resnet-models-with-their-advantages-d932fb7c7d17. [Accessed 4 July 20021].
28. Zhang, Aston, Zachary C. Lipton, Mu Li, and Alexander J. Smola. "Dive into deep learning", Springer, 2021.
29. Sharma, Neha, Vibhor Jain, and Anju Mishra. "An analysis of convolutional neural networks for image classification." in International Conference on Computational Intelligence and Data Science, Noida, India, 2018.
30. Sladojevic, Srdjan, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. "Deep neural networks based recognition of plant diseases by leaf image classification." Computational intelligence and neuroscience 2016.
31. Jmour, Nadia, Sehla Zayen, and Afef Abdelkrim. "Convolutional neural networks for image classification." In 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Pp. 397-402. IEEE, 2018.
32. Bisandu, Desmond Bala. "Design science research methodology in Computer Science and Information Systems." International Journal of Information Technology Vol. 5, No. 4 Pp. 55-60, 2016.

33. Anbsaw Belete Mekonnen, “A deep learning-based approach for potato leaf diseases classification”, Master's thesis in Natural Science Deber Berhan University, Deber Berhan Ethiopia: Unpublished, 2020.
34. Hemanth kumar, "Introduction to Micro Web Framework Flask," 3 5 2021. [Online]. Available:https://medium.com/featurepreneur/introduction-to-micro-web-framework-flask-78de9289270b. [Accessed 1 6 2021].
35. https://scikitlearn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html, [Accessed 10 October 2020].
36. Kimber, Clarissa T. "Origins of domesticated sorghum and its early diffusion to India and China." New York:]ohn Wiley & Sons, Inc, 2000.
37. Smith, C. Wayne, and Richard A. Frederiksen, eds. “Sorghum: Origin, history, technology, and production”. Vol. 2. Pp. 261-307, John Wiley & Sons, 2000.
38. Tuinstra, Mitchell R. "Food-grade sorghum varieties and production considerations." Journal of Plant Interactions, Vol. 3, No. 1, Pp. 69-72, 2008.
39. Chala, C. B. "Sorghum [Sorghum bicolor (L.)] Breeding in Ethiopia: Review." Journal of Biology, Agriculture and Healthcare, Vol. 8, No. 21, Pp. 81-90, 2018.
40. Demeke Mekonnen and F. Di Marcantonio. "Analysis of incentives and disincentives for sorghum in Ethiopia." MAFAP, FAO, Rome, 2013.
41. Mbwika, J. M., H. Odame, and E. K. Ngungi. "Feasibility study on Striga control in sorghum." Nairobi, African Agricultural Technology Foundation. Majestic printing works, Nairobi, Kenya. 78pp (2011).
42. DeMilliano, Walter Alphonsus Johannes, Richard A. Frederiksen, and G. D. Bengston. "Sorghum and Millets Diseases A Second World Review," International Crops Research Institute for the Semi-Arid Tropics, Vol. 378, No. 46, 1992.
43. Tsedaley, Binyam, Girma Adugna, and Fikre Lemessa. "Distribution and importance of sorghum anthracnose (Colletotrichum sublineolum) in southwestern and western Ethiopia." Journal of Plant Pathology, Vol. 15, No. 3, Pp. 75-85, 2016.
44. Rahman, Hidayat, Nadeem Jabbar Ch, SanaUllah Manzoor, Fahad Najeeb, Muhammad Yasir Siddique, and Rafaqat Alam Khan. "A comparative analysis of machine learning approaches for plant disease identification." Advancements in Life Sciences, Vol. 4, No. 4 Pp. 120-126, 2017.

45. Samuel, Anjorin Toba, Oyerinde Akeem Abolade, and Okafor Obiageli Evelyn. "Status of pests and diseases of sorghum and their management practices by Fadama III participating farmers in Abuja, Nigeria." Journal of Agricultural Extension and Rural Development, Vol. 12, No. 2, Pp 36-47, 2020.
46. R. L. Christopher and P. Ramasamy, the Biology and Control of Sorghum Diseases, Kansas: American Society of Agronomy Crop Science Society of America Soil Science Society of America, 2019.
47. H. Goitoom, Biological Control of Anthracnose of Sorghum using Trichoderma isolates under in vitro conditions, Master's thesis in Social Science, Addis Abeba University, Addis Abeba Ethiopia, Unpublished, 2019.
48. Das, Ankita, Amar Kumar Dey, and Manisha Sharma. "Leaf disease detection, quantification and classification using digital image processing." International Journal of Innovative Research in Electrical, Electronics, Instrumentation, and Control Engineering, vol. 5, no. 11, pp. 45-50, 2017.
49. Tropics, International Crops Research Institute for the Semi-Arid. “Sorghum disease a world review,” in International Workshop on Sorghum Diseases, Hyderabad, India, 1978.
50. Upadhyaya, Hari D., Yi-Hong Wang, Rajan Sharma, and Shivali Sharma. "SNP markers linked to leaf rust and grain mold resistance in sorghum." Molecular breeding, Vol. 32, No. 2, Pp. 451-462, 2013.
51. Rehman, Tanzeel U., Md Sultan Mahmud, Young K. Chang, Jian Jin, and Jaemyung Shin. "Current and future applications of statistical machine learning algorithms for agricultural machine vision systems." Computers and Electronics in Agriculture, Vol. 156, P. 585-605, 2018.
52. Gomes, Juliana Freitas Santos, and Fabiana Rodrigues Leta. "Applications of computer vision techniques in the agriculture and food industry: a review." European Food Research and Technology, Vol. 235, No. 6, Pp. 989-1000, 2012.
53. Li, Yang, and Yuhang Zhang. "Application Research of Computer Vision Technology in Automation." In International Conference on Computer Information and Big Data Applications (CIBDA) Guiyang, China, Pp. 374-377, 2020.
54. Foglia, Mario M., and Giulio Reina. "Agricultural robot for radicchio harvesting." Journal of Field Robotics, Vol. 23, No. 6-7, Pp. 363-377, 2006.

55. Vázquez-Arellano, Manuel, Hans W. Griepentrog, David Reiser, and Dimitris S. Paraforos. "3-D imaging systems for agricultural applications—a review." Sensors, Vol. 16, No. 5, 2016.
56. Zhu, Yanjun, Zhiguo Cao, Hao Lu, Yanan Li, and Yang Xiao. "In-field automatic observation of wheat heading stage using computer vision." Biosystems Engineering, Vol. 143, Pp. 28-41, 2015.
57. Wiley, Victor, and Thomas Lucas. "Computer vision and image processing: a paper review." International Journal of Artificial Intelligence Research, Vol. 2, No. 1, Pp. 28-30, 2018.
58. Gonzale, Rafael C. "Digital image processing third edition." Upper Saddle River, New Jersey: Pearson Prentice Hall, 2008.
59. Tyagi, Vipin. "Understanding digital image processing". India CRC Press, 2018.
60. Abdullah, Ahmed, Wahid Palash, Ashiqur Rahman, K. Islam, and A. Alim. "Digital image processing analysis using Matlab." American Journal of Engineering Research (AJER), Vol. 5, No. 12, Pp. 143-147, 2016.
61. Rao, Raghuveer M., and Manoj K. Arora. "Overview of image processing." In Advanced image processing techniques for remotely sensed hyperspectral data, pp. 51-85. Springer, Berlin, Heidelberg, 2018.
62. Panigrahi, Kshyanaprava Panda, Himansu Das, Abhaya Kumar Sahoo, and Suresh Chandra Moharana. "Maize leaf disease detection and classification using machine learning algorithms." In Progress in Computing, Analytics, and Networking Advances in Intelligent Systems and Computing, Odisha, 2020.
63. Bhadur, Ganesh, and Rajneesh Rani. "Agricultural Crops Disease Identification and Classification through Leaf Images using Machine Learning and Deep Learning Technique: A Review." in International Conference on Intelligent Communication and Computational Research, Jalandhar, 2020.
64. Anjali, Y. Rajkumar and J. Kamaldeep, "A Comparative Study of Image Segmentation Methods," International Journal of All Research Education and Scientific Methods, Vol. 4, No. 7, Pp. 232-235, 2017.
65. Karim, Abdul Amir Abdullah, and Rafal Ali Sameer. "Comparing the Main Approaches of Image Segmentation." *Journal of Science, Vol.* 58, No. 4, Pp. 2211-2221, 2017.

66. K. Dilpreet and K. Yadwinder, "Various Image Segmentation Techniques," International Journal of Computer Science and Mobile Computing, Vol. 3, No. 5, Pp. 809-814, 2014.
67. Kornilov, Anton S., and Ilia V. Safonov. "An overview of watershed algorithm implementations in open source libraries." Journal of Imaging, Vol. 4, No. 10, Pp. 1-15, 2018.
68. Bala, Anju. "An improved watershed image segmentation technique using MATLAB." International Journal of Scientific & Engineering Research, Vol. 3, No. 6, Pp. 1-4, 2012.
69. Choras, Ryszard S. "Image feature extraction techniques and their applications for CBIR and biometrics systems." International journal of biology and biomedical engineering, Vol. 1, No. 1, Pp. 6-16, 2007.
70. E. Umapathy, S. Prasanna and Sripriya, "Various approaches of color feature extraction in leaf diseases under image processing," International Journal of Engineering & Technology, Vol. 7, No. 2, Pp. 712-717, 2018.
71. Gabor, Dennis. "Theory of communication. Part 1: The analysis of information." Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering, Vol. 93, No. 26, Pp. 429-441, 1946.
72. Kumar, Arun, Vinod Patidar, Deepak Khazanchi, and Poonam Saini. "Role of feature selection on leaf image classification." Journal of Data Analysis and Information Processing, Vol. 3, No. 2, Pp. 175-183, 2015.
73. Marmol, Urszula. "Use of Gabor filters for texture classification of airborne images and LIDAR data." Archives of Photogrammetry, Cartography and Remote Sensing, vol. 22, pp. 325-336, 2011.
74. De Menezes, Richardson Santiago Teles, Rafael Marrocos Magalhaes, and Helton Maia. "Object recognition using convolutional neural networks." In *Recent Trends in Artificial Neural Networks, Pp. 1-10, 2019.*
75. Van Klompenburg, Thomas, Ayalew Kassahun, and Cagatay Catal. "Crop yield prediction using machine learning: A systematic literature review." Computers and Electronics in Agriculture, Vol. 117, Pp. 2-16, 2020.

76. Liakos, Konstantinos G., Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. "Machine learning in agriculture: A review." Sensors in Agriculture, Vol. 18, No. 8, Pp. 2-29, 2018.
77. Baştanlar, Yalin, and Mustafa Özuysal. "Introduction to machine learning." MicroRNA Biology and Computational Analysis, Vol. 1107, Pp. 105-127, 2014.
78. Chollet, Francois. "Deep learning with Python". Shelter Island: Manning Publications Co, 2017.
79. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature, Vol. 521, No. 7553, Pp. 436-444, 2015.
80. Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and Electronics in Agriculture, Vol. 145, Pp. 311-318, 2018.
81. Rawat, Waseem, and Zenghui Wang. "Deep convolutional neural networks for image classification" Neural Computation, Vol. 29, Pp. 2352–2449, 2017.
82. Deng, Li, and Dong Yu. "Deep learning: methods and applications." Foundations and trends in signal processing, Vol. 7, No. 3–4, Pp. 197-387, 2014.
83. Du, Xuedan, Yinghao Cai, Shuo Wang, and Leijie Zhang. "Overview of deep learning." In 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, 2016.
84. Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." In 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017.
85. Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. "Convolutional neural networks: an overview and application in radiology." Insights into imaging, Vol. 9, No. 4, Pp. 611-629, 2018.
86. Si, Lei, Xiangxiang Xiong, Zhongbin Wang, and Chao Tan. "A Deep Convolutional Neural Network Model for Intelligent Discrimination between Coal and Rocks in Coal Mining Face." Mathematical Problems in Engineering, 2020.
87. Sakib, Shadman, Nazib Ahmed, Ahmed Jawad Kabir, and Hridon Ahmed. "An Overview of Convolutional Neural Network: Its Architecture and Applications," 2018.
88. O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." 2015.

89. Murphy, John. "An overview of convolutional neural network architectures for deep learning." Microway Inc, Pp. 5-20, 2016.
90. Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." In 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017.
91. Sharma, Ochin. "A new activation function for deep neural network." In International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), India, 2019.
92. Wang, Yingying, Yibin Li, Yong Song, and Xuewen Rong. "The influence of the activation function in a convolution neural network model of facial expression recognition." Applied Sciences, Vol. 10, No. 5, Pp. 1-20, 2020.
93. Feng, Jianli, and Shengnan Lu. "Performance analysis of various activation functions in artificial neural networks." In Journal of Physics: Conference Series, Vol. 1237, No. 2, 2019.
94. Nwankpa, Chigozie, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. "Activation functions: Comparison of trends in practice and research for deep learning." *Glasgow 2018.*
95. Bačanin Džakula, Nebojša. "Convolutional Neural Network Layers and Architectures." In Sinteza 2019-International Scientific Conference on Information Technology and Data Related Research, Singidunum University, Serbia, 2019.
96. Gholamalinezhad, Hossein, and Hossein Khosravi. "Pooling Methods in Deep Neural Networks, a Review." 2018.
97. Le, Quoc V., Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y. Ng. "On optimization methods for deep learning." in Proceedings of the 28th International Conference on Machine Learning, Washington, USA, 2011.
98. Coşkun, Musab, Özal YILDIRIM, U. Ç. A. R. Ayşegül, and Yakup Demir. "An overview of popular deep learning methods." European Journal of Technique, Vol. 7, No. 2, Pp. 165-176, 2017.
99. Arsenov, Andrey, Igor Ruban, Kyrylo Smelyakov, and Anastasiya Chupryna. "Evolution of convolutional neural network architecture in image classification problems." In Selected

Papers of the XVIII International Scientific and Practical Conference on IT and Security (ITS 2018).–CEUR Workshop Processing, Pp. 35-45. 2018.

100. Islam, Taohidul, Manish Sah, Sudipto Baral, and Rudra Roy Choudhury. "A faster technique on rice disease detection using image processing of affected area in agro-field." In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 62-66. IEEE, 2018.

101. Chen, Junde, Defu Zhang, Yaser A. Nanehkaran, and Dele Li. "Detection of rice plant diseases based on deep transfer learning." Journal of the Science of Food and Agriculture, Vol. 100, Pp. 3246–3256, 2020.

102. S. Mohammad and S. Wahyudi, "Convolutional neural network for maize leaf disease image classification," Telecommunication, Computing, Electronics and Control, Vol. 18, No. 3, Pp. 1376-1381, 2020.

103. B. Prakruti, S. Sanat, S. Anshul, S. Dineshkumar and P. Srinivasu, "Identification of Diseases in Corn Leaves using Convolutional Neural Networks and Boosting," in 8th International Conference on Pattern Recognition Applications and Methods, Mumbai, India, 2019.

104. Priyadharshini, Ramar Ahila, Selvaraj Arivazhagan, Madakannu Arun, and Annamalai Mirnalini. "Maize leaf disease classification using deep convolutional neural networks." Neural Computing and Applications, Vol. 31, No. 12, Pp 8887-8895, 2019.

105. Afifi, Ahmed, Abdulaziz Alhumam, and Amira Abdelwahab. "Convolutional Neural Network for Automatic Identification of Plant Diseases with Limited Data." Plants, Vol. 10, No. 1, 2021.

106. Da Silva Abade, Andre, Ana Paula GS de Almeida, and Flavio de Barros Vidal. "Plant Diseases Recognition from Digital Images using Multichannel Convolutional Neural Networks." in 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Prague, 2019.

107. Ramya, V. and M. Anthuvan Lydia. "Leaf disease detection and classification using neural networks." International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, No. 11, Pp. 207-209, 2016.

108. Sibiya, Malusi, and Mbuyu Sumbwanyambe. "A computational procedure for the recognition and classification of maize leaf diseases out of healthy leaves using convolutional neural networks." AgriEngineering, Vol. 1, No. 1, Pp. 119-131, 2019.

109. Gad, Ahmed Fawzy, Ahmed Fawzy Gad, and Suresh John. “Practical computer vision applications using deep learning with CNNs”. Berkeley: Apress, 2018.

110. Deng, Qi, Yun Wang, and Shaobo Ji. "Design science research in information systems: A systematic literature review 2001-2015." In international conference on information resources management, Ontario, 2017.

111. Mandran, Nadine, and Sophie Dupuy-Chessa. "Supporting experimental methods in information system research." In 2018 12th International Conference on Research Challenges in Information Science (RCIS), Pp. 1-12. IEEE, 2018.

112. Ardakan, Mohammad Abooyee, and Kaveh Mohajeri. "Applying design research method to IT performance management: Forming a new solution." Journal of Applied Sciences, Vol. 9, No. 7, Pp. 1227-1237, 2009.

113. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems, Vol. 25. Pp 1097-1105, 2012.

114. Dobbin, Kevin K., and Richard M. Simon. "Optimally splitting cases for training and testing high dimensional classifiers." BMC medical genomics, Vol. 4, No. 1, Pp. 1-8, 2011.

115. World bank group, "The Growth Acceleration and How to Pace It," Washington, DC, February 2016.

116. Tian, Hongkun, Tianhai Wang, Yadong Liu, Xi Qiao, and Yanzhou Li. "Computer vision technology in agricultural automation—A review." Information Processing in Agriculture, Vol. 7, No. 1, Pp. 1-9, 2020.

117. Akram, Shazia, Mehraj-Ud-Din Dar, and Aasia Quyoum. "Document Image Processing- A Review." International Journal of Computer Applications, Vol. 10, No. 5, Pp. 35-40, 2010.

Appendix-I

Dear Evaluator,

This evaluation form is prepared aiming at measuring to what extend does sorghum leaf classification system is useable and acceptable by end users in the area of agricultural expert. Therefore, you are kindly requested to evaluate the system by labeling (X**)** symbol on the space provided for the corresponding attribute values for each criteria of evaluation. I would like to appreciate your collaboration in providing the information.

Note: - The values for all attributes in the table are rated as: Excellent=5, Very good =4, Good=3, Fair= 2 and Poor =1.

No	Criteria evaluation	Poor	Fair	Good	Very Good	Excellent
1	The prototype system user friendly					
2	The attractiveness of the system					
3	Response time of the system					
4	The correctness of the prototype in the classification of sorghum leaf					
5	Applicability of the system					
6	How do you rate the significance of the system in the domain area?					

Appendix-II

```
##importing different libraries

import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
os.environ['KERAS_BACKEND'] = 'tensorflow'
import matplotlib.pyplot as plt
from import itertools
```

## Reading an image and resize

```
Original_ds="C:\\Users\\Dagu\\Desktop\\dataset\\all"

Resized_ds="C://Users//Dagu//Desktop//resized//all "

image =size(list)
img_width,img_height=224,224
for file in list:
   im=Image.open(path1 + '\\'+file)
   img_1=im.resize((img_width,img_height))
   img_1.save(path2 +'\\'+ file, 'jpeg')
```

## changing into one dimensional array

```
image_array=array([array(Image.open(‘C://Users//Dagu//Desktop//resized//all ' + '\\'+
im2)).flatten() for im2 in new_data],'f')
```

## Label encoding

```
label=np.ones((image),dtype=int)
label[0:999]=0
```

```
label[1000:1999]=1
label[2000:2999]=2
label[3000:3999]=3
data,label=shuffle(image_array, label,random_state=4)
```

## Splitting the dataset into training and testing and normalizing the input data

```
x_train,x_test, y_train,y_test=train_test_split(x, y,test_size=0.3, random_state=2)
x_train=x_train.reshape(x_train.shape[0],*(224,224, 3))
x_test= x_test.reshape(x_test.shape[0], *(224,224,3))
x_train /=255
x_test /=255
```

## Importing keras module

```
from keras.models import Sequential
from keras.callbacks import ReduceLROnPlateau
from keras.layers.normalization import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, confusion_matrix
from keras.layers import Conv2D, Flatten, Dense, MaxPool2D , Activation, Dropout
keras.optimizers import Adam
number_of_classes = 4
image_size=(224,244,3)
model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_rows,img_cols, 3)),
  layers.Conv2D(16, 3, padding='same', activation= LeakyReLU),
  layers.MaxPooling2D(),
  layers.Dropout(0.5),

  layers.Conv2D(32, 3, padding='same', activation= LeakyReLU),
  layers.MaxPooling2D(),
  layers.Dropout(0.5),

  layers.Conv2D(64, 3, padding='same', activation= LeakyReLU),
  layers.MaxPooling2D(),
  layers.Dropout(0.5),

  layers.Conv2D(128, 3, padding='same', activation= LeakyReLU),
  layers.MaxPooling2D(),
  layers.Dropout(0.5),

  layers.Flatten(),
  layers.Dense(512, activation='relu'),
  layers.Dropout(0.5),
  layers.Dense(number_of_classes,activation='softmax')
])
model.summary()
```

## Compile the model

```
Optimizer = Adam(lr=0.001,beta_1=0.9, beta_2=0.999, epsilon=1e-07)
model.compile(loss='categorical_crossentropy', Optimizer=optimizer, metrics=['accuracy'])
learning_rate_reduction = ReduceLROnPlateau(monitor='value_accuracy',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

## Data transformation

```
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=False,
        vertical_flip=False)
datagen.fit(x_train)
```

## Training the model

```
epochs =100
batch_size=64
history = model.fit_generator(datagen.flow(x_train, Y_train, batch_size=batch_size),
                              epochs = epochs,
                              verbose = 1,
                              validation_data=(x_test, Y_test),
                              steps_per_epoch=len(x_train) / batch_size)
```

```
Epoch 1/100
88/88 [==============================] - 166s 2s/step - loss: 1.3423 - accuracy: 0.40
38 - val_loss: 0.7483 - val_accuracy: 0.6617
Epoch 2/100
88/88 [==============================] - 87s 982ms/step - loss: 0.7337 - accuracy: 0
.6931 - val_loss: 0.6258 - val_accuracy: 0.7475
Epoch 3/100
88/88 [==============================] - 91s 1s/step - loss: 0.6216 - accuracy: 0.747
2 - val_loss: 0.5763 - val_accuracy: 0.7533
Epoch 4/100
88/88 [==============================] - 87s 977ms/step - loss: 0.5126 - accuracy: 0
.7962 - val_loss: 0.5933 - val_accuracy: 0.7567
Epoch 5/100
88/88 [==============================] - 87s 985ms/step - loss: 0.5008 - accuracy: 0
.7956 - val_loss: 0.4902 - val_accuracy: 0.7950
Epoch 6/100
```

```
88/88 [==============================] - 86s 972ms/step - loss: 0.3638 - accuracy: 0
.8564 - val_loss: 0.6000 - val_accuracy: 0.7725
Epoch 7/100
88/88 [==============================] - 89s 1s/step - loss: 0.3115 - accuracy: 0.871
8 - val_loss: 0.5911 - val_accuracy: 0.7742
Epoch 8/100
88/88 [==============================] - 89s 1s/step - loss: 0.2814 - accuracy: 0.890
7 - val_loss: 0.7081 - val_accuracy: 0.7742
Epoch 9/100
88/88 [==============================] - 87s 982ms/step - loss: 0.2029 - accuracy: 0
.9240 - val_loss: 0.6150 - val_accuracy: 0.8217
Epoch 10/100
88/88 [==============================] - 86s 977ms/step - loss: 0.1313 - accuracy: 0
.9582 - val_loss: 0.5986 - val_accuracy: 0.8050
Epoch 11/100
88/88 [==============================] - 86s 969ms/step - loss: 0.0922 - accuracy: 0
.9721 - val_loss: 0.6298 - val_accuracy: 0.8283
Epoch 12/100
88/88 [==============================] - 94s 1s/step - loss: 0.1041 - accuracy: 0.959
6 - val_loss: 0.6273 - val_accuracy: 0.8217
Epoch 13/100
88/88 [==============================] - 88s 993ms/step - loss: 0.0731 - accuracy: 0
.9762 - val_loss: 0.6362 - val_accuracy: 0.8150
Epoch 14/100
88/88 [==============================] - 87s 977ms/step - loss: 0.1023 - accuracy: 0
.9627 - val_loss: 0.6769 - val_accuracy: 0.8292
Epoch 15/100
88/88 [==============================] - 84s 952ms/step - loss: 0.0274 - accuracy: 0
.9946 - val_loss: 0.7079 - val_accuracy: 0.8225
Epoch 16/100
88/88 [==============================] - 87s 980ms/step - loss: 0.0330 - accuracy: 0
.9921 - val_loss: 0.7049 - val_accuracy: 0.8333
Epoch 17/100
88/88 [==============================] - 84s 952ms/step - loss: 0.0165 - accuracy: 0
.9970 - val_loss: 0.8689 - val_accuracy: 0.8167
Epoch 18/100
88/88 [==============================] - 102s 1s/step - loss: 0.0325 - accuracy: 0.98
86 - val_loss: 0.7744 - val_accuracy: 0.8217
Epoch 19/100
88/88 [==============================] - 89s 1s/step - loss: 0.0158 - accuracy: 0.998
4 - val_loss: 1.0294 - val_accuracy: 0.7992
Epoch 20/100
88/88 [==============================] - 86s 971ms/step - loss: 0.0176 - accuracy: 0
.9945 - val_loss: 0.9055 - val_accuracy: 0.8150
```

```
Epoch 21/100
88/88 [==============================] - 87s 989ms/step - loss: 0.0447 - accuracy: 0
.9896 - val_loss: 0.8792 - val_accuracy: 0.8075
Epoch 22/100
88/88 [==============================] - 95s 1s/step - loss: 0.0699 - accuracy: 0.973
0 - val_loss: 0.8243 - val_accuracy: 0.8042
Epoch 23/100
88/88 [==============================] - 86s 977ms/step - loss: 0.0420 - accuracy: 0
.9836 - val_loss: 0.7801 - val_accuracy: 0.8317
Epoch 24/100
88/88 [==============================] - 87s 990ms/step - loss: 0.0189 - accuracy: 0
.9957 - val_loss: 0.8715 - val_accuracy: 0.8317
Epoch 25/100
88/88 [==============================] - 86s 974ms/step - loss: 0.0115 - accuracy: 0
.9974 - val_loss: 0.8687 - val_accuracy: 0.8167
Epoch 26/100
88/88 [==============================] - 90s 1s/step - loss: 0.0159 - accuracy: 0.997
1 - val_loss: 0.8449 - val_accuracy: 0.8417
Epoch 27/100
88/88 [==============================] - 89s 1s/step - loss: 0.0069 - accuracy: 0.998
8 - val_loss: 0.7646 - val_accuracy: 0.8442
Epoch 28/100
88/88 [==============================] - 96s 1s/step - loss: 0.0058 - accuracy: 0.998
6 - val_loss: 0.8007 - val_accuracy: 0.8433
Epoch 29/100
88/88 [==============================] - 83s 941ms/step - loss: 0.0041 - accuracy: 0
.9990 - val_loss: 0.8251 - val_accuracy: 0.8467
Epoch 30/100
88/88 [==============================] - 84s 954ms/step - loss: 0.0044 - accuracy: 0
.9989 - val_loss: 0.7707 - val_accuracy: 0.8400
Epoch 31/100
88/88 [==============================] - 83s 938ms/step - loss: 0.0041 - accuracy: 0
.9987 - val_loss: 0.7559 - val_accuracy: 0.8392
Epoch 32/100
88/88 [==============================] - 83s 941ms/step - loss: 0.0031 - accuracy: 0
.9992 - val_loss: 1.0297 - val_accuracy: 0.8142
Epoch 33/100
88/88 [==============================] - 87s 986ms/step - loss: 0.0070 - accuracy: 0
.9986 - val_loss: 0.9002 - val_accuracy: 0.8233
Epoch 34/100
88/88 [==============================] - 84s 946ms/step - loss: 0.0053 - accuracy: 0
.9987 - val_loss: 0.7699 - val_accuracy: 0.8400
Epoch 35/100
```

```
88/88 [==============================] - 85s 960ms/step - loss: 0.0034 - accuracy: 0
.9992 - val_loss: 0.9291 - val_accuracy: 0.8325
Epoch 36/100
88/88 [==============================] - 83s 944ms/step - loss: 0.0037 - accuracy: 0
.9993 - val_loss: 0.7708 - val_accuracy: 0.8325
Epoch 37/100
88/88 [==============================] - 84s 953ms/step - loss: 0.0043 - accuracy: 0
.9987 - val_loss: 1.0025 - val_accuracy: 0.8267
Epoch 38/100
88/88 [==============================] - 90s 1s/step - loss: 0.0057 - accuracy: 0.998
7 - val_loss: 0.8636 - val_accuracy: 0.8325
Epoch 39/100
88/88 [==============================] - 83s 941ms/step - loss: 0.0037 - accuracy: 0
.9991 - val_loss: 0.9010 - val_accuracy: 0.8358
Epoch 40/100
88/88 [==============================] - 85s 964ms/step - loss: 0.0041 - accuracy: 0
.9989 - val_loss: 0.8021 - val_accuracy: 0.8450
Epoch 41/100
88/88 [==============================] - 83s 940ms/step - loss: 0.0042 - accuracy: 0
.9991 - val_loss: 0.9154 - val_accuracy: 0.8317
Epoch 42/100
88/88 [==============================] - 84s 947ms/step - loss: 0.0036 - accuracy: 0
.9987 - val_loss: 0.8296 - val_accuracy: 0.8375
Epoch 43/100
88/88 [==============================] - 84s 953ms/step - loss: 0.0027 - accuracy: 0
.9990 - val_loss: 0.8517 - val_accuracy: 0.8442
Epoch 44/100
88/88 [==============================] - 83s 940ms/step - loss: 0.0036 - accuracy: 0
.9987 - val_loss: 1.0077 - val_accuracy: 0.8175
Epoch 45/100
88/88 [==============================] - 87s 980ms/step - loss: 0.0038 - accuracy: 0
.9988 - val_loss: 0.8309 - val_accuracy: 0.8467
Epoch 46/100
88/88 [==============================] - 84s 951ms/step - loss: 0.0031 - accuracy: 0
.9991 - val_loss: 0.9237 - val_accuracy: 0.8383
Epoch 47/100
88/88 [==============================] - 85s 958ms/step - loss: 0.0025 - accuracy: 0
.9989 - val_loss: 0.8604 - val_accuracy: 0.8417
Epoch 48/100
88/88 [==============================] - 83s 941ms/step - loss: 0.0025 - accuracy: 0
.9989 - val_loss: 0.8343 - val_accuracy: 0.8383
Epoch 49/100
88/88 [==============================] - 87s 984ms/step - loss: 0.0026 - accuracy: 0
.9989 - val_loss: 0.8914 - val_accuracy: 0.8317
```

```
Epoch 50/100
88/88 [==============================] - 87s 982ms/step - loss: 0.0026 - accuracy: 0
.9991 - val_loss: 0.9154 - val_accuracy: 0.8417
Epoch 51/100
88/88 [==============================] - 83s 942ms/step - loss: 0.0028 - accuracy: 0
.9988 - val_loss: 0.8722 - val_accuracy: 0.8375
Epoch 52/100
88/88 [==============================] - 98s 1s/step - loss: 0.0031 - accuracy: 0.998
9 - val_loss: 0.9332 - val_accuracy: 0.8333
Epoch 53/100
88/88 [==============================] - 90s 1s/step - loss: 0.0030 - accuracy: 0.998
7 - val_loss: 0.9451 - val_accuracy: 0.8275
Epoch 54/100
88/88 [==============================] - 96s 1s/step - loss: 0.0032 - accuracy: 0.998
6 - val_loss: 0.9449 - val_accuracy: 0.8267
Epoch 55/100
88/88 [==============================] - 98s 1s/step - loss: 0.0027 - accuracy: 0.998
7 - val_loss: 1.0029 - val_accuracy: 0.8250
Epoch 56/100
88/88 [==============================] - 90s 1s/step - loss: 0.2294 - accuracy: 0.920
7 - val_loss: 0.6864 - val_accuracy: 0.7792
Epoch 57/100
88/88 [==============================] - 109s 1s/step - loss: 0.1605 - accuracy: 0.94
65 - val_loss: 0.8973 - val_accuracy: 0.7950
Epoch 58/100
88/88 [==============================] - 101s 1s/step - loss: 0.0362 - accuracy: 0.98
92 - val_loss: 0.8927 - val_accuracy: 0.8392
Epoch 59/100
88/88 [==============================] - 102s 1s/step - loss: 0.0111 - accuracy: 0.99
65 - val_loss: 1.0499 - val_accuracy: 0.8225
Epoch 60/100
88/88 [==============================] - 115s 1s/step - loss: 0.0098 - accuracy: 0.99
78 - val_loss: 0.8654 - val_accuracy: 0.8458
Epoch 61/100
88/88 [==============================] - 94s 1s/step - loss: 0.0039 - accuracy: 0.999
2 - val_loss: 0.9126 - val_accuracy: 0.8400
Epoch 62/100
88/88 [==============================] - 98s 1s/step - loss: 0.0035 - accuracy: 0.998
7 - val_loss: 0.9281 - val_accuracy: 0.8308
Epoch 63/100
88/88 [==============================] - 98s 1s/step - loss: 0.0043 - accuracy: 0.998
7 - val_loss: 0.9979 - val_accuracy: 0.8408
Epoch 64/100
```

```
88/88 [==============================] - 94s 1s/step - loss: 0.0033 - accuracy: 0.998
6 - val_loss: 0.9094 - val_accuracy: 0.8450
Epoch 65/100
88/88 [==============================] - 98s 1s/step - loss: 0.0026 - accuracy: 0.998
6 - val_loss: 0.9290 - val_accuracy: 0.8458
Epoch 66/100
88/88 [==============================] - 110s 1s/step - loss: 0.0020 - accuracy: 0.99
91 - val_loss: 0.9798 - val_accuracy: 0.8442
Epoch 67/100
88/88 [==============================] - 108s 1s/step - loss: 0.0022 - accuracy: 0.99
92 - val_loss: 0.9115 - val_accuracy: 0.8425
Epoch 68/100
88/88 [==============================] - 102s 1s/step - loss: 0.0018 - accuracy: 0.99
93 - val_loss: 0.9823 - val_accuracy: 0.8375
Epoch 69/100
88/88 [==============================] - 97s 1s/step - loss: 0.0019 - accuracy: 0.998
9 - val_loss: 0.9273 - val_accuracy: 0.8467
Epoch 70/100
88/88 [==============================] - 100s 1s/step - loss: 0.0014 - accuracy: 0.99
93 - val_loss: 1.0201 - val_accuracy: 0.8317
Epoch 71/100
88/88 [==============================] - 99s 1s/step - loss: 0.0031 - accuracy: 0.998
6 - val_loss: 0.9060 - val_accuracy: 0.8458
Epoch 72/100
88/88 [==============================] - 99s 1s/step - loss: 0.0018 - accuracy: 0.999
0 - val_loss: 0.9668 - val_accuracy: 0.8400
Epoch 73/100
88/88 [==============================] - 100s 1s/step - loss: 0.0020 - accuracy: 0.99
87 - val_loss: 0.9253 - val_accuracy: 0.8467
Epoch 74/100
88/88 [==============================] - 109s 1s/step - loss: 0.0019 - accuracy: 0.99
88 - val_loss: 0.9623 - val_accuracy: 0.8433
Epoch 75/100
88/88 [==============================] - 102s 1s/step - loss: 0.0024 - accuracy: 0.99
89 - val_loss: 0.9499 - val_accuracy: 0.8417
Epoch 76/100
88/88 [==============================] - 111s 1s/step - loss: 0.0026 - accuracy: 0.99
86 - val_loss: 0.9521 - val_accuracy: 0.8358
Epoch 77/100
88/88 [==============================] - 111s 1s/step - loss: 0.0022 - accuracy: 0.99
88 - val_loss: 0.9300 - val_accuracy: 0.8458
Epoch 78/100
88/88 [==============================] - 106s 1s/step - loss: 0.0018 - accuracy: 0.99
90 - val_loss: 0.9299 - val_accuracy: 0.8458
```

```
Epoch 79/100
88/88 [==============================] - 89s 1s/step - loss: 0.0019 - accuracy: 0.998
7 - val_loss: 0.9287 - val_accuracy: 0.8467
Epoch 80/100
88/88 [==============================] - 94s 1s/step - loss: 0.0016 - accuracy: 0.998
8 - val_loss: 0.9289 - val_accuracy: 0.8467
Epoch 81/100
88/88 [==============================] - 103s 1s/step - loss: 0.0017 - accuracy: 0.99
87 - val_loss: 0.9752 - val_accuracy: 0.8475
Epoch 82/100
88/88 [==============================] - 100s 1s/step - loss: 0.0023 - accuracy: 0.99
87 - val_loss: 0.9854 - val_accuracy: 0.8458
Epoch 83/100
88/88 [==============================] - 100s 1s/step - loss: 0.0014 - accuracy: 0.99
90 - val_loss: 0.9536 - val_accuracy: 0.8467
Epoch 84/100
88/88 [==============================] - 102s 1s/step - loss: 0.0017 - accuracy: 0.99
86 - val_loss: 1.0166 - val_accuracy: 0.8383
Epoch 85/100
88/88 [==============================] - 95s 1s/step - loss: 0.0014 - accuracy: 0.999
3 - val_loss: 0.9622 - val_accuracy: 0.8450
Epoch 86/100
88/88 [==============================] - 102s 1s/step - loss: 0.0016 - accuracy: 0.99
94 - val_loss: 0.9817 - val_accuracy: 0.8442
Epoch 87/100
88/88 [==============================] - 108s 1s/step - loss: 0.0016 - accuracy: 0.99
90 - val_loss: 0.9780 - val_accuracy: 0.8450
Epoch 88/100
88/88 [==============================] - 100s 1s/step - loss: 0.0017 - accuracy: 0.99
87 - val_loss: 1.0472 - val_accuracy: 0.8367
Epoch 89/100
88/88 [==============================] - 111s 1s/step - loss: 0.0020 - accuracy: 0.99
86 - val_loss: 0.9996 - val_accuracy: 0.8325
Epoch 90/100
88/88 [==============================] - 101s 1s/step - loss: 0.0018 - accuracy: 0.99
88 - val_loss: 0.9729 - val_accuracy: 0.8467
Epoch 91/100
88/88 [==============================] - 103s 1s/step - loss: 0.0015 - accuracy: 0.99
88 - val_loss: 1.0292 - val_accuracy: 0.8425
Epoch 92/100
88/88 [==============================] - 98s 1s/step - loss: 0.0017 - accuracy: 0.998
8 - val_loss: 0.9608 - val_accuracy: 0.8442
Epoch 93/100
```

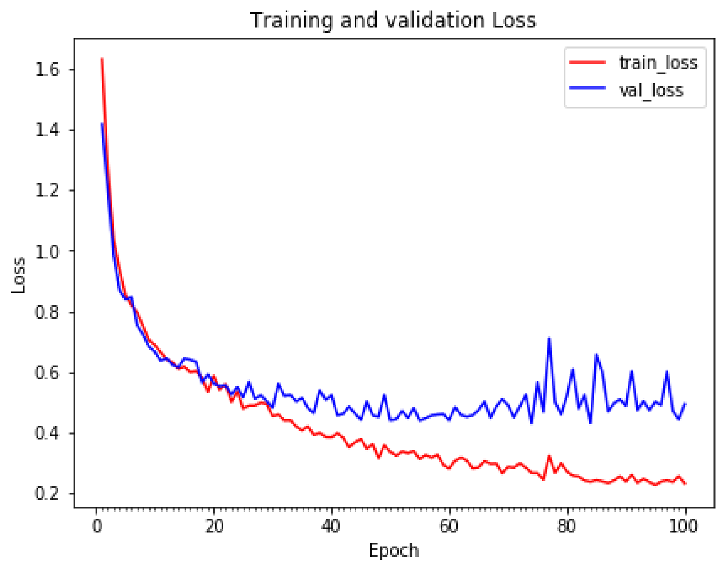
```
88/88 [==============================] - 98s 1s/step - loss: 0.0013 - accuracy: 0.998
8 - val_loss: 1.0509 - val_accuracy: 0.8442
Epoch 94/100
88/88 [==============================] - 115s 1s/step - loss: 0.0012 - accuracy: 0.99
90 - val_loss: 1.0017 - val_accuracy: 0.8442
Epoch 95/100
88/88 [==============================] - 109s 1s/step - loss: 0.0016 - accuracy: 0.99
87 - val_loss: 1.0104 - val_accuracy: 0.8467
Epoch 96/100
88/88 [==============================] - 104s 1s/step - loss: 0.0013 - accuracy: 0.99
92 - val_loss: 1.0252 - val_accuracy: 0.8442
Epoch 97/100
88/88 [==============================] - 107s 1s/step - loss: 0.0012 - accuracy: 0.99
93 - val_loss: 1.0365 - val_accuracy: 0.8450
Epoch 98/100
88/88 [==============================] - 107s 1s/step - loss: 0.0015 - accuracy: 0.99
87 - val_loss: 1.0291 - val_accuracy: 0.8483
Epoch 99/100
88/88 [==============================] - 117s 1s/step - loss: 9.8192e-04 - accuracy:
0.9991 - val_loss: 1.0540 - val_accuracy: 0.8475
Epoch 100/100
88/88 [==============================] - 116s 1s/step - loss: 0.0013 - accuracy: 0.99
87 - val_loss: 1.0488 - val_accuracy: 0.8467
```

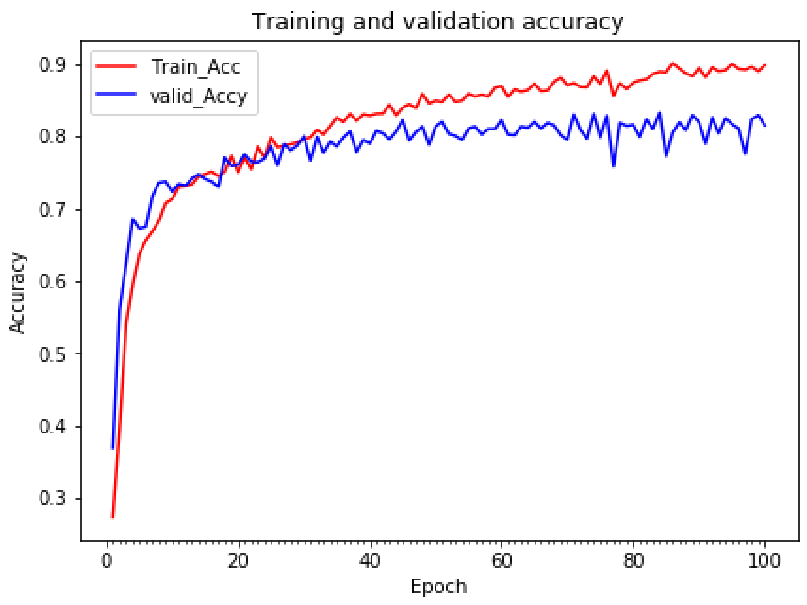
```
## Plotting Loss Curves
plt.figure(figsize=[8,6])
plt.plot(history.history['loss'],'r',linewidth=2.0)
plt.plot(history.history['val_loss'],'b',linewidth=2.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=15)
plt.xlabel('Epochs ',fontsize=13)
plt.ylabel('Loss',fontsize=13)
plt.title('Loss Curves',fontsize=13)

## Plotting Accuracy Curves
plt.figure(figsize=[8,6])
plt.plot(history.history['accuracy'],'r',linewidth=2.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=2.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=15)
plt.xlabel('Epochs ',fontsize=13)
plt.ylabel('Accuracy',fontsize=13)
plt.title('Accuracy Curves',fontsize=13)
```

## Plotting Confusion Matrix

```
classes=['Anthracnose','Healthy','leaf_blight','rust']
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
target_names = []
for key in train_generator.class_indices:
    target_names.append(key)
Y_pred = model.predict(test_generator)
Y_pred_classes = np.argmax(Y_pred,axis=1)
# Y_true = np.argmax(test_generator,axis= 1)
confusion_mtx = confusion_matrix(test_generator.classes, Y_pred_classes)
plot_confusion_matrix(confusion_mtx, classes)
```


Training and validation Loss
train_loss
val_loss
Loss
Epoch


Training and validation accuracy
Train_Acc
valid_Accy
Accuracy
Epoch