ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Tomato Leaf Diseases Detection and Classification using Convolutional Neural Network (CNN)

MASTER'S THESIS

By

Zewdu Tiumay

June 20, 2020

Adama, Ethiopia

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Tomato Leaf Diseases Detection and Classification using Convolutional Neural Network (CNN)

MASTER'S THESIS

By

Zewdu Tiumay

A THESIS SUBMITTED TO SCHOOL OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Office of Graduate Studies

Adama science and Technology University

Advisor: Mesfin Abebe (Ph.D.)

June 2020

Adama, Ethiopia

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

Tomato Leaf Diseases Detection and Classification using Convolutional Neural Network (CNN)

MASTER'S THESIS

By

Zewdu Tiumay

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTING
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

PRESENTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF
MASTERS OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

OFFICE OF GRADUATE STUDIES
ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

Advisor: Mesfin Abebe (PhD)

June 2020
Adama, Ethiopia

Declaration

I hereby declare that this MSc thesis is my original work and has not been presented in any other university and that all sources of materials used for the thesis have been fully acknowledged.

Name: Zewdu Tiumay

Signature: _____

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

Name: mesfin Abebe (PHD)

Signature: _____

Date of submission: June 20, 2020

Approval page

We, the undersigned, member of the Board of Examiners of the final open defense by Zewdu Tiumay have read and evaluated his/her thesis entitled: “Tomato Leaf Diseases Detection and Classification using Convolutional Neural Network (CNN)” and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of Masters in Computer Science and Engineering

1.	_____	_____	_____
	Name of Student	Signature	Date
2.	_____	_____	_____
	Supervisor/Advisor	Signature	Date
3.	_____	_____	_____
	Chairperson	Signature	Date
4.	_____	_____	_____
	Internal Examiner	Signature	Date
5.	_____	_____	_____
	External Examiner	Signature	Date
6.	_____	_____	_____
	School Dean	Signature	Date
7.	_____	_____	_____
	School Dean	Signature	Date
8.	_____	_____	_____
	Post graduate Dean	Signature	Date

Acknowledgment

First and foremost, I'd like to thank God next to God I'd like to thanks my adviser Mesfin Abebe (Ph.D). Without his patience and understanding over the years, I would not have made it this far. His mentorship has helped me not only succeed academically but has also taught me much in my personal life. I have been very lucky to have him as my advisor. I would also like to thank my co-adviser Tilahun Melak (Ph.D.), who has provided more hands-on help and guidance than I could have ever asked for, all with a great sense of humor.

I would also like to thanks my parents, for their understanding, love, and support. I'm proud to have such wonderful people not only be my parents but also friends. My brothers, Getahun, Tsegaye, and zeme they have always had my back, and been willing to listen to any rants I could dish out .finally and mostly I'd like to thanks, my wife.

Table of Contents

List of Figure

List of Tables

List of Equations

List of Abbreviations and Acronyms

CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Architecture
DL	Deep Learning
FC	Fully Connected
GPU	Graphical Processing Unit
LCD	Liquid-Crystal Display
ML	Machine Learning
RAM	Random Access Memory
RCNN	Region- Convolutional Neural Network
SVM	Support Vector Machine
YOLO	You Only Look Once
ANN	Artificial neural network
Adam	Adaptive learning rate optimization algorithm
CPU	Central processing unit
CNTK	Cognitive tool kit
Conv	Convolution
CL	Convolution layer
ConvNets	Convolutional neural networks
FN	False negatives
FP	False positives
FCL	Fully connected layer
GHz	Giga hertz
GB	Gigabytes
GPU	Graphical processing unit
HD	High definition
PL	Pooling layer
PPV	Positive predictive value
P	Precision

RAM	Random access memory
R	Recall
RELU	Rectified linear unit
train_accuracy	Training accuracy
train_loss	Training loss
TN	True negatives
TP	True positives
val_accuracy	Validation accuracy
val_loss	Validation loss
BCE	Binary Cross-Entropy
VGG16	Visual geometry group
CA	classification accuracy
GCP	Google cloud platform
IDE	integrated development environment

Abstract

Tomato (Lycopersiconesculentum Mill.) is one of the most grown vegetable plants in the world, second to potato. It originally came from the tropical area from Mexico to Peru. Tomato is grown in many parts of the country and also among the most important vegetable crops and its production has shown a marked increase. It became the most profitable crop providing a higher income to small scale farmers compared to other vegetable crops. There is a number of limits tomato yield some of those are caused by bacteria, vires, and fungus. Tomato leaf diseases are used to detect and classify symptoms of plant diseases, detection of tomato leaf diseases through the naked eye is ineffective, especially because there are numerous diseases. Therefore, we need to develop automatic tomato leaf diseases detection and classification. In this study we designed TomdiseasesNet model architecture based on inseption v3, VGG16, and mobile net architecture to perform tomato leaf diseases detection and classification using RGB, Grayscale, and augmented input image dataset. Deep learning become the most accurate and precise paradigms for the detection of plant diseases. Leaves of infected crops are collected and labeled according to the disease. Processing of image is performed along with pixel-wise operations to enhance the image. It is followed with feature extraction the classification of patterns of captured leaves in order to identify tomato plant leaf diseases. Six classifier labels are used as *Tomato Bacterial Spot, Tomato Early Blight, Tomato Sectorial Leaf spot, Tomato Leaf mold, Tomato Yellow Leaf, and Healthy* leaf. The features extracted are fit into the neural network with 200 epochs, 70/30 splitting ratio, and 0.0001 learning rate. TomdiseasesNet model achieved a testing accuracy 99.68%, Training accuracy 99.97%, and validation accuracy 99.78. The overall accuracy obtained 99.68% hence tomato plant leaf diseases infected by *Tomato Bacterial Spot, Tomato Early Blight, Tomato Sectorial Leaf spot, Tomato Leaf mold, and Tomato Yellow Leaf* can be detect the tomato plant leaf diseases and classify the types of diseases with a high rate of accuracy.

Keywords: ***Deep Learning, Convolutional Neural Networks, Tomato Leaf Diseases Classification.***

1. CHAPTER ONE: INTRODUCTION

1.1. Background of the Study

Tomato (Lycopersiconesculentum Mill.) is one of the most grown vegetable plants in the world, second to potato. It originally came from the tropical area from Mexico to Peru [1]. Tomato is grown in many parts of the country and also among the most important vegetable crops and its production has shown a marked increase. It became the most profitable crop providing a higher income to small scale farmers compared to other vegetable crops [2]. In Ethiopia possessing different agro-ecological conditions it grows at an altitude between 700 and 2000 meters above sea level [3]. Most intensive production is found in the rift valley, mainly along the Awash River valley and the lakes region [3] [4] this plant is grown in a 0.458 M ha area with 7.277 M MT production and 115.9 MT/ha productivity. The tomato plant is cultivated in all the seasons but typically during winter and summer seasons. The plant cannot resist severe frost. It nurtures well under an average monthly temperature range of 21^{o}C to 23^{o}C but commercially it may be grown at temperatures ranging from 18°C to 27°C Temperature and Light intensity affect the pigmentation, fruit-sets and nutritive value of the tomato fruits [5].

Its use as a found originated in Mexico and spread throughout the world following the Spanish colonization of the Americas [1]. It is important for human health including minerals, vitamins C and E, B-carotene, lycopene, flavonoids, organic acids, Phenolic, and chlorophyll. Tomato has medicinal values and is used for blood purification and curing digestive ailments. Thus, the scientific community has recently become interested in the analysis of nutrients in tomato [6].

Tomato is one of the most caring food in Ethiopia and it is popularly used for both Business and fast food and home use purposes. [1] The fresh produce is sliced and used as salad. It is also cooked for making ‘wait’. The processed products like tomato paste, tomato juice, tomato catch-up, tomato soup, and whole peel-tomato are produced in the country for local market and export. [1]There is a vast potential for the internal market for domestic fresh tomato fruit primarily in densely populated urban areas and the processing industries in the foreign market, such as Djibouti and Somalia [7]. It is one of the world’s major vegetables with a total area and production of 4.4 million ha and 115 metric tons, respectively [7]. As it is a relatively short duration plant and gives a high yield, it is economically attractive and the area under cultivation is increasing daily. Its position in the whole world is after potato both in area and production [7]

Figure 1-1 Samples of tomato plant leaf diseases; [8] [9]

Figure 1.1 shows these four diseases mostly appear on a tomato plant. In Ethiopian most common method used to detect the technique of the farmers is using naked eyes, which required experience that is depends on the farmer's knowledge. Another method is getting advice from export which is following by only 1% of the farmers due to its highest cost. Another economical option is discussed in this thesis that is using deep learning (CNN) in image processing techniques [5]. The diseases affect the plant at different growth stages in the field. Couse of tomato plant leaf diseases; Bacterial, Fungus, and viruses like Late blight (phytophthorianfestans) has a property of occurring due to the availability of suitable environmental conditions, which encompass high moisture and low temperature for tomato late blight in the surrounding GamoGofa in southern Ethiopia areas. Tomato diseases disease occurs year after year in this area and causes considerable yields losses, 63.7 - 100% in tomato fields in the GamoGofa in southern Ethiopia not only in Gamogofa or southern regen All part of Ethiopian which oucur between 18^0C to 27^0C [10].

Several factors limit tomato yield. These include pests and diseases. Plant Yield losses of 100% are common, particularly when tomatoes are infected early in development. The average global plant losses of all diseases combined were approximately 12.8% of the potential production but in tomato plant alone was subjected to a 21.8% loss. [7] In our country Ethiopia, the tomato plant disease caused a 100% or complete loss on the unimproved local cultivar, and 67.1% on susceptible cultivars [11].

On the world 170.8 million tons of tomatoes were produced in 2014, and with a yield potential of up to 48.1 tons/ha [13]. In 2016 cropping season, tomato production in Ethiopia 913,013 tons harvested from 9767.78 ha of land [10]. However the production of the crop is constrained by several biotic and abiotic factors. Hence, the average fruit yield is 15-30 ton/ha [23]. Among

biotic constraints, fungal diseases are major factor affecting the production and productivity as well as quality of the crop. Many diseases are attacking this crop, and can cause 15-95% crop loss both in lowland and highland areas [20].

Early blight and Late blight caused by phytophthora infestans is one of the moset significant constraints to tomato productions, caused up to 90% of crop losses in cool and wet weather conditions

Table 1-1 Tomato production quantity in Ethiopia from 2008-2017 *[12]*

Year	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Hectare	5,342	4,593	5,338	7,256	7,237	7,257	5,026	9,524	6,299	7,933
Tone	41,815	40,426	55,635	81,738	55,514	39,373	30,700	65,209	28,365	41,333

Deep learning, Digital image processing, and image analysis technology based on the advances in microelectronics and computers has many applications in agriculcher (Biology) and it circumvents the problems that are associated with traditional photography. Nowadays some kinds of tools this tool helps to improve the images from microscopic to the telescopic range and also offers scope for their analysis. In local cultivar detect the plant diseases and classification by naked eye observation of experts is the main approach. But, this requires continuous monitoring of experts that might be prohibitively expensive in large farms, further, in developing countries, farmers may have to go a long distance to contact experts, this makes consulting experts too expensive and time-consuming and farmers are unaware of exactly the types of diseases its check indirect from the chemical tray and error it's exactly treated or not. Automatic detection and classification of plant leaf diseases is an important research topic as it may prove benefits in monitoring plants farm, and thus automatically detect the diseases from the symptoms that appear on the plant leaves. This enables machine vision to provide image-based automatic detection and classification. The digital camera or any captured device is used to capture the image of the leaf. The detection and classification of leaf diseases have been done based on a deep learning approach. Tomato plant diseases may be broadly classified using CNN either infections or noninfectious. The demand for a high level of safety and superior quality in agricultural products is of prime concern. The foundation of quality assessment is dependent upon features of leafs such as its. [13]

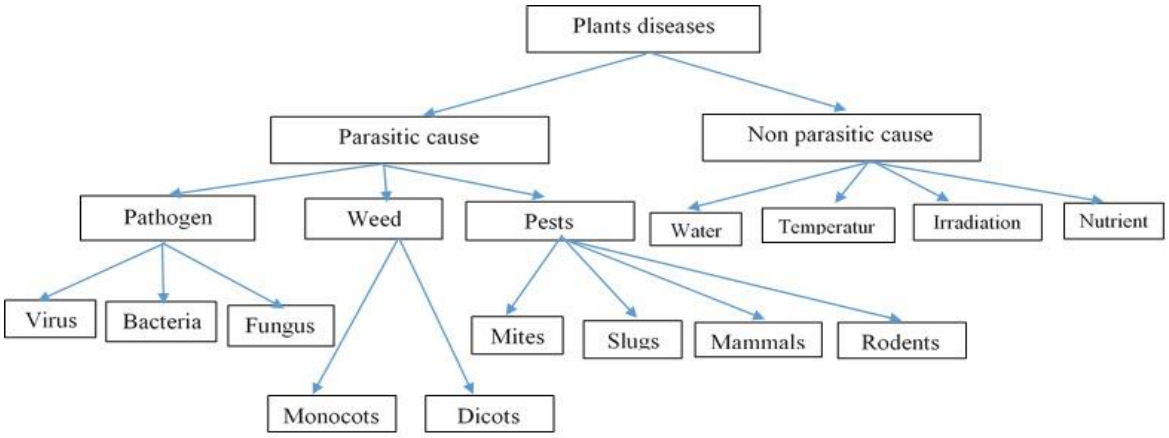

Figure 1-2 categorizations of plant diseases. [13]

1.1.1. Leaf Diseases and Symptoms

The quality of leaves defines the degree of excellence or a state of being free from defects, deficits, and substantial variations. [14] [15] Plant leaf diseases can be mostly categorized based on the nature of their key causal agent (i.e. infectious and noninfectious). Infectious leaf diseases are instigated by a pathogenic organism such as a bacterium, mycoplasma, fungus, virus, nematode, viroid, etc. an infectious agent is proficient in reproducing within or on its host and scattering from one susceptible host to another leaf host. On the other hand, noninfectious plant diseases owe their origin to critical growing conditions, the disadvantageous relationship between moisture and oxygen, excesses of temperature, toxic constituents in the soil or atmosphere, and deficiency of an essential mineral. Noninfectious causal agents are not contagious and not proficient in reproducing within the host.

Disease and their symptoms and descriptions.

1.2. The Motivation of the Study

Plant disease detection is a critical topic that has been studied over the years and is motivated by the need to produce healthy food. So now a day's some kinds of tools image processing, deep learning (conventional neural network algorithm) are the tools to detect and classify the plant leaf diseases. In our country farmers and investors detect the disease by naked eye observation of experts in the traditional approach adopted in practice for diseases. Some tomato diseases control all farms by wind within 24 hours so I motivated to study this area or topic. Because of losing cost, time, and manpower, and so on in the agriculture sector, one of the major problems in the plant diseases. The plant diseases caused by various factors such as viruses, bacteria, fungus, etc. all diseases are detected or recognize by using image processing techniques. In this topic so much

researches. The researcher reviewed research that has some gap the first one most reviewed papers are collect the dataset from the public web site from plant village and classify two-class healthy and diseases this two-class easily classify by the naked eye but not classify disease types. This research solves the plant disease to detect and classify types of diseases.

1.3. Statement of the Problem

Plant diseases have turned into a dilemma as it can cause a significant reduction in both the quality and quantity of agricultural products. In Ethiopia, the disease caused a 100% tomato plant loss on the unimproved local cultivar, and 67.1% on susceptible cultivars [11].

Agriculture is the leading sector of the economy in Ethiopia [10]; the land is cultivated by old production techniques, which lead to different structural problems. In agriculture disease, recognition is a major challenge this study focuses on tomato leaf disease detection and classification. Therefore, suitable actions have to be taken to control diseases on agricultural products while reducing; manpower, the use of chemicals to control the diseases, and losses money. In the plant, many diseases are caused by viruses, bacteria, fungus. [5] Diagnosing and detection of these plant leaf diseases are very important to cure and control the spreading of diseases. The method of diagnosing these plant diseases is based on the knowledge of farmers and experts. Nodaway's image processing and ML become the key technique for the diagnosis of various features of the plant in the areas of agriculture because it minimizes confusion and helping the expert in avoiding the abuses during diagnosis and control of plant disease has impacted society and world olden times. Plant diseases have caused enormous economic losses in each one of the countries. The plant despair by so many reasons one of this reason is a disease this plant protects the result of plant despair changes the weather condition, no food for human and animal To develop an automatic diseases detection system that takes images of the leaf via any digital camera and detects the disease of the leaf using image processing techniques and deep learning then produces a proper solution for the detected disease. ***Research Question***

- Which are suitable image processing techniques for diagnosing tomato disease?
- How to develop an automatic Tomato disease identification system based using an image analysis technique?
- Which possible algorism is classifying the tomato plant leaf diseases?
- What are the significant features that can be used to identify tomato disease?

1.4. The Objective of the Study

1.4.1. General Objective

The general objective of this study is to design and develop an efficient and more accurate tomato leaf disease detection and classification model using the Convolutional Neural Network for tomato leaf disease.

1.4.2. Specific Objectives

To achieve the main goal of this research, the following specific objectives are carried out of the study:

1. Review literature on previous studies made on plant disease identification by using image processing and deep learning techniques.
2. Collect healthy and infected Tomato leaf images directly from tomato farm and plant village.
3. Preparing the necessary data set by using different mechanisms such as image preprocessing, splitting, and Augmentation.
4. Design a convolutional neural network model to detect and classify the disease.
5. Train the proposed model by using the collected dataset.
6. Validate and Test the proposed model with a validation dataset.
7. Evaluate the effect of learning parameters including learning rate, batch size, number of epochs as well as train, validation and test split ratio

1.5. Scope and Limitation of the Study

1.5.1. The Scope of the Study

The study focuses on designing and implementing a model that can automatically detect if a tomato plant leaf is infected by one of the five types of disease and healthy, then identify or classify which type of disease, after the detection and classification of the disease.

1.5.2. The Limitation of the Study

The limitation of this proposed research focuses on five types of tomato leaf diseases *Tomato Bacterial Spot, Tomato Early Blight, Tomato Sectorial Leaf spot, Tomato Leaf mold, Tomato Yellow Leaf curl*. The proposed system mainly focuses on only the tomato plant leaf, not including Flower, Root, fruit, and others.

1.6. Significance of the Study

The study provides different benefits for different stakeholders and other researchers. After the evaluations of the result, it can be applied in different areas:

- Governmental and private industries working in agriculture benefit from the study in other way farmers and investors also beneficiaries of this study.
- The study opens a door for different researchers in Ethiopia, who have the interest to work on agro Technologies.
- The model also can be applied to other plant disease detection and classification which can be detected based on RGB and Grayscale image.
- Private industries working in agriculture can also get benefit from the study.

1.7. Organization of the Thesis

The rest of this thesis is organized as follows. Chapter two presents the review of literature on the domain of Tomato plant, Tomato Bacterial and fungus diseases, machine learning, deep learning, and studies that are conducted in those domains. At the end of chapter two reviews of studies that directly related to this thesis are discussed and summarized. In chapter three the methodologies that are used to conduct this thesis are discussed briefly including data collection methods, materials, and tools which have been used, selection of appropriate model, the architectural design of the proposed model, and evaluation techniques are discussed. Chapter four deals with the proposed model and experimental parameters on tomato plant leaf disease detection and classification remedy using infected and healthy leaf images of Tomato. In addition to these different experimental results with detailed discussions are presented. Finally, the conclusion and recommendations are presented in chapters five and six.

2. CHAPTER TWO: LITERATURE REVIEW AND RELATED WORK

2.1. Introduction

This chapter focuses on diseases that attach tomato leaf's Digital image processing techniques and approaches for real-world applications. In digital image processing, every system starts with image acquisition. After the images are captured there are several processes that we follow to reach the desired goal of a machine vision system. In this chapter, we explain the science image-based deep learning.

2.2. Tomato Plant Leaf Diseases

Tomatoes (*Solanum Lycopersicum*) can be grown on almost any moderately well-drained soil type. A good supply of organic fertilizer can increase yield and production. Tomatoes and related vegetables, such as potatoes, peppers, and eggplants, should not be planted on the same land more than ones in three years. Ideally, any cover crop or crop with tomatoes should be members of the grass family. Corn, an excellent rotation crop with tomatoes supply a large amount of organic matter and does not promote the growth of disease organisms that attack tomatoes. Certified seeds and plants are recommended and should be used whenever possible.

Table 2-1Example of Tomato leaf diseases that attack Ethiopian Horticulture

Fungal and Bacterial Diseases	***Characteristic symptoms***	***Other part affected***
Bacterial wilt	Yellowing followed by leaf necrosis and wilting of entering plant; longitudinal cutting of stem reveals brown vascular discoloration and eventual hollowness	The whole part of a plant
Early blight	On stems appears as a lesion with dark border and gray center; can encompass and girdle steam causing the death of lower canopy leaves	The whole part of the plant except roots
Leaf mold	Can result in considerable leaf defoliation unlikely resulting in plant death	The whole part of a plant
Septoria leaf spot	Dark elongated spots develop on petioles and steam leading to loss of lower canopy, and unthrifty plants; concern with overwintering inoculum for the next season	The whole part of the plant except roots
Yellow leaf curl	A devastating viral diseases of cultivated tomato in tropical and subtropical regions worldwide, and now present in the southern US; denoted by plant stunting and pronounced chlorotic leaves that curl upward	The whole part of the plant except roots

2.2.1. Bacterial Wilt

Bacteria blight is a serious disease caused by *Ralstonia solanacearum*. This bacterium survives in the soil for extended periods and enters into the roots through wounds and water in transplanting, cultivation period, or insects and through natural wounds where secondary roots emerge. Disease development is favored by high temperatures and moisture. The bacteria multiply rapidly inside the water-conducting tissue and surrounding all part of the plant, filling it with slime. This results in the rapid wilt of the plant, while the leaves stay green. If an infected stem is cut crosswise, it will look brown and tiny drops and yellowish.

2.2.2. Early Blight

This disease is caused by the fungus (*Alternaria tomatophiland A.solani*) is a scientific name and is first observed on the plants as small, brown lesions mostly on the older foliage. Leaves Spots enlarge and concentric rings in a bull's-eye pattern may be seen in the center of the diseased area. The tissue surrounding the spots may turn yellow. If high temperature and humidity occur at this time, much of the foliage is killing. Sometimes girdle the plant disease if they occur near the soil line (collar rot). On the fruits and flowers, lesions attain considerable size, usually involving nearly the entire fruit. Concentric rings are also present on the fruit and stem. The fungus survives in the soil, affecting the plant at the seedling or transplanting time, on a volunteer tomato plant and other solanaceous hosts, such as potato, eggplant, and black nightshade.

2.2.3. Leaf Mold

This disease is caused by the fungus (*passiflora fulva*) caused leaf mold. It is first observed on older leaves near the soil where air movement is poor and humidity is comfortable for the fungus. The initial symptoms are pale green or yellowish spots on the leaf surface, which enlarge and turn a distinctive yellow. Under humid temperature conditions, the spots on the leaf surfaces become covered with a gray, velvety growth of the spores produced by the fungus. When the infection is severe, the spots coalesce, and the foliage is killing. Occasionally, the fungus attacks stem blossoms flowers and fruits. Green and mature fruit can have a black, leathery rot on the stem but not in leaf end. The fungus survives on plant residue and in the soil. Spores are spread by rain, wind, animals, or tools. Seeds can be contaminated. The fungus is dependent on high relative humidity and high temperature for disease development.

2.2.4. Septoria Leaf Spot

This destructive disease of tomato foliage, petioles, and stems these diseases are not affected fruit the diseases are caused by the fungus (*Septoeialvecopersici)*. Infection usually occurs on the lower leaves near to the soil, after the plant begins to set fruit. Numerous small, circular spots with dark borders surrounding a beige-colored center appear on the first leaves. Tiny black specks, which are spore-producing bodies, can be seen in the center of the spots. Severely spotted leaves turn is yellow, die and fall off the plant. The fungus is most active when temperatures range from 68 to 77^0 F, the humidity is high, and rainfall or overhead irrigation wets the plants. Defoliation weakens the plant, reduces the size and quality of the yield, and exposes the fruit to sunscald. The fungus is not soil-borne but can overwinter on plant residue from the previous plant, decaying vegetation, and some wild hosts related to the tomato.

2.2.5. Late Blight (several genotypes)

Late blight is a potentially serious disease of tomato; it's caused by the fungus (*Phytophthora infestans)*. Late blight is especially damaging during low temperatures, wet weather. The fungus can affect the whole plant parts. In the Young leaf, lesions are small and appear as dark, water-soaked spots. These leaf spots will quickly survive and a white mold will appear at the margins of the affected area on the lower surface of leaves. Complete defoliation browning and shriveling of leaves and stems can occur within 14 days from the first symptoms. Fungal spores are spread between plants and farm area by rain and wind. A combination of daytime temperatures in the upper 70s ^{0}F with high humidity is ideal for infection

2.2.6. Tomato Yellow Leaf Curl

is not seed-borne but is transmitted by whiteflies. This disease is extremely damaging to fruit yield in tomato. Whiteflies may bring the disease in ton the farm area from infected weeds nearby, such as various nightshades and jimsonweed. After the infection of tomato plants on the leaf may be symptomless for as long as 2-3 weeks. Symptoms in tomato plants are the upward curling of leaves, yellow leaf margins, smaller leaves than normal, plant stunting, and flower drop. If tomato plants are infected by yellow leaf curl early in their growth, there may be no fruit formed. Infected, but will show no symptoms.

2.3. Computer Vision

Image is a binary representation of visual information that can be represented into the two-dimensional axis which has height, and width, and it can also be represented as a collection of pixels that form visual information, such as drawings, pictures, logos, and graphs. In the case of this study, there are two types of image sets.

2.3.1. Training Data (Image)

Training data (image set) is a data that is used to train or fit a model, and it is not recommended to use training data, because it is more likely biased since the model already has seen the data during training the model.

2.3.2. Test Data (Image)

It is a set of unbiased or unseen data used to test our model and used to provide an unbiased evaluation of our final model. Test data should be data that is different from the data that is used for training the model. Using test data helps our model to be more robust and effective because the model evaluates the result of unseen data.

2.4. Image Processing

Digital image processing is a branch of computer vision in which images are converted into an array of integers or pixels [16][[17] or processing a digital image by using a digital computer so that we can get an enhanced image which we can extract useful information from it

2.5. Image Pre-Processing

The pre-processing techniques are accomplished to make the image applicable for further processing. The image is resized in the preprocessing step of plant disease detection [18]. The processing of images consists of image enhancement, color conversion, removal of noise [30]. In Image enhancement, the quality of the image is enhanced to increase the visuality. In color space conversion, the RGB image is converted into greyscale using various color models such as CIELAB, YCbCr, and HSV. For noise removal, various filters are used [19]. The RGB image converted into CIELAB, YCbCr, and HSV because RGB is a device-dependent color space, and the image processing system needs images in device-independent color space models [20]. After performing the resizing, color space conversion, and enhancement, Histogram equalization methods are used to designate intensities [21].

2.6. Image Segmentation

In Image Segmentation, the image is segmented into various parts based on the similarity between various features. The part having the same features are grouped [22], [23] [24]. The image can be analyzed easily with segmentation. Image segmentation can be Local segmentation in which a specific part of the image is considered and Global segmentation in which the whole image is considered [25].

2.7. Feature Extraction

The feature extraction step retains the various image features that are passed to the classifier as its input. Classifier easily classifies the image data into clusters by using features. Features can be grouped into classes:

Local Features: - Geometric features such as concave or convex, endpoints, etc. Belongs to the category of local features.

Global Features: - Topological features such as connectivity, number of holes, etc. belong to the category of global features [26].

2.8. Image Detection

Detection is the process of detecting or noticing something in the real world. So Detection in this study is used to refer to detecting if a tomato plant leaf is infected with diseased or healthy

2.9. Deep Learning over Machine Learning Approach

2.9.1. Machine Learning Technique

Machine learning is a branch of Artificial intelligence which is a way of making machines intelligent to learn from a bunch of learning algorithms and apply what they have learned to make brilliant decisions like a human being, but as long as they are still a machine and in some situations, they need human intervention, because they work only for what they are designed for no less no more. That's why we need deep learning, which gives us a bit better performance.

2.9.2. Deep Learning

Deep learning is an advanced class of machine learning algorithms that have utilized several sequential layers. Each layer uses the output of the preceding layer as input. The learning process can be unsupervised, supervised, or semi-supervised. LeCun et al define deep learning as a

representation learning method [27]. Representation learning algorithms make optimizations to find the most convenient way to represent the data [28]. Deep learning does not have to divide the feature extraction and the classification because of the model automatically extracts the features while training the model. It is used in many research areas such aces as image processing, image restoration, speech recognition, natural language processing, and bioinformatics. Deep learning is a subpart of Machine Learning, for individual definitions.

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

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

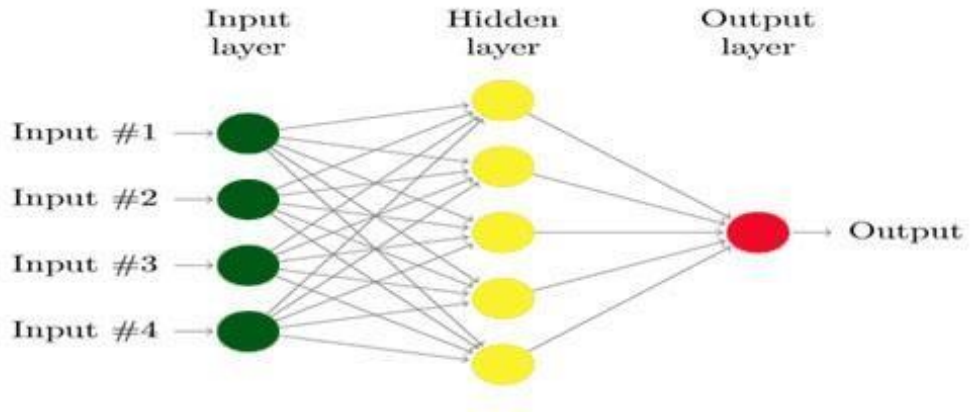

Figure 2-1neural network layer

2.9.2.1. Role of Deep Learning in Image Processing

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

2.9.2.2. Convolutional Neural Network (CNN)

A convolutional neural network is preferred as a deep learning method in this study. CNN, which can easily identify and classify objects with minimal pre-processing, is successful in analyzing the visual image and can easily separate the required features with its multi-layered structure. It consists of four main layers: - convolutional layer, pooling layer, activation function layer, and fully connected layer. Fig 2.2 shows a general CNN architecture. Layers of CNN are: - Convolution layer, Activation layer (e.g. using ReLu), pooling layer, Dropout Layer, fully connected Layer

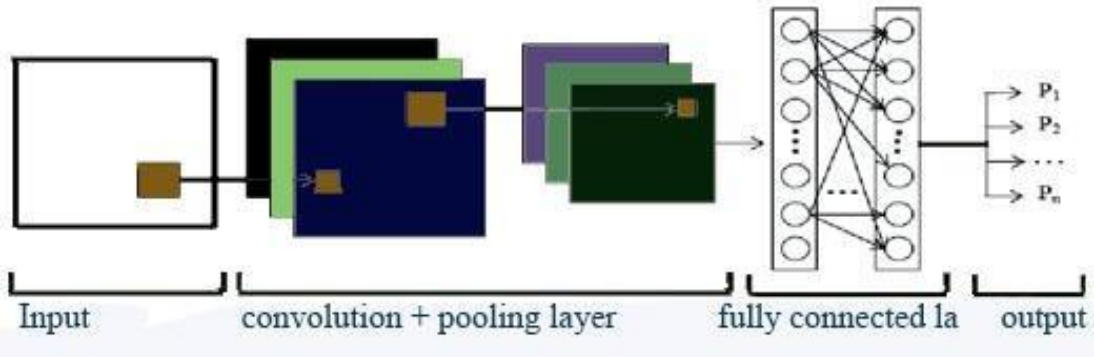

Figure 2-2 A general CNN architecture

(a) Convolution Layer

A filter or kernel is used to run over the image in fixed gap intervals called strides. Selecting the size of stride is crucial to achieving the desired results. During running the filter over the image, the dot product of filter with part of the image on which filter lies is calculated. Then the sum of all values of the product matrix is copied to the corresponding position in the convolved feature map matrix. Thus, we get a reduced dimension feature map of the image. Filters may be of many kinds, where each filter is used to extract a different kind of feature from the image e.g. one filter may be responsible for extracting one kind of feature from the image based on shapes, colors, edges and another filter may be used to extract feature based on color intensities.

(b) Parameters which helps in adjusting CNN's performance

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

- ***Padding***: - It is a process of adding zeroes around the border of the original image symmetrically. This helps us obtain the feature map output to be of a size as per our requirement. Commonly it is used to preserve the dimension of the image after convolution.
- ***Filter:*** - These are also called kernels. These may be of many types. Each filter increases the depth of the output generated after convolution. So, if we are using 3 filters then the depth of the output will be 3. There are 3 parameters on which the output of convolution depends on stride, Depth, and padding. We must finely tune these parameters to obtain the desired output.

(c) Dimension Reduction in CNN

While convolution operation, the dimension of the input image is reduced after a filter matrix is convoluted over the input image. It also increases the depth of the image as this helps in identifying the regions in the image. E.g. a9*9 RGB image with 3, 3*3 kernels at stride 1(per channel), will produce an RGB feature of 7*7*3. Here 3rd dimension represents the depth of the convolved image.

(d) Pooling Layer

This layer operates a small kernel on the image at a fixed stride. It is used to pick the pixel with the highest intensity and discard other pixels. The resultant matrix will be a reduced dimensional matrix of the feature image. This helps in reducing the unnecessary sparse cells of the image which are of no use in classification. Max pooling helps reduce the dimensionality of the network (or image), but this may cause some information loss. The concept behind these is that adjacent or nearby pixels can be approximated by the maximum information-carrying pixel.

(e) Activation Layer

This layer mostly uses ReLu, softmax, sigmoid as an activation. Function ReLu is a function that is used to set all negative values to zero and keeps positive value as it is. This step is followed by the convolution and pooling layer. In numerous fields such as computer vision, they are becoming the state of art achieving near human or better performance. They sound interested but designing a CNN is a herculean task in itself. Till now there is no fixed formula for the design of CNN. Many researchers have come up with general suggestions such as. But they don't always hold, as the task-dependent as importantly on data as it does on the algorithm. CNN exploits the

spatial hierarchical feature of data, extracting features, and helps classify them into different classes. This has led to the development of a stream of data augmentation and pre-processing to increase the data, as more data allows for the chance of better training and avoiding overfitting. This helps build models that are more robust to new samples as we try to make it mare generalized to noise at the training phase.

(f) Dropout Layer

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

(g) Fully Connected Layer

The structure from top to down usually forms a pyramid structure, the number of parameters in these layers keep on converging until they finally reach the number of desired classes. Increasing the number of hidden units in the layer can increase the learning ability of the network, but there is a saturation of the increase in the accuracy of the network. There is no formulation of the units you choose as it is a hit and trial usually. These also pile up with the fully connected subset of the network. Most of the network in research usually performs well with the number of units in multiples of 64. Two to three layers networks are good if enough patterns are being passed to the network after flattening of the convolutional layers.

2.10. Related Works

Here, we take some of the papers related to plant leaf disease detection and classification using various advanced techniques, and some of them shown below.

Usama Mokhtar [29] the authors are present as an efficient method that identifies whether a tomato leaf is healthy or infected. The image given as input was first pre-processed by removing the un wanted background and the noise present was eliminated with the help of erosion technique. Gray level Co-occurrence Matrix (GLCM) was used for texture feature extraction from the enhanced image. The support vector machine (SVM) classifier was trained using different kernel functions and the performance has been evaluated using the N-fold cross-validation technique. The proposed system has achieved an accuracy of 99.83% using the linear

kernel function with the SVM classifier. Even though the obtained accuracy is high, it is not sufficient enough to predict or differentiate between the health of diseased leaves. Also, the type of diseases was not identified.

S.D. Khiladi [30] the authors are presented various segmentation, feature extraction, and classification techniques that identify and detect the type of diseases using the diseased image to conduct classification. The leaf image given as input to the system was pre-processed by something enhancing the image by performing histogram equalization. To obtain the affected area using different segmentation techniques like K-Means clustering have been proposed. The feature was then extracted from the segmented region and calculated using GLCM. After feature extraction, the diseases can be detected with the help of artificial neural networks (ANN) or Buck propagation neural networks. The drawback of segmenting the image using K-Means clustering is that the process proposed was semi-automated as the user has to explicitly select the cluster which contains the diseased part.

Juan F.Molina et al. [31] presented a color-based strategy to detect an early blight disease or any kind of infection on tomato leaf. The auteur proposed method based on the categorization of tomato leaves with the help of color descriptors. Color characterization is done using color structure descriptors i.e. color histogram by quantifies the amount of significant color using the Hue-MaxMin method, scalable color descriptor i.e. leading spatial color distribution and color layout description or color pattern variation using YCbCr color space transform. After calculation of all these features or descriptor values, a novel strategy based on nested leaf one out cross-validation method is used to achieve a better classification ratio. The individual descriptor configuration evaluation performed using an inner loop permit, while the out loop measures performance assessment between different descriptors. The author concluded that the color structure descriptor provided better accuracy than other methods.

Usama Mokhtar [32] author presents the image processing technique for tomato leaves disease detection. In the image acquisition phase, digital images of infected tomato leaves are collected which include two types of tomato diseases namely: Early blight and powdery mildew. In the preprocessing phase, some techniques are applied for image enhancement, smoothness; remove noise, image resizing, image isolation, and background removing. The author introduced the

Gabor wavelet transformation and SVM for the identification and classification of tomato diseases. In the feature extraction phase with the help of Gabor wavelet transform feature vectors are obtained for the next classification phase. In the classification phase, the support vector machine (SVM) is trained for identifying the category of tomato diseases. The input of SVM is feature vector and corresponding classes, whereas the outputs are the decision that that detects tomato's leaf disease. SVM is employed using Invmult Kernel, Cauchy kernel, and Laplacian kernel function. Grid search and N-fold cross-validation techniques are used for performance evaluation metrics.

Sachin D [33] authors described disease detection, in which image processing is the first step for obtaining an image in digital form and pre-processing to remove noise and another object from image. Pre-processing also converts RGB images into gray images using equation f(x) =0.29889*R+0.5870*G+0.114*B and makes histogram equalization. Image segmentation is done using boundary and spot detection algorisms for finding infected parts of the leaf. Classifications of objects are done using the k-means clustering method. Otsu threshold algorithm is used for thresholding which creates binary images from gray images. With the help of feature extraction color texture, shape, morphology, edges are used in plant disease detection. Leaf color extraction using the H&B component and color Co-occurrence method is feature extraction methods in image processing. Classifications of diseases are done using an artificial neural network (ANN) and Backpropagation network.

K Muthukannan et al. [34] defined a fuzzy rule-based technique using the color feature for the classification of the disease of tomato leaf. Firstly, the gradient operator is used for preprocessing to suppress noise or small function in the image. Two important features i.e. mean and standard deviation are extracted from a cropped image with different image size samples in the last, Fuzzy inference system using fuzzy rules with selected color features classified the image region as healthy and highly affected disease portion of the plant leaf. Where orange color indicates a healthy portion of the image, yellow and red color indicated the highly affected region of the leaf. The author concluded the performance of fuzzy rule-based classification is satisfactory. The experiment results show the proposed method can detect leaf diseases with little computational effort.

Prajakta Mitkal [35] author present image processing techniques for detecting diseases of Sugarcane leaf. The author chooses 6 types of disease of the experiment, they are Brown spot, Downy mildew, sugarcane mosaic, red stripe, red rot, and downy fungal. In image acquisition, images are captured in better quality resolutions with a format such as TIF, PNG, JPEG, and BMP for image-analysis. In preprocessing RGB image are converted to grayscale and unwanted part of data from the image is removed. Segmentation locates a healthy area of the given image which contains green pixels and potentially infected area. Three algorithms namely SVM, Nonlinear SVM, and Multiclass SVM are used in feature extraction for disease detection.

David Hughes et al. [36], used GoogleNet and AlexNet models to train 54,306 images from the plant village website, in which GoogleNet performs better and consistently with a training accuracy of 99.35%. But in this study, the accuracy degrades to 31.4% when it is tested using images taken under conditions different from images used for training the model. In this study, we have used three train test split distribution of 75/25, 60/40, and 70/30 in percent with three types of image types which are RGB color images, grayscale images, and segmented images

Mosisa Desalegn [37]authors present used deep learning approach wheat ret diseases detection of leaf Authors choose three types of diseases of the experiment they are yellow Rust, leaf rust, and stem rust. In image acquisition, images are collected from three agricultural research institutes in Ethiopia namely, Kolomsa Agricultural Research Institute, Bishoftu Agricultural Research Institute, and Ambo Agricultural Research Institute. In pre-processing used RGB image, Grayscale image, and color segmented image he develops MosNet model based on CNN architecture. Classification of diseases is done using CNN and archived high accuracy 99.76 but in this paper have only two class healthy and diseases because of the shortage of dataset

Critique of sardogan et al. (2018) shows the implementation of learning vector Quantization algorithm along with the CNN in classifying the tomato leaf diseases with using the RGB values of the pixels in the images to convolution every single image. The LVQ algorithm used full 500 features vectors to train and test methods acquired from the initial images. Additionally, max pooling and ReLU activation function are used to improve the model performance which in return classify the image accurately. Another study

Zhang et al. (2019) used CNN to classify vegetable leaf diseases, by using three-channel convolutional neural network (TCCNN) for every single RGB color on the infected leaf. Finally, softmax, classifier layer classifies the diseases.

Table 2-2 Summary of related work

Author	***Title***	***Methodology used***	***Accuracy***	***Gap***
Usama Mokhtar	SVM-based detection of tomato leaves diseases". In: Intelligent Systems	-Gray level Co-occurrence Matrix (GLCM) -Support vector machine (SVM) classifier -N-fold cross-validation technique. For evaluation	99.83%	-the obtained accuracy is high, it is not sufficient enough to predict or differentiate between healthy of diseased leaves. Also, the type of diseases was not identified
InH Sabrol and K Satish	Tomato plant disease classification in digital images using classification tree	-Otsu's method for image segmentation -Supervised learning techniques for classification	97.3%	-overfitting in case of noisy data -the amount of control that the user has over the model is relatively less.
Mosisa Desalegn	Deep learning approach wheat rust disease detection.	-MosNet models used to train -is using the train from scratch	99.76%	-the obtained accuracy is high, it is not sufficient enough to predict or differentiate between healthy of diseased leaves. Also, the type of diseases was not identified
Serdogan, m., Tuncer	Plant leaf disease detection and classification based on LVQ Algorithm	Learning vector Quantization (LQV)		Cannot classify different classes of diseases Dataset size is too small
Santhosh Kumar, S, and Raghaven dra, B.K. (2019)	Diseases detection of various plant leaf using image processing Techniques: A Review	Image processing techniques		To many features for single image No performance comparotion

Some researchers used so many techniques for disease detection and classification like SVM, Naïve Byes, DT, KNN, Random Forest, AdaBoost, Fuzzy classifier, NN ruled based classification. Every technique has certain limitations. Those are used public datasets, and others focused on only two classes healthy or not healthy. They are not classified as the type of diseases so to overcome the drawback of these different techniques fusion of various techniques is a good idea. In this proposed approach, we fused on the blending of three techniques to get accurate results with the fastest speed.

This system developed to capture the image. Also, it is found that using technology-based data preprocessing and data augmentation is used to remove over-feting and has good results and better performance. This research is only capable of detecting five types of disease classes and healthy tomato leaves. To detect other types of classes of diseases, more and more new data has to be used to train the proposed network model.

3. CHAPTER THREE: RESEARCH METHODOLOGY

3.1. General Approach

The methodological analysis of diagnosing the tomato leaf diseases is shown in Figure.3.1 and includes the following steps, such as

- Review literature on previous studies made on plant disease identification by using image processing and deep learning techniques.
- Collect healthy and infected Tomato leaf images directly from tomato farm and plant village web site
- Preparing the necessary data set by using different mechanisms such as image preprocessing, splitting, and Augmentation.
- Design a convolutional neural network model to detect and classify the disease.
- Train the proposed model by using the collected dataset.
- Validate and Test the proposed model with a validation dataset.
- Evaluate the effect of learning parameters including learning rate, batch size, number of epochs as well as train, validation and test split ratio

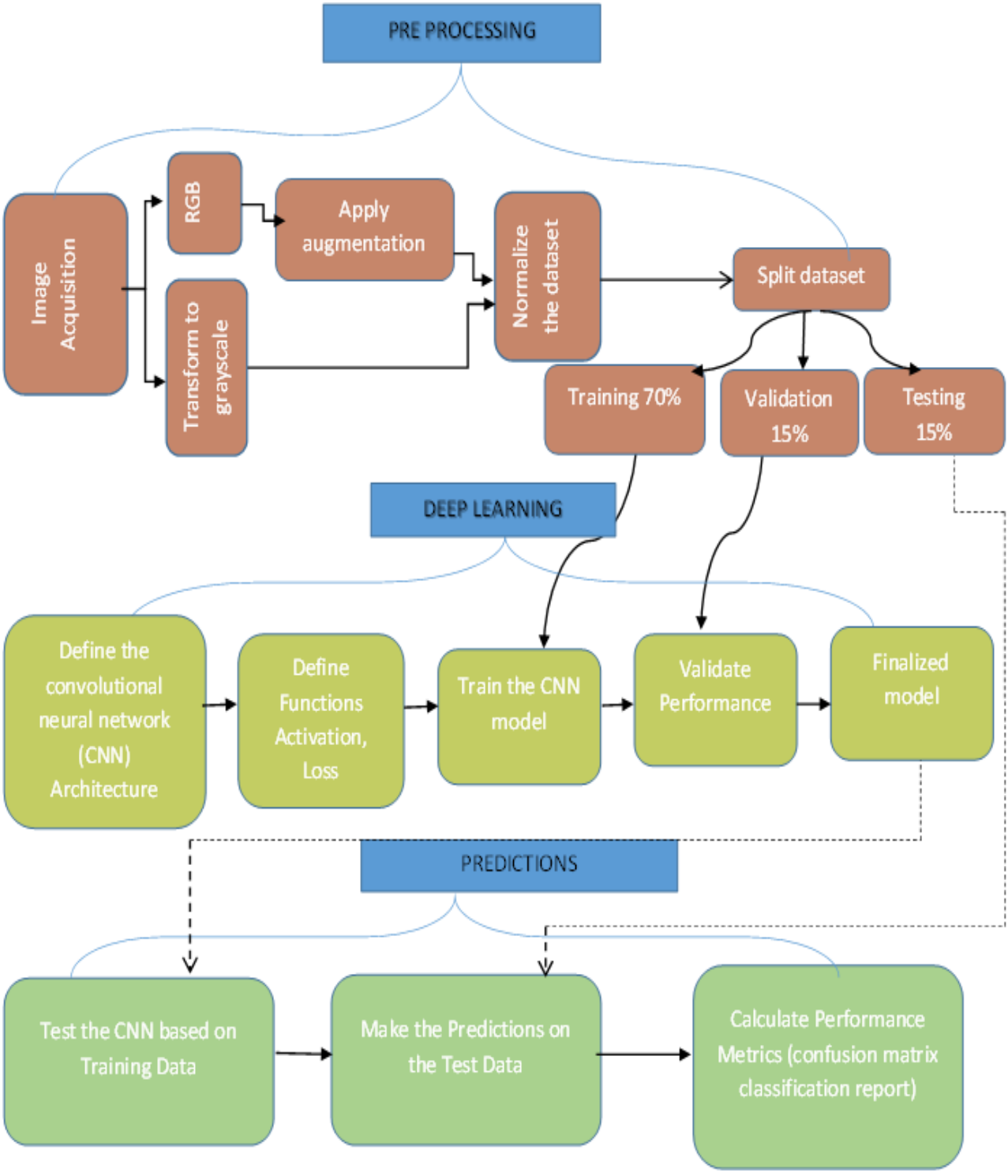

Figure 3-1Block Diagram Of research flow

3.2. Problem Formulation

This part identifies the main limitation in the current research related to tomato plant leaf detection and classification and finds out the possible solution for the identified problem. The research is focused on tomato plant leaf disease detection, classification, and medical remedy. The main aim of this study is to predict the condition of tomato leaf which has a short life span. To achieve this, a deep learning-based convolutional neural network model will be trained on the images, collected from a digital camera, and labeled according to the researcher's visual perspective of the image. The performance of the Deep learning model will be evaluated based on the Confusion matrices and classification accuracy.

3.3. Literature Review

By searching different related literature (Internet, Books, Journals, etc.) will be reviewed to understand the concept of image processing and CNN how they applied to solve the related problem in the prior research. To achieve this research objective the most recent and current literature will be reviewed. During the revision of this literature, the gap of the previous solution will be identified to use input for this proposed solution. This is the important and necessary content where all possible reference and journal related to research is investigated and analyzed.

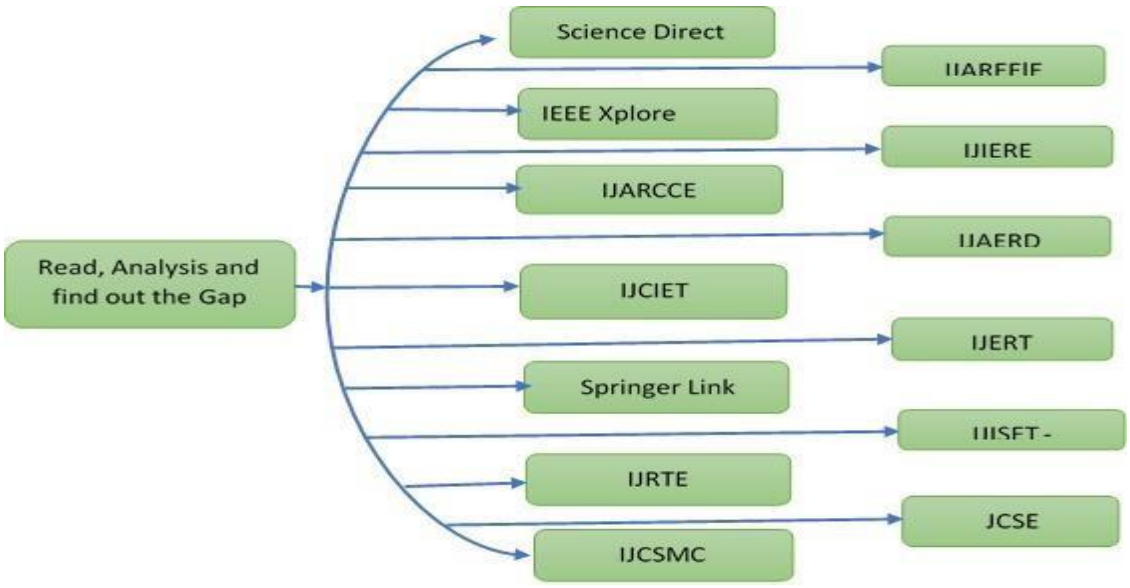

Figure 3-2 Digital library for the proposed research

3.4. Material and Tools

3.4.1. Software Tools

Investigation of available software tools with their libraries is conducted to select the appropriate tool for the implementation of the CNN algorithm for tomato image classification. During the investigation, we have seen that there are tools that are general for both deep learning and machine learning algorithms and specific only for one of them. Before selecting the tools, we have considered some criteria which are helpful to select the appropriate software tools with their corresponding libraries. The other criteria are to select tools with enough learning materials such as free video tutorials, existing experience, and the other one is the tools that must be used in machines with limited resources (like CPU only). Software tools that we have used to implement the CNN algorithm are python as a programing language with TensorFlow and Keras libraries on anaconda environment. These tools fulfill all the consideration criteria's and they are used in python which is familiar to us. **Training from scratch**

Anaconda is used for the implementation of the model and it is a free and open-source distribution of the python and R programming languages for image processing, data science, machine learning, and related applications that aim to add and simplify package management and deployment. It contains different IDE's which are used to write the coding part such as Qt Console, Jupyter Notebook, Visual studio, and Spyder. We have used the Jupyter notebook and visual studio to implement the coding part. It is easy to use and run in a web browser.

TensorFlow is a free and open-source library developed by Google and it is currently the most famous and fastest deep learning library [38]. It can be used in any desktop which runs Windows, macOS, Linux, in the cloud as a service and mobile devices like iOS and Android. The architecture of TensorFlow works for the preprocessing of the data, building the model, train the model, and estimate the model. All the Computation in TensorFlow involves tensors (n-dimension array) that represent all kinds of data. TensorFlow also uses a graph framework for graphical representation of the series of computations during the training. It has two distributions for CPU and Google collab.

Keras is a high-level neural network API that is written in python which runs on the top of TensorFlow either Theano or Microsoft Cognitive Toolkit (CNTK). It is very simple to develop a Model, user-friendly, easily extensible with python and most importantly it contains pre-trained CNN models such as VGG16 and Inception that we during the experiment. It allows easy and fast prototyping, and support both CNN and ENN or the combination of the two [38]

Visio 2016 is used for designing the system architecture. This tool was used to design a diagram and to create a Gant chart easily with ready-made templates and helping to simplify complex information

Figure 3-3 some open source libraries and our thesis Environment

3.4.2. Hardware Tools

Any digital camera was used to capture the sample images from the field. To implement the CNN algorithm with the selected software tools a very slow machine with CPU Intel(R) Core(TM) i7-7500 CPU @ 2.70GHz 2.90GHz processor, memory 8 GB.

3.5. Initialization

Choosing the proper initialization for each layer is the key to achieving high performance. Appropriately initializing the weights can be significantly influential to how easily the network learns from the training as it effectively preselects the initial position of the model parameters for the loss function to optimize in designing the model to experiment with the network parameters are initialized with Xavier sometimes called Glorot initialization. The key idea behind this initialization scheme is to make it easier for a signal to pass through the layer during forward as well as backward propagation, for a linear activation this also works nicely for sigmoid activations because of the interval where they are unsaturated is roughly linear as well.

3.6. Image Acquisition and Dataset

A dataset was created with images captured using smart phone 13Mga pixel resolution. These image were acquired from three different farms cultivating Meki, Alemtena, and awash melkasa during three different time periods (July-august, October-November, and April-May) in order to minimize the loss of diseases features due to specular reflection. The obtained image were manually segmented using adobe software to remove background information. This is done to eliminate the learning of features from background information and evaluate the sensitivity of the deep learning model in learning specific features of the disease. The images were preprocessed to the accepted input dimensions of the deep learning models (specifically 256 x 256 x 3). The dataset created using these preprocessed images consisted of six different classes' this are five types of diseases and healthy tomato leaf. As shown in table 3.1. The performance of the deep learning models depends heavily on the quantity of the data. The lack of enough input data results in overfitting of the learnt model which will perform poorly in case of validation. As per the literature, there is no clarity in determining the number of images required for classification [39]

The dataset for this experiment does not exist and has to be created by capturing the images of the tomato leaf different tomato plant farm and from plant village public web site. The dataset contains images with several diseases and healthy in tomato leaf plants. In this system, we extracted our dataset from the well-known plant village dataset and simply taking pictures using a digital camera that contains nearly 10640 images of 5 types of tomato plant leaf

diseases. Our dataset contains samples for 5 types of tomato plant leaf disease in addition to a healthy leaf, 6 classes in total as follows [40] 1, class Bacterial Spot. 2, Early Blight. 3, Healthy. 4, Septoria Leaf spot. 5, Leaf mold. 6, Yellow Leaf curl

Table 3-1 the disease name and number of corresponding images in plant village dataset

Disease name		***Tomato Bacterial Spot***	***Tomato Early Blight.***	***Tomato Healthy***	***Tomato Septorial Leaf spot***	***Tomato Leaf mold***	***Tomato Yellow Leaf curl***
Number of images	Plant village data	2017	900	1491	1671	852	3109
	Collected data	100	100	100	100	100	100

3.7. Image Pre-processing

Our collected dataset and Plant village datasets are arbitrarily sized RGB image. As most of the neural network models assume a square shape input image, it is reshaped to 256x256 pixels maintaining a uniform aspect ratio. Removing low-frequency background noise, normalizing the intensity. Image processing involved cropping of all the images manually, making the square around the leaves It is ensured that the image contains all the required information for feature extraction.

3.8. Image Augmentation

In most cases of image classification research, the dataset is very large. Data augmentation is a process of increasing the number of training data points in a dataset by generating more data from the RGB (original dataset) [41] it is important to increase the number of datasets. It helps the network to learn more complex features from the data and prevent the problem of Over-fitting. In this study, various data augmentation technique has been performed on the original images dataset.

3.9. Feature Extraction

After the preprocessing, the disease portion from the image is extracted. This leaf disease area is treated as a region of intersect for image processing. Then, further feathers are extracted based on the disease symptoms that are used to detect the disease types.

3.10. Optimization Algorithm

Below we are going to describe the most common optimization algorithm we can find why important to is optimized our computation. Because if we have thousands or even millions of images, the backpropagation step will be very slow because we won't update the weights and biases until we have computed gradient descent for the whole data set

3.10.1. Mini-batch Gradient Descent and Stochastic Gradient Descent

Instead of waiting until all computations have finished, we can update when a small number of images have been used to learn. We can split the whole data into small pieces, which we call mini-batches. And instead of computing gradient descent for the full data set, we can compute it, at each interaction, for a mini-batch, but, most important, we will update all weights and biases with the values we get using the only batch at each step (also called an epoch). If we use only, instead of m images for each mini-batch, only one image, we have the Stochastic Gradient Descent. This is the one that we have been used in this thesis, as can be seen at the previous line, of code optimized. SGC if m is equal to the size of the full data set, we have the same version of gradient descent as we didn't use any optimization, call Batch GD. The SGD has an obvious disadvantage it doesn't cover to a global minimum instead of that, the cost variation over time has a lot of noise, but the average is also decreasing .and this instead of the regular version of GD

3.10.2. Other Optimization Algorithms

There are several optimizations we can use trying to increase the performance without decreasing a lot of accuracy of our network. Whereas there is a lot of different optimization algorism [42], the most common and cited in papers are maybe GD with momentum, RMS prop, and adam

3.11. Performance Measurement Methods

3.11.1. Loss Function

the loss function is an important factor into account when solving a classification or recreation problem and Its to take into the crucial element [43] in convolutional neural networks, which is used to measure the inconsistency between the predicted value and the actual label. It is a positive value, where the robustness of the model increases along with the decrease of the

value of loss function. The loss function is the hardcore of empirical risk function as well as a significant component of the structural risk function.

3.11.2. Cross-Entropy

Is the default loss function to use for binary classification problems it is intended for use binary classification where the target values between 0 and 1. Mathematically it is the preferred loss function under the inference framework of maximum likelihood. It is the loss function to be evaluated first and only changed. It will calculate a score that summarizes the average difference between the actual and predicted probability distributions for predicting class1. The score is minimized loss and a perfect cross-entropy value is 0. It can be specified as the loss function in Keras by specifying binary cross-entropy at compiling the mathematical computation behind [44] cross-entropy is tough, but fortunately, it is easy to understand and implement it.

3.12. Regularization Techniques

Neural networks are prone to overfit, i.e. they model the training data too well and performed poorly on unseen data. One way to overcome this problem is to apply regularization. The two most popular regularization techniques are dropout and early stopping.

3.12.1. Dropout

This proposed as an effective way to regularize deep NNs by introducing stochasticity in neuron activations. Specifically, dropout randomly drops the value of neurons with a certain rate and sets the values to zero during network optimization. As a result, the network can generalize better and is less prone to overfitting.

3.12.2. Early Stopping

As explained earlier, an overfitting model performs well on training data, but poorly on unseen data. During training, the performance of the network on unseen data is evaluated with a validation set, i.e. a part of the dataset that is representative of the test set but that has not yet been seen by the model. An overfitting network increases the validation error while the training error steadily decreases. By keeping track of the lowest validation error after each epoch, one can halt the training when the model performance on the validation set hasn’t improved for some epochs.

3.12.3. Regularization Approaches

Deep and large enough neural networks can memorize any data. During training, their accuracy on the training set typically converges towards perfection while it degrades on the test set. This phenomenon is called overfitting. These different regularization approaches used to handle this overfitting problem.

3.13. Evaluation Metrics to Evaluate the Accuracy of Model

This experiment uses the F1 score and the accuracy to evaluate the performance of the model. The model is trained on 80% of the data while 15% is used for validation and 5% for testing. The metrics used to evaluate the model in this classification task are

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

Accuracy**: -** the model will be calculated as the percentage of correct prediction of the top class (the class having the highest probability as indicated by the CNN model) and the target class assigned by the author before-hand is the same. It can be represented by the below e

Equation 3-1 Accuracy

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

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

Precision: - is calculated as the fraction of true positives (TP) from the sum of the relevant classes, i.e. the sum of the true positives and the false positives. It can be represented by the following formula

Equation 3-2 Precision

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

Recall: **-** is calculated as the fraction of true positives from the sum of True positives and False Negatives. It can be represented by the below

Equation 3-3 Recall

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

***F1 score*:-**will be used in this experiment as the dataset is imbalanced. F1 score is interpreted as the harmonic mean of precision and Recall. It can be represented by the below

Equation 3-4 F1 Score

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

***A confusion matrix*: -** is also an evaluation metric that is used to describe the performance of a classification task. Precision, Recall, and Accuracy can all be calculated with the help of a confusion matrix

Table 3-2 confusion matrix

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

4. CHAPTER FOUR: RESEARCH DESIGN

4.1. Introduction

This chapter focuses on the design of the proposed work and its experimental parameters. Specifically, the design of the proposed model and descriptions, how features are extracted, and classification is performed in the proposed model and other pertained models by using the technique called training from scratch are described briefly.

4.2. Model Selection

A deep learning algorithm which is CNN is chosen on different literature that was conducted in computer vision especially in image classification. CNN's represent an interesting method for adaptive image processing. The algorithm is used for feature extraction, classification, training, testing as well as for evaluating the accuracy of the model. CNN's take a data, without the need for separate pre-processing or feature extraction stage and with the preprocessing data. In addition to these, feature extraction and classification stages occur naturally within a single framework.

As the main advantage of using the CNN algorithm for the detection and classification of tomato plant leaf diseases, it is more and automated than classical machine learning algorithms for [44]. In the classical machine learning algorithm, there is a need to develop different algorithms for different problems, therefore it uses more handcrafted algorithms, but on CNN once we developed an algorithm for the detection of Bacterial and fungus for tomato plant [45].

4.3. Overview of Tomato Leaf Diseases Detection

Before starting the detection, process two main phases are done: Training and Testing. In the training phase of the network, the CNN algorithm accepts size normalized images contained in the training and validation dataset. In this phase, more training data are generated to fit the CNN model by using data augmentation techniques specified in Table 4.1 the following figure illustrates the overall architecture of the process of tomato detection and classification using CNN algorithm.

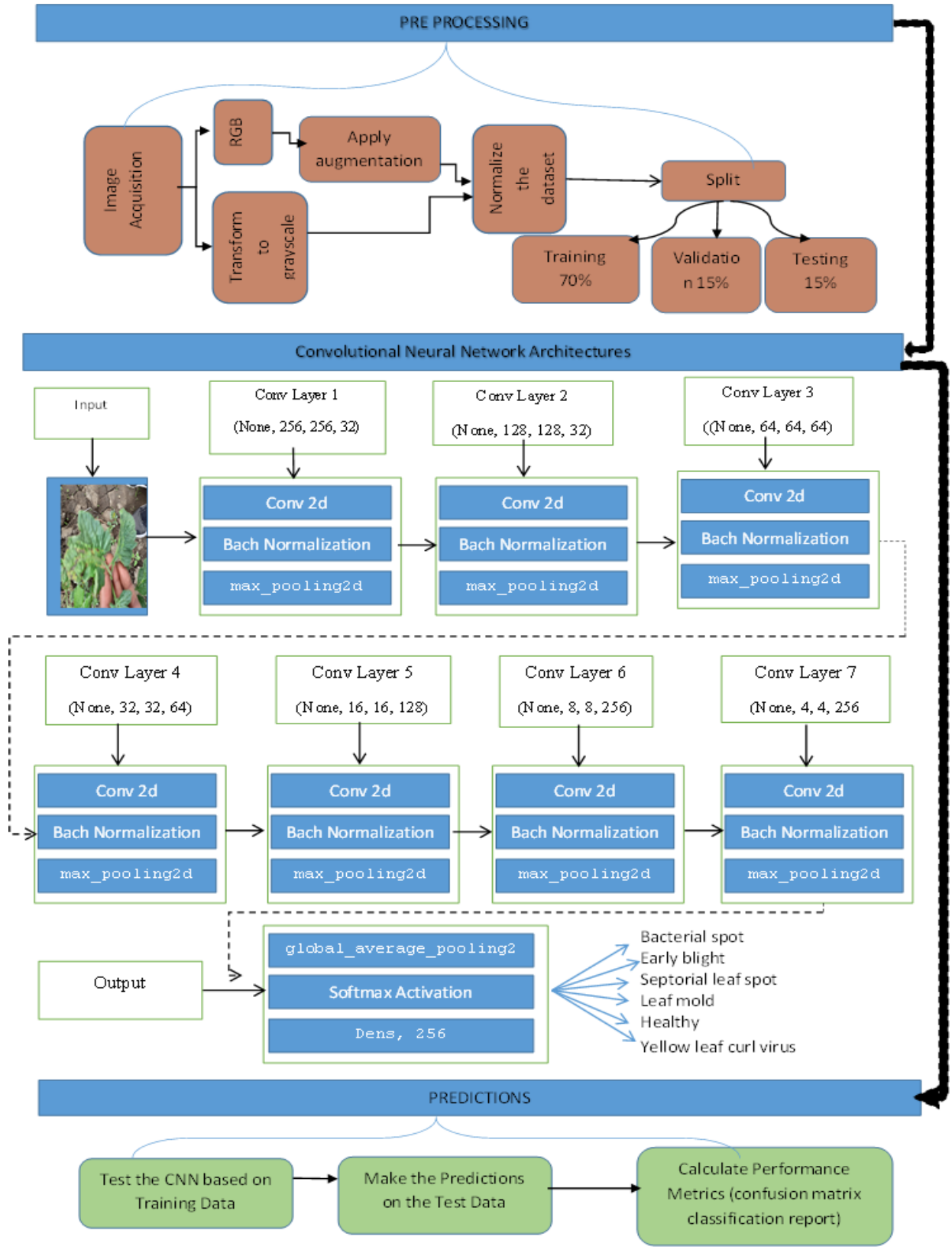

Figure 4-1Block diagram of the detection and classification of tomato leaf disease

The model uses the augmented data for training and the original validation data to give performance metrics. Inside of the CNN layers, useful features of each image are extracted and classification based on the extracted feature is performed and the process is called model training. During the training of the model, we can access the performance of the process is called by using the validation dataset which is used to measure the performance of the model. After accessing the performance of the model, the model which best performed is saved and used as a predictive model. Then the testing phase is performed by giving unseen during the training to the predictive model. The model finally gives class prediction which is the probability of the image belongs to one of the given class during the training.

4.4. Convolutional Neural Network Architectures

A lot of convolutional architectures have been developed from the 1990s. In the following, this study has proposed TomDiseaseNet convolutional neural network architecture, which is implemented from scratch. This research is proposed to find the tomato plant leaf diseases and classify each type of disease. The next step is to make a pair of tomato plant leaf disease and train the model. After identifying the disease obtain accuracy again. For that, we also prepared an architecture which is called as TomDiseasesNet

4.4.1. TomDiseasesNet

TomDiseasNet architecture is specially designed to identify the disease form of the tomato plant leaf image. As a shown figure of TomDiseaseNet Architecture, an image of tomato plant leaf image converts into 256*256*3, where 3 are for a color image. TomDiseaseNet contains 5 convolutional layers and 3 fully connected layers. In TomDiseaseNet, Relu is applied after very convolution and a fully connected layer and dropout are applied before the first and the second fully connected layer. This network has 800,166 total parameters and needs 1.1 billion computation units in a forward pass.

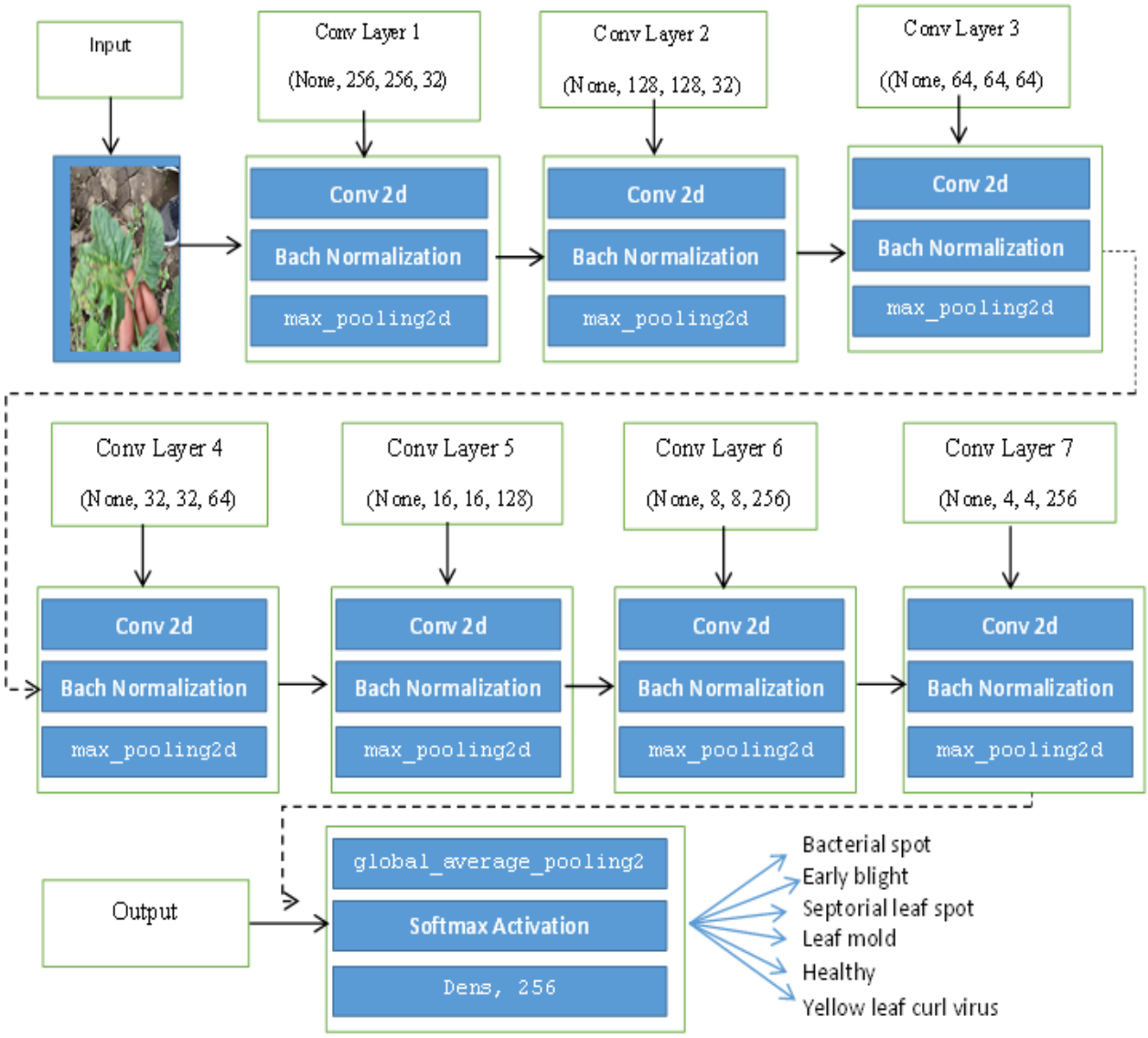

Figure 4-2 TomDiseaseNet Architecture

4.5. TomDiseaseNet Architecture Overview

The winners of the ILSVRC 2015 image classification challenge introduced a new concept called a deep residual learning framework that made it possible to train hundreds of layers and still achieve compelling performance without worries of performance saturation [44]. Residual networks allow that the additional layer does not change the output of the layer from which it receives input. This is known as identity mapping [45].

The training error of the overall module with the additional identity layers is the same as the error would be without the additional layers. Therefore, the additional identity mapping layer

is used only for feature learning on top of the already available input. Since the additional layer is learning only outlined in this paper, it is a 7 layers deep residual learning framework and is illustrated in figure 4.2. By using input image size (W1), receptive field size (F), stride (S), and amount of zero paddings (P) we can compute the spatial size of the output volume in each layer [46]. The following equation gives the exact output volume size of all the layers in the TomdiseasesNet model.

Equation 4-1 $output\ size\ (W2) = (W1-F+2P)/S+1$

Where: W1 is the size of the input volume, F is filter size, P is the number of zero paddings, and S is the stride.

The initial spatial size of the input volume (W1) is 256*256*3 and it gets changed after some convolution operations, the initial size of the filter F and Stride S are 5*5*3 and 2 respectively and these sizes are changed after some convolution and pooling operations, and there is no zero padding P in our network and the value of P is always zero throughout the model. In the following table, all of the parameters in each layer are described according to Equation (4.1) given above.

The spatial parameters have mutual constraints.

For example, when the input volume has size W1=20, no zero-padding is used p=0, and the filter size is F = 3, then it would be impossible to use stride s = 2 since Equation (4.1) gives 5.6, which is F = 3, then it would be impossible don't fit neatly and symmetrically across the input. We have considered this set of parameters to be valid during the process of resizing images contained in our dataset. If this arrangement is not considered, libraries that are used to implement the TomDiseaseNet model will throw an exception or it will zero pad the rest of the area or it will resize the image to make it fit.

4.5.1. Input Layer

The input layer of our TomDiseaseNet model accepts grayscale and RGB images of size 256*256*3 with six different classes (Bacterial Spot. 2, Early Blight. 3, Healthy. 4, Sectorial Leaf spot. 5, Leaf mold. 6, and Yellow Leaf curl). This layer only passes the input to the first convolution layer without any computation. Therefore, there are no learnable features and the number of parameters in this layer is 0.

4.5.2. Convolution Layer

Convolutional layer is the main and first layer of the CNN architecture. It is used to extract the characters from the input image. It is used to extract the characters from the input image. The convolution layer maintains the association connects pixels by learning image features from input data. The mathematical operation takes two input image matrices and 3x3 kernel filter. The 5x5 image pixel values and the convolution filter of 3x3. The features after finishing the multiplication of the corresponding matrix. The CNN adds and divides the result by the total number of pixel and then create a map and put the value of the filter at that place. After that, it moves the feature to every other position of the image and obtains the output of the matrix and repeats the steps for the other filters. This layer moves the filter to every possible position on the image.

In the TomdiseasesNet model there are seven convolutional layers:-

- The first convolution layer of the model filters the 256 x 256 x 3 input image by using 32 kernels with a size of 5 x 5 x 3 with a stride of 2 pixels. Since (127-5)/2+1=62, and since this layer has a depth of K = 32, the output volume of this layer is 62 x 62 x 32. The product of the output volume gives a total number of neurons in the layer (first Conv layer) which is 123.008. Each of 62 * 62 * 32 neurons in this volume is connected to a region of size 5 x 5 x 3 in the input volume. To control the number of parameters in convolution layers parameter sharing is used. If we see the input there are 128 * 128 * 32 = 524,288 neuron in the first convolution layer and each has 5 * 5 * 3 = 75 weights and 1 bias. Together this adds up to 524,288 * 75 = 39,321,600 parameters in the first layers of the model. This number is very high and impossible to implement in our machine. Here comes the concept of parameter sharing which is one of the advantages over the traditional neural network. When we use parameter sharing if one feature is used in some spatial location let say (x, y), then it should also be useful to compute at some other position (x2, y2). In other words, denoting a single a 2-dimensional slice of depth as a depth slice, which is a volume of size 62 x 62 x 32 has 32 depth slices, each of size 5 x 5, and we are making the neurons in each depth slice to use the same weight and bias. With this parameter sharing the first convolution layer in the TomdiseasesNet model has only 32 unique sets of weights (one for each depth

slice) for a total of 32 * 5 * 5 * 3 = 2,400 unique weights or 24,432 parameters by adding 32 bias and all 62 * 62 neurons in each depth slice have the same parameters. The output volume and parameters of each learnable layer in the TomdiseasesNet model are described in Table 4.1 below.

- The second convolutional layer takes as input the output (pooled output) of the first convolutional layer and filters it by using 32 kernels of size 3 x 3 x 32.
- The fourth convolutional layers have 64 kernels of size 32 x 32 x 64 and the fifth convolutional layer also has 64 kernels of size 3 x 3 x 64. All the convolutional layers of the TomdiseasesNet model use softmax nonlinearity as activation functions. Softmax is chosen because it is faster than other nonlinearities such as the train to train deep CNNs with gradient descent [47].

4.5.3. Pooling Layer

This layer compresses the image into a smaller size by taking the maximum value from the filtered image and obtains a matrix out of it. It also controls overfitting.

In the TomdiseasesNet model there are seven max-pooling layers after the first, second, and fourth convolution layers of the proposed model.

- The first max-pooling layer reduces the output of the first convolution layer with a filter of size 128 x 128 and stride 32.
- The second max-pooling layer takes as an input the output of the second convolutional layers and pools by using 64 x 64 filters of stride 32 the same as in max pooling.
- The third max-pooling layer has a filter of size 32 x 32 with stride 64. This layer has no learnable features and it only performs downsampling operation along the spatial dimension of the input volume, hence the number of parameters in these layers is 0.
- The fourth max-pooling layer has a filter of size 16x16 with stride 128.
- The fifth max-pooling layer has a filter of size 8x8 with stride 256.
- The sixth max-pooling layer has a filter of size 4x4 with stride 256
- The seventh max-pooling layer has a filter of size 2x2 with stride 256.

4.5.4. Fully Connected (FC) Layer

In the TomdiseasesNet model there are two fully connected layers including the output layer. The first fully connected layers have 64 neurons each and the final layer which is the output

layer of the model has only one neuron. The first FC layer accepts the output of the sevens Conv layer after converting the 3D volume of data into a vector value (Flattening). This layer computes the class score and the number of neurons in the layer predefined during the development of the model. It is the same as ordinary neural networks and as the name implies, each neuron in this layer is connected to all the numbers of layers.

4.5.5. Output Layer

The output layer is the last (the third FC layer) of the model and it has 1 neuron with a softmax activation function. Because the model is designed to classify six classes called categorical classification.

Table 4-1 Summary of tomdiseasenet model parameters

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
norm_1 (BatchNormalization)  (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
norm_2 (BatchNormalization)  (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        51264
_________________________________________________________________
norm_3 (BatchNormalization)  (None, 64, 64, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
norm_4 (BatchNormalization)  (None, 32, 32, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
norm_5 (BatchNormalization)  (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         33024
_________________________________________________________________
norm_6 (BatchNormalization)  (None, 8, 8, 256)         1024
_________________________________________________________________
```

```
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 256)         590080
_________________________________________________________________
norm_7 (BatchNormalization)  (None, 4, 4, 256)         1024
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
global_average_pooling2d_1   (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
=================================================================
Total params: 800,166
Trainable params: 798,502
Non-trainable params: 1,664
```

As we can see in the table above the TomdiseasesNet model have 800,166 extremely small parameters when we compare to the other deep learning architectures such as AlexNet which have 60 million parameters, VGG 138 million parameters, and GoogLeNet 4 million parameters; therefore, they need a huge computational power to train those models from scratch and they need a very large amount of data. But the TomdiseasesNet model is trained with a minimum amount of resources and data and it performs very well.

4.6. Feature Extraction

Is the main stage of most image classification problems for the classification stage it started, important features that are used to classify the image are extracted by the CNN algorithm? In the existing system using some features parameters like color feature, this feature is considered because their visual color difference identifies whether the crop is infected by the disease or not by a human vision in the previses system our proposed model (TomdiseaseNet) is automated features for tomato plant leaf image classification. Deep learning enables machines to learn from experience and understand. Deep learning learns pathogens and classifiers naturally not like machine learning, leading to the effectiveness of machine learning in such situations, with deep learning, accuracy is unlimited. By gathering knowledge from the experience, the methodology maintains a strategy from the requirement for people to ideas empowers the computer to learn complex ideas by building them out of fewer complexes. Figure4.3 below shows the lower levels of a convolutional neural network are responsible for feature extraction.

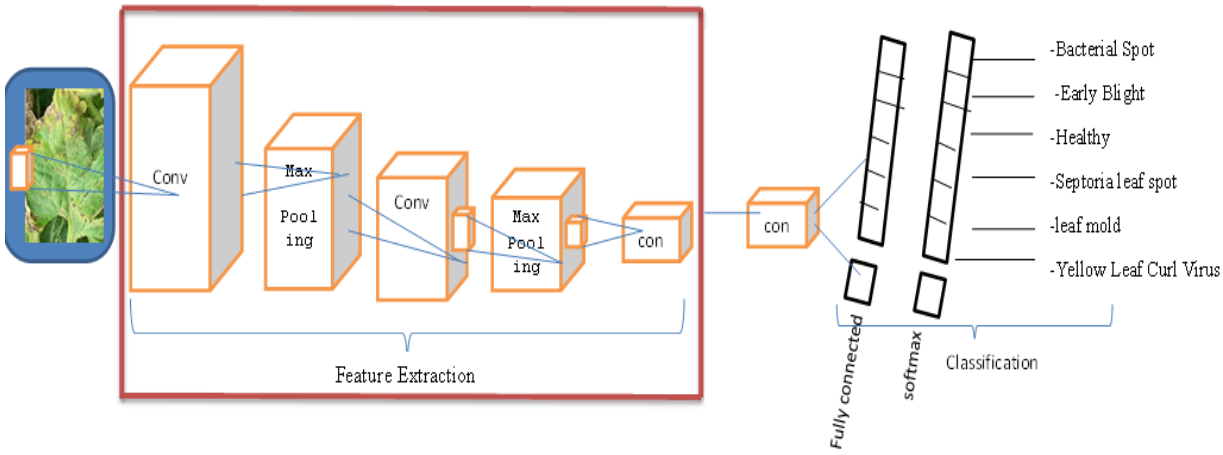

Figure 4-3Feature Extraction on the TomdiseaseNet model

4.7. Classification Using TomdoiseasesNet Models

Classification layer is a fully connected layer that yields a vectors of K measurement where k is the count of classes that the system will have the capacity to classify. The vector will have the probabilities for each one of the class of any picture being presented. Neurons layer have full link with all actuations in the past layer, as found in neural network architectures. Their actuations can therefore be calculated through a matrix product after a bias is applied. [41]. The last FC layer of TomdiseasesNet learning model usually utilizes a Softmax function to give the classifications. Figure 4.4 shows the classification layer which comprises of full connection and a classifier function. [41]

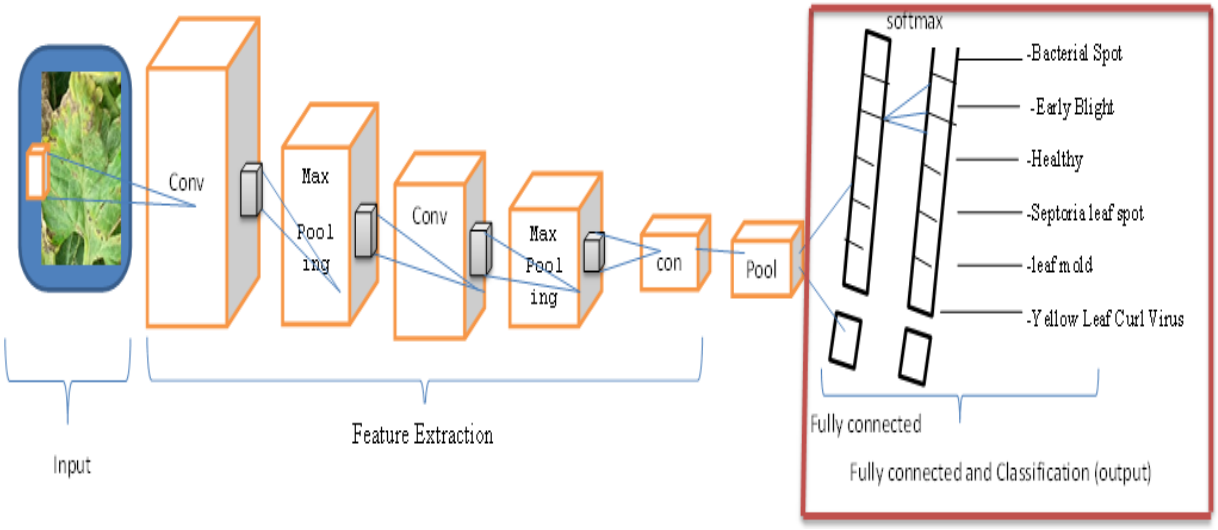

Figure 4-4Classification Layer

4.8. Dataset

The data has been collected from the plant village Disease Classification challenges. It has been downloaded the dataset consists of leaves of various categories of diseases found in types of diseases. In this dataset each category consists of:- Bacterial Spot 2127, Early Blight 1000, Healthy 1591, Septoria leaf spot 1771, leaf mold 952, Yellow Leaf Curl Virus 3209. A total of 10040 images of Tomato plant leaf diseases are used. There are a total of 5 kinds of diseases which are used to categorize input images the data structure format is shown in figure 4.3

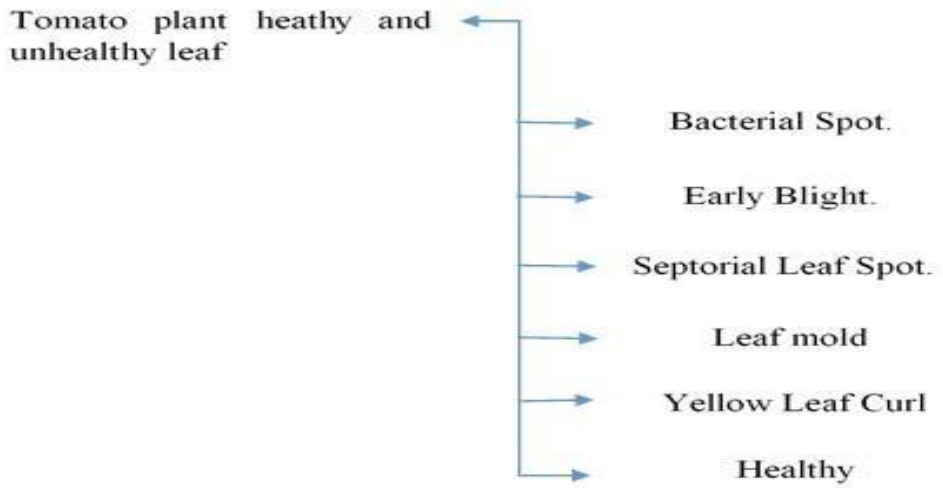

Figure 4-5input Dataset Structure

4.8.1. Sample Input Image Dataset

The images which are collected in the dataset are all of the different kinds of plant. Images having leaves of the same kind of diseases are categorized into one folder. Later when the model is training if any new input data comes having the same features as of these data images, then that image will also get included in the existing dataset. And the model gets smarter by the new features of the input image data is shown in figure 4.4

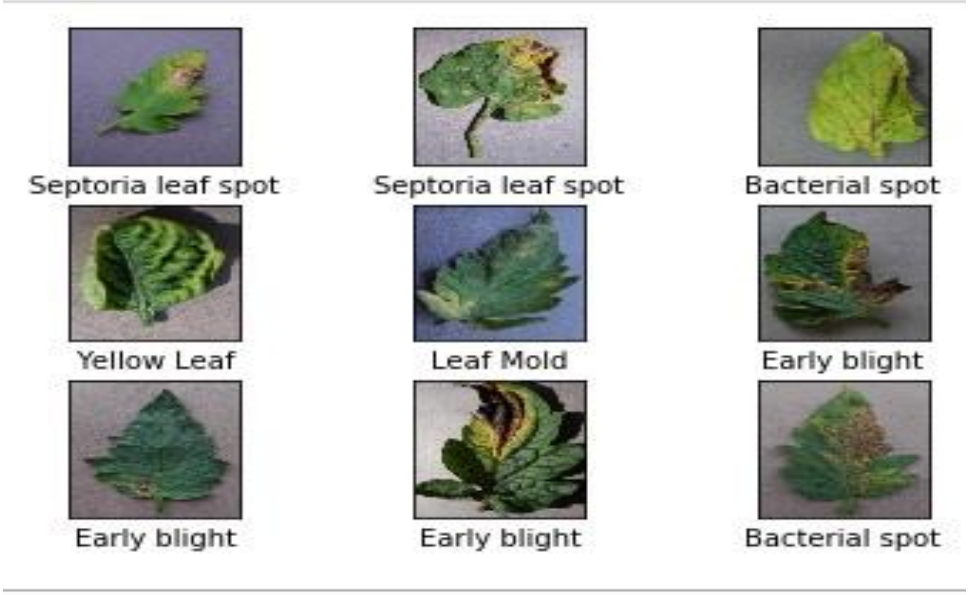

Figure 4-6 sample image from the defined dataset

4.9. Performance Evaluation

The loss function is the most important element in CNN, which helps us evaluate our model's performance. It is a value that varies between 0 and 1, which measures the inconsistency between the predicted and actual values. As the loss tends to 1 it implies the model is not doing well and as it tends to 0, it is showing that the model is predicting our values well.

4.10. TomdiseasesNet Model Hyperparameters

All the convolutional neural network models employed had so many parameters; meaning there were so many possible changes in the architecture. In addition, training them with the huge dataset hand took quite a long time. Performing hyperparameter optimization in our case would have been a total overkill, especially with the computing resources we had available to us. Therefore, we utilized some common hyperparameter values and looked further into techniques to achived a better model evaluation. Table 4.2 highlights the hyperparameter values used across all four CNN models. Hyperparameters is configurations that are external to the learning algorithm whose value is set before the training process begins. There is no standard rule to choose the best hyper-parameters for a given problem [48]. Therefore, a lot of experiments are conducted to choose the hyper-parameters. In the following, hyper-parameters that are chosen for the model are described.

- ***Dataset type***

  RGB image

  Grayscale image

  RGB Augmented image

- ***Deep learning architecture:***

  TomDiseaseNet

- ***Training mechanism***

  Training from Scratch

- ***Training testing set distribution***

  Train: 80%, Test: 5%

  Train: 70%, Test: 15%

  Train: 60%, Test 20%

- ***Optimization algorithms***

  The TomdiseasesNet model is trained using a gradient descent optimization algorithm to minimize the error rate and backpropagation of error algorithm is used to update the weights. Gradient descent is by far the most popular and the most widely used optimization algorithm in deep learning researches [48], [49]. At the same time, every state-of-the-art Deep Learning library contains implementations of gradient descent optimization algorithm such as Keras (used in this thesis). It updates the weight of the model and tunes parameters, therefore, minimize the loss function. To optimize the gradient descent, the Adaptive Moment Estimation (Adam) optimizer is used [50]. Adam computes adaptive learning rate is the most common optimization algorithm used today for training deep neural networks. As it is straightforward to implement, computationally very efficient and is very effective in dealing with large data and parameters. Each parameter and it uses squared gradients to scale learning rate, it also uses the moving average of the gradient.

- ***Learning rate***

  Because backpropagation was used to train the TomdiseasesNet model, the learning rate is used during weight update. It controls the amount of weight to update during backpropagation [68]. The challenging part was to choose the proper learning rate during our experiment. In our experiment, we have seen that a learning rate with a

value of too small take longer to train than a value of larger. But when we give a smaller value the model is more optimal than a model with a learning rate of larger. The experiment was done by using a learning rate of 0.01, 0.001, and 0.0001. Then, learning rate 0.0001 is the optimal one for all of the experiments even if it takes longer to train.

- ***Loss function***

  The loss function is directly related to activation functions that are used in the output layer the last fully connected layer of the model that type of problem we are trying to solve in the TomdiseasesNet model, we are using categorical cross-entropy Categorical cross-entropy used for single label categorization. This is only one category is applicable for each data point. Its as loss function to minimize. Categorical cross-entropy used for single label categorization. This is only one category is applicable for each data point. It is a softmax loss. It is a softmax activation plus binary cross-entropy loss if we can use this loss, we will train a CNN to output it is used for a multiple class. We have used Binary Cross-Entropy (BCE) loss as a loss function for our model. Even though there is another loss function such as Categorical Cross-Entropy (CCE), Mean Squared Error (MSE), but binary cross-entropy is the recommended choice of loss function for binary classification [51], [52]. It performs well for the model's output probabilities. it measures the distance between the actual output and the desired output. The experiment was done by using BCE loss.

- ***Activation function***

  Experiments are conducted by using two different activation functions: SoftMax and Sigmoid in the TomdiseasesNet model and Sigmoid performs better. In the output layer of the model, the sigmoid activation function is used because it is the choice for a binary classification problem [68, 71].

- ***Number of epochs***

  The number of epochs is the number of times the whole training data is shown to the network while training. Is the number of iterations the entire training dataset passes forewarned and backward through the model or the network in our experiment, the model was trained by using different epochs starting from 50 to 200? During the training, we have seen that when we used too small or too large an epoch, the model

gets overffiting between the training error and validation error. After many experiments, the model gets optimal with epoch fifty (50).

- ***Batch size***

  The total number of training input data we pass into training is present a single set is divided into small subsets called mini-batches to update the parameters. However, to prevent any loss in accuracy and generality of the model, the model is trained in the whole dataset, divided into mini-batches, for several epochs.

Table 4-2 Summary of hyper-parameters used during model training

Parameter	***Value***
Epoch	200
Bach size	32
Activation Function	Softmax, Sigmoid
Loss Function	BCE, CCE
Optimization algorithm	Adam
Learning rate	0.0001

5. CHAPTER FIVE: IMPLEMENTATION

5.1. Introduction

The section provides detail on how the entire implementation of the research was done for creating an efficient discuses tomato leaf disease detection and classification model. It also discusses how the treatment plan was suggested based on the classification.

5.2. Environment

The research was implemented using python 3.7.7 using Jupyter notebook as the main integrated development environment (IDE). Python language was selected as there is a lot of support from an active community for image classification using Tensorflow with Keras. The study was initially started on a local windows system but was eventually moved to the Google cloud platform (GCP) due to a shortage of resources. A Deep Learning images instance was created on the cloud, which is optimized for deep learning and has all the required packages pre-installed.

5.3. Experimental setup

Their scenarios are considered during the experiment of this thesis. The first two scenarios are classifying the images by transfer learning approach and the third scenario is proposing a CNN model based on the VGG16 architecture. During transfer learning the well-known CNN architecture which is VGG16 [53] and Inception V3 [54] that win the largest image classification competition are chosen. The models are trained by thousands of images and more than two classes. TomdiseasesNet model is a modified version of the VGG16 model by dramatically decreasing 138 million parameters in 800,166 (Table 4.1).

5.4. Dataset Preparation

Since the images had different proportions, their size was modified to a 256x256 pixel proportion. This resolution was the same satisfactory for both maintaining the image features, as well as keeping computational time to a minimum. All images were originally saved in an RGB 3-channel, and a small script was set up to assign a numerical value as a label to each image, according to its prefix. Therefore, images that had fungal and bacterial as prefixes were assigned with the value ‘0’ and images that had healthy prefixes were assigned with the value 1.

The images are loaded into the memory alphabetically due to their order in the containing folder. If we split the dataset as it is in training, validation, and testing, certainly, at least the training and

testing dataset will not contain sufficient or any images from one or the other class. If the testing set derives from the loaded images of the loaded dataset, the set will contain multi-class images; therefore we will not be able to measure the performance of the classifier correctly. Therefore the order of the image was defined by the name of each file, a shuffling method based on a random-number generator is used to randomly recorder the images.

5.4.1. Data Split

The dataset was split into three sections: A training portion dedicated to training the classifier a validation portion that would allow the training process to improve and a testing portion that is completely hidden from the training process and will be used for validating the classifier on unknown data. All split create that contain 70/30 ratio of the class of leaves image to avoid the unnecessary bias that would incur if a dataset would contain images from only one category

Python's random. Shuffle generator was used to shuffle the order of images and avoid having datasets with simpler external conditions. This way, images of leaves with different shapes, angles, levels of diseases, main leaf color, brightness, ambient lighting, etc.., are all included in all categories to achieve the highest possible variability. Variability in the dataset ensures that the model will be trained in the most generalized fashion possible, and will be evaluated and tested under all conditions [55]

The firs data split takes place by removing 15% of the total dataset and saving it for later uses as testing. The remaining 85% of the dataset is then split again into 70/15 ratio, resulting in the following portions, as seen in Table 1

Table 5-1Table1 Data splitting percentages and purpose.

Dataset	***Training***	***Validation***	***Testing***
% of total dataset	70%	15%	15%
Number of images	7447	1594	1594

```
Train size( tomato_Bacterial spot ): 1488
Validation size( tomato_Bacterial spot ): 319
Test size( tomato_Bacterial spot ): 319
Train size( tomato_Early blight ): 696
Validation size( tomato_Early blight ): 149
Test size( tomato_Early blight ): 149
Train size( tomato_healthy ): 1113
Validation size( tomato_healthy ): 238
Test size( tomato_healthy ): 238
Train size( tomato_Leaf Mold ): 666
Validation size( tomato_Leaf Mold ): 142
Test size( tomato_Leaf Mold ): 142
Train size( tomato_Septoria leaf spot ): 1239
Validation size( tomato_Septoria leaf spot ): 265
Test size( tomato_Septoria leaf spot ): 265
Train size( tomato_Yellow Leaf ): 2245
Validation size( tomato_Yellow Leaf ): 481
Test size( tomato_Yellow Leaf ): 481
```

Figure 5-1 Splitting data set result

The validation set is used during the training process to help the algorithm update its weights appropriately so that it can improve its performance and avoid overfitting. After the model has finished training, we run the testing data and verify if it has classified the test images correctly, thus creating the need to keep the testing set hidden.

5.4.2. Data Augmentation

In this study, one of the solutions has been verified for improvement of tomato plant diseases detection and classification using leaves, in the case of small datasets, data augmentation is a way of countering the challenge of having insufficient data on a TomdiseasesNet model. Data augmentation is often used to improve generalization properties. We can multiply the size of the dataset with data augmentation. We will use it for creating an image with different modifications. The new image has an obvious disadvantage they will be redundant. But this method works, as can be seen in the literature [56], [57]. It doesn't work as well as getting new and different images, but it can be improved our network. Different modifications can be applied to our data set [56].Crop: the size of the different images we have used is very different. We must use a default (256x256) size that can be also taken by any digital cameras. this size must be enough to catch tomato leaf diseases. Rotation: it must be done with a uniform random generation, with values between 0 and 25 degrees. Brightness range: mirroring images is another common

technique used for increasing the size of data sets, with also uniform randomly generated values between 0.2 and 1.5. Zoom: also a uniform random generation was used, with values in the zoom range 1. Width and height shift: it must be done with a uniform random generation, with 0.1, 0.1 widths, and height shift range respectively. A horizontal and vertical flip is used to increase the size of the data. Color shifting: that will change the value of some or all of the channels RGB in our image, with uniform random generation. We can use a limit value of 20% trying to not make too darker to white the image. The goal of data augmentation not only increasing the dataset. It is to benefit of learning the important features of each image at different rescale and positions.

Sample code for Data Augmentation

```
In [33]: from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(            # this is the augmentation configuration used for training
        rotation_range=25,
        zoom_range=.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        fill_mode='nearest',
        brightness_range=[0.5,1.5],
        vertical_flip=True,
        horizontal_flip=True
        )

val_datagen = ImageDataGenerator()             # for val/testing only rescaling function
```

Figure 5-2 Sample code of data augmentation

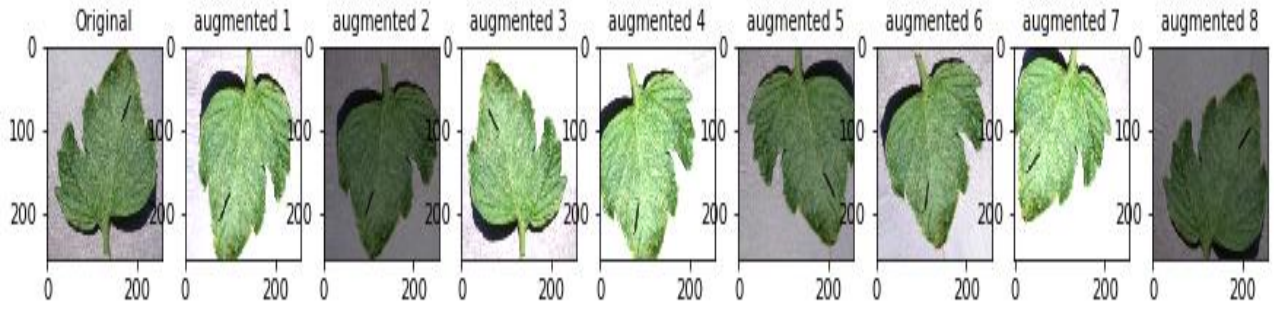

Figure 5-3 Augmented Images Using 1 Original Image

5.5. Image Preprocessing

Load the dataset: The images are loaded recursively based on their pathname. Depending on the prefix of their file name, a label value of '0' or '1' is appended to each loaded image. Color conversion: Two functions for color conversion were implemented in this thesis: a conversion from the BGR to RGB color-space, where the order of the image's color-space, where the images is lose all the color information and maintain only the brightness and saturation for each pixel. ***Data normalization***: is achived by finding the difference of the mean from every pixel, after that separating the outcome by the standard deviation. The dataset values are normalized within the (0, 1) range to ensure that the loss function, which is usually not convex, finds the global minimum as easy as possible. Reducing the range of input for training values also assists the convergence of the backpropagation algorithm. Dataset shuffling: The dataset is being shuffled by python's randomly Shuffle method, which is an order-shuffling algorithm based on a random number generator. After the application, the order of the images is originally was alphabetical, now mixed throughout the dataset. Dataset splitting: the dataset is being split in the following fashion: First 15% of the total number of dataset is held out and set as a testing dataset, second 15% of the dataset is held out and set the validation set, then the remaining dataset is splitting again the85/15 fashion, where the 15% portion is used as the validation dataset. The rest is used for training dataset the algorithm. The training and validation sets are used for the training of the algorithm, while the testing dataset is used only after the classifier is trained, to make predictions. The following are Simple code of image pre-processing and data set the splitting

```
img = image.load_img(img_path, target_size=(256, 256))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.
```

Figure 5-4sample code of single image pre-processing

```
for cls in classes_list:
  path = os.path.join(original_dataset_dir, cls)
  fnames = os.listdir(path)
  train_size = math.floor(len(fnames) * 0.7)
  validation_size = math.floor(len(fnames) * 0.15)
  test_size = math.floor(len(fnames) * 0.15)
```

Figure 5-5sample code of data splitting

5.6. Build a TomDiseasesNet Model

```
In [12]: model = models.Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', padding="same", input_shape=(256, 256, 3)))
model.add(BatchNormalization(name='norm_1'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_2'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (5, 5), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_3'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_4'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3),  padding="same",activation='relu'))
model.add(BatchNormalization(name='norm_5'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (1, 1), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_6'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_7'))
model.add(MaxPooling2D((2, 2)))

model.add(GlobalAveragePooling2D())
model.add(Dense(6, activation='softmax'))
model.summary()
```

Figure 5-6 Sample code of TomdiseasNet model

- TomdiseasesNet or CNN (Convolutional neural network or ConvNet) is a class of deep neural networks, commonly applied to analyzing visual imagery. Here is the TomdiseasNet model's example of CNN with few layers using Conv2D -2D convolutional layer, Bach Normalization andMaxPooling2D application of a moving window across 2D input space.
- Hyperparameters are set before training they represent the variables that determine the neural network structure and how it is trained.
- Conv2D with hypermarameters mentioned above: Conv2D (kernel size, (filters, and filters), input_shape = (256, 256, 3) with activation function for each layer as a softmax

- MaxPooling2D layer to reduce the spatial size of the incoming features; 2D input space: MaxPooling2D ((2, 2))
- Do the same increasing the kernel size: 4-8-16-32-64-128-256
- Provide last fully connected layer which specifies the number of classes of tomato leaf diseases and healthy 6. softmax activation function outputs a classified image: Dense (6, activation=' softmax')
- Print the summary of the model.

5.6.1. Parameters for layers

- Convolutional layer filter size filters the number of filters should depend on the complexity of the dataset and the depth of the neural network. A common setting to start with is [256, 128, 64, 32, 16, 8, and 4] for seven layers.
- Kernel size = number of filter = a small window of pixels at a time (3x3) which will be moved until the enter image is scanned. If the image is smaller than 256x256, work with a smaller filter of 1x1;
- Width and height of images were already providing. 2D convolutional layers take a three-dimensional input, typically an image with three color channels such as RGB;
- Max pooling is used which further reduces the size of the convolved image and prevents from overfitting. It is the application of a moving window across a 2D input space, where the maximum value within that window is the output: 2x2

5.6.2. Parameters to fit the model

- Epoch is the number of pass of the entire training dataset when an entire dataset is passed forward and backward through the neural network. It describes the number of images (items) and the algorithm sees the entire dataset and the algorithm has seen all samples in the training dataset. The number of epochs is 200 give because the smaller tests were slow to make progress. This number of an epoch is enough scope for the loss function to converge if it is possible. This also means that we are giving enough time to the architecture to learn. It takes around 15 seconds per epoch which is 15*100=1500 seconds. A single number of epochs are too big to feed to the computer at once we divide it into several smaller batches.

6. CHAPTER SIX: RESULT AND DISCUSSION

6.1. Introduction

This chapter describes the implementation of the classification process of Bacterial spot, early blight, Healthy, Leaf Mold, Sectorial leaf spot, and Yellow leaf disease on tomato plants by using the CNN algorithm, which was specified in detail in the previous chapter. In this chapter, all the experimentation details such as the results of each experiment and discussion of these results are presented briefly. The result of the experiment is shown in different graphs and tables.

6.2. Experimental Result

To classify the image, the color feature of the image was as described in detail in section 4.4. The main reason that the color feature is chosen to classify the image was that when we look at the image. We can simply say the image is healthy, diseased, and types of diseased. Three different classification scenarios are conducted during the experiment to test the classification performance.

The first two scenarios are based on pre-trained CNN models and the third one is by using the TomdiseasesNet model like most of the deep learning classification algorithms, our experiments have two main phases. The first one is the training phase and the second one is the testing phase. In the training phase, data is repeatedly presented to the classifier, while weights are updated to data that has never seen or test data by the classifier to test the performance of the classification algorithm. In the following, we will see the experimental result in detail.

6.3. Detection and classification using TomdiseasesNet model

6.3.1. Detection of Tomato Leaf Diseases by using the Collected Dataset

Our collected data set implement using the TomdiseasesNet model. Tomdoseas model was designed by modifying the VGG16 architecture which can give a promising result. The model has a total of 8 layers, seven convolutions, and one dense layer. It accepts a size of 256x256 color images we have conducted in this study and gives an output of 6 classes. The TomdiseasesNet model is trained with a total number of 10040 images before data augmentation during the training is applied in the dataset. A lot of experiment during the training is applied in the dataset. A lot of experiments are conducted by the TomdiseasesNet model by changing the

ratio of training and testing dataset, by using different learning rates, and finally by different activation functions.

6.3.1.1. Changing Training and Testing Dataset Ratio.

The result that is obtained from the experiment of the TomdiseaseNet model by using a different ratio for the training and the testing split is presented in the following table by using the classification accuracy metrics in the form of percentage for the train, validation, and test data separately.

Table 6-1 Result of an experiment by using different training and testing dataset ratio

Training Split	Test Split	Accuracy			Loss		
		Trainir	Validation	test	Training	validation	test
70	30	93.8%	95.4%	97.5%	20.4%	18.2%	22%
80	**20**	**98.49%**	**98.48%**	**97.86%**	**4.3%**	**5.7%**	**6.25%**
60	40	95.3%	96.5%	96.6%	14.1%	10.1%	7.2%

The TomdiseasesNet model is giving a promising result with different training and testing dataset ratios as we can see in table 6.2. From those experiments using 80% for training, 15% of validation, and 5% of testing is a better performance that other than two. Using ratio 8:2 means using 80% of the whole dataset is for training and 5% of the whole dataset is used for testing. In addition to this validation data 15% of the whole dataset.

6.3.1.2. Changing Learning Rate.

The result that is obtained from the experiment of the TomdiseasesNet model is by using different learning rates is presented in the following table by using the classification accuracy metrics in the form of percentage for the train, validation, and test data separately. As we can see in the following result giving higher learning rates has less accuracy than that of smaller learning rates. Therefore, a learning rate of 0.0001 is considered optimal in the TomdiseasesNet model.

Table 6-2 Result of the TomdiseasesNet model by using different learning rate

Learning Rate	***Accuracy***			***Loss***		
	Training	***Validation***	***Test***	***Training***	***Validation***	***Test***
0.1	94.6%	91.4%	74.7%	21.7%	20.1%	60.4%
0.01	94.15%	95.42%	97.78%	16.7%	13.5%	8.2%
0.001	**98.49%**	**98.48%**	**97.86%**	**4.3%**	**5.7%**	**6.25%**

6.3.1.3. Using Different Activation Function.

The result that is obtained from the experiment of the TomdiseasesNet model is by using different activation functions is presented in the following table by using the classification accuracy metrics in the form of percentage for the train, validation, and test data separately.

Table 6-3 Result of the TomdiseasesNet model by using different activation functions

Activation functior	***Accuracy***			***Loss***		
	Training	***Validation***	***Test***	***Training***	***Validation***	***Test***
SoftMax	94.6%	95.2%	97.1%	15.2%	12.3%	8.6%
Sigmoid	**98.8%**	**99.14%**	**98.49%**	**0.0004%**	**0.0057%**	**0.006%**

As we can see in table 6.4, using sigmoid activation function in the last fully connected layers or the output layer for binary classification is better than SoftMax which is more preferred for multiclass classification problems

Finally, the TomdiseasesNet model successfully classifies the given image with training accuracy if 98.8% and means test accuracy of 98.49% by using a learning rate of 0.001, output layer activation function of sigmoid, and train and test dataset ratio of 8:2.

6.4. Detection of Tomato Leaf Diseases by using the Public Dataset

6.4.1. Result Analysis for the TomdiseasesNet Tomato Leaf Diseases Detection Model

As we can see in the following plot (Figure 6.3), at the beginning of the training the value of the training accuracy line gets 90% and the value of the validation accuracy line is around 91% then both values are getting higher up to epoch 5. After epoch 5 both lines pass 99.18% and increases very slowly. When we see the training loss and validation loss curves in Figure 6.3 plot both decreases linearly from 0.2 up to 0.025 starting from the first epoch to epoch 200. After epoch 10 both validation and training loss line pass 0.05 which decreases highly starting from the beginning of the training and didn't pass 0.025 which is the smallest value in the training.

Finally, we can see that the validation accuracy is in sync with the training accuracy and validation loss is in sync with training loss. The validation loss and training accuracy curves are nearly linear, and on the other hand, the validation loss and training loss are nearly linear. The curves are showing that there is no overfitting in the proposed model because the validation accuracy us increasing and the validation loss is decreasing and most importantly there is no much gap training and validation accuracy and also there is no much gap between training and validation loss. Therefore we can say that our model's generalization capability becomes much well since the loss of the validation set was only slightly more compared to training loss.

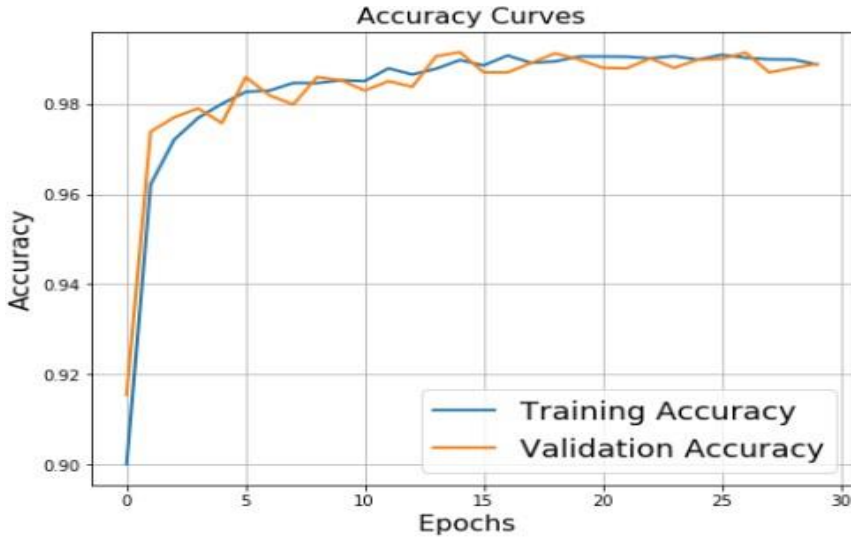

Figure 6-1 Training and validation accuracy of the TomdiseasesNet model

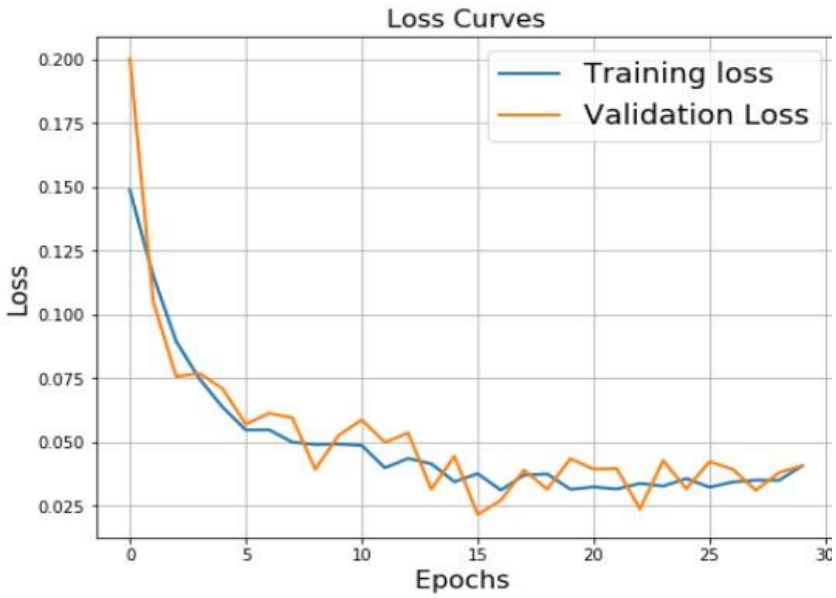

Figure 6-2 Training and validation loss of the TomdiseasesNet model

Table 6-4. Mean accuracy and loss of the TomdiseasesNet model

Metrics	***Accuracy***			***Loss***		
	Training	***Validation***	***Test***	***Training***	***Validation***	***Test***
Value	99.11%	98.14%	98.12%	0.04%	0.157%	0.16%

The result that is obtained from the experiment of the TomdiseasesNet model is presented in the following table by using the classification accuracy metrics in the form of percentage for the train, validation, and test data separately.

6.5. Tomato Leaf Diseases Classification Result

For tomato leaf classification task, the network had to distinguish between 6 class four three networks were trained and tested: - one CNN is RGB image the second is grayscale the third is an image and the last is augmented image both networks were trained for 200 epochs using Adem optimizer and learning rate of 0.0001. The loss function was binary cross-entropy due to binary classification. The training and testing set contained 9041 and 1594 images respectively. The confusion matrices of both networks are illustrated in figure 6.3 and figure 6.4. From both confusion matrices, it is obvious that the classification accuracy of both networks is almost

identical. The only difference is the number of false-negative networks of RGB image has less false negatives. Recall and precision are used as classification evaluation metrics

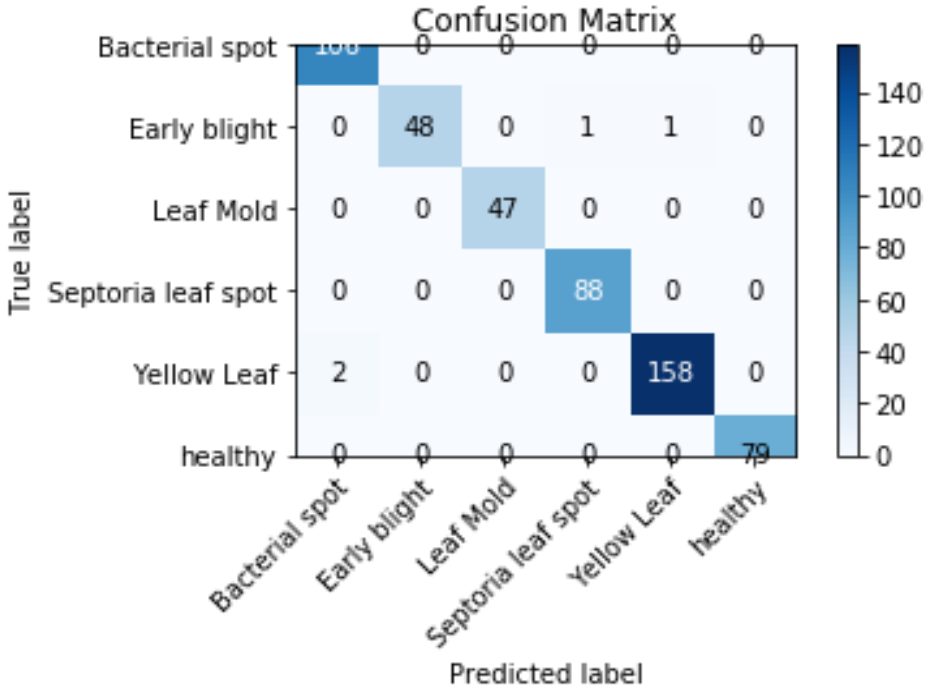

Figure 6-3Confusion matrix of the CNN with RGB images

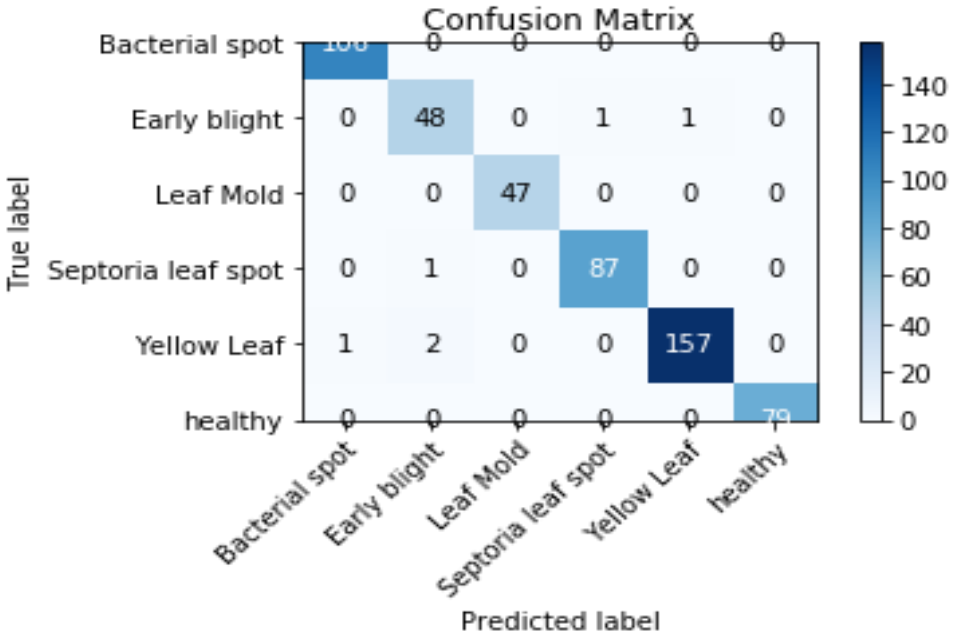

Figure 6-4Confusion matrix of the CNN with Grayscale images

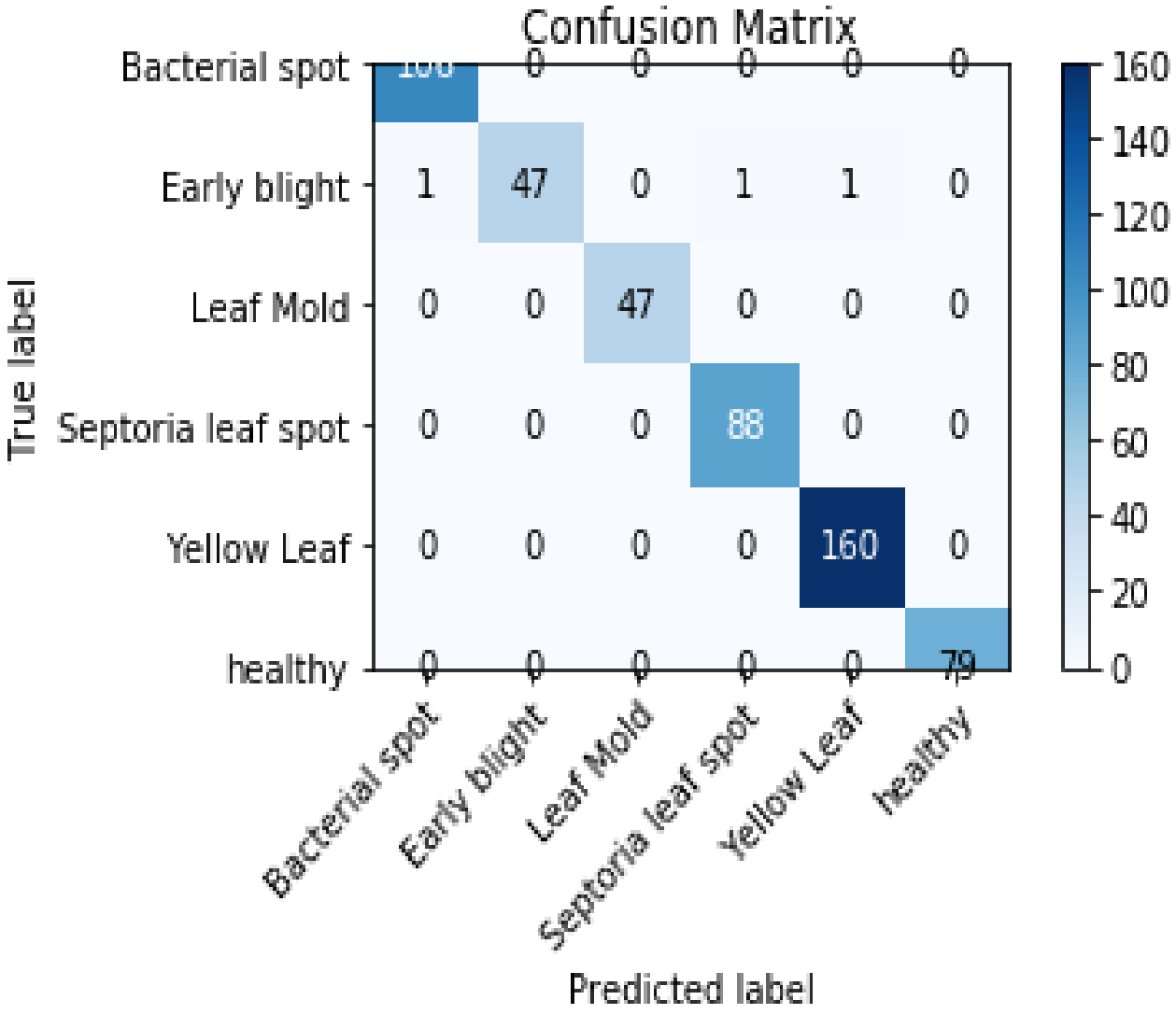

Figure 6-5 Confusion matrix of the CNN with Augmented images

After performing all the evaluation on the Grayscale and RGB image dataset on the TomdiseasesNet model, experiments with the top result are discussed below. The TomdiseasesNet model is evaluated using different parameters that can affect the efficiency of the model. Theses parameters are several epochs, train test split ratio, learning rate, and dropout ratio which are explained briefly in chapter 4. Different parameter ratios are used for the evaluation purpose and the most successful ratios are discussed in this chapter.

6.5.1. Summary for Evaluation of TomdiseasesNet Model

Table 6-5 ***Summary for the*** *TomdiseasesNet* ***model***

No	***Dataset type***	***Epochs***	***Learning Rate***	***Test Ratio***	***Accuracy***	***Loos***
1	Grayscale	50	Adam(0.001)	20%	93.34%	15%
		30	Adam(0.00001)	15%	90.45%	24%
		100	Adam(0.001)	5%	94.25%	14%
		200	Adam(0.001)	20%	95.62%	12%
2	RGB	50	Adam(0.001)	20%	95.81%	0.026%
		30	Adam(0.001)	30%	93.47%	0.013%
		100	Adam(0.0001)	15%	98.43%	0.011%
		200	Adam(0.0001)	15%	99.18%	0.0003%
3	Augmented	**50**	Adam(0.001)	20%	98.81%	0.006%
		30	Adam(0.001)	30%	99.11%	0.0043%
		100	Adam(0.0001)	15%	99.43%	0.0041%
		200	**Adam(0.0001)**	**15%**	**99.88%**	**0.0003%**

The above table summarizes top results found for the evaluation of the TomdiseasesNet model on two types of datasets which are used for comparison. This data set types are the grayscale image type, which is types of images with only one color channel, the RGB image types, which are images containing more than 16million colors in one RGB image.

6.5.2. Evaluating TomdiseasesNet Model on Grayscale Images.

Table 6-6 TomdiseasesNet model on Grayscale Images

No	***Epochs***	***Learning Rate***	***Test Ratio***	***Accuracy***	***Loos***
1	50	Adam(0.001)	20%	90.45%	15%
2	30	Adam(0.00001)	20%	93.55%	24%
3	100	Adam(0.0001)	15%	94.25%	14%
5	**200**	**Adam(0.0001)**	**15%**	**95.62%**	**12%**

1. Model Evaluation with parameters, Epoch of 200, Learning Rate Adam (0.0001), and Test data Ratio of 20%, which results with an accuracy of 95.62% from above table 6.7

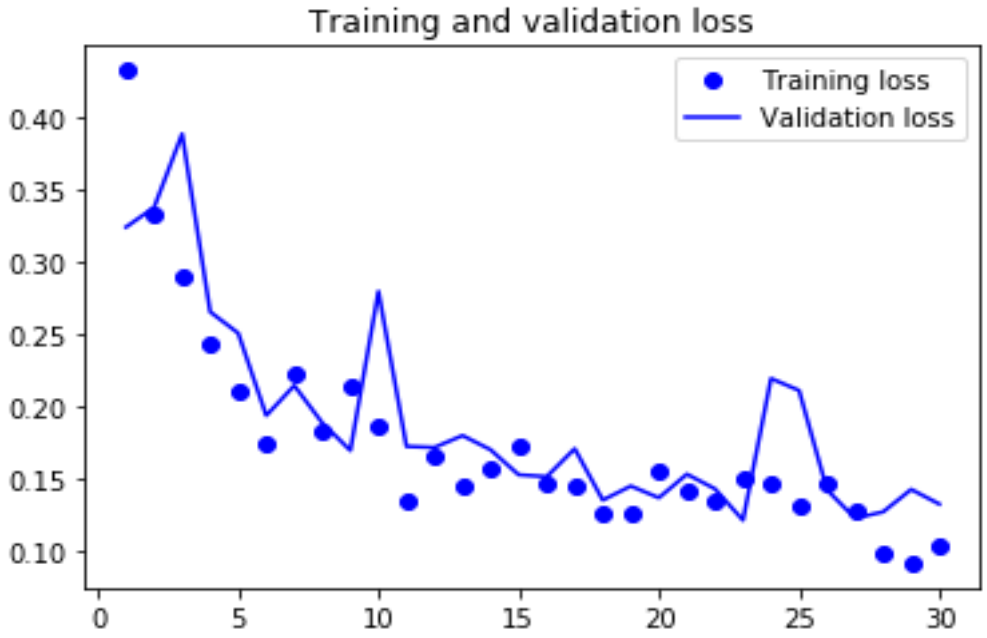

Figure 6-6Accuracy Graph of Train vs Validation on the grayscale image

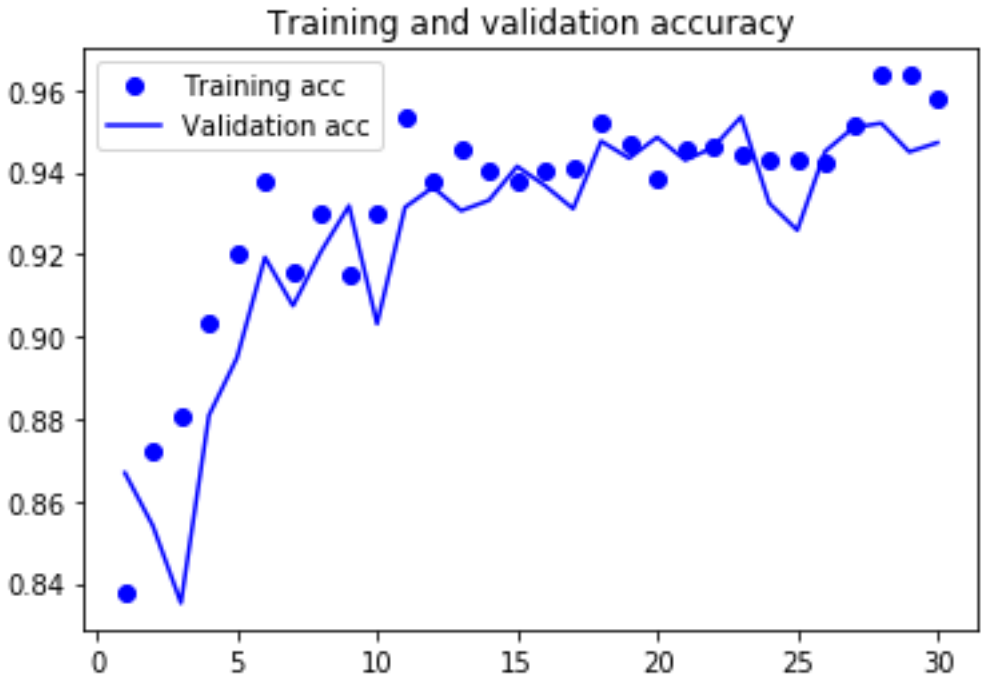

Figure 6-7Loss Graph of Train vs Validation on grayscale image

2. model Evaluation with parameters with Epoch of 100, learning rate adam (0.00001), Test data ratio of 15%, which results with an accuracy of 94.25%% from Table 6.7

6.5.3. Discussion for TomdiseasesNet Model on Grayscale Images

Grayscale images are images with only one color channel, which cannot hold enough information to be processed or extracted from the image data. Evaluating the same model on the same data set in table 6.7 by changing their parameters that can affect the efficiency of the model, the effect of the learning rate discussed in results one and two in the table. The only parameter changed from the two results is their Learning rate which 0.001 is used for the first result and 0.00001 is used the second result, which results with an accuracy of 90.45% and 93.55% respectively. This shows that the learning rate result which is equal to 0.001 has decreased to 0.00001in result two. The Learning rate decreases, the accuracy of the model decreases proportionally.

6.5.4. Evaluation TomdiseasesNet Model on RGB

Table 6-7Result Summary of TomdiseasesNet model on RGB image

No	***Epochs***	***Learning Rate***	***Test Ratio***	***Accuracy***	***Loos***
1	50	Adam(0.001)	20%	0.9581%	0.024%
2	30	Adam(0.001)	20%	93.47%	0.013%
3	100	Adam(0.0001)	15%	0.9811%	0.038%
4	**200**	**Adam(0.0001)**	**15%**	**0.9918%**	**0.0003%**

The above table summarizes the top five results found when evaluating the TomdiseasesNet model on RGB image using different learning parameters and their effects on the model as we can learn from the table each parameter change has its effect on the result the model is delivering with the training time needed and accuracy of the model.

6.5.5. Result Comparison of RGB images using Learning Rate and Split Ratio

From table 6.8 we can take results on rows 2, 3, and4 which have different parameters in which result on row 2 has LR of 0.001, results on row 3 with LR of 0.0001 and result on row 4 with LR of 0.0001. By comparing the three results for their learning rate we can conclude the effect of the learning rate on the model Table 6.8. Row 4 results evaluated using 200 numbers of epochs with a test ratio of 15% which result in an accuracy of 99.18**%.**

```
In [138]: score,accuracy =model.evaluate(test_batches,verbose=1)
          print("Test score is {}".format(score))
          print("Test accuracy is {}".format(accuracy))
          score,accuracy =model.evaluate(train_batches,verbose=1)
          print("train score is {}".format(score))
          print("train accuracy is {}".format(accuracy))

          160/160 [==============================] - 120s 747ms/step
          Test score is 0.0003963100607506931
          Test accuracy is 0.9918495416641235
          746/746 [==============================] - 596s 799ms/step
          train score is 4.6690334443155734e-07
          train accuracy is 0.9997091293334961

In [139]: score,accuracy =model.evaluate(valid_batches,verbose=1)
          print("Validation score is {}".format(score))
          print("Validation accuracy is {}".format(accuracy))

          160/160 [==============================] - 128s 801ms/step
          Validation score is 0.00010006267257267609
          Validation accuracy is 0.9951932430267334

In [140]: model.save('./saved model/zewud100.h5')
```

Figure 6-8 ***results of the*** *TomdiseasesNet* ***model on RGB image***

As we can see from the classification report of the above figures the model achieves a 99.97% training accuracy but it has misclassifying data which is explained below the confusion matrix

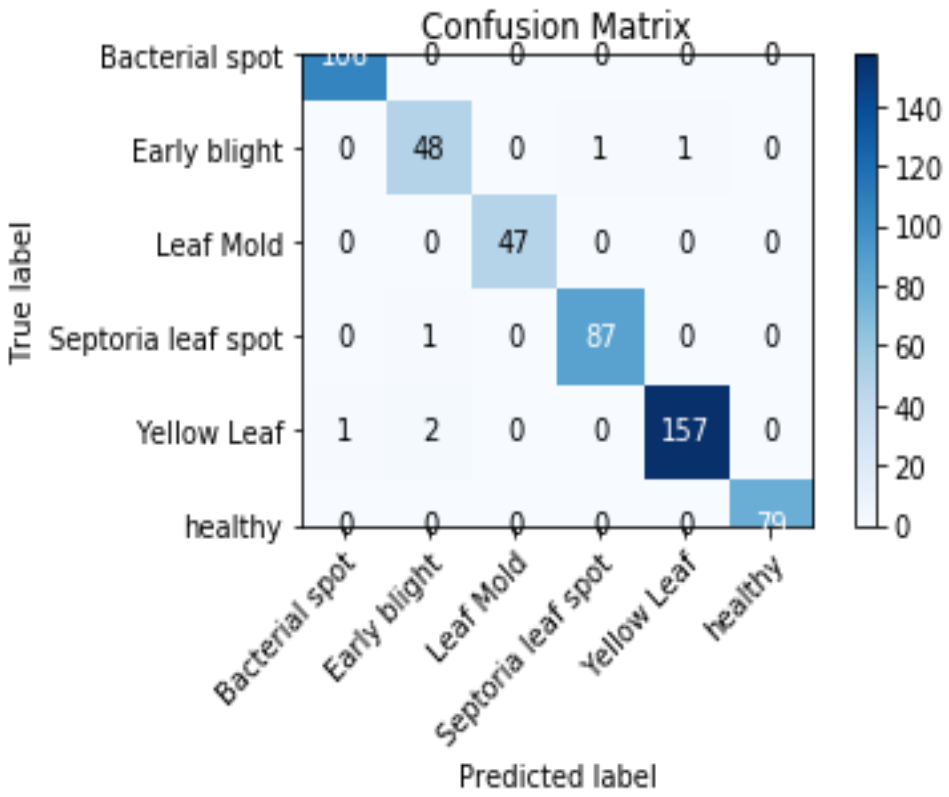

Figure 6-9 Confusion matrix of the CNN with RGB images

The above confusion matrix shows that the model works perfectly when tested on the images it above already knows or trained on. But it has some misclassifying images on test data which are six misclassifying images from the total 1595 testing data the model misclassified three yellow Leaf, one Sectorial Leaf Spot, and two Early blight.

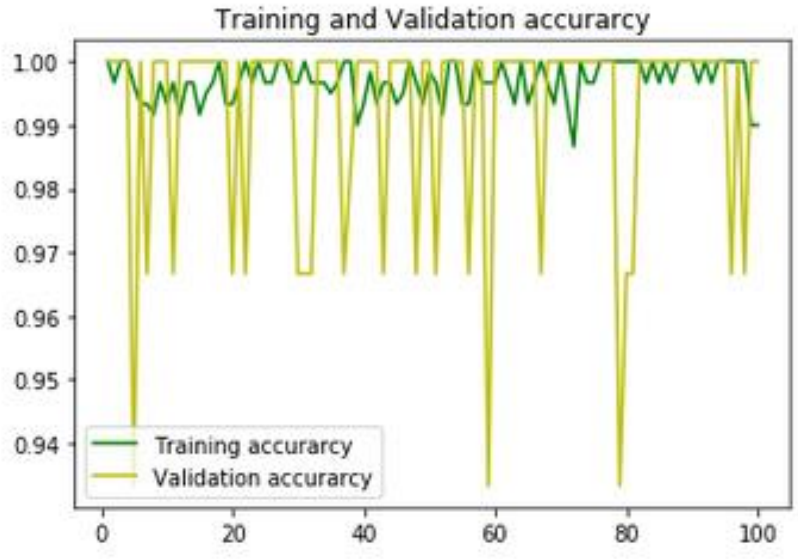

Figure 6-10 Accuracy Graph of Train vs Validation on RGB image

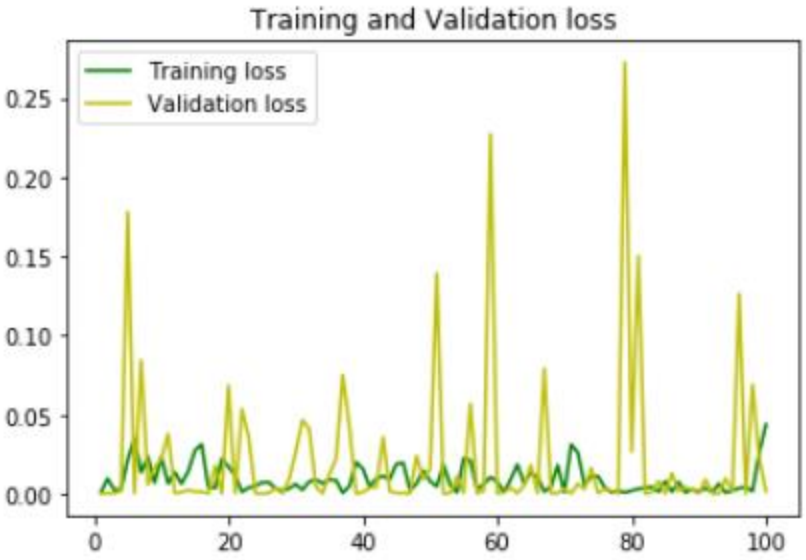

Figure 6-11Loss Graph of Train vs Validation on RGB image

The Loss graph shows that the model loss which starts from around 0,3(30%) has gradually decreased and tends to almost zero and this is what we are looking for and at the same time if we see the accuracy to 0.9997(99.97%). In both graphs, both the training and validation data have no significant distance between them. Table 6.8 row 4 results are evaluated using 200 number of epoch with test ratio 15% and LR of 0.0001, which result in an accuracy of 99.18%.

6.5.6. Evaluation TomdiseasesNet Model on Augmented Image

Table 6-8 Result Summary of TomdiseasesNet model on augmented image

No	***Epochs***	***Learning Rate***	***Test Ratio***	***Accuracy***	***Loos***
1	50	Adam(0.001)	20%	0.9781%	0.014%
2	30	Adam(0.001)	20%	0.9647%	0.0123%
3	100	Adam(0.0001)	15%	0.9911%	0.038%
4	**200**	**Adam(0.0001)**	**15%**	**0.9968%**	**4.4863%**

The above table summarizes the top five results found when evaluating the TomdiseasesNet model on RGB image using different learning parameters and their effects on the model as we can learn from the table every parameter change has its effect on the result the model is delivering with the training time needed and accuracy of the model.

6.5.7. Result Comparison of Augmented Images using Lr and Split Ratio

From table 6.9 we can take results on rows 2, 3, and4 which have different parameters in which result on row 2 has LR of 0.001, results on row 3 with LR of 0.0001 and result on row 4 with LR of 0.0001. By comparing the three results for their learning rate we can conclude the effect of the learning rate on the model Table 6.8. Row 4 results evaluated using 200 numbers of epochs with a test ratio of 15% which result in an accuracy of 99.68**%.**

```
score,accuracy =model.evaluate(test_batches,verbose=1)
print("Test score is {}".format(score))
print("Test accuracy is {}".format(accuracy))
score,accuracy =model.evaluate(train_batches,verbose=1)
print("train score is {}".format(score))
print("train accuracy is {}".format(accuracy))
```

```
160/160 [==============================] - 100s 624ms/step
Test score is 4.486341822484974e-06
Test accuracy is 0.9968652129173279
746/746 [==============================] - 465s 624ms/step
train score is 0.0
train accuracy is 0.9997316002845764
```

```
score,accuracy =model.evaluate(valid_batches,verbose=1)
print("valid score is {}".format(score))
print("valid accuracy is {}".format(accuracy))
```

```
160/160 [==============================] - 105s 657ms/step
valid score is 0.061402082443237305
valid accuracy is 0.9978055953979492
```

Figure 6-12 results of the TomdiseasesNet model on augmented image

As we can see from the classification report of the above figures the model achieves a 99.68% training accuracy but it has misclassifying data which is explained below the confusion matrix

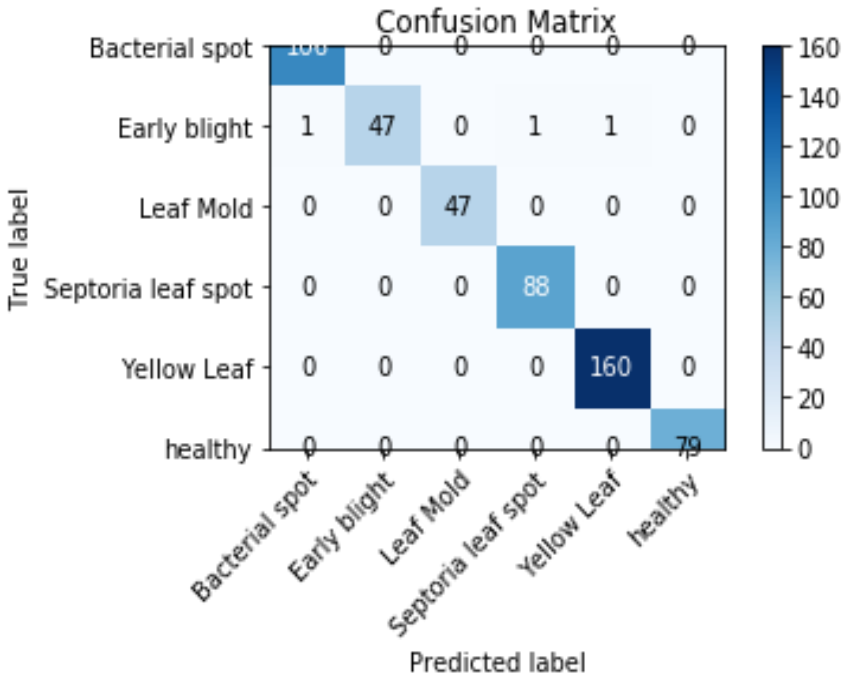

Figure 6-13 Confusion matrix of the CNN with Augmented images

The above confusion matrix shows that the model works perfectly when tested on the images it above already knows or trained on. But it has some misclassifying images on test data which are six misclassifying images from the total 1595 testing data the model misclassified three Early blight. In these three Early blight leaves are affected by multiple diseases.

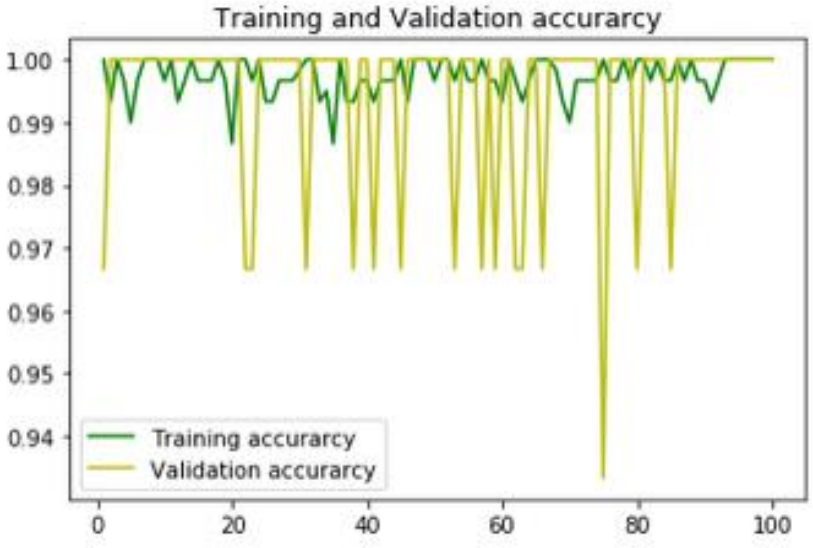

Figure 6-14Accuracy Graph of Train vs. Validation on Augmented image

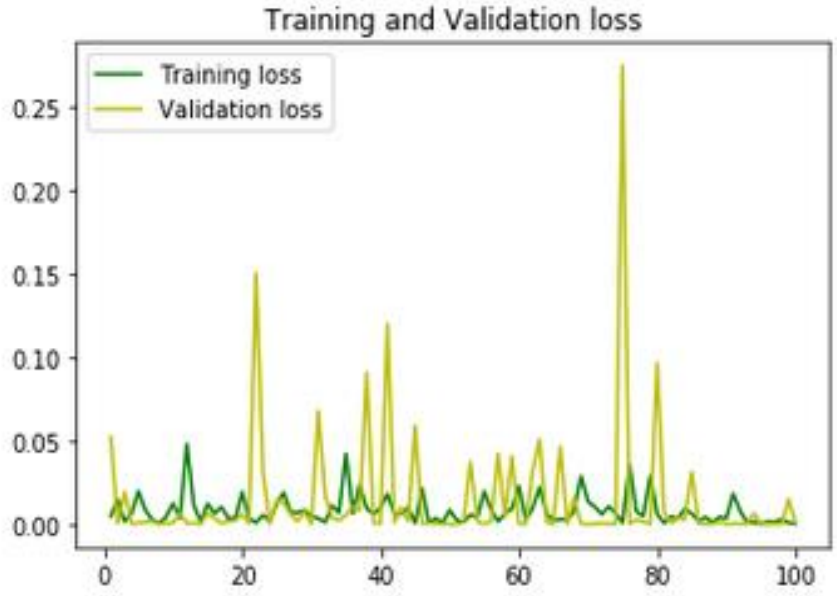

Figure 6-15 Loss Graph of Train vs Validation on Augmented image

The Loss graph shows that the model loss which starts from around 4.489(4%) has gradually decreased and tends to almost zero and this is what we are looking for and at the same time if we see the accuracy to 0.9968(99.68%). In both graphs, both the training and validation data have no significant distance between them. Table 6.9 row 4 results are evaluated using 200 number of epoch with test ratio 15% and LR of 0.0001, which result in an accuracy of 99.68%.

6.6. Visualization

TomdiseasesNet model is how works and what exactly learns we choose to visualize intermediate activations is consists of displaying the feature maps that are the output by various convolution and max-pooling layer in a network, given a several inputs. The output of a layer is often called its activation, the output of the activation function. It gives a view of input is decomposed into the different filters learned by the network. Shows in Figure 6.11 full-color image and Figure 3.12 Grayscale image every channel encodes relatively independent features, so the proper way to visualize these feature maps is independently plotting the contents of every channel. The full-color model learned to identify the disease spots, the grayscale method is the other hand did not learn how to locate the disease but instead learned the shape of the leaf and some patterns in the background.

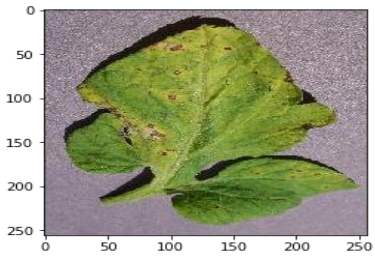

Figure 6-16 RGB input image

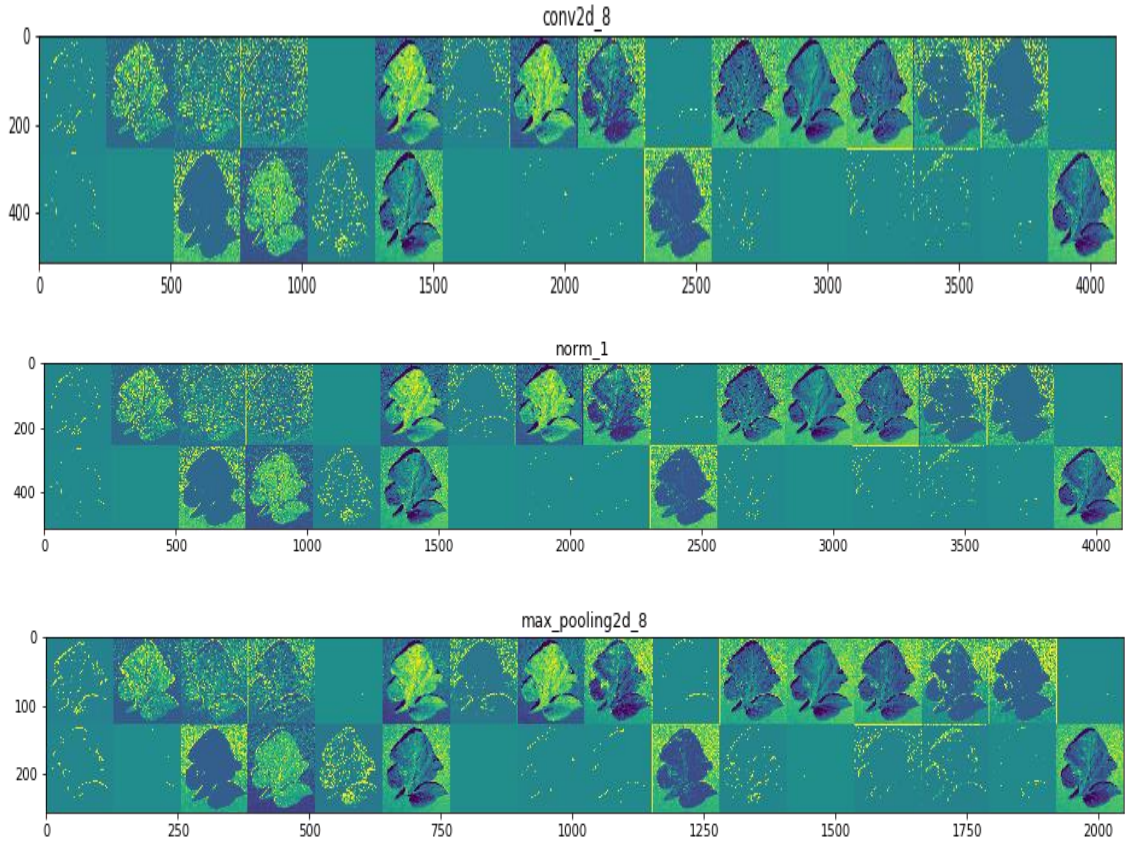

Figure 6-17 Full-color activation

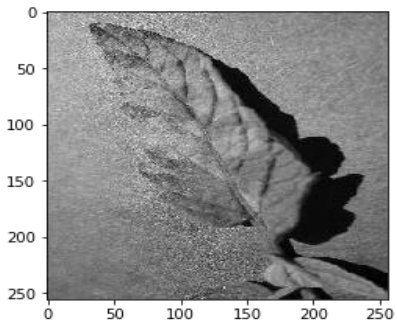

Figure 6-18 grayscale input image

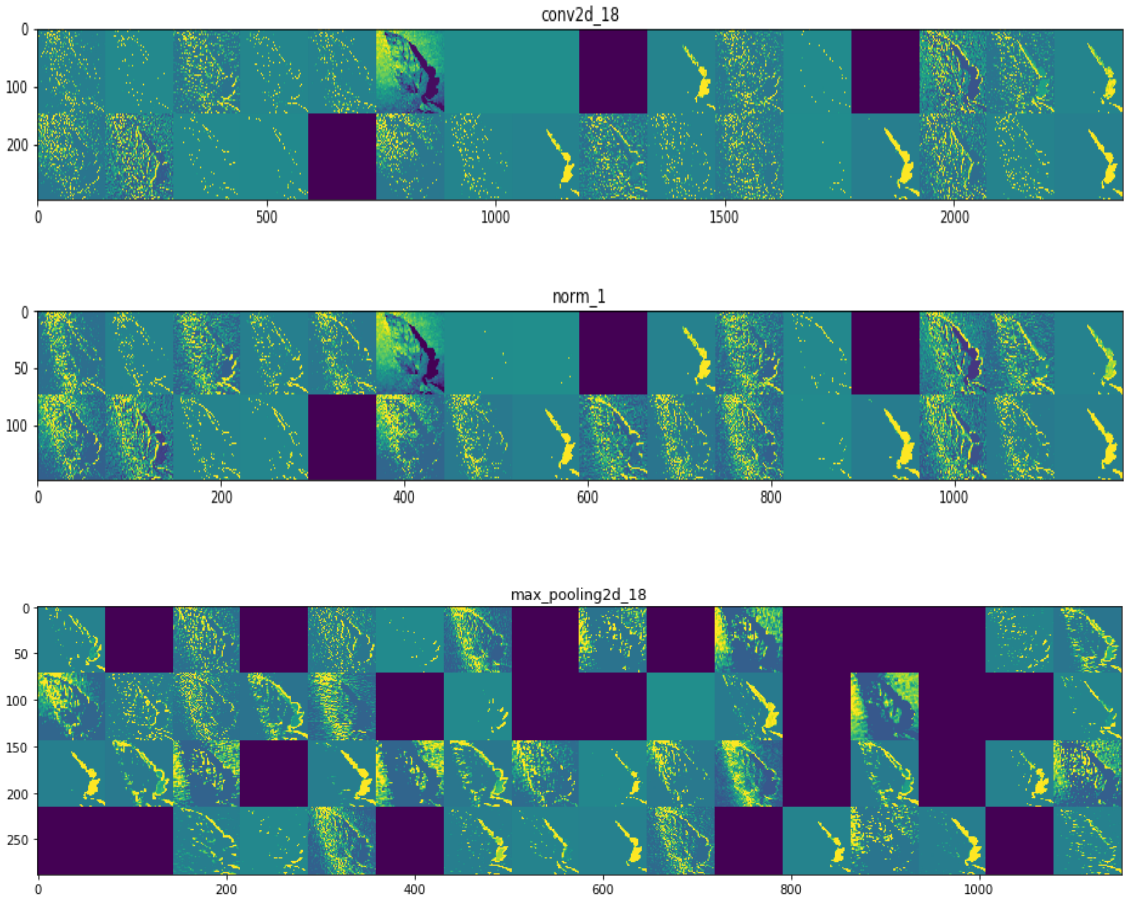

Figure 6-19Grayscale activation

Hidden Layer Output Visualization. Figure 2 visualizes the hidden layer output [58] for each layer were an input image of tomato early blight and its generated intermediate outputs are summarized. In our trained model, some of the intermediate outputs in the shallow layers (conv1, Conv7) highlight the yellow and brown lesions that are apparent within the image. However, in the deeper layer, owing to the convolution and pooling layers, the image size is too small to interpret whether such extracted features have been retained. Moreover, the global average pooling layer converts images to a feature vector that eliminates the spatial information, making it highly difficult to understand how the features are handled in extracted features positively contribute to the classification of the input image to the correct disease class or are used for a reason to deny other possibilities (e.g., a furry tail raises a possibility of an image containing a cat or a dog but certainly not a car). Hence, understanding what the CNN has learned by only exploring the intermediate output is insufficient.

7. CHAPTER SEVEN CONCLUSION AND FUTURE WORKS

7.1. Conclusion

Nowadays Tomato production is suffered from several problems, different tomato diseases like *Tomato Bacterial Spot, Tomato Early Blight, Tomato Sectorial Leaf spot, Tomato Leaf mold, Tomato Yellow Leaf curl* which reduces the production sometimes loss 100% on the unimproved local cultivar and quality of tomato yield. Besides, the shortage of diagnostics tools in developing countries like Ethiopia has a devastating impact on their development and quality of life. Therefore, there is an urgent need to detect the disease at an early stage with affordable and easy to use technological solutions. To make early identification of the diseases, we have TomdiseasesNet and implemented a deep learning approach by using the Convolutional Neural Network and image processing algorithm. We have presented a TomDiseasesNet model to detect and classify different tomato leaf diseases and healthy using leaf images of the tomato as an input. The TomDiseasesNet model can be used as a tool to detect leaf diseases of tomato.

This study discussed the CNN model by applying different important factors that can affect the model designed in the study. Three dataset types are used in the study to conduct the experiments for the TomdiseasesNet model, and this data set types are: - RGB data set, a data set with three-channel images Grayscale image data set a data set which only contains one channel images, and augmented image dataset a data set which is RGB and applying different augmented technics shush as rotation, zoom, width shift, height shift, shear range, fill mode, brightness range, vertical flip, and horizontal flip. TomdiseasesNet model has achieved an accuracy of 95.62% with 100 epochs. 0.0001 learning rate using grayscale image data set. This result is improved when the model is trained on the RGB data image data set, which climbed to an accuracy of 99.18% finally after the augmented image dataset is archived belter performance than the other archived 99.68% with 200 training epochs, the learning rate of 0.0001.

Finally, the most important factors that have impacts on the efficiency and accuracy of the model are several training epochs and learning rate the number of epochs is the number of iterations the model learns the whole data set. It is nor fixed that is how many epochs a model should be trained, but it obvious that the model learns more as it is trained more and more having a limit to stop somewhere and depends on the model architecture, type, and size of the data set. This study is archived with the highest accuracy with 200 epochs for the TomdiseaseNet model.

7.2. Summary

In this chapter explain about Tomato leaves detection and classification are presented. The methodology for the training and for the testing of leaf images is explained. The detection rates and result were also given and explained. This study used real-life and public image dataset. The testing results show us this study can be used in deep learning approach detects tomato leaves diseases. This thesis illustrates that the characteristics of the selected parameters for neural network are feasible and practical in the classification of tomato plant leaf diseases.

7.3. Recommendation

To be more precise on the result of this study, it would be better to apply it to types of crop diseases that can be identified by their colors from the crops in a large and complex real-world environment. The very critical issue we recommend to move this study one step forward especially in Ethiopia is that there should be farmers, investors, a private and governmental institution that can help the agricultural sector move forward to a modern agricultural area for Ethiopia

7.4. Future Work

- In the future the progress of identifying the plant diseases has been further modified to analyse a multiple leafs and various part of the plant.
- Reduce the computation and the size of deep model for small machine.
- Further improvement can be made to the neural network system to factor dataset scalability and more environmental diversities.
- Finally, it would be interesting to use the neural network system to segment contours a given image of our dataset in the future work more images of different tomato plant diseases needs to be collected.

.

.

8. Reference

[1] L. D, ""Research experiences in tomato production"," Ethiopia Agricultural Research Organization, Addis Ababa., 2002.

[2] Z. Y. H. E. Dessalegne L, "Agronomic studies in tomato: Horticulture research and development in Ethiopia. Proceedings of the Second National Horticultural Workshops of Ethiopia," *Addis Ababa, Ethiopia,* p. 163.

[3] B. T, ""Response of tomato cultivars differing in growth habit to nitrogen and phosphorus fertilizers and spacing on vertisol in Ethiopia."," ActaAgri 2: 1., 2008.

[4] H. Y. Kassa B, ""Tuber yield loss assessment of potato cultivars with different levels of resistance to late blight. In: Proceedings of the 3rd Annual conference of Crop Protection Society of Ethiopia"," in *Crop Protection Society of Ethiopia*, Ethiopia Adiss Abeba, 1996.

[5] M. &. S. S. Abera, ""Review on Disease Management Practice of Tomato Wilt Caused Fusariumoxysporum in Case of Ethiopia"," *Journal of Plant Pathology & Microbiology.,* Vols. 10.4172/2157-7471.1000460., p. 9, 2018.

[6] W. T. a. A. Desalegn Regassa, "Tomato (Lycopersiconesculentum Mill) varieties evaluation in Borana zone, Yabello district, southern Ethiopia.," *Plant Breeding and Crop Science,* vol. no_10 , pp. 206-210, 2016.

[7] M. M. Aklile, "Performance evaluation of tomato varieties for the irrigation production system in Mecha District of west Gojiam ZoneAmhara Region, Ethiopia.," 2016.

[8] S. H. R. B. Ravi Shankar, "A PRACTICAL GUIDE TO IDENTIFICATION AND CONTROL OF TOMATO DISEASES".

[9] [Online]. Available: https://hgic.clemson.edu/factsheet/tomato-diseases-disorders/.

[10] A. &. M. A. D. &. G. S. Mengistu, "Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques. International Journal of Database Theory and Application," *International Journal of Database Theory and Application,* vol. 10.14257/ijdta.2016.9.4.07. , pp. 79-88, 2016.

[11] G. A. Mengesha, "integrated the management of tomato late blight [Phytophthorainfestans (Mont.) de bary] through host plant resistance and reduced frequency of fungicide application in gamoegofa zone, southern Ethiopia," *HARAMAYA UNIVERSITY,* 2017.

[12] [Online]. Available: https://www.tilasto.com/en/topic/geography-and-agriculture/crop/tomatoes/tomatoes-area-harvested/ethiopia.

[13] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," vol. 145, p. 311–318, 2018.

[14] S. S. P. P. Shivkumar Bagde, "Artificial Neural Network Based Plant Leaf Disease Detection," *IJCSMC*, vol. 4, no. ISSN 2320–088X, 2015.

[15] H. MR, "Environmental factors affecting plant productivity. Achieving efficient use of rangeland resources," *Montana State University Agricultural Experiment Station, Bozeman*, pp. 27-36, 1988.

[16] "Converting YOLO* Models to the Intermediate Representation (IR)," [Online]. Available: https://docs.openvinotoolkit.org/latest/ docs MO DG prepare model convert model tf specific Convert YOLO From Tensorflow.html.

[17] J. T. M. W. a. T. J. G. P. Walstra, "Second Edition Second Edition",, no. June. 2005..

[18] C. G. D. a. M. K. Wanjale, "A modern approach for plant leaf disease classification which depends on leaf image processing," in *International Conference on Computer Communication and Informatics (ICCCI)* , -, 2017.

[19] P. U. G. a. M. K. O. h. Ms. Kiran R. Gavhale, "Unhealthy region of citrus leaf detection using image processing techniques," in *International Conference for Convergence of Technology, IEEE.*, 2014.

[20] K. K. a. R. dagade, ""Disease detection and diagnosis on plant using image processing-A review," *International Journal of Computer Applications*, vol. Vol.108, no. 13, pp. 36-38, 2014.

[21] Trimi Neha Tete and Sushma Kamlu, "Plant disease detection using different algorithms," in *"Proceedings of the Second International Conference on Research in Intelligent and Computing in Engineering*, 2017.

[22] Dilpreet Kaur and Yadwinder Kaur, "Various image segmentation techniques: A Review," *International Journal of Computer Science and Mobile Computing(IJCSMC)*, vol. 3, no. 5, pp. 809-814, 2014.

[23] Nida M. Zaitoun and Musbah J. Aqel, "Survey on image segmentation techniques," *Procedia Computer Science*, vol. 65, pp. 797-806, 2015.

[24] T. M. a. A. D. Lahouaoui Lalaoui, "New method for image segmentation," *Procedia Social and Behavioral Sciences*, pp. 1971-1980, 2015.

[25] P. J. J, "Various segmentation techniques in image processing," *International Journal of Engineering Science and Computing*, vol. 1.7, no. 6, pp. 12781-12783, 2017.

[26] G. K. a. P. K. Bhatia, "A detailed review of feature extraction in image processing," in *IEEE*, 2014.

[27] Y. B. a. G. H. LeCun, "Deep Learning," *Nature*, vol. 521, pp. 436-44, 2015.

[28] E. O. G. a. M. K. H. Durmus¸, ""Disease detection on the leaves of the tomato plants by using deep learning," in *In Agro-Geoinformatics, IEEE 6th International Conference on, pp.*

1-5, 2017.

[29] U. M. e. al, "SVM-based detection of tomato leaves diseases," *Springer*, pp. 641-652, 2015.

[30] S. D. K. a. A. B. Patil, "Plant Disease Detection Using Image Processing," in *International Conference on Computing Communication Control and Automation*, Feb. 2015.

[31] G. R. B. C. G. F. F. H. Molina JF, "Automatic detection of early blight infection on tomato crops using a color based classification strategy," *XIX IEEE Symposium on Image, Signal Processing, and Artificial Vision. (2014) pp. 1–5*, pp. 1-5, 2014.

[32] M. A. S. A. A. E. H. H. Usama Mokhtar, "Tomato leaves diseases detection approach based on support vector machines," *IEEE*, 2015.

[33] A. B. P. Sachin D. Khirade, "Plant Disease Detection Using Image Processing" 2015 IEEE International Conference on Computing Communication Control and Automation," vol. 15, pp. 978-14799-6892-3/15 .

[34] L. P. Muthukannan K, "Fuzzy inference system based unhealthy region classification in plant leaf image," *ComputInfEng*, p. 2103–2107 .

[35] P. P. M. N. P. B. M. P. a. P. PrajaktaMitkal, "Leaf Disease Detection and Prevention Using Image processing using Matlab," *International Journal of Recent Trends in Engineering & Research (JETER)* , vol. 02, no. 02, February– 2016.

[36] H. D. P. S. M. Mohanty Sharada P., "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 7, p. 1419 , 2016.

[37] M. Dessalegn, "Applying a Deep Learning Approach for Wheat Rust Disease Detection," p. 99, 2019.

[38] [. François, "Deep Learning with Python," *New York: Manning Publications*, 2017.

[39] B. JGA, "Factors influencing the use of deep learning for plant disease recognation.," *Biosyst Eng*, pp. 84-91, 2018.

[40] w. r. 2. https://www.kaggle.com/emmarex/plantdisease, "www.kaggle.com," kaggle, 2018. [Online]. Available: https://www.kaggle.com/emmarex/plantdisease, was retrieved 20/12/2018. .

[41] C. François, "Deep Learning with Python," *New York: Manning Publications*, 2017.

[42] S. RUDER, "algorithm, Sebstian ruder an overview of gradient optimization," [Online]. Available: http://ruder.io/optimization-gradient -desent/.

[43] B. J. Mccaffrey, " Neural Network Cross Entropy Error," 2019. [Online]. Available: https://visualstudiomagazine.com/articles/2014/04/01/neural-network-crossentropyerror.aspx. [Accessed: 16-May-2019]. .

[44] —. T. v. F. a. P. v. N.]”Google, "Machine Learning Crash Course Google Developers," 2019. [Online]. Available: https://developers.google.com/machinelearning/crash-course/classification/true-falsepositive-negative..

[45] S. K. a. Z. (2014), "A Very Deep Convolutional Networks for LargeScale Image Recognition," *arXiv preprint arXiv,* vol. 1409.1556, 2014.

[46] C. G. D. a. M. K. Wanjale, "A modern approach for plant leaf disease classification which depends on leaf image processing," in *International Conference on Computer Communication and Informatics (ICCCI), IEEE*, 2017.

[47] T. N. T. a. S. Kamlu, "Plant disease detection using different algorithms," in *Proceedings of the Second International Conference on Research in Intelligent and Computing in Engineering* , Vol.10, pp-103-106, 2017.

[48] P. a. G.Adam, "Deep learning a racttioner’s Approch, Sebastopol:," *O’Reilly Media,* 2017.

[49] J. L. E. M. A. R. D. O. F. B. R. H. S. A. N. a. N. D. R. Miikkulainen, "Evolving deep neural networks, ”in Artificial intelligence in the Age of Neural Networks and Brain Computing," *Elsvier,* pp. 293-31, 2019.

[50] B. M. a. M. Streeter, "Delay-tolerant algorithm for asynchronous distributed online learning,” Advances in Neural Information Processing Systems," pp. 2915-2923, 2014.

[51] P. a. G.Adam, "Deep learning a racttioner’s Approch," *Sebastopol: O’Reilly Media,* 2017.

[52] C. Francois, "Deep Learning with python," *New York: Manning publication,* 2017.

[53] K. S. a. A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arxiv preprint arxiv,* pp. 1-14, 2014..

[54] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. a. A. R. C. Szegedy, "Going deeper with convolutions IEEE conference on computer vision and pattern recognition," in *IEEE* , 2015.

[55] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla and D. Stefanovic, "Solving Current Limitations of Deep Learning Based Approaches for Plant Disease Detection.," vol. 11, p. 939, 2019.

[56] S. Dielemgan, "Classifying plankton with deep neural networks".

[57] A. Ng, "Deep learning specialization. Master deep learning and break into ai.".

[58] D. P. H. a. M. S. S. P. Mohanty, "Using deep learning for image-based plant disease detection,," *Frontiers in Plant Science,* vol. vol. 7, p. 1419, 2016.

[59] ‖. 2. [. A. /. BI vs. Big Data vs. Data Mining, 19-Apr2019. [Online]. Available: https://selecthub.com/business-intelligence/bi-vs-big-data-vs-data-mining.

[60] K. S. a. A. Zisserman, "very deep convolutional networks for large-scale image

recognition," *arXiv preprint arXiv,* pp. 1-14, 2014.

[61] W. D. B. a. M. M. B. P. A. Miceli, " Isolating Random and Bias Covariances in Tracks.," 2018.

8. Appendixes:

8.1. Appendix I: Tomato Leaf Dataset

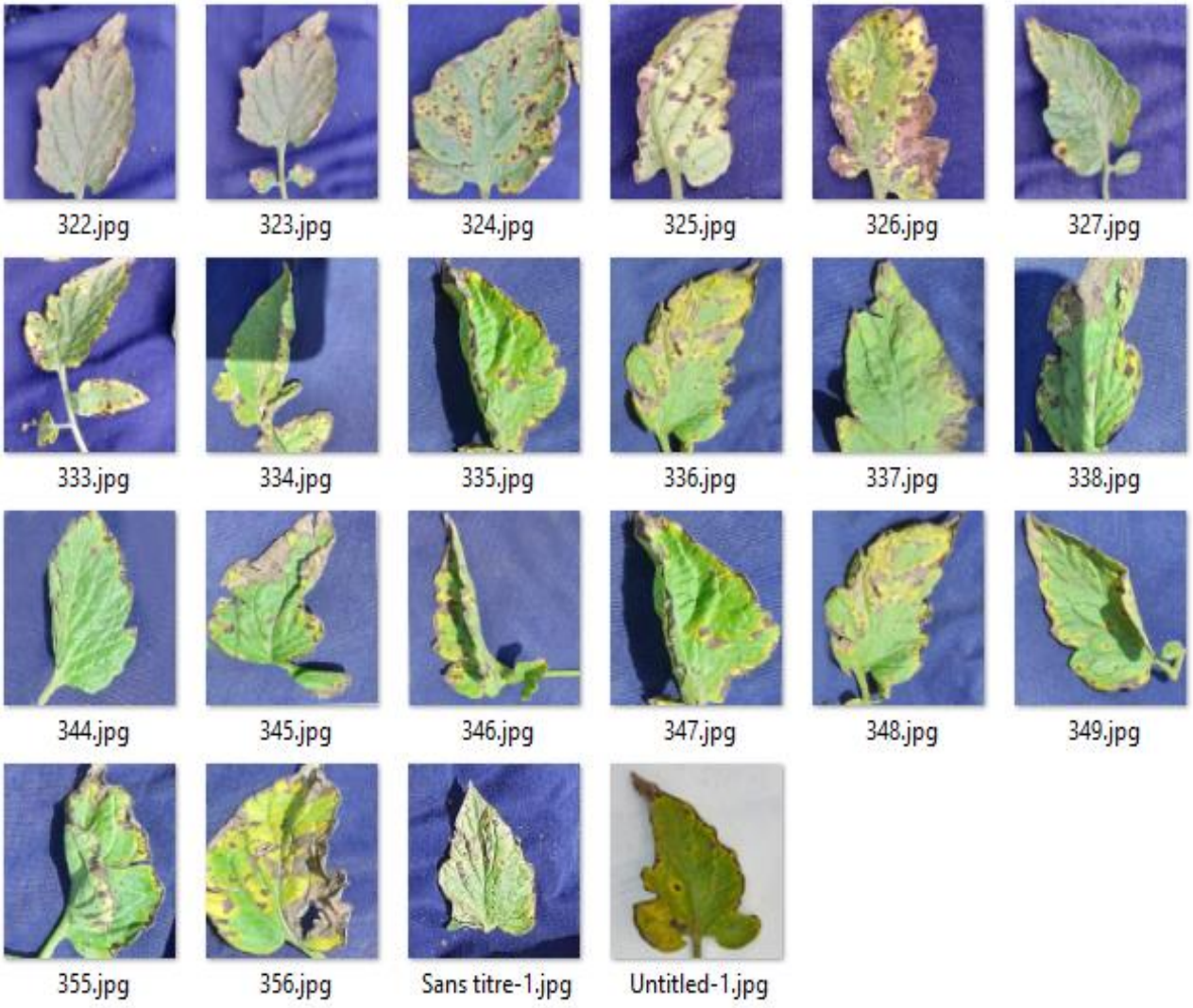

Figure Sample dataset from collected different tomato farm around Awash Melkasa, Meki and Alemtena

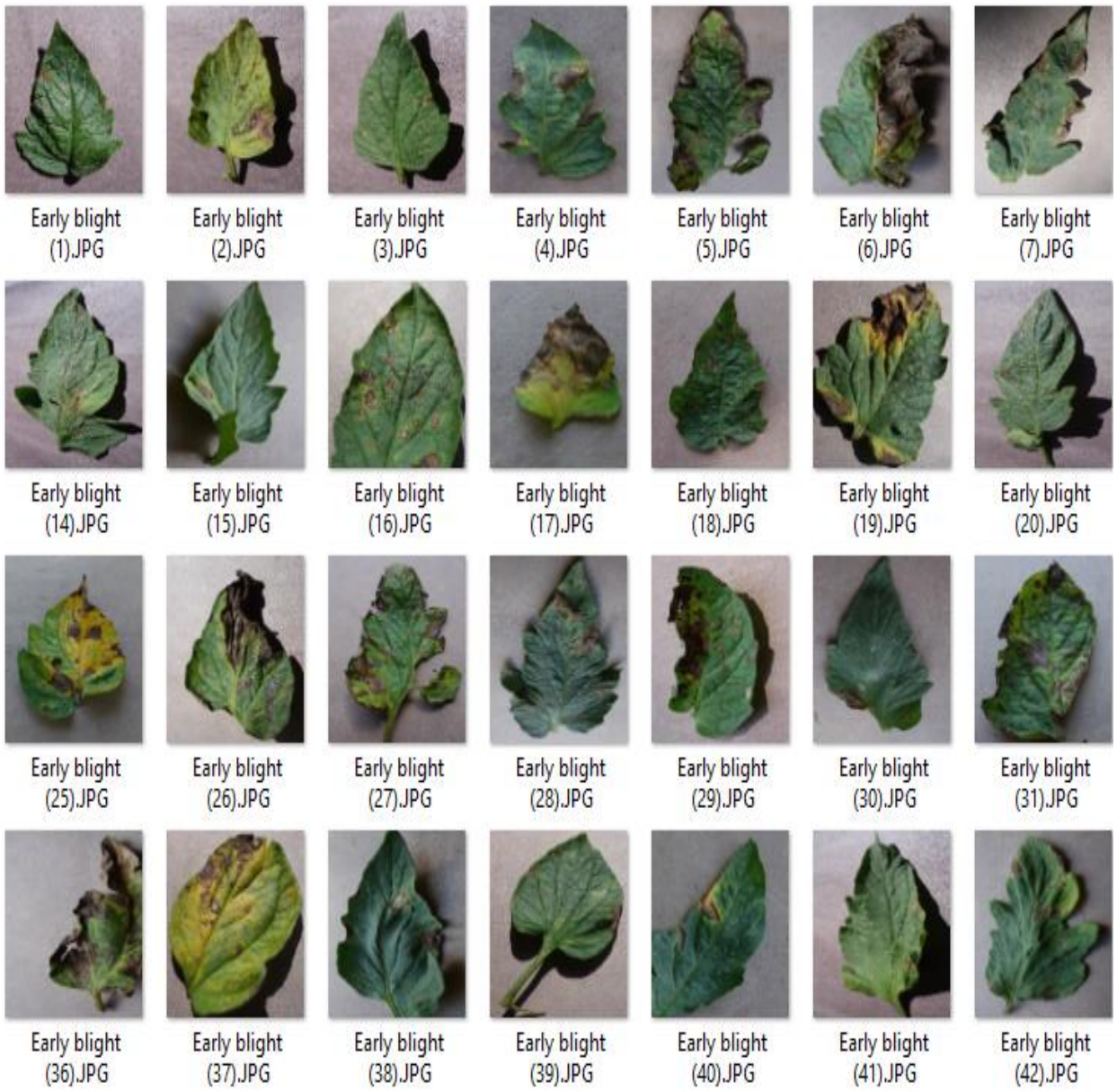

Figure Sample from plant village public website

8.2. Appendix II: Experiment of TomdiseasesNet Model

8.2.1. Important python libraries used

In [1]:
```
import numpy as np
import keras
from keras import backend as k
from keras.layers import Dense, Dropout, GlobalAveragePooling2D, Reshape, Activation, Conv2D, Input, MaxPooling2D, BatchNormaliza
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy, categorical_accuracy
from keras.preprocessing.image import ImageDataGenerator
from keras import models, regularizers
from keras.applications import imagenet_utils
from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard
import tensorflow as tf
import os
import seaborn as sn
import cv2
from random import randint
%matplotlib inline
```

```
Using TensorFlow backend.
```

8.2.2. Python code for Reading and augmented Image data

In [2]:
```
train_path = "../dataset_splitss/train"
valid_path = "../dataset_splitss/validation"
```

In [3]:
```
train_batches = ImageDataGenerator(preprocessing_function=keras.applications.mobilenet_v2.preprocess_input).flow_from_d
test_batches = ImageDataGenerator(preprocessing_function=keras.applications.mobilenet_v2.preprocess_input).flow_from_di
```

```
Found 7451 images belonging to 6 classes.
Found 1595 images belonging to 6 classes.
```

In [33]:
```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(              # this is the augmentation configuration used for training
        rotation_range=25,
        zoom_range=.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        fill_mode='nearest',
        brightness_range=[0.5,1.5],
        vertical_flip=True,
        horizontal_flip=True
        )

val_datagen = ImageDataGenerator()               # for val/testing only rescaling function
```

8.2.3. TomdiseasesNet model (CNN model) for the RGB image dataset

```
model = models.Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', padding="same", input_shape=(256, 256, 3)))
model.add(BatchNormalization(name='norm_1'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_2'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (5, 5), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_3'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_4'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3),  padding="same",activation='relu'))
model.add(BatchNormalization(name='norm_5'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (1, 1), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_6'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_7'))
model.add(MaxPooling2D((2, 2)))

model.add(GlobalAveragePooling2D())
model.add(Dense(6, activation='softmax'))

model.summary()
```

8.2.4. TomdiseasesNet model (CNN model) for Grayscale image dataset

```
model = models.Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', padding="same", input_shape=(256, 256, 1)))
model.add(BatchNormalization(name='norm_1'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_2'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (5, 5), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_3'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_4'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3),  padding="same",activation='relu'))
model.add(BatchNormalization(name='norm_5'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (1, 1), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_6'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), padding="same", activation='relu'))
model.add(BatchNormalization(name='norm_7'))
model.add(MaxPooling2D((2, 2)))

model.add(GlobalAveragePooling2D())
model.add(Dense(6, activation='softmax'))

model.summary()
```

8.2.5. Both RGB and Grayscale Model Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
norm_1 (BatchNormalization)  (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
norm_2 (BatchNormalization)  (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 64)        51264
_________________________________________________________________
norm_3 (BatchNormalization)  (None, 64, 64, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
norm_4 (BatchNormalization)  (None, 32, 32, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 128)       73856
_________________________________________________________________
norm_5 (BatchNormalization)  (None, 16, 16, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 256)         33024
_________________________________________________________________
norm_6 (BatchNormalization)  (None, 8, 8, 256)         1024
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 4, 4, 256)         590080
_________________________________________________________________
norm_7 (BatchNormalization)  (None, 4, 4, 256)         1024
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 1542
=================================================================
Total params: 800,166
Trainable params: 798,502
Non-trainable params: 1,664
```

8.2.6. Python Code for Fitting or Running the Model

```
model.compile(Adam(lr=.001), loss='binary_crossentropy', metrics=['accuracy'])
```

```
tb_counter  = len([log for log in os.listdir(os.path.expanduser('~/Desktop/new/color/')) if 'local' in log]) + 1
tensorboard = TensorBoard(log_dir=os.path.expanduser('~/Desktop/new/color/') + 'zewdu_' + '_' + str(tb_counter),
                          histogram_freq=0,
                          write_graph=True,
                          write_images=True)

history=model.fit_generator(train_batches,
                    steps_per_epoch=50,
                    validation_data=test_batches,
                    validation_steps=1,
                    epochs=200,
                    verbose=1,
                    callbacks = [])
```