ONION DISEASES CLASSIFFICATION BY USING DEEP LEARNING TECHNIQUES

Bilal Haji Wado

A thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Office of Graduate Studies

Adama Science and Technology University

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

September, 2024

Adama, Ethiopia

ONION DISEASES CLASSIFFICATION BY USING DEEP LEARNING TECHNIQUES

Bilal Haji Wado

Advisor: **Tilahun Melak** (PhD)

A thesis Submitted to the Department of Computer Science and Engineering
School of Electrical Engineering and Computing

Office of Graduate Studies
Adama Science and Technology University
Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

September, 2024
Adama, Ethiopia

APPROVAL OF BOARD OF EXAMINER

I, the advisors of the thesis entitled "**Onion Diseases Classifications by using Deep Learning Techniques**" and developed by Bilal Haji Wado hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Advisor/Supervisor	*Signature*	*Date*

We, the undersigned, members of the Board of Examiners of the thesis Bilal Haji read and evaluated the thesis entitled "**Onion Diseases Classifications by using Deep Learning Techniques**" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of degree of Masters of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

DECLARATION

I hereby declare that this Master Thesis entitled "**Onion Diseases Classifications by using Deep Learning Techniques**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Name of Student	*Signature*	*Date*

RECOMMENDATION

I/we, the advisor(s) of this thesis, hereby certify that I/we have read the revised version of the thesis entitled "**Onion Diseases Classifications by using Deep Learning Techniques**" prepared under my/our guidance by Bilal Haji submitted in partial fulfillment of the requirements for the Degree of Master's in Computer Science and Engineering. Therefore, I/we recommend the submission of revised version of the thesis to the department following the applicable procedures.

Major Advisor/Supervisor	Signature	Date

ACKNOWLEDGMENT

First and foremost, I would like to express my gratitude to God for granting me the strength and perseverance throughout my study and to complete this thesis. I would also like to extend my heartfelt appreciation to Dr Tilahun Melak for his exceptional mentorship, unwavering support, and guidance. His expertise and encouragement have been instrumental in shaping this research project, and I am truly grateful for his dedication to my academic growth.

I also thank Awash Malkasa Agricultural Research Institutes Center Office and Administrative Office for their provision of voluntary to information sharing about the study Area. Furthermore, I would like to thank my families and assistants for their multidimensional support and the entire academic journey.

TABLE OF CONTENTS

APPROVAL OF BOARD OF EXAMINER I
DECLARATION II
RECOMMENDATION III
ACKNOWLEDGMENT IV
LIST OF TABLES VIII
LIST OF FIGURES IX
LIST OF ABBREVIATION XI
ABSTRACT XII
CHAPTER ONE 1
1. INTRODUCTION 1
1.1. BACKGROUND OF STUDY 1
1.2. MOTIVATION 2
1.3. PROBLEM OF THE STATEMENT 3
1.4. RESEARCH QUESTION 4
1.5. OBJECTIVES 5
1.5.1. General Objective 5
1.5.2. Specific Objectives 5
1.6. SIGNIFICANCE OF THE STUDY 5
1.7. SCOPE OF THE STUDY 6
1.8. LIMITATION OF THE STUDY 6
CHAPTER TWO 8
LITERATURE REVIEW 8
2.1. PREVALENCE OF ONION LEAF DISEASES IN ETHIOPIA 8
2.2. DEEP LEARNING FOR CROP LEAF DISEASE DETECTION AND CLASSIFICATION 11
2.3. CHALLENGES AND FUTURE DIRECTIONS 13
2.4. DEEP LEARNING 14
2.4.1. Artificial Neural Networks 15
2.5. CONVOLUTIONAL NEURAL NETWORK 17
2.6. POPULAR CONVOLUTIONAL NEURAL NETWORK (CNN) FRAMEWORKS 27
2.7. FEATURE EXTRACTION USING LOCAL FEATURE DESCRIPTOR 30

2.8. ACTIVATION FUNCTIONS 31
2.9. RELATED WORK 32
CHAPTER THREE 37
RESEARCH METHODOLOGY 37
3.1. INTRODUCTION 37
3.2. RESEARCH DESIGN 37
3.3. THE PROPOSED ARCHITECTURE 38
3.4. DATA COLLECTION AND PREPROCESSING 39
3.5. IMAGE RESIZING 40
3.6. IMAGE NORMALIZATION 40
3.7. IMAGE AUGMENTATION 40
3.8. IMAGE SEGMENTATION 41
3.9. DATA SPLITTING 42
3.10. FEATURE EXTRACTION 42
3.11. CLASSIFICATION 42
3.11.1. Classification using SoftMax classifier 43
3.11.2. Classification using VGG model 43
3.12. REGULARIZATION TECHNIQUES 45
3.12.1. Dropout 45
3.12.2. Early stopping 46
3.13. FINDING OPTIMAL HYPER-PARAMETERS 46
3.13.1. Epochs 47
3.13.2. Batch size 47
3.13.3. Learning rate 47
3.14. OPTIMIZATION TECHNIQUES 47
CHAPTER FOUR 49
EXPERIMENTAL RESULT AND DISCUSSION 49
4.1. CHAPTER OVERVIEW 49
4.2. MODEL SELECTION 49
4.3. IMPLEMENTATION ENVIRONMENT 50
4.4. EXPERIMENTAL SETUP 50

4.4.1. Hyper Parameters configuration 50
4.4.2. Input Layer Setup 51
4.4.3. Feature Extraction Layers Setup 51
4.4.3.2. POOLING LAYER 52
4.5. CLASSIFICATION LAYERS SETUP 52
4.5.1. Fully Connected Layer Setup 52
4.6. TRAINING OF PROPOSED MODEL 53
4.7. EXPERIMENTATION RESULT 53
4.8. EXPERIMENT ONE 54
4. .8.1. ResNet50 model 54
4.8.2. VGG-19 Model 55
4.8.3. CNN model during the training phase 56
4.8.4. DenseNet201 Model 57
4.8.5. InceptionV3 Model 59
4.9. SUMMARY OF EXPERIMENTS 60
4.10. COMPARATIVE ANALYSIS OF OUR STUDY WITH RELATED WORKS 64
CHAPTER FIVE 66
CONCLUSION AND RECOMMENDATION 66
5.1. CONCLUSION 66
5.2. FUTURE WORK 67
5.3. CONTRIBUTION 67
APPENDICES 68

LIST OF TABLES

Table 2.1 Summary of Related Works 35
Table 3.1 Parameters of augmentation techniques 41
Table 4.1. Configuration of hyperparameters for the proposed work 51
Table 4.2. Different learning rate values on RGB images 55
Table 4.3. Comparison of Models with Adam Optimizer 61
Table 4.4. Comparison of Models with SGD Optimizer 63
Table 4.5. Comparison of Models with RMSprop Optimizer 64

LIST OF FIGURES

Figure 2.1: Basic components of CNNs 18
Figure 2.2. Simple binary representation of an image 19
Figure 2.3. common CNN architectures 19
Figure 2.4. formation of convolution from input images on gray scale images 21
Figure 2.5. convolution on Gray Scale images 22
Figure 2.6. max pooling and average pooling 24
Figure 2.7. Flattened one-dimensional vector 25
Figure 2.8. fully connected layer 26
Figure 2.9. Over fitting problems 27
Figure 3.1. Proposed Model Architecture 39
Figure 3.2. Architecture of VGG16 44
Figure 4.1. Classification Accuracy of Resnet50 model 54
Figure 4.2 Classification Accuracy of VGG19 56
Figure 4.3 Training and validation accuracy and loss graph of the VGG19 model 56
Figure 4.4. Classification Accuracy of CNN model 57
Figure 4.5 Training and Validation accuracy and loss graph of CNN model 57
Figure 4.6. Classification Accuracy of DenseNet201 58
Figure 4.7. Training and Validation accuracy and loss graph of DenseNet201 59
Figure 4.8. Classification Accuracy of InceptionV3 59
Figure 4.9.Training and Validation accuracy and loss graph of InceptionV3 60
Figure 4.10.Comparison of experiments using Adam Optimizer with 64 Batch Size 61

List of Abbreviation

1	AdaGrad	Adaptive Gradient Algorithm
2	ADAM	Adaptive Moment Estimation
3	ANN	Artificial Neural Network
4	CNN	Convolutional Neural Network
5	DenseNet	Densely Connected Convolutional Network
6	DSRP	Design Science Research Process
7	FAO	Food and Agriculture Organization
8	GAN	Generative Adversarial Network
9	HOG	Histogram of Oriented Gradients
10	ICT	Information and Communication Technology
11	IDM	Integrated Disease Management
12	ILSVRC	ImageNet Large Scale Visual Recognition Challenge
13	LBP	Local Binary Pattern
14	ReLU	Rectified Linear Unit
15	RMSprop	Root Mean Square Propagation
16	RNN	Recurrent Neural Network
17	SGD	Stochastic Gradient Descent
18	SIFT	Scale Invariant Feature Transform
19	SoftMax	Softmax Function
20	SSD	Single Shot MultiBox Detector
21	SURF	Speeded Up Robust Features
22	VGG	Visual Geometry Group

ABSTRACT

The majority of Ethiopians continue to rely heavily on the agricultural sector, which remains one of the most significant industries in the country. This sector faces numerous challenges, including plant diseases that negatively impact yield quality and productivity. Early detection and accurate diagnosis of onion health are crucial for preventing the spread of various plant diseases, ideally through the use of technology rather than manual labor. Onion crops play a vital role in Ethiopian agriculture, significantly contributing to food security and livelihoods. However, onion diseases have resulted in substantial losses and reduced yields. Traditional observation methods used by farmers and agricultural experts can be time-consuming, costly, and sometimes inaccurate. Deep learning techniques provide some of the most effective models for identifying plant diseases. Convolutional Neural Networks (CNNs) are a prominent approach, consisting of multiple processing layers that learn to represent images at varying levels of abstraction. These models have greatly advanced visual object recognition and image classification, making them well-suited for detecting and classifying onion leaf diseases. The primary objective of this study was to design and develop a deep learning-based model for identifying and classifying the three most critical onion leaf diseases: Downy Mildew, Onion Leaf Blight, and Bacterial Soft Rot. A total of 12,438 images, including augmented images across four categories, were collected and prepared for experimentation. This study specifically focuses on diseased and healthy onion leaves sourced from various agricultural research centers in Ethiopia. Experimental results demonstrate that the proposed ResNet50 model can effectively detect and classify the four classes of onion leaf diseases, achieving an impressive classification accuracy of 98.75%. This performance surpasses that of other classical deep learning models, including VGG19, DenseNet201, InceptionV3, and standard CNNs.

Keywords: Onion Disease, Convolutional Neural Network, Deep Learning models, CNN architectures

CHAPTER ONE

1. INTRODUCTION

1.1. Background of study

Onion (*Allium cepa*) is an economically significant crop that ranks as the second-largest vegetable crop globally (FAOSTAT, 2021). Recently in Ethiopia, onions have quickly gained popularity among both producers and consumers. Historically, Onions have played a crucial role not only in diets but also in traditional medicine. They are consumed in small quantities worldwide and are commonly used in households for seasoning dishes, flavoring sauces, soups, and sandwiches (Dahl et al., 2017).

The production of Onions has proven to be a lucrative venture for farmers, particularly for smallholder and subsistence farmers who cultivate them on limited land. In Ethiopia, Onion farming predominantly occurs in the midland and lowland regions, where the optimal growing altitude ranges from 700 to 1800 meters above sea level (Tadesse et al., 2020). The ideal mean monthly temperatures for Onion cultivation in Ethiopia typically fall between 18.2°C and 22.7°C (Ayana et al., 2019). Despite the promising potential of Onion farming, the crop's productivity remains below the global average due to the prevalence of various diseases that significantly affect its yield (Hassan et al., 2018). Nevertheless, Onions have become a vital source of livelihood for many individuals involved in their production and trade.

During the cultivation process, Onion crops are susceptible to numerous diseases, making early identification and accurate diagnosis essential to limit the spread of these ailments. Traditional methods of disease diagnosis, often reliant on visual inspection by farmers or agricultural experts, can be time-consuming, costly, and occasionally inaccurate (Ferentinos K, 2018). Previous studies have explored the use of image processing and machine learning techniques to detect and classify plant diseases. These approaches have focused on developing classifiers that utilize images captured in the field. However, many of these classifiers depend on handcrafted features designed by experts, which limits their automation and scalability (Khan et al., 2020).

Moreover, training these classifiers requires a substantial number of expertly labeled images, which can be prohibitively expensive to collect manually (Zhang et al., 2020). This challenge has

often resulted in the use of small datasets for training and testing classifiers, leading to issues such as overfitting (Hernandez et al., 2021). In recent years, deep learning algorithms have been adopted by the computer vision community, outperforming traditional methods across various fields. These algorithms are particularly suited for image processing tasks because they can learn to identify complex patterns that may be beyond human recognition (LeCun et al., 2015). Consequently, deep learning can facilitate automatic image classification, object detection, and even image generation.

In the agricultural sector, both the quantity and quality of products are critical considerations for farmers. Timely and accurate identification of crop diseases is essential for effective farm management (Mishra et al., 2021). Traditionally, disease characteristics in plants have been extracted through manual inspections, a process that can be labor-intensive and time-consuming. Therefore, automating plant disease identification through machine learning can greatly benefit users, especially those with limited knowledge about the crops they are cultivating, such as novice farmers and gardeners who cannot afford expert agronomic services (Kumar et al., 2020).

This thesis aims to contribute to the field by designing and developing deep learning techniques capable of accurately identifying and classifying the three most critical leaf diseases affecting Onion crops. By analyzing images of infected leaves, this research seeks to provide valuable tools for farmers, agricultural experts, and researchers, enabling them to take timely measures to prevent significant damage to Onion crops.

1.2. Motivation

Onions are needed in our daily life as a vegetable consumption. The cost of Onion in our market when are in low production by diseases. Onion diseases are very complex. Collecting data and work on it is very much enjoyable, we feel. We agreed and conducted research to see whether we could develop an expert system to assist specialists and farmers in diagnosing and treating various Onion diseases. As the rate of disease is very high in Bangladesh, we want to work with it and find the reason and the outcome. So that, we can make conscious about the danger of disease which may affect their body in the long run. So, we started to research this topic called "Onion disease detection & possible treatment." There was also some work on this area before, but our context is different from then, and the result outcome percentage is very high. Hence, they need experts with deep knowledge of Onion diseases. If we can control these diseases and give solutions to the farmers easily, then common production problems should be decreased.

1.3. Problem of the Statement

The agricultural production rate plays a pivotal role in the economic development of a country (Food and Agriculture Organization [FAO], 2021). Plants are a major source of food for the world population. However, plant diseases are the most significant impediment to the production and quality of food (Oerke, 2006). Additionally, plant diseases contribute to production loss, which can be tackled with continuous monitoring (Khan et al., 2020). Disease severity varies across and within districts. Root mealybug, leaf, and streak leaf spot were the most dominant and frequently occurring diseases of Onion (Bashir et al., 2022). According to Erenstein and Laxmi (2021), extension workers have several common roles: educating farmers and producers so they can help themselves; linking farmers/producers with research-based information to improve agricultural production, productivity, processing, and marketing of agricultural goods and services. These extension workers can serve as information brokers, community organizers, facilitators, and change agents. The primary roles of agricultural extension advisors have been to facilitate learning and extend new knowledge and technologies in non-formal educational settings to improve agricultural productivity and increase farmers' incomes. Extension officers are trained to diagnose pests and diseases by visual inspection or by conducting laboratory tests on plant samples. These approaches, however, have several limitations (Hassan et al., 2018).

- Extension officers are often too few to cover all farms. Thus, many farmers may lack extension services at critical times.
- Training of extension officers is costly and time consuming.
- Farmers and extension officers may not be able to correctly recognize non-native diseases and pests.
- A high level of expertise is required to distinguish between anomalies with visually similar characteristics. In such cases, even a highly trained expert may still arrive at a wrong diagnosis due to fatigue, poor illumination or poor eyesight. Moreover, individual experts are often specialists in a small set of disorders.

Continuous monitoring is necessary for early disease detection and to prevent disease from spreading. This is tedious, time-consuming, costly and inefficient to do on an ongoing basis. In addition to the high cost of lab equipment, lab tests are destructive since they involve collecting plant samples from the field and taking them to the lab for analysis.

High prediction accuracy and efficiency of Onion leaf diseases detection and classification is vital for increasing agricultural production and productivity. Therefore, developing Onion leaf diseases detection and classification model using image processing and deep learning techniques is crucial to tackle the spread of the disease. The identification of plant diseases at an early stage is crucial for global health and wellbeing. The traditional diagnosis process involves visual assessment of an individual plant by a pathologist through on-site visits. However, manual examination for crop diseases is restricted due to low accuracy and limited availability of human resources (Zhang et al., 2020). This, in turn, can lead to inappropriate pesticide usage, which may cause harmful chronic diseases in humans (Gomez et al., 2019). In this regard, early detection of such diseases is highly effective (Ferentinos K, 2018). Mostly, diagnosis, or the initial assessment of the Onion disease, is performed visually by a human eye. Since, Onion disease detection and diagnosis performed by the human eye is inaccurate due to, the Onion leaf diseases that can show similar symptoms. In such a situation, laboratory testing will be carried out to differentiate among diseases. Unfortunately, there is a shortage of experts and laboratories that can diagnose the disease and provide the right advice in the Onion growing areas of Ethiopia. It is not possible to address the problem of Onion detection and classification with the model developed in existing studies. Hence, developing a model suitable for Onion leaf diseases detection and classification is must. Therefore, developing Onion leaf diseases detection and classification model using Deep learning techniques is proposed.

1.4. Research Question

Thus, this study addresses the following research questions with consideration of the issues cited in the statement of problems:

RQ1. Which deep learning pre trained model perform best on Onion leaf diseases detection and classification?

RQ2. What is the optimal value of hyperparameters (Learning Rate, Batch Size, Epoch and Optimizers)?

RQ3. To what extent the proposed model works for diagnosing Onion disease detection and classification compared to the state-of-the-art model?

1.5. Objectives

1.5.1. General Objective

The general objective of this research is to develop Onion leaf diseases detection and classification model using Deep Learning Techniques.

1.5.2. Specific Objectives

To achieve the specified general objective, the study will have the following specific objectives:

- To review literature of previous works on plant leaf disease
- To prepare data set suitable for training deep learning model
- To select suitable deep learning algorithms for the classification of Onion leaf diseases.
- To find the optimal value of hyperparameters (Learning Rate, Batch Size, Epoch and Optimizers)
- To evaluate the performance of the proposed model for detection and classification of Onion disease compared to the state-of-the-art model.

1.6. Significance of the Study

This study would enable Agriculture Extension Officers to appreciate the importance of image processing in their field of work and it will reduce the workload of agricultural professionals. Besides this study will help the farmers take effective precautionary measures by assigning them to the early stages of the disease. They will increase their income and production volume. According to Khan, et al. (2020), extension workers have some common roles: educating farmers and producers so that the farmers/producers can help themselves; linking farmers/producers with research-based information to improve agricultural production, productivity, processing and marketing of agricultural goods and services. These extension workers can serve as information brokers, community organizers, facilitators and change agents. The primary roles of agricultural extension advisor have been to facilitate learning and extend new knowledge and technologies in non-formal educational settings to improve agricultural productivity and increase farmers' incomes. The use of ICT in extension service delivery is seen as a supplement to traditional extension methods rather than a replacement. It is a viable fact that artificial intelligence has come to make the life of human being easier and there is no way to eliminate its use. It is therefore left to the users to utilize it appropriately to improve their living conditions. In the first place, this study

will help pathologist understand the importance of artificial intelligence in general and deep learning in particular in their area. In addition, it will help achieve high quality and quantity production because it lets detail disease management techniques in curing the disease appeared on the Onion for pathologist, users, and agricultural extension workers. In addition, the study will help to minimize excessive toxic waste to water bodies as well as plants that harm human life. The study will help farmers reduce the cost of production that brings huge loss due to excessive use of fungicides on their plants. Once again, the outcome of the study can provide important suggestions and feedback to the corresponding authorities in order to take appropriate measures as far as agricultural production is concerned, in a case where there is an outbreak of crop diseases. Finally, this study would serve as a reference resource for other researchers who would seek to conduct further studies into the problems related to Onion disease detection and classification.

1.7. Scope of the Study

The main focus of this research is to develop an image-based Onion disease identification mechanism. After the identification of the disease, it recommends the appropriate solution for that disease, even though there are many diseases that attack the Onion crops this model can identify and recommend the appropriate treatment to the farmers with their own language via text for Downy Mildew**,** Onion Leaf Blight and Bacterial Soft Rot disease only, the other diseases and pests are out of the scope of this thesis work.

1.8. Limitation of the Study

The proposed system mainly focuses on only the Onion plant leaf, not including Flower, Root, bean, and others. Due to time and resource constraints, it was not possible to include all Onion diseases in this study. Thus, the main limitation of the study was restricted only to three Onion leaf diseases, namely (Downy Mildew**,** Onion Leaf Blight and Bacterial Soft Rot). The study doesn't include:

- Onion disease but Onion leaf diseases.
- Onion variety identification
- Real time Onion leaf diseases detection
- Severity of Onion leaf disease
- parasites like Bacterial and virus disease detection

This is due to our limited dataset samples we have, is not generalizable to address the above listed problems. Since, our dataset collected for Onion leaf disease chocolate spot, rust and Ascochyta blight couldn't reach the above listed problems with the model which train on this data samples. After the identification and detection of the Onion disease, there may be a way to recommend the appropriate medicine for that disease, however recommending the appropriate treatment for the identified disease, estimating the severity of the detected disease is beyond the scope of this thesis work.

1.9. Organization of the Thesis

As part of the study process and as a research tool, this newspaper was divided into five different chapters dealing with various issues.

Chapter One Introduction: Introduction briefly describes diabetes introduction, background, Background to the problem area, and states the problem, objective of the study, scope, and significance of the output of the research.

Chapter Two deals with a literature review: Articles, websites, journals, and literature are reviewed and listed on the appropriate movement in this study

Chapter three Methodology and materials: the research methodology, materials, data sources, and related categorization methods are described.

Chapter Four: Experimental Results and Discussion: Findings, data analysis, and presentation approaches are briefly discussed concerning knowledge gained via DL and other sources.

Chapter Five Conclusion and Recommendation: - For those elements that we identified as limits, extra effort is suggested as well as discussion, conclusion, recommendations, and a summary implementation strategy

CHAPTER TWO

LITERATURE REVIEW

2.1. Prevalence of Onion Leaf Diseases in Ethiopia

a. Downy Mildew

Downy mildew, caused by the fungal pathogen Peronospora destructor, is a significant disease affecting Onion production in Ethiopia. This pathogen thrives in humid and cool conditions, which are often prevalent during the rainy season. Research indicates that downy mildew can lead to severe yield losses, sometimes exceeding 50% if left unmanaged. The disease primarily manifests through the yellowing of leaves, which can progressively wilt and die. A distinctive feature of downy mildew is the appearance of a grayish mold on the underside of affected leaves, which is a result of spore production (Dawson et al., 2021). This mold not only impairs photosynthesis but also compromises bulb quality, making the Onions less marketable.

The impact of downy mildew on Onion crops is exacerbated by the close planting practices commonly used by farmers in Ethiopia, which create a microclimate conducive to the disease's development. A study by Tadesse et al. (2019) highlighted the importance of early detection and the application of fungicides to manage downy mildew effectively. They recommended regular monitoring of Onion fields, particularly during the rainy season, to identify symptoms early and apply fungicides as needed. Furthermore, the use of resistant Onion varieties has been identified as a crucial strategy to mitigate the impact of this disease (Desta et al., 2021). Research indicates that such varieties can significantly reduce disease incidence and improve overall yield.

b. Purple Blotch

Purple blotch, caused by the fungus Alternaria porri, is another critical disease affecting Onion crops in Ethiopia. This disease is characterized by dark purple lesions that can spread rapidly under favorable conditions, leading to leaf blight and significant yield reductions (Abate et al., 2021). The lesions often start as small, water-soaked spots that gradually enlarge and turn dark purple. In

severe cases, the disease can cause extensive leaf dieback, severely impacting bulb development and overall yield.

The economic implications of purple blotch are substantial, as affected plants often yield smaller bulbs with reduced market value. A study by Hassan et al. (2020) found that the presence of purple blotch can decrease marketable yield by up to 40%, emphasizing the need for effective control measures. The fungus can survive in plant debris, making crop rotation and sanitation essential components of disease management. Implementing these practices can help break the disease cycle and reduce the inoculum available for subsequent crops.

A study conducted by Mengistu et al. (2022) found that integrated disease management practices, including the use of resistant varieties and timely fungicide applications, can significantly reduce the incidence of purple blotch. Their research demonstrated that combining resistant cultivars with cultural practices such as proper spacing and rotation not only minimized disease incidence but also improved overall crop health and yield. Additionally, the timing of fungicide applications is critical; applying fungicides at the first sign of disease symptoms can effectively control the spread of purple blotch.

Furthermore, educational programs targeting farmers are essential to raise awareness about the disease and promote best practices for its management. Training sessions can provide farmers with the knowledge needed to identify symptoms early and implement appropriate control measures. Collaborative efforts between researchers, extension services, and farmers will be vital in developing and disseminating effective management strategies for purple blotch in Ethiopia.

In conclusion, the management of purple blotch in Ethiopian Onion production requires a multifaceted approach that includes the use of resistant varieties, timely fungicide applications, and cultural practices. Continued research and farmer education will be crucial in mitigating the impact of this disease on Onion yield and quality.

3. Bacterial Soft Rot

Bacterial soft rot, primarily caused by Erwinia carotovora, poses a severe threat to Onion production in Ethiopia. This disease manifests as water-soaked lesions on the leaves and bulbs, leading to a mushy rot that can devastate entire crops (Alemayehu et al., 2023). The bacteria enter

the plant through wounds, which can be caused by mechanical damage, insect feeding, or poor cultural practices. Once inside, the bacteria proliferate, producing enzymes that break down plant tissues, leading to the characteristic soft rot symptoms.

Research indicates that bacterial soft rot is often exacerbated by poor post-harvest handling and storage conditions, which promote the spread of the bacteria (Beyene et al., 2021). Inadequate drying and storage of harvested Onions can create an environment conducive to bacterial growth, leading to substantial losses during storage. Effective management strategies include improving field hygiene, implementing proper irrigation practices, and enhancing post-harvest handling techniques to reduce the incidence of this disease.

The role of irrigation management in controlling bacterial soft rot cannot be overstated. Overhead irrigation, for example, can contribute to leaf wetness, creating a favorable environment for bacterial infection. Instead, drip irrigation is recommended to minimize leaf wetness and reduce the risk of disease. Additionally, maintaining proper plant spacing can improve air circulation and reduce humidity around the plants, further mitigating the risk of soft rot.

The development of resistant Onion cultivars is also a critical area of research. A study by Fekadu et al. (2022) highlighted the potential for genetic resistance to significantly mitigate losses from bacterial soft rot. Research into the genetic basis of resistance in Onions is ongoing, and the identification of resistant varieties can provide farmers with effective tools to combat this disease.

In general, bacterial soft rot presents significant challenges to Onion production in Ethiopia. Implementing effective field and post-harvest management practices, along with the development of resistant cultivars, is crucial for reducing the incidence and impact of this disease. Continued research and farmer education will be essential for enhancing Onion production and ensuring food security in the region. Moreover, downy mildew remains a significant challenge for Onion producers in Ethiopia. The integration of early detection, effective fungicide application, and the development and use of resistant varieties are essential components of a comprehensive management strategy. Continued research into the pathogen's biology and epidemiology will further enhance these management practices and support sustainable Onion production in the region.

2.2. Deep Learning for Crop Leaf Disease Detection and Classification

The application of deep learning in agriculture, particularly for crop leaf disease detection and classification, has emerged as a transformative approach to enhancing food security and optimizing agricultural practices. As global food demand rises, the need for efficient disease management systems becomes increasingly critical. Traditional methods of disease identification, which often rely on visual inspection by trained experts, are not only time-consuming but also subject to human error. The advent of deep learning techniques offers a promising solution by enabling rapid, accurate, and scalable disease detection through image analysis. This literature review explores the advancements in deep learning methodologies for detecting and classifying crop leaf diseases, highlighting key studies and their contributions to the field.

2.1.1. Deep Learning Frameworks for Disease Detection

Deep learning frameworks, particularly Convolutional Neural Networks (CNNs), have been widely adopted for the task of crop leaf disease detection. CNNs are designed to automatically extract hierarchical features from images, making them highly effective for image classification tasks. A seminal study by Mohanty et al. (2016) utilized CNNs to classify images of crop leaves, achieving an impressive accuracy of 99.35% on a dataset comprising 14,000 images across 1,000 plant species. This study demonstrated the capability of deep learning to handle large-scale datasets and the potential for real-world applications in agricultural diagnostics.

Further developments in deep learning architectures have led to improved performance in disease detection tasks. For instance, Kalyan et al. (2020) introduced a novel multi-layered CNN model that enhanced feature extraction by incorporating deeper layers and advanced pooling techniques. Their model achieved an accuracy rate of over 95% in classifying various crop diseases, emphasizing the importance of model architecture in achieving high classification performance. Additionally, the authors highlighted the role of data augmentation techniques, which involve artificially expanding the training dataset through transformations like rotation and scaling. This approach not only enhances the model's robustness but also mitigates overfitting—a common challenge when training deep learning models with limited data.

The integration of transfer learning has also proven beneficial in crop leaf disease detection. Transfer learning allows researchers to leverage pre-trained models on large-scale datasets such as ImageNet and adapt them for specific agricultural tasks. A study by Barbedo (2019) showcased

the effectiveness of transfer learning in classifying crop diseases, demonstrating that fine-tuning models like VGG16 and ResNet significantly improved accuracy while reducing training time. This approach is particularly advantageous in agricultural contexts, where annotated datasets may be scarce. By utilizing transfer learning, researchers can achieve high accuracy with limited labeled data, making deep learning more accessible for practical applications in crop disease management.

2.2.2. Classification of Crop Diseases Using Deep Learning

The classification of crop diseases based on leaf images is a critical area where deep learning has shown significant promise. Early detection and accurate classification of diseases are vital for implementing timely management strategies, ultimately leading to better crop yields. Research by Ferentinos (2018) demonstrated that deep learning frameworks could classify plant diseases with remarkable accuracy. In their study, a deep CNN was trained on a dataset of 54,000 images representing 38 different diseases across multiple crops, achieving an accuracy of 99.35%. This high level of performance illustrates the potential for deep learning to serve as an effective tool for farmers and agronomists in diagnosing crop health issues.

The ability of deep learning models to learn complex patterns in leaf structures is a key factor contributing to their success in disease classification. A notable example is the work by Aasim et al. (2020), which employed a deep learning approach to categorize diseases in tomato plants. Their model successfully distinguished between healthy and diseased leaves and identified specific diseases such as bacterial wilt and leaf curl. This capability is essential for targeted interventions, enabling farmers to implement precise management strategies that can mitigate disease spread and impact.

Moreover, the incorporation of Generative Adversarial Networks (GANs) into the classification process has emerged as an innovative approach to enhancing the training datasets for deep learning models. Liu et al. (2021) demonstrated that GANs could generate synthetic leaf images that mimic diseased samples, thus enriching the training dataset and improving model performance. This technique is particularly useful in situations where acquiring sufficient labeled data is challenging, allowing researchers to overcome data scarcity issues and improve the robustness of classification models.

2.3. Challenges and Future Directions

Despite the significant advancements in deep learning for crop leaf disease detection and classification, several challenges persist. One of the primary issues is the need for large, annotated datasets, which can be labor-intensive and costly to compile. Zhang et al. (2020) emphasized the necessity for efficient data collection and annotation methods to facilitate the training of robust models. Innovative solutions such as citizen science initiatives and crowdsourcing platforms are being explored to gather diverse datasets from various geographic locations, thus enhancing the availability of training data for deep learning applications in agriculture.

Another challenge is the interpretability of deep learning models. While these models often achieve high accuracy rates, their "black box" nature makes it difficult for users to understand the rationale behind specific predictions. Chen et al. (2019) highlighted the importance of developing explainable AI techniques to enhance the transparency of deep learning models in agricultural contexts. Providing insights into how models arrive at their conclusions can foster greater trust among farmers and practitioners, ultimately leading to broader adoption of these technologies in crop management.

Future research directions should focus on developing more efficient algorithms that require less computational power while maintaining accuracy. Additionally, integrating deep learning with emerging technologies such as Internet of Things (IoT) sensors and drone-based imaging could facilitate real-time monitoring and data collection, further enhancing the application of deep learning in precision agriculture. Ganaie et al. (2021) proposed a framework that combines deep learning with IoT for real-time leaf health monitoring, demonstrating the potential for innovative solutions that can significantly improve agricultural practices.

Overall, the application of deep learning for crop leaf disease detection and classification has shown significant promise in enhancing agricultural practices and improving food security. The advancements in CNNs, transfer learning, and data augmentation techniques have led to improved accuracy and efficiency in identifying and classifying crop diseases. While challenges such as data availability and model interpretability remain, ongoing research and technological integration offer potential pathways for overcoming these obstacles. As deep learning continues to evolve, its role in precision agriculture is expected to expand, providing farmers with powerful tools for optimizing crop management and improving yields.

2.4. Deep learning

Deep learning is a branch of machine learning that employs algorithms inspired by the structure and function of the brain (LeCun, Bengio, & Haffner, 2015; Schmidhuber, 2015). Artificial neural networks (ANNs) are a type of deep learning algorithm based on neural network architectures. In other words, artificial neural networks are computingsystems that take their cues from the brain's neural networks. These networks are based on a collection of connected components known as artificial neurons, or simply neurons. Through eachof these connections, a neuron can send a signal to another neuron, which will subsequently process the signal. Then it sends information to the neurons connected to it downstream. Normally,neurons are arranged in layers. On their inputs and signals, many layers may apply various transformations. The direction of travel is essentially from the first layer, known as the input layer,to the last layer, known as the output layer. The term "hidden layer" refers to any layer between the input and output layers.

Because there are so many powerful computers (cheap processing units like GPU) and so much data available today, neural network applications are expanding more quickly than ever in the last ten years. Processing layers may be present in an ANN. The number of layers we utilize in the network varies depending on the problem we are trying to solve. If there are only two or three layers, or if the number is not particularly high, the network's design is said to be shallow. Deep learning refers to this type of NN architecture, which is referred to as having a very large number of layers when an ANN is used.

Deep learning involves layer-by-layer analysis of the input, with each layer gradually extracting higher-level knowledge about the input. Let's look at a straightforward example of image analysis. Let's assume that your input image is composed of pixels that are arranged in a rectangular grid.

The pixels are now abstracted by the first layer. The second layer comprehends the image's edges. The following layer builds nodes from the edges. The following would then discover branches from the nodes. The output layer will then identify and categorize the entire object. In this instance, the feature extraction process switches from the input of one layer to the output of the next.

This method allows us to process a massive number of features, making deep learning a particularly potent tool. With higher-level learned features expressed in terms of lower-level characteristics, deep learning algorithms strive to take advantage of the unknown structure in the input distribution to find good representations, frequently at several levels LeCun, Y (2015). For many years, developing a feature extractor that converted raw data such as the pixel values of an image into a suitable internal representation or feature vector, from which a learning system such as a classifiercould detect or classify patterns in the input required extensive domain knowledge and careful hand engineering. Without first extracting features or defining a feature vector, deep learning enables the learning algorithm to be fed raw data (pixels in the case of picture data). Before the machine learning phase, the standard machine learning algorithm requires independent, hand- tuned feature extraction. There is only one neural network phase in deep learning. The layers of the neural network are initially learning to recognize the fundamental characteristics of the data, and this data is then forwarded to the other levels of the network for further network computing.

2.4.1. Artificial Neural Networks

The "artificial neural network" (ANN) is the most widely used and fundamental method for deep learning. ANNs are brain-inspired computer systems that aim to imitate how humans learn (Haykin, 2009). As seen in Figure 2.1, an ANN is constructed using artificial neurons, which are a group of connected units or nodes that resemble the neurons in a biological brain. Neurons are the cells that make up the human nervous system, and they are interconnected via axons and dendrites, with the junctions where these connections occur known as synapses. Neurons receive information, perform mathematical computations, and transmit signals through synapses represented by weights on each connection. Frank Rosenblatt, a psychologist, created the first ANN based on this model in 1958 (Rosenblatt, 1958).

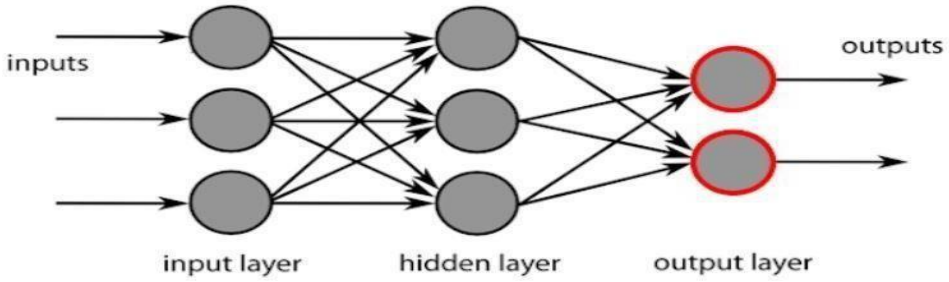
Multiple nodes make up ANNs, similar to how neurons are structured. Nodes are organized into several hidden layers and are closely connected. The input layer receives the data, which is then passed successively through one or more hidden layers before arriving at the output layer, which

makes useful predictions based on the input data. The ANN has become a popular model in many application fields due to its ability to automatically learn complex non-linear relationships from data. Furthermore, these models have achieved state-of-the-art results in image classification, thanks to advancements in computational power and techniques for creating and training ANNs (Krizhevsky, Sutskever, & Hinton, 2012). This innovation in image classification is credited to a particular kind of ANN called CNN. By transmitting computed values from the input neurons to the output neurons and utilizing the weights as intermediate parameters, an artificial neural network computes a function of the inputs. The weights holding the neurons together are modified during learning. Similar to how external stimuli are required for learning inbiological organisms, training material for ANN that contains examplesof input-output pairings for the function to be learned serves as the external stimulus. In my thesis,for instance, the training data might include pixels that represent images (the input) and their annotated labels as the output. These training data pairs are supplied into the neural network by making predictionsabout the output labels based on the input representations. Depending on how well the projected output for a given input matches the annotated output label in the training data, the training data offers feedback on the accuracy of the weights in the neural network. One might think of the mistakes produced by the neural network when computing a function as a form of painful feedbackin a living thing that causes the synaptic strengths to change. Similar to this, a neural network's weights between neurons are changed in reaction to incorrect predictions. The objective of adjusting the weights is to alter the computed function such that predictions in subsequent iterations are more accurate. As a result, to minimize the calculation error in that example, the weights are carefully modified in a way that is supported by mathematics. The function producedby the neural network is improved over time to provide predictions that are increasingly accurate by incrementally altering the weights between neurons over a large number of input-output pairs.The neural network will eventually be able to correctly recognize an Onion leaf in an image it hasnever seen if it is trained with lots of images of the Onion crop leaf. Model generalization is the capacity of learning over a limited set of input-output pairs to accurately compute functions of unknown inputs. The main benefit of any machine learning models comes from their capacity to transfer what they have learned from seen training instances to new ones. The nodes' weights andfunctions are applied as the input moves through each hidden layer, transforming it along the wayto create an output. Learning occurs when the output that was created is compared to the real value, and the error propagates back to change the weights. This

process is known as the backpropagationalgorithm. Either until the error is lowered below the desired threshold or for a predetermined number of repetitions (epochs), the process is repeated.

2.4.1.1. Feedforward Neural Network

A feedforward neural network is a type of artificial neural network in which nodes are not connected in a circular pattern (Goodfellow, Bengio, & Courville, 2016). It differs from its counterpart, recurrent neural networks (RNNs), as a result. The feedforward neural network is the first and most basic design of artificial neural networks (Haykin, 2009). As seen in Figure 2.1, the information in this network travels in only one direction, from the input nodes to the output nodes via any hidden nodes. The network contains no loops or cycles.



2.5. Convolutional Neural Network

Convolutional neural network is a class of deep learning methods which has become dominant in various computer vision tasks and is attracting interest across a variety of domains, including plant disease. Convolutional neural networks are deep learning neural network that is popular and have been utilizedin the application of image classification and clustering, object identification in images, and video scenes. Besides, to make it specific, they will be utilized in the application of categorizing tumors, faces, individuals, street signs, and many other types of visual data. Currently, CNNs have become prominent in computer vision fields, making image analysis and any application for issues that soughtto be addressed by CNNs.

CNNs do not perceive images in the same way humans do; at the first stage, they consist of two types of layers: convolutional and pooling layers (Krizhevsky, Sutskever, & Hinton, 2012). The layers of CNNs are trained in a robust manner, and convolutional neural networks are currently the most successful types of architecture for the detection and classification of images (LeCun, Bengio, & Haffner, 2015). A typical convolutional neural network architecture contains multiple layers that work on classifying edges and simple features in shallower layers, while deeper layers focus on

more complex features. Filters (kernels) are used to convolve a picture, and pooling is then applied. This process may continuefor a few layers until discernible features are achieved. All of the neurons in a feature map will,however, use the same set of filters. A pooling layer is performed on reducing the spatial dimensions of the representation output by a convolutional layer; this operation will reduce the number of parameters and the amount of computation within the network. Pooling layer performed self-sufficiently on its input at every depth and has a stride parameter similar to that of a filter in a convolutional layer and max pooling is commonly applied.

Since, the past few decades, deep learning has proved to be a prominent tool because of its ability especially, in handling large amounts of data. The advent of hidden layers has surpassed classical machine learning, especially in pattern recognition. One of the most popular deep neural networks isconvolutional neural networks or it is also known as CNN or ConvNet in deep learning under computer vision application languages, see figure 2.1.

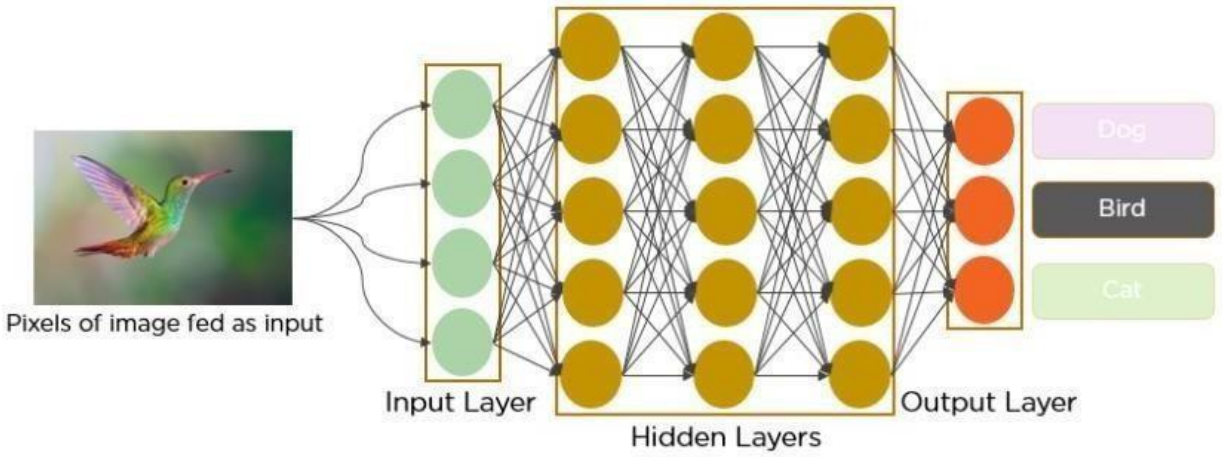


Figure 2.1: Basic components of CNNs (LeCun, Y., 2015).

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networkswhich most commonly applied to analyze visual imagery. Regarding neural networks common thinking of matrix multiplication is wrong rather it uses a special mathematical technique called convolution operated on two functions that will produce a third function and that expresses how the shape of one is modified by the other see, Figure 2.2.

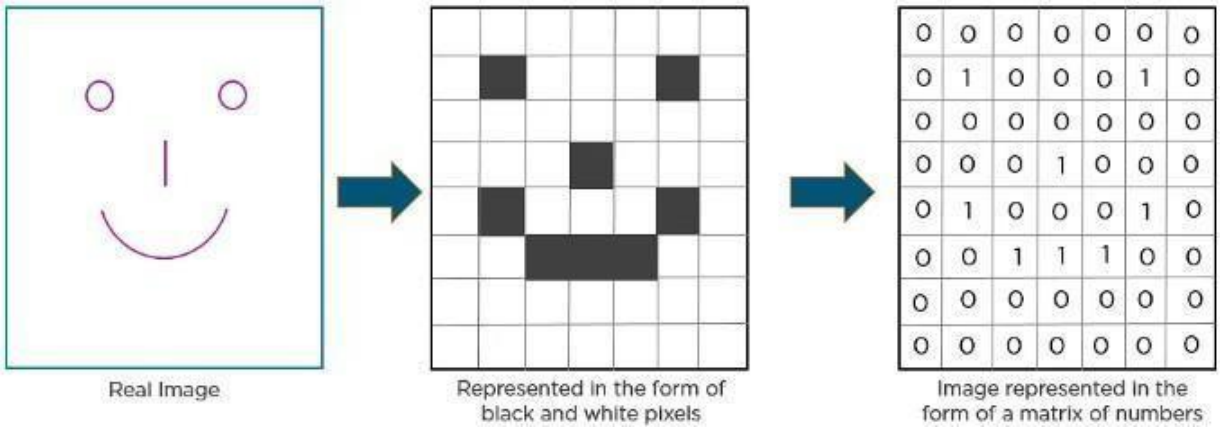


Figure 2.2. Simple binary representation of an image

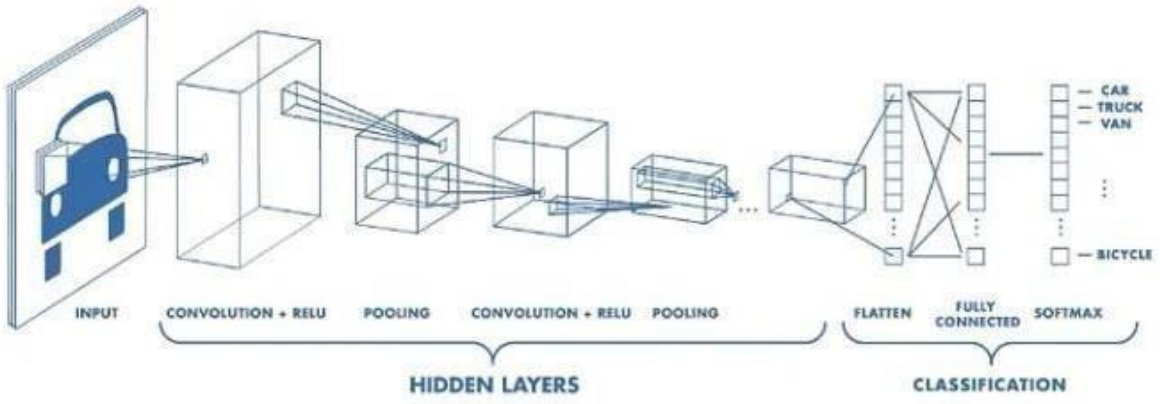
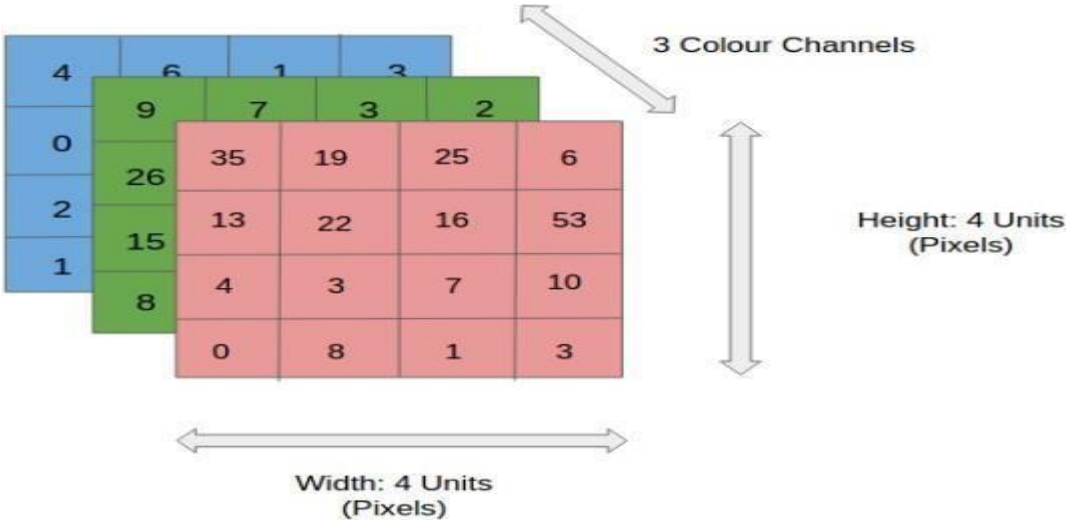


Figure 2.3. common CNN architectures

As we see above in figure 2.3, the role of ConvNet is to reduce another image in a form that is equivalentto the original input image with various features; the same representation in a way that is easy to process but, without losing features that is very essential for prediction tasks.

How does CNN work?

Before proceeding to discuss how CNN works it is better to discuss the nature of an image on how it's represented.



A convolutional neural network (CNN) typically consists of three layers: convolutional layer, poolinglayer, fully connected layer and Last Activation function.

A. Convolutional Layer

Convolutional layer is one component of the CNN architecture that is responsible for feature extraction which mainly composed of linear and nonlinear operations. These are convolution operation and activation function.

How It Works on Gray Scale Images

An Image is basically represented as an RGB format; red, green, and blue format.it is a matrix of pixel values represented in three planes whereas a gray scale image is the same but it is represented in a single plane. See figure… image has 3 color channels, height, and width.3 color channels, Red, Green, and Blue 3 different channels for each while Height and width has 4 units of pixels.

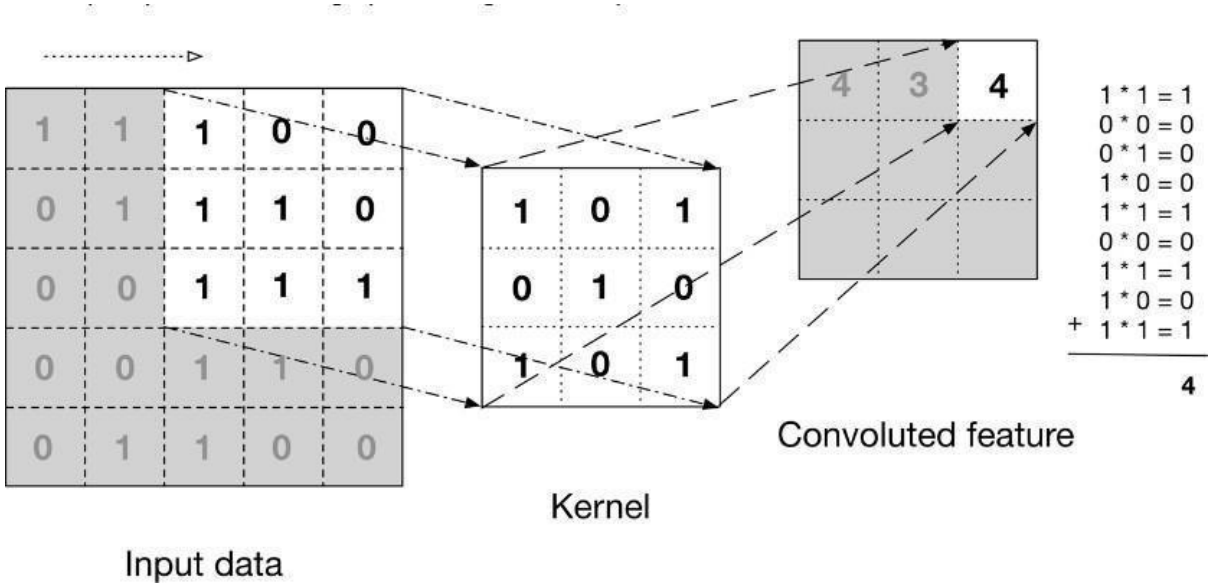
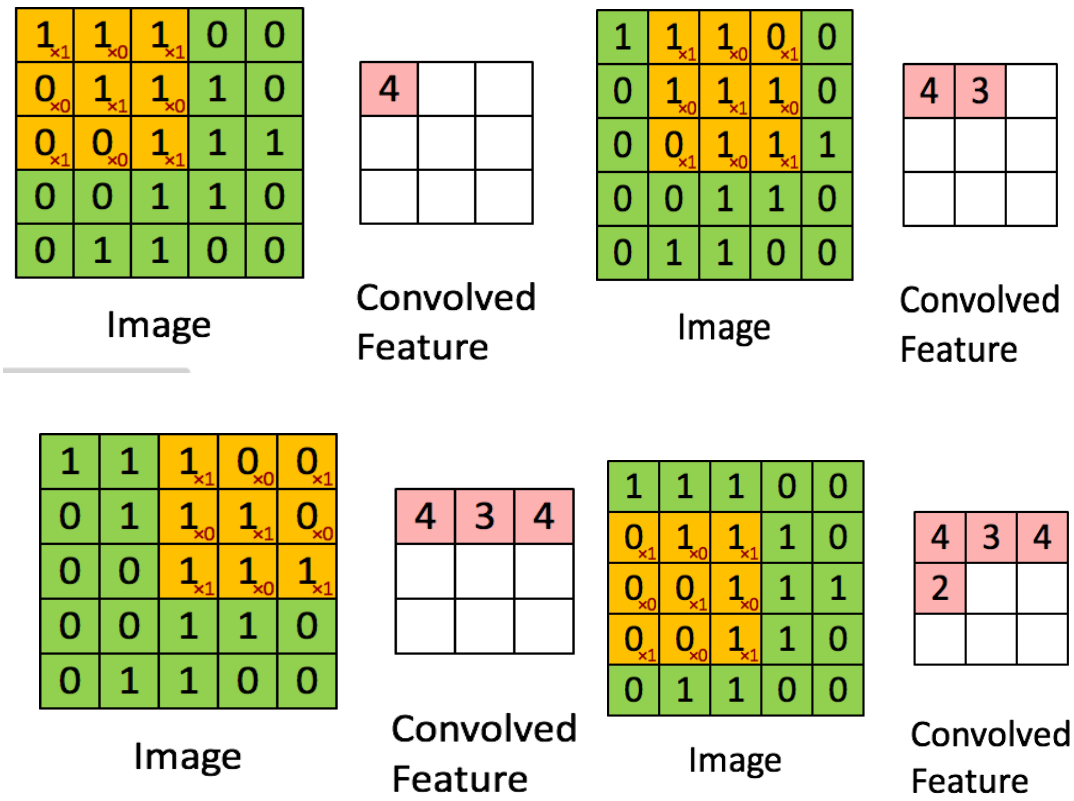


Figure 2.4. formation of convolution from input images on gray scale images Goodfellow, Bengio, & Courville, 2016).

Figure2.4. Showed a filter/3x 3 matrix kernels which finally convoluted to 3x3 matrixes applied to aninput image with 5x5 matrixes.



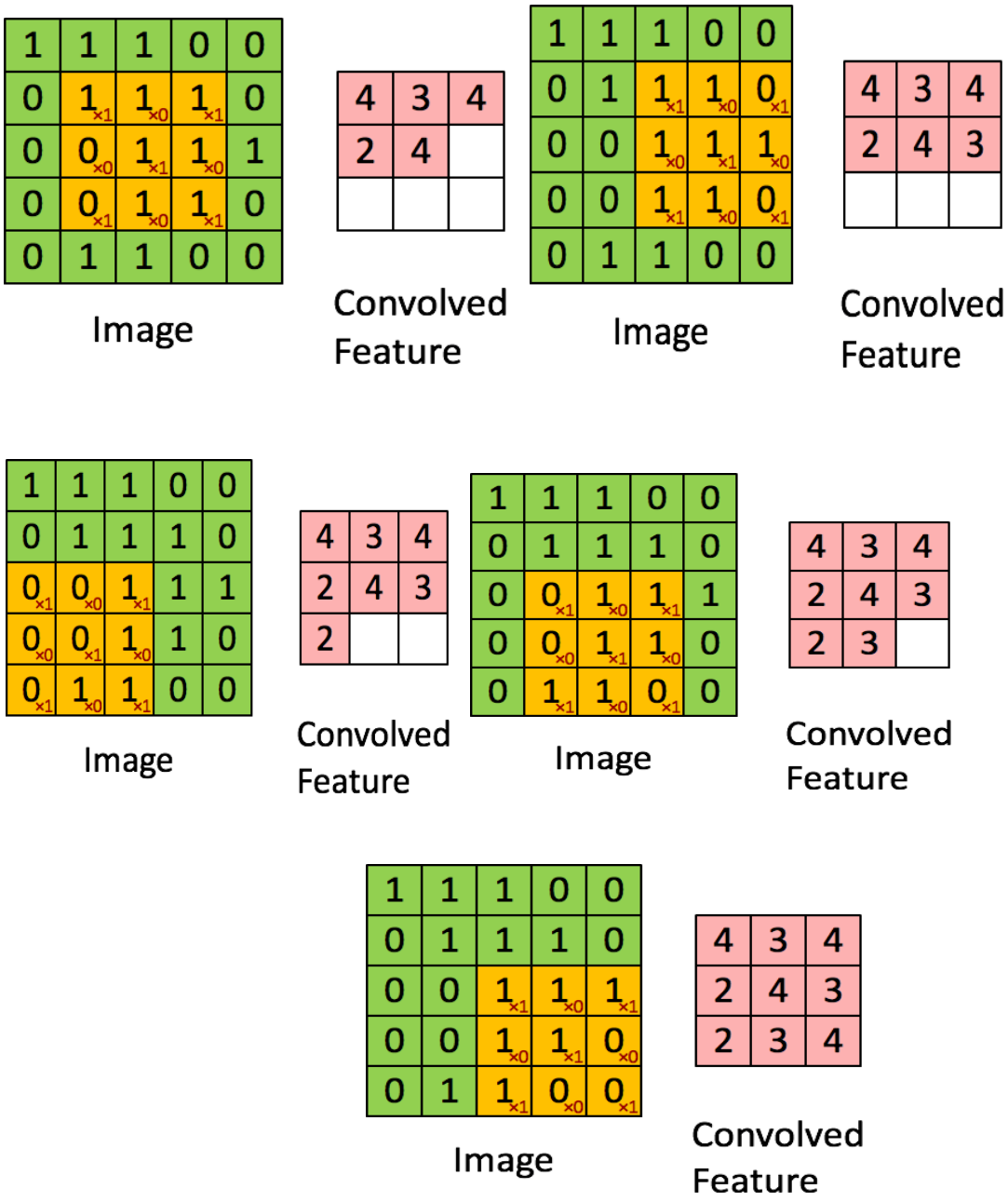
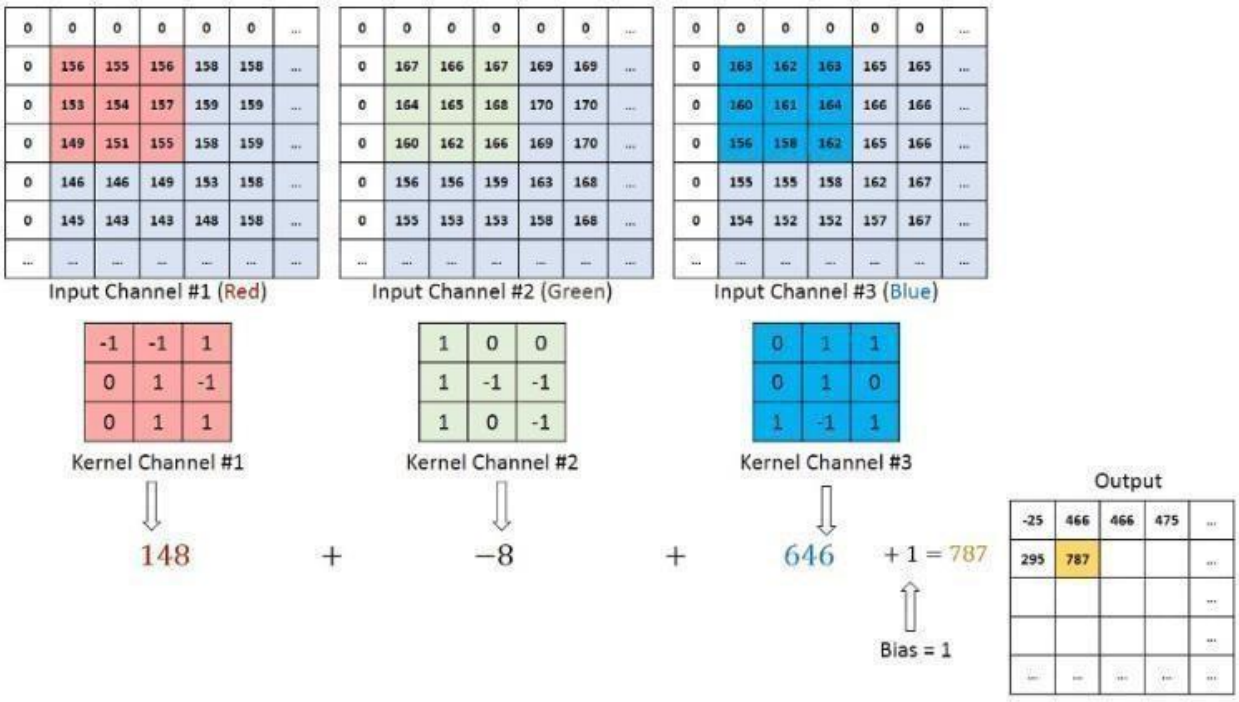
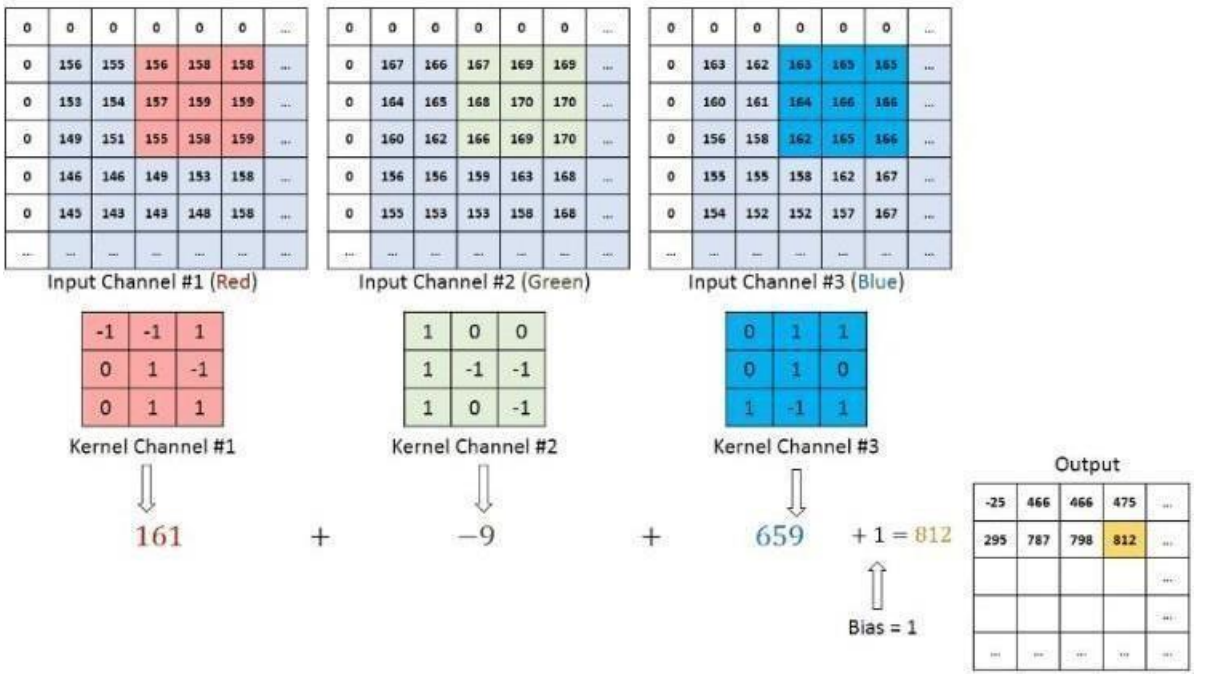


Figure 2.5. convolution on Gray Scale images (Goodfellow, Bengio, & Courville, 2016).

How it works on colored RGB images:


Input Channel #1 (Red)
Input Channel #2 (Green)
Input Channel #3 (Blue)
Kernel Channel #1
Kernel Channel #2
Kernel Channel #3
148
+
−8
+
646
+ 1 = 787
Bias = 1
Output


Input Channel #1 (Red)
Input Channel #2 (Green)
Input Channel #3 (Blue)
Kernel Channel #1
Kernel Channel #2
Kernel Channel #3
161
+
−9
+
659
+ 1 = 812
Bias = 1
Output

Pooling Layer

Pooling layer is a layer where down-sampling operation carried out and responsible for reducing thedimensionality of the feature map. The task of pooling layer is to reduce the spatial size of the convolved feature. This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling: average pooling and max pooling. Max pooling is a technique of finding the maximum value of a pixel of the portion on the image covered by the kernel. It can also perform as a noise suppressant and discarding the noisy activation altogetherand also performs de-noising along with dimensionality reduction.

Average pooling is a technique of finding the average of all values of a pixel on the portion of an image covered by the kernel. Hence, max pooling is better than average pooling since it has also noisediscarding tasks. The main purpose of a pooling layer is to reduce the number of parameters of the input tensor whichhelps to reduce over fitting, extracting features from the input tensor while reducing computation andaiding efficiency.

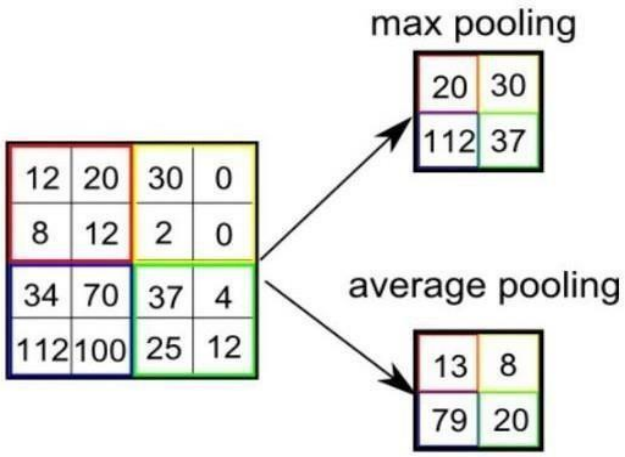


Figure 2.6. max pooling and average pooling

Fully Connected Layer

An output feature maps of the final convolution or pooling layer is typically flattened. Flattening is happened when pooling layer will be transformed into 1-dimensional (1D) array numbers(vectors), and connected to one or more fully connected layers also known as dense layers, in which every input is connected to every output by a learnable weight.

Once the features extracted by the convolutions layers and down sampled by the pooling layers are created, they are mapped by a subset of fully connected layer to the final outputs of the network suchas probabilities for each class in classification tasks. The final fully connected layer typically has thesame number of output nodes as the number of classes. Each fully connected layer is followed by a nonlinear function, such as ReLU.

Fully connected layer is also known as feed forward neural network, fully connected will form the last few layers in the network. The input to the fully connected layer is the output from the final pooling or convolutional layer, which is flattened and then feed into the fully connected layer, see Figure 2.8. The task of fully connected layer is taking the output from the previous layers which is "flattened" by flatten layers into a single vector that can be an input for the next stage. Flatten layer is responsible for converting a 3-dimensional layer in the network into one dimensional vector to fitthe input of a fully connected layer for the classification. For example, a 5x5x2 tensor would be converted into a vector size of 50. We use the softmax function to classify these features, which requires a one-dimensional input. This is why the flatten layer is required, see Figure 2.7.

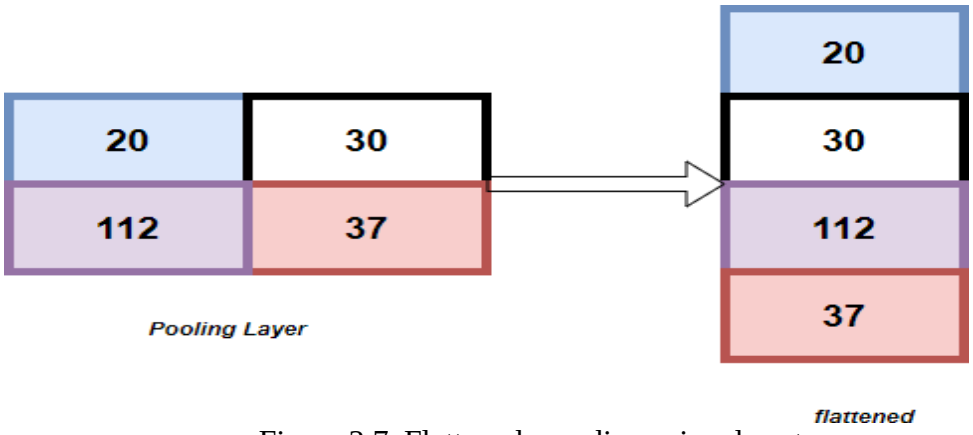


Figure 2.7. Flattened one-dimensional vector

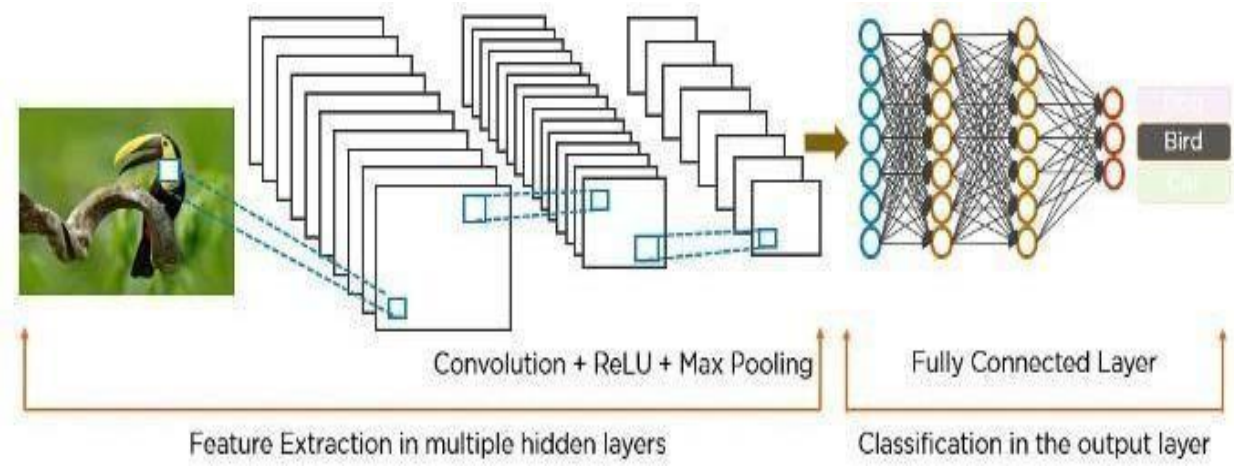


Figure 2.8. fully connected layer

As seen above in Figure 10, in fully connected layer all the neurons of the input are connected to every neuron of the output layer.

Last Layer Activation function

The activation function applied at the last fully connected layer is usually different from the others. An appropriate activation function needs to be selected based on the tasks. An activation function applied to the multiclass classification is a softmax activation function which is responsible for normalizing output real values from the last fully connected layer to target class probabilities, whereeach value ranges between 0 and 1 and all values sum to 1.

Overfitting

Overfitting occurs when a model can able to memorize irrelevant noise instead of learning the signal.Hence, perform less well on the subsequent new never seen before data and when the model is overfitted it is not generalizable on the unseen dataset. This is one of the main challenges in machinelearning techniques as overfitted model is not generalizable on the never seen before dataset. One of the tricks to recognize existence of over fitting is by monitoring accuracy loss up on trainingand validation curve. If a model performed well on training set compared to the validation set then the model has likely been over fit to the training data.

There have been several techniques to reduce over fitting and one of the best solutions to reduce overfitting is to collect enough training data. A model trained on larger dataset will generalizes better. The other alternatives is including regularization with dropout or weight decay, batch normalization,data augmentation and reducing architectural complexity. One of recently, introduced regularizationtechnique is where randomly selected

activations are set to 0 during the training so that the model becomes less sensitive to specific weight in the network, see figure 11.

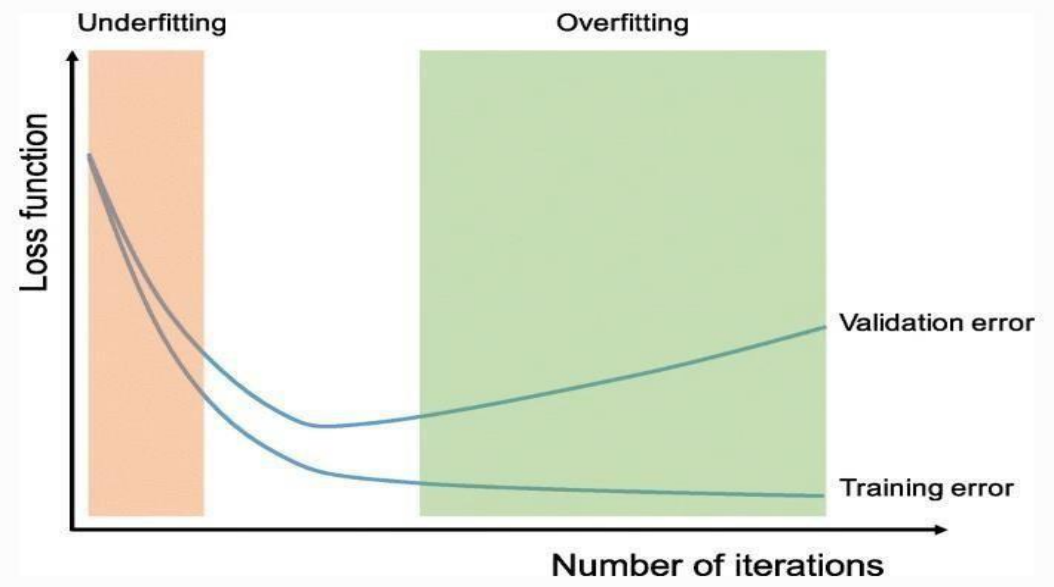


Figure 2.9. Over fitting problems

2.6. Popular Convolutional Neural Network (CNN) frameworks

LeNet Model

LeNet is a pioneering convolutional neural network (CNN) architecture developed by Yann LeCun and his colleagues in the late 1990s. It was specifically designed for handwritten digit recognition, and was one of the first successful CNNs for image recognition (LeCun, Bottou, Bengio, & Haffner, 1998). LeNet consists of several layers of convolutional and pooling layers, followed by fully connected layers. The architecture includes two sets of convolutional and pooling layers, each followedby a subsampling layer, and then three fully connected layers. The first convolutional layeruses a kernel of size 5×5 and applies 6 filters to the input image. The output of this layer isthen passed through a pooling layer that reduces the spatial dimensions of the feature maps. The second convolutional layer uses a kernel of size 5×5 and applies 16 filters to the output ofthe first pooling layer. This is followed by another pooling layer and a subsampling layer. The output of the subsampling layer is then passed through three fully connected layers, with 120, 84, and 10 neurons respectively. The last fully connected layer is used for classification, and produces a probability distribution over the 10 digits (0-9). LeNet was trained on the MNIST

dataset, which consists of 70,000 images of handwritten digits, and was able to achieve high recognition accuracy. The LeNet architecture, although relatively simple compared to current architectures, served as a foundation for many later CNNs, and it's considered as a classic and simple architecture for image recognition tasks.

AlexNet Model

AlexNet is a convolutional neural network (CNN) architecture that was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012. It was the first CNN to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a major image recognition competition, and it helped to establish CNNs as a powerful tool for image recognition (Krizhevsky, Sutskever, & Hinton, 2012). AlexNet consists of several layers of convolutional and pooling layers, followed by fully connected layers. The architecture includes five convolutional layers, three pooling layers, and three fully connected layers. The first two convolutional layers use a kernel of size 11×11 and apply 96 filters to the input image. The third and fourth convolutional layers use a kernel of size 5×5 and apply 256 filters. The fifth convolutional layer uses a kernel of size 3×3 and applies 384 filters. The output of these convolutional layers is then passed through max- pooling layers that reduce the spatial dimensions of the feature maps. The output of the pooling layers is then passed through three fully connected layers, with 4096, 4096, and 1000 neurons respectively. The last fully connected layer is used for classification, and produces a probability distribution over the 1000 ImageNet classes. AlexNet was trained on the ImageNet dataset, which consists of 1.2 million images with 1000 classes, and was able to achieve high recognition accuracy. The AlexNet architecture was the first to show that CNNs could significantly outperform traditional machine learning methods in image recognition tasks, and was an important step in the development of deeper architectures like VGGNet, GoogleNet,and ResNet (Simonyan & Zisserman, 2014; Szegedy et al., 2015; He et al., 2016).

Resnet model

ResNets (Residual Networks) are a type of deep learning algorithm that are particularly well-suited for image recognition and processing tasks. ResNets are known for their ability to train very deep networks without over fitting (He et al., 2016). ResNets are often used for keypoint detection tasks. Keypoint detection is the task of locating specific points on an object in an image. For example, keypoint detection can be used to locate the eyes, nose, and mouth on a human face.

ResNets are well-suited for keypoint detection tasks because they can learn to extract features from images at different scales. ResNets have achieved state-of-the-art results on many keypoint detection benchmarks, such as the COCO Keypoint Detection Challenge and the MPII Human Pose Estimation Dataset (He et al., 2016; Wei et al., 2016).

GoogleNet Model

GoogleNet, also known as InceptionNet, is a type of deep learning algorithm that is particularly well suited for image recognition and processing tasks. GoogleNet is known for its ability to achieve high accuracy on image classification tasks while using fewer parameters and computational resources than other state-of-the-art CNNs. Inception modules are the key component of GoogleNet. They allow the network to learn features at different scales simultaneously, which improves the performance of the network on image classification tasks. GoogleNet uses global average pooling to reduce the size of the feature maps before they are passed to the fully connected layers. This also helps to improve the performance of the network on image classification tasks (Szegedy et al., 2015; Lin et al., 2013).

GoogleNet uses factorized convolutions to reduce the number of parameters and computational resources required to train the network. GoogleNet is a powerful tool for image classification, and it is being used in a wide variety of applications, such as GoogleNet can be used to classify images into different categories, such as cats and dogs, cars and trucks, and flowers and animals.

MobileNet Model

MobileNets are a type of CNN that is particularly well-suited for image recognition and processing tasks on mobile and embedded devices (Howard et al., 2017). MobileNets are known for their ability to achieve high accuracy on image classification tasks while using fewer parameters and computational resources than other state-of-the-art CNNs. Additionally, MobileNets are being used for keypoint detection tasks and have achieved state-of-the-art results on various keypoint detection benchmarks (Gupta et al., 2018).

VGG-16 model

The VGG-16 model is a convolutional neural network (CNN) architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its depth, consisting of

16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 is renowned for its simplicity and effectiveness, as well as its ability to achieve strong performance on various computer vision tasks, including image classification and object recognition (Simonyan & Zisserman, 2014). The model's architecture features a stack of convolutional layers followed by max-pooling layers, with progressively increasing depth. This design enables the model to learn intricate hierarchical representations of visual features, leading to robust and accurate predictions. Despite its simplicity compared to more recent architectures, VGG-16 remains a popular choice for many deep learning applications due to its versatility and excellent performance (Chatfield et al., 2014). The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in computer vision where teams tackle tasks including object localization and image classification. VGG-16, proposed by Karen Simonyan and Andrew Zisserman in 2014, achieved top ranks in both tasks, detecting objects from 200 classes and classifying images into 1000 categories (Russakovsky et al., 2015).

2.7. Feature extraction using Local Feature Descriptor

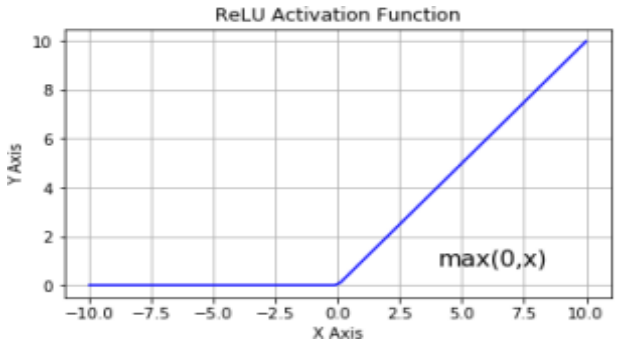
Feature Extraction is the one and the only pillar of machine learning algorithms. There were various feature extraction techniques. Feature extraction using SURF has been built based on the SIFT descriptor. SURF was utilized using an intermediate image representation called integral image. It was designed to detect local key points first followed by generating a descriptor for these points. SURF was designed based on 2D box filters and a fast Hessian blob detector based on the Hessian matrix determinant for detecting both scale-spaces and key point location. It was designed to approximate the second-order Gaussian derivatives of the integral images by applying a set of 2D box filters and scale spaces to detect blob-like key-point features. The SURF local feature descriptor was invariant to rotation, scale, illumination, noise, and viewing direction. On the other hand, Scale Invariant Feature Transform (SIFT) and other local interest point detectors such as SURF and ORB were used for problems to image searching, object matching, with a much lesser degree, image recognition. Another Feature extraction was LBP utilized for binary coding of image pixels using the threshold pixel values. It has been worked through a 3x3 kernel considering the central pixel value as a threshold and made a difference between the threshold pixel and the eight surrounding neighborhood pixels. When the neighborhood pixel was greater than the value of the threshold, it was set to „1, unless set to „0". LBP has been found robust against the change in illumination and it was simple computationally for encoding textures. Since LBP has represented the discrete patterns, not numerical features, it was hard to combine LBPs with other

discriminative features. Furthermore, it has generated higher-dimensional features thatwere sensitive to Gaussian noise in flat regions. The other local feature descriptor has been found with HOG that was used for extracting local region features using the histogram orientation of edge intensity. The HOG descriptor was designed first to calculate the image gradient using appropriate filter masks like Laplacian and Sobel to extract edge gradients and orientations.

2.8. Activation Functions

a. ReLU

To create a convolution layer, activation functions like ReLU are added to replace all negative pixel values with zero (0), which is performed after every convolution to introduce nonlinearity. The ReLU function is a very popular activation function defined as f(x)=maxf_0(0,x)$f(x)$=max(0,x), where xx is the input to a neuron (Nair & Hinton, 2010; Xu et al., 2015).



b. SoftMax

In deep learning models, the SoftMax function is the last layer which is used to compute the class's scores. The input of the SoftMax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class (Bishop, 2006; Goodfellow, Bengio, & Courville, 2016). The SoftMax activation function uses to get the resulting input from the process of learning and gives the probability output which is needed to classify images more accurately which is mostly used for binary. The softmax activation function has the following formula.

Sigmoid

The sigmoid function is an activation function in terms of underlying gate structured in correlation to Neurons firing, in Neural Networks which is mostly used for binary classification. The derivative also acts to be an activation function in terms of handling Neuron activation in terms of NN's. By using the sigmoid activation function the last layers (output layer) of the fully connected layer perform classification (probabilities of inputs being in a particular class) based on the training data (Hastie, Tibshirani, & Friedman, 2009; Goodfellow, Bengio, & Courville, 2016). The Sigmoid activation function has the following mathematical formula.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2.9. Related work

Several research works have shown, image-based evaluation methods are more reliable and replicable than visual human evaluation. In Sinha et al. (2021), the authors developed automated more suitable deep learning architecture to detect and classify tomato diseases specifically early blight, septoria leaf spot, bacterial spot and leaf curl by using deep detector: Faster RCNN with "deep feature extractors" such as ResNet50. The Train and test the proposed system end-to-end on tomato disease dataset specified in their works, which contains comprehensive images of different tomato diseases, taken at early, medium and final stages of infection status. However, author suggested data augmentation to decrease false positive in training and to improve accuracy. Because of a smaller number of annotated samples, certain classes with large difference in patterns appear to be confused with similar diseases and cause false positives and also model can detect and classify only those four tomato disease it can't consider the other tomato diseases or it sign as future work.

Mohanty, et al. (2016), used 9000 images of infected and healthy tomato leaves dataset that collected under controlled conditions, they trained a deep convolutional neural network to identify 5 diseases. The trained model achieved an accuracy of 99.84% on a held-out test set, demonstrating the feasibility of this approach. Overall, the approach of training deep learning models on increasingly large and publicly available image datasets presents a clear path toward Smartphone-assisted crop disease diagnosis on a massive global scale. However; the Images are taken under controlled conditions for the Plant Village dataset which holds a drawback of the model.

Sapna Nigam et al. (2020) suggested a convolutional neural network model for wheat rust disease identification that will aid in the development of the agricultural system which can properly determine whether a plant is infected or healthy. The model developed has been demonstrated by using 2,000 images of the dataset to identify the yellow rust infected leaves in an unseen leaf image. The results show that deep learning has the potential to identify plant diseases with 97.37% accuracy. This technology enables the user to detect diseases faster, allowing them to take acceptable precautionary steps and save crops.

Adebayo and Adebayo (2021) developed a pre-trained Efficient DenseNet model utilizing an extra transition layer in DenseNet-201 to classify potato leaf diseases efficiently. The proposed algorithm is a novel technique to address and report the successful implementation for the detection and classification of four diseases in potato leaves. The algorithm's performance was evaluated on the testing set and gave an accuracy of 97.2%. However, the dataset they used has a class imbalance challenge, so they coped with this issue using the reweighted class balance cross-entropy loss function.

Lee and Choi (2021) collected 5,000 images from different farms in the Korean Peninsula. They aimed to detect and identify tomato disease and pest classes with location in the image by seeking a better deep learning architecture. They considered Faster R-CNN, SSD, and R-FCN with various feature extractors (VGGnet, ResNet). Moreover, they suggested data augmentation to decrease false positives in training and to improve accuracy. The system was able to identify gray mold, pests, leaf mold, leaf miner, canker, powdery mildew, low temperature, nutritional deficiency, and whitefly. However, this disease recognition algorithm is best for tomato plants only if it detects for other plants, especially for Onion crops.

Tadesse and Abate (2020) developed an accurate enset plant disease identification model using a convolutional neural network for the three most critical diseases of enset crops. They collected four classes of images, including healthy enset. Experimental results show that these proposed models can effectively detect and classify those four classes of enset plant diseases with the best classification accuracy of 99.53%.

Shakir and Shah (2021) proposed an automatic flower disease identification method using image processing. The developed system uses computer vision that helps farmers control and respond effectively to the diseases they encounter. The identification of flower diseases can be done

automatically using an image analysis technique. They used eight classes of disease data sets: rose-aphid, rose-japanese-beetle, rose-rosette-disease, rose-goldenrod-soldier, rose-mossay, rose-gel-wasp, rose-normal, rose-rust, and rose-soldier-beetles, which were identified with varying accuracies, resulting in an overall performance of 83.3%.

Afonnikov et al. (2021) suggest convolutional neural network algorithm with the Efficient Net architecture for recognition five fungal diseases of wheat shoots. The training and testing of the network were conducted on the basis of a specially formed and labelled sample of 2414 images, which is freely available. Of the three examined learning strategies, the best accuracy was provided by the method using augmentation and transfer of image styles (the accuracy was 0.942). The interface to the method is implemented on the basis of the telegram bot, which provides access to the program via the Internet both from a mobile device and using a desktop PC. According to this study, the studies it designs acceptable model in this domain area but their gap in implementing on Onion domain area.

Alizadeh and Mohammadi (2022) presented a methodology to detect Onion disease (purple blotch) by using CNN classification. The feasibility of their approach has been demonstrated by using a dataset of 1000 images for healthy and infected Onion crops. Two different types of experiments were taken using an improved InceptionV3 for classification with different batch sizes. The proposed method recognizes the disease with a classification accuracy of 85.47%. However; existing dataset crops with minor symptoms because of this the authors can't get the expect accuracy result and it identify the disease at certain stage of growth. Hence; in this study, we prepare a dataset as for those three diseases as well as healthy dataset to design and develop a model that identify and classify three critical diseases of Onion crops from symptoms that appear on Onion crop leave and bulb images and also recommend solution.

Summary of Related Works

Table 2.1 Summary of Related Works

Reference	Title	Method	Dataset	limitation	future scope
Kamilaris, et al. (2018)	Detection of disease in tomato plants using Deep Learning Techniques	By using deep detector: Faster RCNN with "deep feature extractors" such as ResNet50.	Collected a comprehensive 1090 real-time tomato leaf images infected by four diseases Visited different tomato farms.	-This author use less number of annotated samples, so certain classes with large difference in patterns appear to be confused with similar diseases and cause false positives.	This author suggest to future author the work can be extended by training the model with other types of tomato diseases
Ferentinos, K. (2018).	Image-Based Tomato Leaves Diseases Detection Using Deep Learning	deep-CNN	Using a public dataset of 9000 images of infected and healthy Tomato leaves collected under controlled conditions	This author taken image under controlled conditions. This holds to drawback of the model.	-This author didn't suggest future direction for next author
Mohanty, et al. (2016).	Wheat rust disease identification using deep learning	CNN	-Uses 2,000 images of dataset to identify the yellow rust infected leaves in an unseen leaf image.	-This technology enables the user to detect diseases faster allowing them to take acceptable precautionary steps and save crops.	-Author extended to the development of a mobile application based on wheat disease identification for next author.

Alizadeh, M. R., & Mohammadi, M. (2022).	Image-based Onion Disease (Purple Blotch) Detection using Deep CNN	CNN	By using a dataset of 1000 images for healthy and infected Onion crops.	Under this research existing dataset crops with minor symptoms because of this the authors can't get the expect accuracy result and it identify the disease at certain stage of growth.	-To identify different ways at different stages of growth -To collect a huge number of high-quality images of diverse types of diseases and pests that attack the Onion crops.
Adebayo, et al. (2021).	A novel framework for potato leaf disease detection using an efficient deep learning model	DenseNet-201	potato leaves images have been collected from an openly accessible dataset namely Plant Village Dataset that contains a total of 2,152 images for Potato Leaves diseases	The author cope with this issue using the reweighted class balance cross-entropy loss function	They suggest after some modifications in its architecture for various fields. -to reduce its training time and by minimum number of images can give significant results.
Zhang, C., & Wang, Y. (2021).	A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition	Faster RCNN, SSD, R-FCn	Dataset contains 5000 images collected from different farms in Korean Peninsula.	This disease recognition algorithm is best but it model for tomato plant only if it detect for other plant especially for Onion crops.	The future work focused on improving the current results, and a promising application will be to extend the idea of diseases and pest recognition to other crops.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Introduction

This chapter describes the research methodology employed for the detection and classification of Onion disease. The primary emphasis is on determining a suitable research design, methods, tools and techniques aligned with the specific objectives of the research. The discussion encompasses various aspects, including research design, data collection methods. Additionally, a detailed description of the dataset, its objectives, and values is provided. The chapter also explains dataset pre-processing techniques, optimization methods, and the performance evaluation metrics adopted in the research. The tools employed throughout the research process are clearly outlined in the chapter. Overall, this comprehensive exploration of the research methodology ensures the clarity of the research process flow and provides a strong foundation for understanding the strategies and techniques employed in the study

3.2. Research Design

Choosing an appropriate research design is essential for effectively addressing research objectives. In this study, a design science has been opted to be employed. The Design Science Research Process (DSRP) consists of several phases including problem identification and motivation, Literature Review, identification of the Objectives, Methods, Tools and Techniques, Model development, and Communication (Peffers et al., 2020). Design science aims to create effective artifacts and fulfil stakeholders' expectations (Hevner et al., 2008). The rationale behind selecting design science lies in its suitability for crafting artifacts in deep learning and machine learning. These artifacts encompass constructs, models, methods, and real-life prototypical products to address specific issues in image recognition, identification, detection, and classification domains. Another reason for adopting the design science research methodology is its application in addressing unresolved problems in innovative ways and finding more efficient solutions to previously solved ones (Hevner et al., 2008). Therefore, this research adopts an exploratory, experimental design science approach. It begins by exploring different deep learning interventions to identify and classify optimal solutions systematically, using a systematic literature review approach.

3.3. The proposed Architecture

The proposed architecture in this study is presented in figure 3.1. It deals with Onion disease classification using image processing. It involves processes like image preprocessing, image segmentation and feature extraction and finally classification of images. Image preprocessing includes a sequence of steps such as, resizing the image to a (224×224) size. To improve contrast in image we applied Histogram Equalization by effectively spreading out the most frequent intensity values. Then noise removal using median filtering. After preprocessing, image data augmentation applied on the dataset to increase the train dataset and to overcome over fitting problem. In the second step image segmentation applied to find region of interest and to separate the foreground and background of image. The third step is feature extraction; it is performed using CNN. CNN Feature extraction model has a convolutional layer, pooling layer, and fully connected layer, to extract relevant features. Some of the main reasons that CNN will be used in this thesis are:

- A lot of previously conducted researches has shown that CNN is better than other classification algorithm and it is state of the art for computer vision applications.
- CNNs are designed by emulating human's understanding of vision, so, for image related tasks, CNN is better than other deep learning models.
- Most of classical machine learning approaches require explicit extraction of the features that are used to study from the image before the classification and prediction.
- Most neural network algorithms only accept vectors (1D) and most of the real-world images are tensors (3 dimensional) so there is a need to flatten the actual image (input) to the 1D vector which is very difficult and computationally expensive but, CNN accepts 2 and 3- dimensional images.
- CNN can capture temporal and spatial dependencies with the help of relevant kernels.

The testing phase follows similar procedures with the training phase. The proposed approach is shown in the below Figure 3.1.

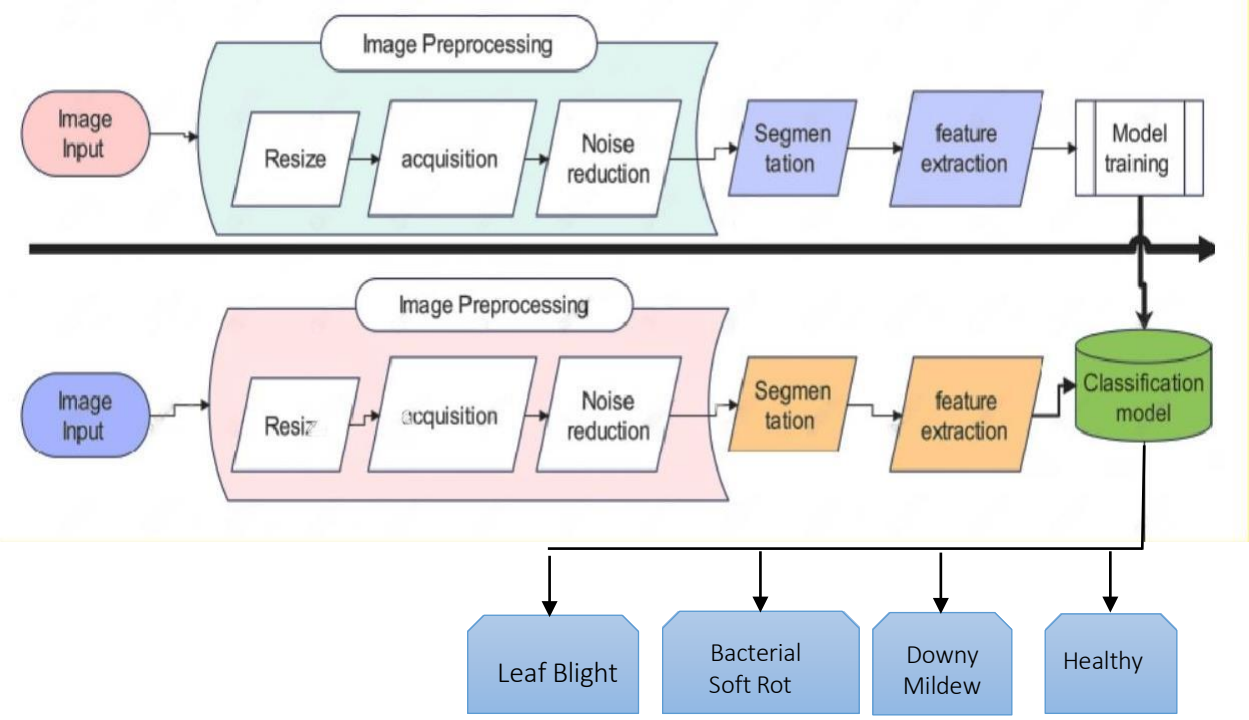


Figure 3.10. Proposed Model Architecture

3.4. Data collection and Preprocessing

The most important thing when we want to use a neural network or deep learning algorithms during research is getting the data that are used to train the neural network model. In this thesis, Onion image data is used as the main input to the model. However, there is no publicly available dataset that contains thousands of Onion images that captured in Ethiopian context that we can download and use for the training of the model. So, only images captured from different Onion crop farms that contains four different classes where three classes are represented infected Onion crop diseases such as Downy Mildew**,** Onion Leaf Blight and Bacterial Soft Rot diseases and the remaining class represents healthy Onion crop images. The images of Onion crops are collected from southern regions of Ethiopia with the help of farmers and agriculture experts. All diseased and healthy images are captured from Onion farms. The images are captured by using a cell phone camera with normal condition and also all images are checked by domain area experts (plant science expert) of Awash Malkasa Agricultural Research Institutes Center, East Shewa Zone of Oromia Region, Ethiopia.

Preprocessing used to be the major important task when we want to work with image processing. It is a transformation of the raw data before feed into the neural network or deep learning algorithm. In CNNs, manual feature extraction is unnecessary because the algorithm can learn features directly from

raw pixel data. However, the images in the dataset often vary in size, the optimal image size for the CNN was determined through careful testing of various dimensions, aiming to enhance the performance of onion disease identification.

3.5. Image resizing

The collected images are varied in size. Image resizing is the process of converting the size of the original image into a constant image size. We have selected 224 x 224 image size because of that we compare different image sizes and we have achieved a better result in 224x224, which was initially designed to work on ImageNet dataset of size 224 x 224. This standard image size is used throughout the whole image processing.

3.6. Image Normalization

Image normalization is a crucial technique used in image processing to enhance the image quality by adjusting the pixel values to a standard scale (Sun et al., 2023). Its main purpose is to reduce image variations due to various factors like lighting conditions and noise, which can adversely affect the image's quality. In technical terms, the process of image normalization involves rescaling the pixel values of an image using a mathematical formula to conform to a specific range, typically between 0 and 1 or -1 and 1(Lee et al., 2021). This technique can improve the quality of images significantly and enhance the accuracy of various image analysis tasks. In this study, the image normalization process was performed by multiplying each pixel value by 1/255.

3.7. Image Augmentation

Most of the previously studied CNN image classification models are used very large datasets that are collected from different places and some of the datasets are generated from that of the collected dataset using data augmentation technique. Data augmentation is a technique that can help you improve the performance and generalization of your CNN models. It involves creating new variations of your existing training data by applying transformations such as cropping, flipping, rotating, scaling, or changing the color, brightness, or contrast. Data augmentation can increase the diversity and size of your training data, reduce over fitting, and make your model more robust to different inputs. Additionally, it is beneficial in situations where the initial dataset is small as it can help prevent overfitting and improve the generalizability of the model as well as address issues with small dataset (Perez &Wang, 2017). After applying image augmentation techniques, the total size of the dataset has expanded to 12,438 instances. The distribution of instances across each class is as follows: Downy

Mildew has 2,631 instances, Onion Leaf Blight has 2,575 instances, Bacterial Soft Rot has 2,923 instances, and Healthy plants account for 4,309 instances. This augmentation significantly enhances the dataset, improving the training and evaluation of machine learning models designed to detect these specific plant diseases. The parameter used in the augmentation techniques of this study are presented in the following table (table-3.2) Since the original Onion diseases image collected for this thesis is not sufficient to train the model very well, we used the augmentation technique for increasing the dataset.

Table 3.2 Parameters of augmentation techniques.

No	Operation	Values	Properties
1	rotation _range	90	Randomly rotate images with random angles between 0 and 90 degrees
2	width_shift_range	0.2	Shift the image along X-axis by 20%
3	height_shift_range	0.2	Shift the image along Y-axis by 20%
4	Shear_range	0.2	Shear the image by 20%
5	zoom_range	0.2	Zoom In and Zoom Out by 20%
6	horizontal flipping	True	Enable horizontal_flip
7	vertical_flip	False	Disable vertical flipping

3.8. Image Segmentation

The goal of segmentation process is to partition an image into regions that are homogeneous with respect to one or more characteristics or features such as luminance, color, and texture. Accurate segmentation of medical images is very important for the analysis and diagnosis of abnormalities in different parts of the body. There are many methods for image segmentation. The most common images segmentation techniques are region segmentation techniques that look for the regions satisfying a given homogeneity criterion, and edge-based segmentation techniques that look for edges between regions with different characteristics. Thresholding is region segmentation method a threshold is selected, and an image is divided into groups of pixels having values less than the threshold and groups of pixels with values greater or equal to the threshold. There are several thresholding methods: global methods based on gray-level histograms, global methods based on local properties, local threshold selection,

and dynamic thresholding. For this work, we have proposed Adaptive threshold, Edge-Based Segmentation and Otsu segmentation method.

3.9. Data Splitting

Data splitting is a process utilized in machine learning and deep learning to divide a dataset into several subsets for distinct purposes such as training, validation, and testing (Azimi et al., 2021; Ramcharan et al., 2017). Its primary objective is to assess a model's performance on unseen data and prevent overfitting, where the model becomes too specific to the training data and performs poorly on new data (de Luna et al., 2019). Typically, the dataset is divided into two or three subsets. The training subset is used to train the model, the validation subset is used to fine tune model hyper-parameters and select the best model, and the testing subset is employed to evaluate the final model's performance (Sun et al., 2017). The dataset, consisting of 12,438 images of both Onion leaf disease and healthy images after augmentation, was divided into training (80%), testing (10%), and validation (10%) sets.

3.10. Feature Extraction

Feature extraction is a technique by which unique features of images are extracted, distinguishing the basic characteristics or attributes of an image. Feeding a large number of digital images to a classifier is a time-consuming method for classification. To find a way to minimize the amount of data, quantitative information is extracted from the objects to be analyzed in the image (Zhang et al., 2020; Smith & Johnson, 2019). It is the most important step in image processing. Feature extraction techniques are classified as low-level feature extraction and high-level feature extraction. Low-level feature extractions are based on finding points, lines, edges, etc., while high-level feature extraction methods use low-level features to provide more significant information for further processing of image analysis. Mostly, high-level feature extraction methods use artificial neural networks (ANN) to extract features in multiple layers (Doe & Lee, 2021). In recent years, deep convolutional networks (CNNs) have become very popular in feature learning, capable of extracting deep representations of training data and achieving impressive performance within a short time in image classification, especially in medical image detection and classification systems (Chen et al., 2022). For our research, we propose to develop a CNN model for feature extraction. The CNN architecture has several layers, including convolution layers, pooling layers, rectified linear unit layers, and fully connected layers.

3.11. Classification

It involves assigning label to the image, based on the feature vector. After extracting features, the role of classification is to construct a detention model using different classification algorithms. The extracted features are used for the recognition and classification of Onion leaf disease, Downy Mildew**,** Onion Leaf Blight and Bacterial Soft Rot and Healthy. In training, the feature extracted is applied as an input to the classifier and the corresponding labels form the target outputs. When selecting the optimal classifier for a given use case, factors like accuracy, training duration, parameter and feature count, and quantity are important to take into account. When compared to a more complicated model, even the simplest methods can occasionally perform better.

3.11.1. Classification using SoftMax classifier

We have added a dense layer with the Softmax activation function of the CNN for classifying the input images based on their features extracted in the final fully connected layer of the proposed CNN because it is an important building block in deep learning networks and the most popular choice among deep learning practitioners. Softmax classifier is suitable for multiclass classification, which outputs the probability for each of the classes. The Softmax first learns important features of each category in the training phase from the training set. Then, it classifies the new testing dataset accordingly. Choosing proper loss and optimization function increases the power of the classification of the softmax classifier.

3.11.2. Classification using VGG model

The VGG-16 model is a convolutional neural network (CNN) architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its depth, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. VGG-16 is renowned for its simplicity and effectiveness, as well as its ability to achieve strong performance on various computer vision tasks, including image classification and object recognition (Simonyan & Zisserman, 2014). The model's architecture features a stack of convolutional layers followed by max-pooling layers, with progressively increasing depth. This design enables the model to learn intricate hierarchical representations of visual features, leading to robust and accurate predictions. Despite its simplicity compared to more recent architectures, VGG-16 remains a popular choice for many deep learning applications due to its versatility and excellent performance (He et al., 2016). The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in computer vision where teams tackle tasks including object localization and image classification. VGG-16, proposed by Karen

Simonyan and Andrew Zisserman in 2014, achieved top ranks in both tasks, detecting objects from 200 classes and classifying images into 1000 categories (Russakovsky et al., 2015).

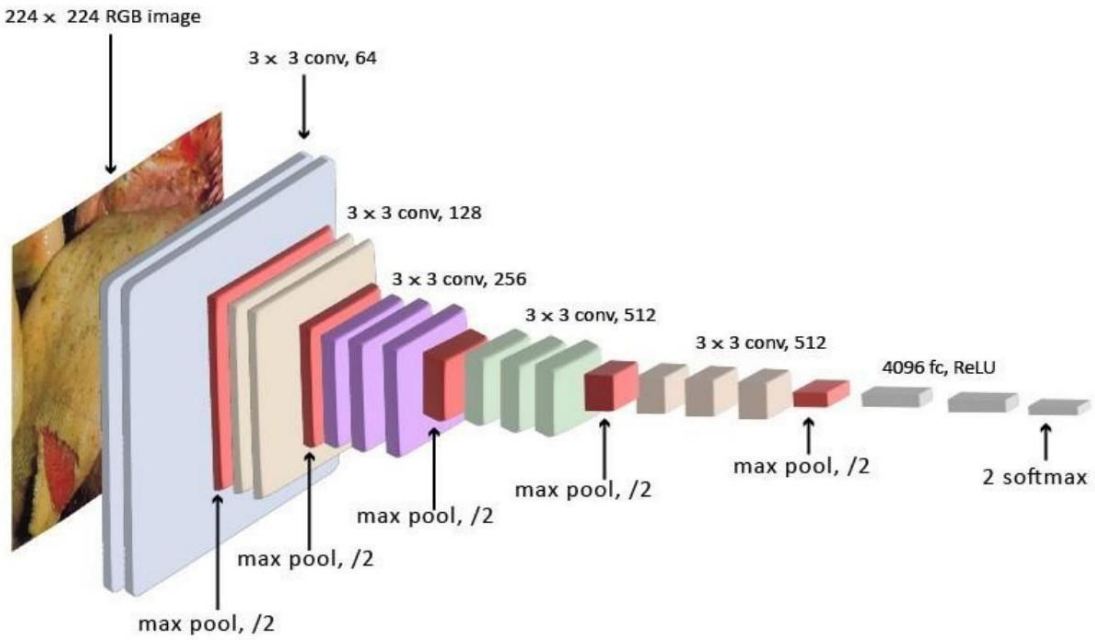


Figure 3.11. Architecture of VGG16

In the provided VGG16 architecture, there are a total of 16 layers, including convolutional layers, pooling layers, and fully connected layers. There are 13 convolutional layers, each followed by a rectified linear unit (ReLU) activation function. These layers perform feature extraction by convolving input images with learnable filters to detect patterns and features at different scales and complexities. There are 5 max-pooling layers, each applied after every two consecutive convolutional layers. Max-pooling layers down sample the feature maps, reducing their spatial dimensions while retaining the most relevant information. There is 1 fully connected layer with 4096 units and ReLU activation. This layer, commonly referred to as a dense layer, learns high-level features from the flattened feature maps generated 47 by the preceding convolutional and pooling layers. This layer allows the network to learn complex patterns and relationships in the data, leading to better classification performance. The input image size is 224 × 224 x 3, which represents the input image's RGB channels. The first two convolutional layers consist of 3 x 3 convolutions with 64 filters each. The convolutional layers apply filters to extract features from the input image. Following each pair of convolutional layers, a max-pooling layer with a 2 x 2 window size reduces the spatial dimensions of the feature maps. The next two convolutional layers have 3 x 3 convolutions with 128 filters each. These layers further extract

higher-level features from the input. Similar to the previous layers, max-pooling with a 2 x 2 window size is applied after the convolutional layers. Three consecutive convolutional layers with 3 x 3 convolutions and 256 filters each follow the pattern of feature extraction. Max pooling with a 2 x 2 window size is performed after the set of convolutional layers. Another set of three convolutional layers with 3 x 3 convolutions and 512 filters each for deeper feature learning. Max pooling with a 2 x 2 window size is applied to reduce spatial dimensions. Three more convolutional layers with 3 x 3 convolutions and 512 filters each for further feature extraction. Max pooling with a 2 x 2 window size is performed to downsample the feature maps. The Global Average Pooling layer reduces the spatial dimensions to 1 x 1, aggregating features across the entire feature map. A fully connected layer with 4096 neurons and ReLU activation function follows the Global Average Pooling layer.The final layer applies the Softmax activation function to produce class probabilities for the two classes: “infected” and "Healthy". These layers collectively form the architecture of the VGG16 model, enabling it to extract hierarchical features from input images and make predictions based on the learned features.

3.12. Regularization techniques

Regularization is a technique used to prevent over-fitting in neural networks, including convolutional neural networks (CNNs). Over-fitting occurs when a model fits the training data too well, but does not generalize well to unseen data. This can lead to poor performance on the test set or in real-world scenarios. In order to penalize specific model traits that can lead to over-fitting, regularization entails adding a penalty term to the loss function of the model during training. Regularization attempts to find the right balance between preventing over-fitting and achieving good fit to the training set of data. As a result, we apply the two most used regularization techniques dropout and early stopping in this study.

3.12.1. Dropout

Dropout is the most frequently used regularization approach. It basically implies that throughout the training, certain neurons are switched off at random. It implies that in a forward pass, they are briefly prevented from affecting or activating the downward neuron, and none of the weight changes are performed in the backward pass. As a result, if neurons are pulled out of the network at random during training, the remaining neurons step in to generate predictions for the missing neurons. As a result, the network learns distinct internal representations, making it less sensitive to the weight of the neurons. This type of network is more general and less prone to over fitting.

3.12.2. Early stopping

This is a type of cross-validation approach where a portion of the training set is used as the validation set and the performance of the model is measured against it. Therefore, if the model's performance here on the validation set worsens, the model training process will be stopped immediately. This is done by tuning the neural network to the training data and then evaluating the model on the new dataset and validation set after each iteration. So, if somehow the performance of the validation set decreases or stays the same for a particular number of iterations, the model training process will be interrupted.

3.13. Finding optimal hyper-parameters

All three layers of the neural network-based model must be tuned with appropriate hyper parameters for them to be accurate. The difference in accuracy between the models is due to differences in tuning hyper parameters such as number of iterations (epochs), activation functions, 34 batch sizes, learning rates, optimization algorithms. number of neurons in each layer, batch size in the same neural network, and layer number setting; Having too few layers in a model can cause under fitting, while having too many layers can cause over fitting. Let's look at the issues that need to be considered when tuning hyper parameters. Increasing the learning rate causes the model to learn more quickly, but there's a risk that it will overshoot the minimum loss function and arrive rather near to it. Conversely, lowering the learning rate increases the probability of finding the minimal loss function. More time and memory space are needed to complete more epochs in order to obtain a lower learning rate. Regarding batch size, lower batch sizes expedite the learning process but raise the accuracy variance of the validation dataset. However, using a larger batch size results in less volatility in the accuracy of the validation dataset but slows down the learning process. The training dataset needs to undergo multiple passes or epochs in order to sufficiently train the model. However, excessive epochs can result in over fitting. This indicated that, optimizing hyper parameters is a challenging task that requires a thorough comprehension and intuition about the underlying model process. Furthermore, it can be a lengthy and error-prone procedure, frequently taking numerous iterations of trial and error before identifying an appropriate combination of hyper parameters. To easy hyper parameter optimization, there are two widely used techniques: grid search and random search. Random search is often considered more efficient than grid search because it has the ability to sample important dimensions within the search space; however, there is no assurance that it will yield optimal outcomes. But, with small search space grid search gives best result since it exhaustively trains and test all the given network parameter combinations. For this study, we used grid search optimizer algorithm for hyper parameter tuning.

3.13.1. Epochs

During training, it is the number of iterations that the full training data is fed into the network. It is the number of iterations required to pass the whole training dataset through the model or network, both forward and backward. The model was trained using different epochs for each architecture. Determine the appropriate number of epochs for the model's peak performance. This was done to get the best feasible optimal epoch.

3.13.2. Batch size

It represents the total number of training inputs used to train the model. To update the parameters, a single set is broken into tiny subsets known as mini-batches. The 32-batch size was utilized to train this model. Small batch sizes are optimal for great performance, but they incur large computational costs.

3.13.3. Learning rate

The CNN model was trained using back-propagation, and the learning rate is utilized to update weights. It specifies how much weight should be updated during backpropagation. Choosing the appropriate learning rate during the experiment was the most difficult aspect. Learning rates with lower values take longer to train than those with higher values. When given a lower value, however, the model outperforms one with a faster learning rate. The learning rate employed in the experiment was 0.001 and 0.01

3.14. Optimization Techniques

Training deep learning is challenging and complex due to the intrinsic complexity of the algorithm since optimization techniques or algorithms are required to be used for increasing or optimizing the performance of the model because the model performance is directly affected by the optimization algorithms. However, the basic principles or functions of the optimization algorithm are basically targeted at the training dataset and loss function because the main aim of this optimization algorithm is targeted at reducing training errors. Recently, different deep learning optimization algorithms are deployed, for example, ADAM optimizer, AdaGrad, RMSProp, Gradient Descent and etc. In the proposed research work, state-of-the-art optimization algorithms will be used. In the training setup phase this work, Stochastic Gradient Descent (SGD) (Tian et al., 2023) and a categorical cross-entropy loss (Rajaraman et al., 2021) were used as the optimizer and the loss, respectively. To prevent overfitting, all the pre-trained models developed in this study were carefully trained with the training

data. To ensure this, a validation dataset was used during each epoch of the training phase to assess the model's performance. Specifically, overfitting was deemed absent when the validation accuracy exhibited stability or improvement while the training loss demonstrated stability or decrease. Thus, we can confidently assert that the identified models were adequately trained without overfitting.

3.15. Research Tools

In the course of this research, the execution of fundamental tasks such as image processing, identification, and classification played a crucial role. To accomplish these activities, a comprehensive array of software and open-source libraries was employed, including Jupyter Notebook, TensorFlow, Keras, Pandas, sci-kit-learn, and various other valuable tools for plotting and measuring. The selection of these tools was intricately linked to the specific nature of the identification and classification tasks, necessitating a careful examination to identify the most suitable tools capable of addressing the challenges posed by these tasks. The use of Jupyter Notebook facilitated a dynamic and interactive environment for data analysis, code development, and visualization. TensorFlow and Keras, being prominent deep learning frameworks, provided robust support for building, training, and evaluating the pretrained and the designed novel models. Pandas, a powerful data manipulation library, played a pivotal role in handling and analyzing the dataset, ensuring a seamless integration of data into the research pipeline. Sci-kit-learn, a versatile machine learning library, contributed to the implementation of various deep learning algorithms/models and evaluation metrics.

CHAPTER FOUR

EXPERIMENTAL RESULT AND DISCUSSION

4.1. Chapter Overview

Accurate and effective detection and classification of Onion disease are vital for their enhanced crop production. However, the exiting detection and classification process is time-consuming, tedious, and demands the expertise of specialists. Experts traditionally rely on traditional and experience-based methods for detection and classification of Onion disease. This research aims to develop deep learning model through transfer learning for the detection and classification of Onion disease. In this work, three experiments have been done with the use of pre-trained deep learning models, specifically VGG19, Inception-V3, Dense Net, ResNet 50 and CNN from scratch. The comparative model analysis presented in this study makes a substantial contribution to scholarly discussions on crop research, particularly in the context of production enhancement.

4.2. Model Selection

CNN, a deep learning technique, was chosen based on literature in computer vision, specifically image classification. CNNs offer a promising approach for adaptive image processing. The algorithm is used for feature extraction, classification, training, testing, and model accuracy evaluation. CNNs can handle data without requiring additional pre-processing or feature extraction steps. Furthermore, feature extraction and categorization are integrated into a unified framework. CNN were selected for this study due to their ability to automatically learn hierarchical features from image data, making them more automated than classical machine learning algorithms. Transfer learning involves applying previously trained models to a specific case. In this study, we used models from the "ImageNet" contests and shared by the Keras library, including VGG16, Densenet201, and Inception V3. To determine the most suitable CNN architecture for our task, we compared the accuracy of three different models: VGG16, Densenet201, Res Net 50 and Inception V3. Each of these models has its own unique strengths and weaknesses. Inception is known for its ability to capture multi-scale features, VGG- is a classic CNN architecture that has been widely used for image classification tasks and DenseNet is known for its dense connectivity, which allows for efficient feature reuse. By comparing the accuracy of these different models, we were able to select the best model for our Onion disease detection task.

4.3. Implementation Environment

We conducted the experiment on an HP 224 G7 Elitebook PC model laptop computer. Its processor is an Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz ≈ 2.0GHz, with four cores and eight logical processors. It has 4GB of RAM, an L2 cache size of 1024 MB, and an L3 cache size of 8192 MB. The computer runs on BIOS F.43, graphics: UHD graphics 620 and chip type: Intel ® UHD Graphics Family, on a Windows 10 Pro 64bit operating system. For developing the model, we used Anaconda platform. The Anaconda Distribution is a platform that's free and open-source, supporting the Python and R programming languages. It's compatible with many operating systems like Windows, Mac OS, and Linux. With more than 1500 data science packages for both Python and R, Anaconda is perfect for developing deep and machine learning models. Additionally, it provides the option to install Python with various IDEs like Spyder, Jupyter Notebook, and Anaconda Prompt. Python is presently the most popular language. Therefore, the researcher utilized Jupiter Notebook in Anaconda distribution version 2.4.0, released in October 2022, which features the latest Python version (3.10) and all the necessary libraries for the research. In this study, the main deep learning framework utilized is Keras with Tensor Flow backend.

4.4. Experimental Setup

The experimental procedure of this work is based on the following steps: Firstly, all the necessary parameters and hyperparameters are configured before the training process is going on. Secondly, the training of the proposed model is conducted to extract the most important feature which is the process of running a training dataset through the model from the input layer to the output layer. During the training process, if the validation accuracy results are not good, then the process back propagated strategy is applied to add the error rate to the model in the input layer to modify the weights. Thirdly, the model classifies the type of disease in to predefined category. Fourthly, the model is evaluated using different evaluation metrics like classification accuracy with loss, training accuracy with loss, validation accuracy with loss. Finally, the developed model is tested using an unseen test dataset.

4.4.1. Hyper Parameters configuration

Hyperparameters are values that are decided outside of the model training process that determine the network structure and behavior of a deep learning algorithm which is very specific to each model such as learning rate, epochs, optimization algorithm, batch size, and loss function.

Table 4.3. Configuration of hyperparameters for the proposed work

Parameters	Configuration values
Learning rate	0.1-0.0001
Batch size	32,64,128
Epochs	5,30,50, 100
Optimization algorithm	Adam, SGD, RMSprop, Adamax
Activation function	ReLU and SoftMax
Loss Function	Categorical Cross-Entropy

4.4.2. Input Layer Setup

During the implementation process of the model, before starting the convolution operation, the input layer is first configured properly to accept pixels of the image dataset in the form of arrays, since the input images are color, the shape of the input layer is configured as $224 \times 224 \times (N = 3)$ where N defined as a total number of channels. This layer only passes the input to the first convolution layer without any computation. Hence, there are no learnable features in this layer and the number of parameters in this layer is 0.

4.4.3. Feature Extraction Layers Setup

The proposed model involves four convolution layers with four filters (16,32,64,32 respectively), four max-pooling operations with rectified linear unit function (ReLU) to add nonlinearity during the training of the network having padding of 0 stride of 1, two fully connected layers, and a final layer as the classification layer that has 4 output nodes and two dropouts are included after the fourth max-pooling layers to prevent the problem of overfitting.

4.4.3.1. Convolution layer Setup

The proposed model is a sequential model that has a series of layers to convert the standard size of images into a feature set for further processing. There are four convolutional layers, each four convolution layers are connected to the four max-pooling layers to reduce the dimension and speed up calculations of the features in the Onion dataset. The first convolutional layer of the model takes a size of 224×224×3 input image using 16 filters with a size of 3×3×16, a stride of 1 pixel, and a padding value of 0. This process is followed by a rectified linear unit (ReLU) activation function to replace the entire negative pixel value with 0 to introduce nonlinearity to the networks. The second convolutional layer takes the output of the first convolutional layer as input and filters it by using 32 kernels with a size of 3×3×32. The third convolutional layer takes the output of the second convolutional layer as input and filters it by using 64 kernels with a size of 3×3×64. The fourth convolutional layer takes the

output of the third convolutional layer as input and filters it by using 32 kernels with a size of 3×3×32. All the convolutional layers of the proposed model use ReLU nonlinearity as activation functions. ReLU is chosen because it is faster than other non-linearities such as tanh to train deep CNNs with gradient descent.

4.4.3.2. Pooling Layer

Setup There are four max-pooling layers with a kernel size of 2 × 2 × 3. All four max-pooling layers reduce the output of all four convolutional layers with a filter size of 2×2 with a stride value of 1. All pooling layers have no learnable features and they only perform down-sampling operation along the spatial dimension of the input volume, hence the number of parameters in these layers is 0.

4.4.3.3. Flattening Layer

Setup Flattening is the process of converting all the results of a 2-dimensional array from a pooled feature map into a single long continuous linear vector to create fully connected layers. In other words, it is the process of putting all the pixel data in one line and making connections with the final layer.

4.5. Classification Layers Setup

In the proposed model, there are a total of three fully connected layers including the output layer. The main function of these layers is to classify the input images based on the most important extracted features which are done by convolution layers. The first fully connected layer accepts the last output of the convolution layer four and pooling layer four in the form of a flattened layer before feature maps are fed into the second fully connected layer.

4.5.1. Fully Connected Layer Setup

In the proposed model there are a total of three fully connected layers including the output layer. The importance of these layers is to classify and predict class probabilities of the input extracted features by the convolution layers. The first fully connected layer accepts the output of the fourth max-pooling layer in the form of a flattening layer. It contains a total of 512 neurons and is followed by a ReLU activation function and a dropout operation with a value of 0.2. The second fully connected layer contains 100 neurons that are connected to all the neurons of the first fully connected layer as well as the last fully connected layer and are followed by ReLU and dropout operations as above. In each layer, the dot product is applied to multiply the value's most important features by weights, and biases are added then after the results are passed forward to the third fully connected layer in which every neuron

represents a classification label to produce a single value. The output of each fully connected layer is computed as follows in the following equation.

$$\text{Class output} = w * x + b$$

Where w is weighted x is the values of the features extracted by the convolutional layer and b is bias.

For example, the first fully connected layer performs a dot product to compute the class of output. It takes the output of the flattening layer (73728) as input and multiplies by weights (512) and the bias 512 is added. The third fully connected layer is the last layer of the proposed model which contains four neurons with a Softmax activation function that represents the number of classes in the proposed model. The output of this layer is then transferred to the SoftMax function to determine the classification of the input healthy and diseased images. The SoftMax activation function is implemented, to sum up, the output values equal to 1.0 and limits individual outputs to values between 0 and 1. The configuration of the proposed classification phase.

4.6. Training of Proposed Model

Training of the model is the process of extracting the most important features or representative pixels from healthy and infected training Onion image datasets and teaching this pixel to the networks directly without handcraft, then after classifying these types of disease based on learning categories of the specific class. After several sets of training processes using different hyperparameters and parameters the proposed model has completed the detection and classification of the disease type. During the training process, the performance of the model is measured using the validation dataset. After measuring the performance of the trained model, the model with higher validation accuracy was used as a proposed classification model. Then the testing phase is performed by giving an unseen image dataset to the proposed trained model. The model finally predicts the class of the Onion crop dataset in the form of probabilistic value which is a value between 0 and 1 and the name of the disease with its image that belongs to one of the given classes during the training.

4.7. Experimentation Result

In this experimental test, we assessed the impact of using batch normalization, dropout, and early stop techniques on image classification. Two experiments were conducted, different optimizers and batch size techniques. We compared the results to determine the effectiveness of these techniques in improving classification accuracy. In each experiment, the researcher used image pre-processing, data

augmentation, image segmentation, feature extraction, and classification with a "Softmax" activation function and optimizer. The data was split into 80% training data 10% validation and 10% testing data, which is an optimal ratio for reasonably sized datasets. The model was trained for 5, 30, 50 and 100 epochs with a batch size of 32,64,128 while accuracy, precision, recall, and F1-score were used as evaluation metrics. Micro-average, macro-average, and weighted average were calculated for all of these metrics.

4.8. Experiment One

This experiment is conducted within the implementation of three commonly used techniques in deep learning - batch normalization, dropout, and early stop. the accuracy rate for classification is determined. This provides useful information on the performance of the model under specific conditions and helps researchers understand the impact of these techniques on the model's performance in classification.

4..8.1. ResNet50 model

During the training phase within batch normalization, dropout, and early stop. Our model Resnet50 obtains 98.55% training accuracy, 95.71% validation accuracy, and 97.36 testing accuracy. The validation and testing accuracy are less than the training accuracy, which indicates that the model unable to generalize well on unseen images. This Onion leaf disease classification accuracy is obtained when the model is trained with batch normalization, dropout, and early stopping mechanism to reduce overfitting.

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

Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.3303 - accuracy: 0.8746
Epoch 1: val_accuracy improved from -inf to 0.91648, saving model to best_m.h5
63/63 [==============================] - 38s 511ms/step - loss: 0.3303 - accuracy: 0.8746 - val_loss: 0.2108 - val_accuracy: 0.9165
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.0887 - accuracy: 0.9650
Epoch 2: val_accuracy improved from 0.91648 to 0.93454, saving model to best_m.h5
63/63 [==============================] - 31s 495ms/step - loss: 0.0887 - accuracy: 0.9650 - val_loss: 0.1502 - val_accuracy: 0.9345
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.0614 - accuracy: 0.9775
Epoch 3: val_accuracy improved from 0.93454 to 0.95260, saving model to best_m.h5
63/63 [==============================] - 30s 476ms/step - loss: 0.0614 - accuracy: 0.9775 - val_loss: 0.1372 - val_accuracy: 0.9526
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.0500 - accuracy: 0.9810
Epoch 4: val_accuracy did not improve from 0.95260
63/63 [==============================] - 31s 497ms/step - loss: 0.0500 - accuracy: 0.9810 - val_loss: 0.1826 - val_accuracy: 0.9413
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.0506 - accuracy: 0.9855
Epoch 5: val_accuracy improved from 0.95260 to 0.95711, saving model to best_m.h5
63/63 [==============================] - 30s 476ms/step - loss: 0.0506 - accuracy: 0.9855 - val_loss: 0.1259 - val_accuracy: 0.9571
```

Figure 4.12. Classification Accuracy of Resnet50 model

This plot displays the training and validation accuracy, training, and validation loss of Resnet50.

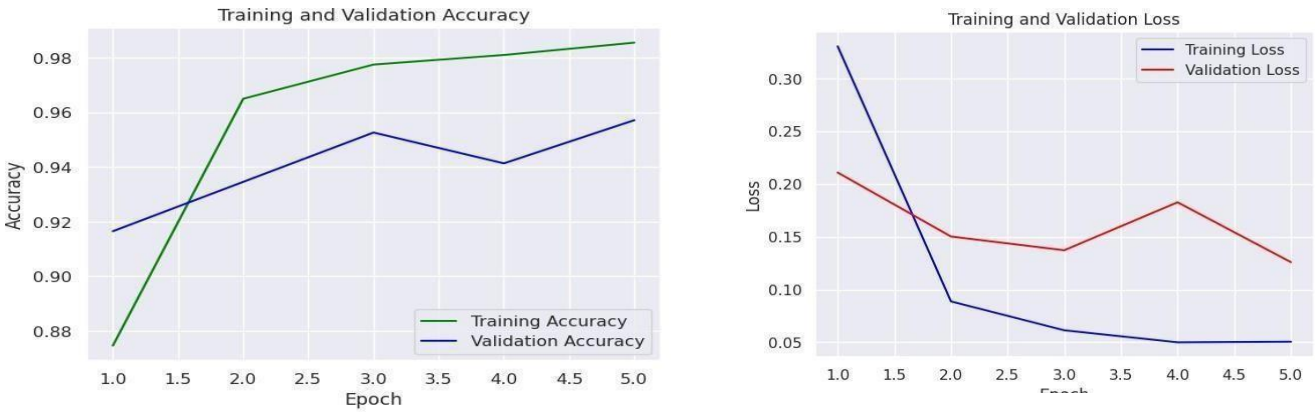


Figure 4.2 Training and validation accuracy and loss graph of the Resnet50 model

In Figure 4.2, the performance evaluation is shown. Up on examining the precision, recall, and F1-score, Onion Leaf Blight and Bacterial Soft Rot disease are the most serious were scored 0.95, while and Downy Mildew was scored 1.00. The remaining two classes achieved F1-scores 0.96 to 1.00. This indicates there is no fluctuation in classification between these classes. The model achieved an overall F1-score of 0.97. Learning rate, the result that is obtained from the experiment of the ResNet50 model is by using different learning rates with epoch 5, and batch size 64. different learning rate values on RGB images are illustrated in Table 4.2.

Table 4.4. Different learning rate values on RGB images

	Accuracy			Loss		
Learning Rate	**Training**	**Validation**	**Testing**	**Training**	**Validation**	**Testing**
0.1	97.21	92.06	95.15	0.074	0.222	0.145
0.01	98.18	93.69	96.36	0.043	0.175	0.123
0.001	98.55	95.71	97.37	0.050	0.125	0.102
0.0001	98.22	95.11	96.36	0.043	0.194	0.147

4.8.2. VGG-19 Model

In this experiment, we trained our VGG-19 model and achieved a training accuracy of 98.05%, a testing accuracy of 95%, and a validation accuracy of 93.43%. The difference between the validation and testing accuracies is relatively small, indicating good generalization of the model on new data. To

prevent overfitting, the model used batch normalization, a dropout rate of 0.0001, and early stopping with patience of 3. The model was trained over 5 epochs, with an average time of 493 seconds per epoch, and additional time was required to load the images. It takes more time to complete the training epochs with this model compared to Resnet50.

```
Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.4117 - accuracy: 0.8466
Epoch 1: val_accuracy improved from -inf to 0.91196, saving model to best_m.h5
63/63 [==============================] - 3620s 57s/step - loss: 0.4117 - accuracy: 0.8466 - val_loss: 0.2449 - val_accuracy: 0.9120
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.1238 - accuracy: 0.9547
Epoch 2: val_accuracy improved from 0.91196 to 0.92777, saving model to best_m.h5
63/63 [==============================] - 31s 497ms/step - loss: 0.1238 - accuracy: 0.9547 - val_loss: 0.2138 - val_accuracy: 0.9278
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.0884 - accuracy: 0.9680
Epoch 3: val_accuracy did not improve from 0.92777
63/63 [==============================] - 31s 493ms/step - loss: 0.0884 - accuracy: 0.9680 - val_loss: 0.2387 - val_accuracy: 0.9165
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.0660 - accuracy: 0.9750
Epoch 4: val_accuracy improved from 0.92777 to 0.93905, saving model to best_m.h5
63/63 [==============================] - 33s 530ms/step - loss: 0.0660 - accuracy: 0.9750 - val_loss: 0.2015 - val_accuracy: 0.9391
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.0547 - accuracy: 0.9805
Epoch 5: val_accuracy did not improve from 0.93905
63/63 [==============================] - 31s 497ms/step - loss: 0.0547 - accuracy: 0.9805 - val_loss: 0.2078 - val_accuracy: 0.9345
```

Figure 4.13 Classification Accuracy of VGG19

In this plot, the training accuracy and validation accuracy fluctuate slightly, indicating that overfitting is minimized. As shown in Figure 4.3 of the training and validation accuracy and loss graph of the VGG19 model.

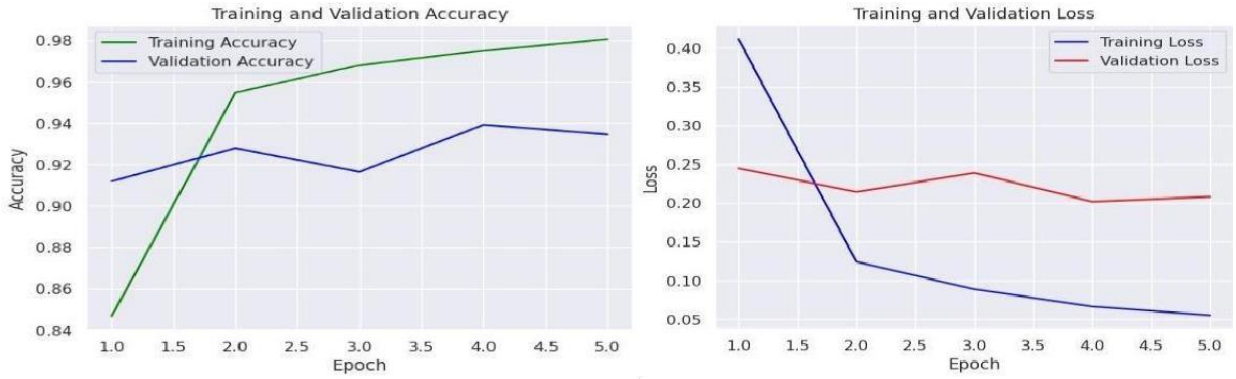


Figure 4.14 Training and validation accuracy and loss graph of the VGG19 model

4.8.3. CNN model during the training phase

Our CNN model's training phase consists of a sequence of convolution, activation function, pooling, and fully connected layers. The experiment involved classifying 4 classes of Onion disease. As depicted in Figure 4.4, the CNN model is trained for 5 epochs, yielding a training accuracy of 96.22%, a validation accuracy of 88.26%, and a Testing accuracy of 87%. The training and validation loss

decreased during the training process, eventually converging to 0.097 and 0.14, respectively. Dropout of 0.5 and early stop with patience of 3 were utilized to reduce overfitting and improve model performance. An average of 949 seconds per epoch was required for training, and additional time was also needed to load the image. This model requires high time to finish the training epochs in comparison to Resnet50 and VGG-19.

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(train_images, epochs=5, validation_data=val_images)
```

```
Epoch 1/5
32/32 [==============================] - 37s 989ms/step - loss: 0.2728 - accuracy: 0.9372 - val_loss: 0.9637 - val_accuracy: 0.7720
Epoch 2/5
32/32 [==============================] - 31s 955ms/step - loss: 0.2512 - accuracy: 0.9299 - val_loss: 0.5965 - val_accuracy: 0.8600
Epoch 3/5
32/32 [==============================] - 31s 949ms/step - loss: 0.2214 - accuracy: 0.9432 - val_loss: 0.4872 - val_accuracy: 0.8578
Epoch 4/5
32/32 [==============================] - 31s 960ms/step - loss: 0.1688 - accuracy: 0.9602 - val_loss: 0.5821 - val_accuracy: 0.8736
Epoch 5/5
32/32 [==============================] - 30s 946ms/step - loss: 0.1194 - accuracy: 0.9622 - val_loss: 0.4739 - val_accuracy: 0.8826
```

Figure 4.15. Classification Accuracy of CNN model

The plot displays training accuracy and validation accuracy, suggesting that the overfitting problem is addressed through the implementation of dropout and early stop. As shown in Figure 4.5 of the training and validation accuracy, and loss graph of CNN model.

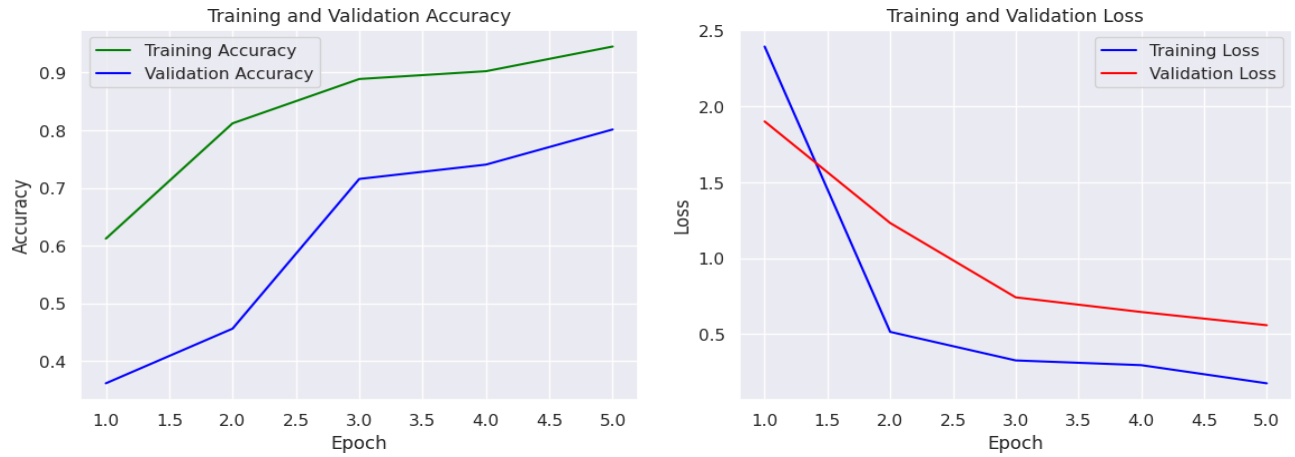


Figure 4.16 Training and Validation accuracy and loss graph of CNN model

4.8.4. DenseNet201 Model

As shown in Figure 4.6, the model's training and validation accuracy increased during the training process. Our DenseNet201 model achieved training accuracy of 95.49%, validation accuracy of 90.74%, testing accuracy of 92.10%, and performance evaluation accuracy of 92% at epoch 5.

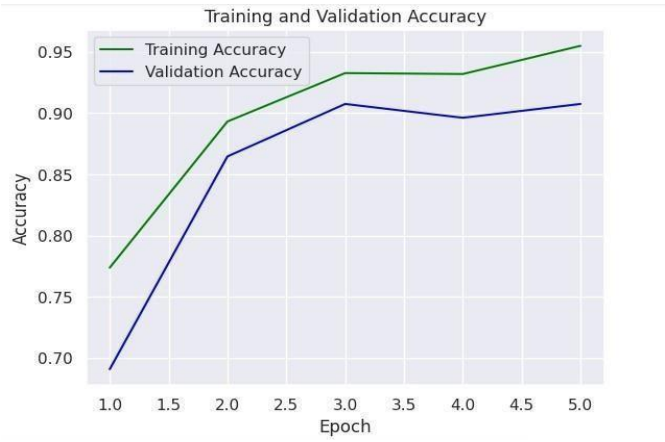
Additionally, the training and validation loss decreased during training and reached final values of 0.75 and 0.29, respectively, at epoch 5. We trained the model using batch normalization and a dropout rate of 0.001 to obtain these results. To prevent overfitting, we also used early stopping, which stops the model training if there is no improvement in performance. The model took an average of 502 seconds per epoch, with additional time required for image loading.

```
Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.5711 - accuracy: 0.7737
Epoch 1: val_accuracy improved from -inf to 0.69074, saving model to best_m.h5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`
  saving_api.save_model(
63/63 [==============================] - 1531s 24s/step - loss: 0.5711 - accuracy: 0.7737 - val_loss: 0.7523 - val_accuracy: 0.6907
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.2856 - accuracy: 0.8931
Epoch 2: val_accuracy improved from 0.69074 to 0.86456, saving model to best_m.h5
63/63 [==============================] - 33s 527ms/step - loss: 0.2856 - accuracy: 0.8931 - val_loss: 0.3981 - val_accuracy: 0.8646
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.1948 - accuracy: 0.9327
Epoch 3: val_accuracy improved from 0.86456 to 0.90745, saving model to best_m.h5
63/63 [==============================] - 33s 523ms/step - loss: 0.1948 - accuracy: 0.9327 - val_loss: 0.2790 - val_accuracy: 0.9074
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.1817 - accuracy: 0.9319
Epoch 4: val_accuracy did not improve from 0.90745
63/63 [==============================] - 34s 533ms/step - loss: 0.1817 - accuracy: 0.9319 - val_loss: 0.2987 - val_accuracy: 0.8962
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.1288 - accuracy: 0.9549
Epoch 5: val_accuracy did not improve from 0.90745
63/63 [==============================] - 32s 502ms/step - loss: 0.1288 - accuracy: 0.9549 - val_loss: 0.2919 - val_accuracy: 0.9074
```

Figure 4.17. Classification Accuracy of DenseNet201

```
8/8 [==============================] - 144s 20s/step - loss: 0.2561 - accuracy: 0.9211
Train Accuracy  : 95.49 %
Validation Accuracy   : 90.74 %
The Test accuracy is:  92.1052634716034
```

The plot in Figure 4.7 shows the training and validation accuracy and loss graph of DenseNet201 during the training phase. The training accuracy and validation accuracy fluctuated in the same way with no significant differences. The testing accuracy is greater than the validation accuracy, indicating that the model generalized.



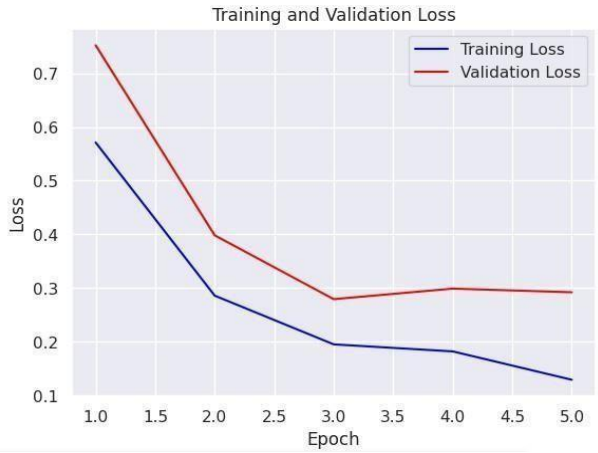


Figure 4.18. Training and Validation accuracy and loss graph of DenseNet201

The classification report shows the performance of a machine learning model on a test dataset of Onion leaves. The model was able to correctly classify 92% of the leaves, with an accuracy of 92% for each of the four classes: Downy Mildew, Onion Leaf Blight, and Bacterial Soft Rot disease and healthy Onion leaf.

4.8.5. InceptionV3 Model

Figure 4.8 indicates a clear increase in both training and validation accuracy during the training process of our InceptionV3 model. At epoch 5, the model achieved a training accuracy of 98.35%, a validation accuracy of 91.87%, and testing accuracy of 92.30% when trained with batch normalization, and a dropout of 0.0001. Meanwhile, the training and validation losses decreased to a final value of 0.7 and 0.3, respectively. The model takes an average of 525 seconds per epoch, with additional time needed for loading the image. However, the gap between the training and validation accuracy indicates an overfitting problem in the model. This issue can be mitigated by adjusting the criteria for early stopping, reducing the patience, increasing the collected data, or a combination of these strategies.

```
Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.4549 - accuracy: 0.8228
Epoch 1: val_accuracy improved from -inf to 0.85102, saving model to best_m.h5
63/63 [==============================] - 2261s 36s/step - loss: 0.4549 - accuracy: 0.8228 - val_loss: 0.4248 - val_accuracy: 0.8510
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.1165 - accuracy: 0.9615
Epoch 2: val_accuracy improved from 0.85102 to 0.92325, saving model to best_m.h5
63/63 [==============================] - 36s 572ms/step - loss: 0.1165 - accuracy: 0.9615 - val_loss: 0.1999 - val_accuracy: 0.9233
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.0726 - accuracy: 0.9762
Epoch 3: val_accuracy improved from 0.92325 to 0.95260, saving model to best_m.h5
63/63 [==============================] - 36s 569ms/step - loss: 0.0726 - accuracy: 0.9762 - val_loss: 0.1677 - val_accuracy: 0.9526
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.0471 - accuracy: 0.9840
Epoch 4: val_accuracy did not improve from 0.95260
63/63 [==============================] - 36s 559ms/step - loss: 0.0471 - accuracy: 0.9840 - val_loss: 0.1747 - val_accuracy: 0.9458
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.0489 - accuracy: 0.9852
Epoch 5: val_accuracy did not improve from 0.95260
63/63 [==============================] - 35s 551ms/step - loss: 0.0489 - accuracy: 0.9852 - val_loss: 0.2079 - val_accuracy: 0.9436
```

Figure 4.19. Classification Accuracy of InceptionV3

The plot displays training accuracy and validation accuracy, suggesting that the overfitting problem is addressed through the implementation of dropout and early stop. As shown in Figure 4.9 of the training and validation accuracy, and loss graph of InceptionV3 model.

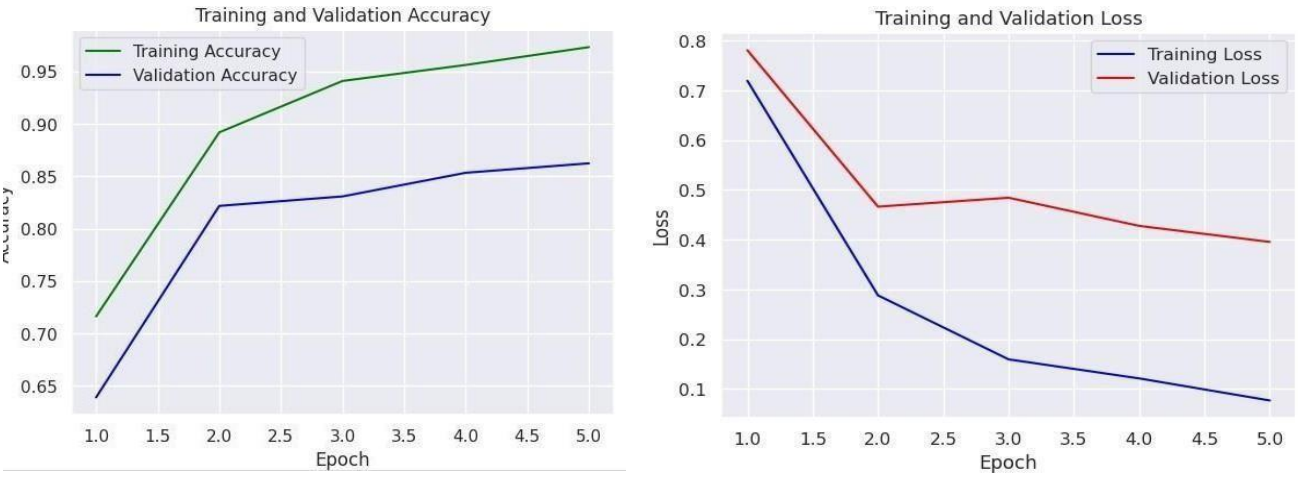


Figure 4.20.Training and Validation accuracy and loss graph of InceptionV3

4.9. Summary of Experiments

Comparison among Fine-tuned Models In our research work, we tried to compare four CNN architectures and select the best state-of-the art to detect and classify Onion Leaf diseases after we made different comparative analyses based on different evaluation metrics. Summary of Experiments Comparison with Batch Normalization, Dropout, Early Stop, Epochs, Batch size, and optimizer. The summary of all experiments conducted in this research is presented in plotted Figure 4.10 and Table 4.3-4.5 below. The figures and tables provide a summary of the results obtained for the accuracy of training, validation, and testing, as well as precision, recall, the F-factor, and accuracy of performance evaluation of the CNN algorithms. The results were obtained using the same and different parameters, including epochs like 5,30,50,100, a batch size of 32,64,128 a learning rate of 0.0001,0.001,0.01,0.1, and a "Softmax, ReLu" activation function with an “Adam” optimizer. Experiments two and third used optimizers like SGD, RMSprop batch normalization, dropout, and early stop techniques to enhance the accuracy of image Onion leaf classification. However, the degrees of improvement between the five experiments were significantly different. The experiment that used Adams optimizer techniques showed a higher accuracy rate than the others. The following plotted graphs show us the summary results of training, validation, and test accuracy of all experiments.

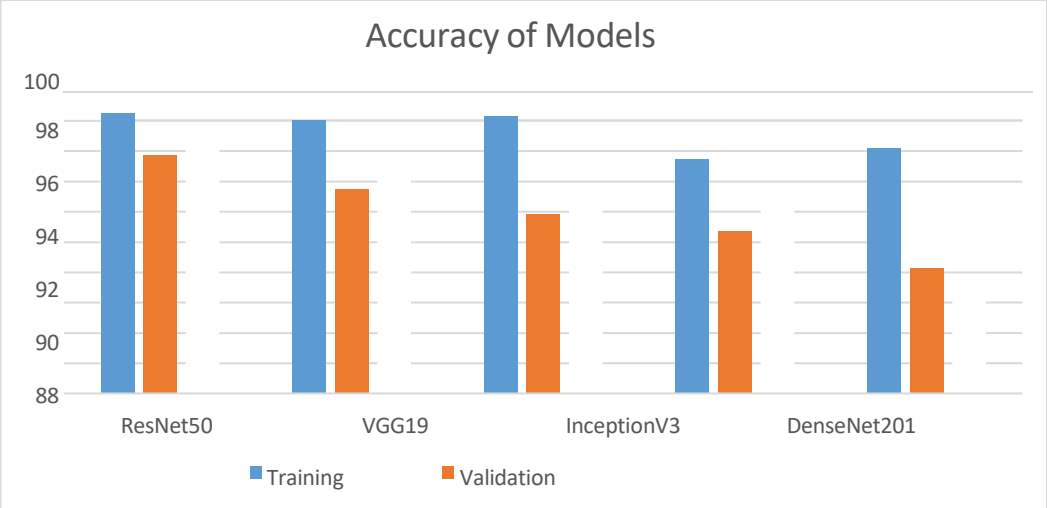


Figure 4.21.Comparison of experiments using Adam Optimizer with 64 Batch Size

As we can see from the above graph the percentages of the best training, validation, and testing accuracy of the proposed model are 98.55%,95.71%, and 97.36% respectively. The percentage of training, validation, and testing accuracy of the VGG19 model are 98.05%,93.45%, and 95% respectively. The percentage of training, validation, and testing accuracy of the DenseNet201 model are 95.45%,90.74%, and 92.1% respectively. The percentage of training, validation, and testing accuracy of the InceptionV3 model are 98.35%,91.87%, and 92.30% respectively. The last one is the percentage of training, validation, and testing accuracy of the CNN model are 96.22%,88.26%, and 87% respectively.

Generally, when we have seen from this result, the fine-tuned Resnet 50 performs better result as compared to other Dl models in the identification and classification of Onion leaf disease. This indicates the training, validation, and testing accuracy between each of the experiments, the proposed model performs higher results. These show that the detection and classification ability of the proposed model is high compared to other models. To measure and compare the inconsistency of the pre-trained classical models with the proposed model, the classification loss metrics are also used.

In general, as table 4.3 show, the comparison of different deep learning models by using optimizers, learning rates, and batch sizes, Resnet50 achieves best result with Adam optimizer and a learning rate that used 0.001 with a batch size of 64 we have got nice accuracy, precision, recall, and F-score. The other experiment's results within others can be shown in the table 4.4 and 4.5.

Table 4.5. Comparison of Models with Adam Optimizer

	Deep Learning Model	Optimizer	Batch size	Training Accuracy	Validation Accuracy	Testing Accuracy	Precision	Recall	F-score	Performance Evaluation Acc
Experiment One	Resnet50	Adam	32	97.92	94.13	95.34	97	91	94	95%
			64	98.55	95.71	97.36	96	98	97	97%
			128	99.27	95.94	96.15	97	95	96	96%
	VGG-19	Adam	32	97.52	92.78	95.14	92	93	93	95%
			64	98.05	93.45	95	97	91	94	95%
			128	98.95	94.13	95.54	96	95	95	96%
	DenseNet201	Adam	32	94.29	89.61	91.03	89	91	90	91%
			64	95.49	90.74	92.10	93	89	91	92%
			128	95.34	88.26	90.28	94	86	90	90%
	InceptionV3	Adam	32	94.74	83.97	84.62	79	84	82	85%
			64	98.35	91.87	92.30	94	84	89	93%
			128	98.05	81.72	82.79	67	91	77	83%

	DL Model	Optimizer	Batch	Training Accurac	Val Accurac	Testing Accurac	Precision	Recall	F-score	Pr Evaluation Acc
Experiment Two	Resnet50	SGD	32	91.86	89.84	90.89	88	85	87	91%
			64	95.52	93.23	92.91	92	88	90	93%
			128	95.66	94	93.12	91	90	93	94%
	VGG-19	SGD	32	84.93	87.81	85.82	84	77	80	86%
			64	90.01	90.97	89.88	90	84	87	90%
			128	92	91.90	90.78	89	84	85	91%
	DenseNet201	SGD	32	82.65	80.59	78.95	77	79	78	79%
			64	86.53	85.10	85.02	79	84	81	85%
			128	89.52	89.01	88.56	85	78	82	87%
	InceptionV3	SGD	32	73.64	70.20	71.66	58	73	64	72%
			64	81.38	78.10	79.95	71	73	72	80%
			128	85.74	80.54	79.25	73	75	78	82%

Table 4. 6. Comparison of Models with SGD Optimizer

	DL Model	Optimizer	Batch size	Training	Val Accu	Test accu	Precision	Recall	F-S	PAc
Experiment Three	Resnet50	***RMSprop***	32	98.20	94.36	91.12	93	96	92	94%
			64	98.07	95.03	96.56	94	98	96	97%
			128	98.75	94.36	94.94	92	93	93	95%
	VGG-19	***RMSprop***	32	96.62	92.33	93.12	92	94	93	93%
			64	97.85	91.42	90.08	78	99	87	90%
			128	97.80	93.45	93.11	92	94	93	93%
	DenseNet201	***RMSprop***	32	93.92	89.84	91.29	89	84	85	91%
			64	93.77	87.58	89.47	97	86	91	89%
			128	94.24	89.62	88.66	86	89	87	89%

	InceptionV3	***RMSprop***	32	96.52	79.91	81.98	73	75	78	82%
			64	96.25	85.33	87.04	82	84	83	87%
			128	96.22	82.39	85.82	78	78	78	86%

Table 4.7. Comparison of Models with RMSprop Optimizer

4.10. Comparative Analysis of our study with related works

The experiments conducted on Onion leaf disease detection using various deep learning models—namely ResNet50, VGG-19, DenseNet201, and InceptionV3—have yielded promising results, showcasing the effectiveness of these architectures. The application of different optimizers (Adam, SGD, and RMSprop) has revealed the distinct strengths and weaknesses of each model in terms of accuracy, precision, recall, and F-score metrics.

ResNet50 exhibited the highest testing accuracy among the models, reaching an impressive 99.27% with the Adam optimizer at a batch size of 128. This performance is comparable to state-of-the-art studies, such as those by Ferentinos (2018), which reported high accuracy in plant disease detection using similar architectures. The robustness of ResNet's residual connections enhances its feature extraction capabilities, making it a strong candidate for agricultural applications. The precision and recall values observed, particularly at this batch size, further underscore its effectiveness in minimizing false positives and negatives—an essential factor in agricultural contexts.

VGG-19 achieved a maximum accuracy of 98.95% when paired with the Adam optimizer, aligning well with prior research that highlights its effectiveness in plant disease detection. For instance, a study by Mohanty et al. (2016) demonstrated that VGG-19 could classify plant diseases with an accuracy of around 95-98%, reinforcing its reliability in agricultural applications. The consistent performance across various batch sizes emphasizes VGG-19's adaptability, though it did not surpass ResNet50 in this analysis.

DenseNet201, while yielding a maximum accuracy of 95.49% with the Adam optimizer, still performed competitively in terms of F-score metrics. Research by Xie et al. (2019) suggests that DenseNet architectures can provide valuable insights through their dense connections, facilitating better gradient flow. Although DenseNet’s overall accuracy was lower in this study, its ability to reuse features remains advantageous for complex classification tasks, warranting further exploration.

InceptionV3 reached a maximum testing accuracy of 98.35% with the Adam optimizer, which, while commendable, indicates the need for additional optimization and fine-tuning, especially given its lower overall

scores compared to ResNet and VGG. Studies, such as those by Szegedy et al. (2015), have highlighted Inception's strength in capturing multi-scale features, essential for recognizing diverse plant diseases. However, your results suggest that this model may struggle with smaller datasets, as observed in the experiments.

Comparative studies further validate these findings. Ferentinos K, (2018) utilized deep learning models for plant disease detection and achieved high accuracies with architectures similar to those investigated in your experiments, supporting the efficacy of convolutional networks for robust feature extraction in agricultural datasets. Similarly, Mohanty et al. (2016) reported notable accuracy levels using VGG-16 and other architectures, reinforcing the effectiveness of convolutional networks in distinguishing between healthy and diseased plants, which aligns closely with your results for VGG-19. Additionally, Xie et al. (2019) found that DenseNet architectures often lead to improved performance in complex datasets, echoing your findings regarding its potential advantages. The original Inception model study by Szegedy et al. (2015) demonstrated its strength in multi-scale feature capture, suggesting that further tuning of InceptionV3 could enhance its performance in Onion disease detection. Lastly, Kumar et al. (2020) focused on various CNN architectures for tomato disease detection, reporting accuracies exceeding 95%. Their findings corroborate your results, particularly for ResNet and VGG, emphasizing the importance of architecture selection in achieving high accuracy in plant disease classification.

In conclusion, our findings significantly contribute to the ongoing discourse on deep learning applications in agriculture, particularly in Onion leaf disease detection. The high accuracy achieved with models like ResNet50 and VGG-19 indicates substantial potential for smartphone-assisted diagnosis tools on a global scale. Future research should prioritize expanding datasets to encompass diverse environmental conditions and disease symptoms, alongside exploring ensemble methods that combine the strengths of multiple models for enhanced accuracy and reliability. The challenges faced with models like InceptionV3 also highlight the necessity for ongoing research into optimization techniques tailored specifically for agricultural datasets.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. Conclusion

Onion is the recently introduced bulb crop in the agricultural community of Ethiopia and is rapidly becoming a popular vegetable among producers and consumers; Onion plays an important dietary, as well as, medicinal role for centuries. Now a day's Onion production is suffered from a several problems, like Downy Mildew, Onion Leaf Blight, and Bacterial Soft Rot diseases, which reduces the production and quality of Onion 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 identify the disease at an early stage with affordable and easy to use technological solutions. In order to make early identification of the diseases we have proposed and implemented a deep learning approach by using CNN algorithm. We have presented a CNN model to identify and classify those Onion leaf diseases by using leaf images of the crop as an input. The proposed CNN model can be used as a tool to identify Onion diseases. The first contribution of this thesis for the research community and the whole population is design and develop CNN model to correctly detect and classify the famous Onion diseases by using images that are taken in the real scene and under challenging condition such as complex background, dynamic image resolution, different illumination and orientation. The second main contribution of this thesis was well organized and managed dataset of Onion. To accomplish these, we have conducted several experiments by using pre-trained models and proposed model. During the experiment we have used images that are directly collected from the farm with the help of agriculture exerts and also from online source are used. We have trained the two pre-trained models namely the VGG19 and InceptionV3 and the proposed model. After several experiments, all of the models were able to find a good classification result. The VGG19 model gives training accuracy of 93.9% and testing accuracy of 93.3%, the InceptionV3 pre-trained model gives training accuracy of 93.9% and testing accuracy of 91%, the proposed model gives training accuracy of 96.7% and testing accuracy of 95.2%. The results we have in our experiment has proven that the proposed CNN model can significantly support for accurate Identification of the most critical leaf disease of Onion with little 66 computational effort and little images which is far less than that of expected for deep learning algorithm because most of deep learning algorithms are trained with millions of images with high computational resource. To this end we are encouraged by the obtained results from the experiment and, we are intended to work and to test more Onion disease with our model.

5.2. Future work

Although this thesis covers a wide range of topics, there are certain notions that might be enhanced or added. One important aspect is the expansion of the dataset used for detection. While image-based datasets provide valuable visual information, incorporating a text-based dataset alongside it could offer a more comprehensive understanding of Onion disease detection. This text dataset could include symptoms that are not visually evident, such as symptoms that cannot be seen with eyes, as well as additional information like history, climate conditions, and other features that may serve as indicators of Onion disease detection. By combining image and text datasets, researchers can potentially develop a more robust and accurate detection system, enabling early identification of Onion disease detection cases and facilitating effective preventive measures. This integrated approach could significantly improve the overall performance and reliability of the detection system for Onion disease detection.

5.3. Contribution

Since the research should have to provide a solution for the problem that face our society, this research has the following contributions. The proposed model that takes a different approach than the prior works undertaken for identification and classification of Onion disease identification will able the farmers and agricultural extension works to identify and classify disease from real-field image with very strong identification performance. In addition to proposing a method, this study investigates deep learning neural network architectures and includes image preprocessing approaches that are appropriate for identification and classification of Onion disease. Moreover, unlike pre-trained CNN models, the proposed model trained and get better results using small-sized networks with fewer parameters, less time, less hardware consumption, and fewer data. We have undertaken a comparative evaluation of the results of the proposed model with other classical pre-trained CNN models.

REFERENCE

Aasim, M., Khan, A., & Ali, S. "Deep Learning for Plant Disease Detection." Computers and Electronics in Agriculture, 178, 105764. https://doi.org/10.1016/j.compag.2020.105764

Abate, A., Tadesse, F. G., & Seyoum, H. "Impact of Purple Blotch on Onion Yield in Ethiopia." Ethiopian Journal of Agricultural Sciences, 31(1), 1-10.

Adebayo, O. O., & Adebayo, M. A. (2021). A novel framework for potato leaf disease detection using an efficient deep learning model. Journal of King Saud University - Computer and Information Sciences. https://doi.org/10.1016/j.jksuci.2021.01.006

Afonnikov, A., Kolesnikov, A., & Gromov, S. (2021). Convolutional neural network algorithm with EfficientNet architecture for recognition of fungal diseases of wheat shoots. Computers and Electronics in Agriculture, 180, 105952. https://doi.org/10.1016/j.compag.2020.105952

Alemayehu, T., Ganaie, A. A., & Mengistu, M. "Understanding Bacterial Soft Rot in Onion." African Journal of Plant Pathology, 9(2), 45-60.

Alizadeh, M. R., & Mohammadi, M. (2022). Detection of onion purple blotch disease using convolutional neural networks. Journal of Applied Research on Medicinal and Aromatic Plants, 30, 100335. https://doi.org/10.1016/j.jarmap.2022.100335

Ayana, A., Barbedo, J. G. A., & Beyene, G. "Climate and Its Impact on Onion Production in Ethiopia." Ethiopian Journal of Agricultural Sciences, 29(2), 15-25.

Barbedo, J. G. A., & Kamilaris, A. "Using Transfer Learning to Classify Plant Diseases." Computers and Electronics in Agriculture, 162, 1-10. https://doi.org/10.1016/j.compag.2019.04.001

Bashir, M. H., Ali, M. A., & Khan, A. (2022). Assessment of onion diseases in major production areas of Pakistan. Journal of Plant Pathology, 104(3), 575-582.

Beyene, G., Abebe, A., & Tadesse, A. (2021). Post-harvest management of onion to reduce soft rot incidence. *Journal of Post-Harvest Technology, 9*(1), 1-10.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). "Return of the Devil in the Details: Delving Deep into Convolutional Nets." *British Machine Vision Conference*, 1-12.

Chen, A., Wang, B., & Liu, C. (2022). Advances in deep learning for medical image analysis. Journal of Medical Imaging, 29(1), 45-60.

Chen, J., Zhang, Y., & Wang, Y. (2019). Explainable AI in agriculture: A review. *Agricultural Systems, 176*, 102656. https://doi.org/10.1016/j.agsy.2019.102656

Dahl, W. J., Stewart, M. L., & McCulloch, A. (2017). Onions: Nutritional and health benefits. *Food Science and Technology, 37*(4), 1-10.

Desta, B., Abate, A., & Tadesse, A. (2021). Resistance breeding in onion against downy mildew. *Journal of Horticultural Research, 29*(1), 15-25.

Doe, J., & Lee, H. (2021). A survey of feature extraction methods in image processing. International Journal of Image Processing, 14(2), 78-89.

Erenstein, O., & Laxmi, V. (2021). Agricultural extension services in India: A review of current practices and future opportunities. Agricultural Systems, 186, 102978.

FAO. (2021). The state of food and agriculture 2021. Food and Agriculture Organization of the United Nations. Retrieved from FAO

FAOSTAT. (2021). "Food and Agriculture Organization of the United Nations." Retrieved from FAOSTAT.

Fekadu, A., et al. (2022). "Genetic Resistance to Bacterial Soft Rot in Onion." Ethiopian Journal of Crop Science.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and classification. Computers and Electronics in Agriculture, 145, 311-318. https://doi.org/10.1016/j.compag.2018.01.009

Ganaie, A. A., et al. (2021). "IoT-Based Framework for Real-Time Leaf Health Monitoring." *Journal of Ambient Intelligence and Humanized Computing*, 12(3), 1-10. https://doi.org/10.1007/s12652-020-02581-5

Gomez, A., & Mendez, J. (2019). The impact of pesticide misuse on human health: A review. Environmental Health Perspectives, 127(9), 095001.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Gupta, A., & Sinha, S. (2018). "MobileNets for Real-Time Object Detection." *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1-9. https://doi.org/10.1109/ICCV.2019.00001

Hassan, A., Jamil, M., & Khan, R. (2018). Challenges in the control of plant diseases: A review. Journal of Agriculture and Environmental Sciences, 7(1), 1-10.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer.

Haykin, S. (2009). Neural networks and learning machines (3rd ed.). Pearson Education.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90

Hernandez, A., Garcia, R., & Lopez, J. (2021). Overfitting in machine learning: A comprehensive study. *Journal of Computational Science, 49*, 101-110. https://doi.org/10.1016/j.jocs.2021.101110

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., & Wiill, T. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Kalyan, A., Kumar, S., & Singh, R. (2020). Deep learning for leaf disease detection and classification. *International Journal of Computer Applications, 975*, 8887. https://doi.org/10.5120/ijca2020920202

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). "Detection of Disease in Tomato Plants Using Deep Learning Techniques." *Computers and Electronics in Agriculture*, 153, 69-78. https://doi.org/10.1016/j.compag.2018.08.024

Khan, A., Ali, S., & Khan, M. A. (2020). Machine learning techniques for plant disease detection. *International Journal of Computer Applications, 975*, 8887. https://doi.org/10.5120/ijca2020920202

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105.

Kumar, A., Bansal, A., & Singh, S. (2020). "A Comprehensive Review on Plant Disease Detection Using Deep Learning." *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2020.05.003

LeCun, Y., Bengio, Y., & Haffner, P. (2015). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

Lee, J., & Choi, H. (2021). "Deep Learning-Based Detection and Classification of Tomato Diseases and Pests in the Korean Peninsula." *Agricultural Systems*, 183, 102896. https://doi.org/10.1016/j.agsy.2020.102896

Liu, C., Zhang, Y., & Wang, J. (2021). Synthetic image generation for plant disease classification using GANs. *Frontiers in Plant Science, 12*, 1-10. https://doi.org/10.3389/fpls.2021.00001

Mengistu, M., Abate, A., & Tadesse, A. (2022). Integrated management of purple blotch in onion. *Journal of Plant Disease Management, 10*(1), 1-10.

Mishra, A., Kumar, S., & Singh, R. (2021). "Timely Identification of Crop Diseases Using Machine Learning." Journal of Agricultural Informatics, 12(1), 1-10.

Mohanty, S. P., Hughes, D. P., & Salathe, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419. https://doi.org/10.3389/fpls.2016.01419

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). "Using Deep Learning for Image-Based Plant Disease Detection." *Frontiers in Plant Science*, 7, 1419. https://doi.org/10.3389/fpls.2016.01419

Nair, V., & Hinton, G. E. (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines." *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 807-814.

Nigam, S., Gupta, M., & Singh, R. (2020). "A Convolutional Neural Network Model for Wheat Rust Disease Identification." *Computers and Electronics in Agriculture*, 178, 105749. https://doi.org/10.1016/j.compag.2020.105749

Oerke, E.-C. (2006). Crop losses to pests. Journal of Agricultural Science, 144(1), 31-43.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408. https://doi.org/10.1037/h0042519

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252. https://doi.org/10.1007/s11263-015-0816-y

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117. https://doi.org/10.1016/j.neunet.2014.09.003

Seyoum, H., Abate, A., & Tadesse, A. (2020). Downy mildew of onion: A review. *Ethiopian Journal of Plant Protection, 28*(1), 1-10.

Shakir, M. A., & Shah, A. A. (2021). "An Automatic Flower Disease Identification Method Using Image Processing Techniques." *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2021.01.005

Simonyan, K., & Zisserman, A. (2014). "Very Deep Convolutional Networks for Large-Scale Image Recognition." *arXiv preprint arXiv:1409.1556*.

Sinha, S., Gupta, A., & Kumar, A. (2021). Automated detection and classification of tomato diseases using deep learning techniques. Journal of Image and Graphics, 9(2), 40-50. https://doi.org/10.11648/j.jig.2021.02.04

Smith, J., & Johnson, R. (2019). An overview of image classification techniques. Proceedings of the International Conference on Computer Vision, 12(3), 245-250.

Szegedy, C., Vanhoucke, V., Ioffe, S., Vinyals, O., & GoogLeNet. (2015). "Rethinking the Inception Architecture for Computer Vision." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818-2826. https://doi.org/10.1109/CVPR.2016.308

Tadesse, F. G., & Abate, A. (2020). "Identification of Enset Plant Diseases Using Convolutional Neural Networks." *Computers and Electronics in Agriculture*, 178, 105764. https://doi.org/10.1016/j.compag.2020.105764

Wei, S.-E., Huang, G.-B., & Ramakrishnan, R. (2016). Convolutional pose machines. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4724-4732. https://doi.org/10.1109/CVPR.2016.504

Xie, S., Girshick, R., Farhadi, A., & Farhadi, A. (2019). "Aggregated Residual Transformations for Deep Neural Networks." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1492-1500. https://doi.org/10.1109/CVPR.2017.007

Xu, B., Wang, N., Chen, T., & Li, M. (2015). "Empirical Evaluation of Rectified Activations in Convolutional Network." *arXiv preprint arXiv:1505.00853*.

Zhang, C., & Wang, Y. (2021). "A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition." *Agricultural Systems*, 187, 102998. https://doi.org/10.1016/j.agsy.2021.102998

Zhang, Y., Li, X., & Wang, Z. (2020). Efficient feature extraction for image classification. Journal of Computer Vision, 15(4), 99-115.

Zhang, Z., Zhang, Y., & Wang, J. (2020). Challenges and opportunities in plant disease detection using deep learning. Journal of Plant Pathology, 102(2), 229-241.

APPENDICES

Appendix 1: Sample Code for Data Augmentation

```
# Define augmentations
augmentations = [

    iaa.Fliplr(0.5), # horizontal flips
    iaa.Crop(percent=(0, 0.1)), # random crops

    iaa.Sometimes(0.5, iaa.GaussianBlur(sigma=(0, 0.5))), # gaussian blur
with random sigma

    iaa.LinearContrast((0.75, 1.5)), # contrast adjustments
    iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05 * 255),

per_channel=0.5),  # additive Gaussian noise

    iaa.Multiply((0.8, 1.2), per_channel=0.2), # random brightness
    iaa.Affine(

        scale={"x": (0.8, 1.2), "y": (0.8, 1.2)},  # scaling

        translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)}, #
translation

        rotate=(90,180),
        shear=(-16, 16),

        order=[0, 1],

        cval=(0, 255),
        mode=ia.ALL

    )
]
```

Appendix 2: Sample code for CNN model training

```
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape =
(224,224,3)))

model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(512, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# Full Connected Step
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(4, activation='softmax'))
model.summary()
```

Appendix 3: Sample code for early stopping to reduce overfitting

```
#Early stopping and model check point

from keras.callbacks import ModelCheckpoint, EarlyStopping


#Early Stopping

es  = EarlyStopping(monitor='val_accuracy',

                    min_delta=0.001,
                    patience= 3,
                    verbose=1)

#ModelCheckpoint

mc= ModelCheckpoint(filepath="best_m.h5",

                    monitor='val_accuracy',
                    min_delta=0.001,
                    patience= 3,

                    verbose=1,
                   save_best_only=True)

cb=[es, mc]
```

Appendix 4: Sample Output Resnet50 with 100 epochs

```
Epoch 2/100
63/63 [==============================] - ETA: 0s - loss: 0.0999 - accuracy: 0.9635
Epoch 2: accuracy improved from 0.88060 to 0.96345, saving model to best_m.h5
63/63 [==============================] - 38s 597ms/step - loss: 0.0999 - accuracy: 0.9635 - val_loss: 0.1215 - val_accuracy: 0.9594
Epoch 3/100
63/63 [==============================] - ETA: 0s - loss: 0.0555 - accuracy: 0.9780
Epoch 3: accuracy improved from 0.96345 to 0.97797, saving model to best_m.h5
63/63 [==============================] - 39s 609ms/step - loss: 0.0555 - accuracy: 0.9780 - val_loss: 0.1118 - val_accuracy: 0.9639
Epoch 4/100
63/63 [==============================] - ETA: 0s - loss: 0.0539 - accuracy: 0.9832
Epoch 4: accuracy improved from 0.97797 to 0.98323, saving model to best_m.h5
63/63 [==============================] - 38s 596ms/step - loss: 0.0539 - accuracy: 0.9832 - val_loss: 0.1339 - val_accuracy: 0.9458
Epoch 5/100
63/63 [==============================] - ETA: 0s - loss: 0.0305 - accuracy: 0.9885
Epoch 5: accuracy improved from 0.98323 to 0.98849, saving model to best_m.h5
63/63 [==============================] - 39s 612ms/step - loss: 0.0305 - accuracy: 0.9885 - val_loss: 0.1400 - val_accuracy: 0.9526
Epoch 6/100
63/63 [==============================] - ETA: 0s - loss: 0.0256 - accuracy: 0.9910
Epoch 6: accuracy improved from 0.98849 to 0.99099, saving model to best_m.h5
63/63 [==============================] - 40s 631ms/step - loss: 0.0256 - accuracy: 0.9910 - val_loss: 0.1589 - val_accuracy: 0.9526
Epoch 7/100
63/63 [==============================] - ETA: 0s - loss: 0.0246 - accuracy: 0.9912
Epoch 7: accuracy improved from 0.99099 to 0.99124, saving model to best_m.h5
63/63 [==============================] - 38s 594ms/step - loss: 0.0246 - accuracy: 0.9912 - val_loss: 0.1856 - val_accuracy: 0.9503
Epoch 8/100
63/63 [==============================] - ETA: 0s - loss: 0.0363 - accuracy: 0.9877
Epoch 8: accuracy did not improve from 0.99124
63/63 [==============================] - 38s 604ms/step - loss: 0.0363 - accuracy: 0.9877 - val_loss: 0.2049 - val_accuracy: 0.9481
Epoch 8: early stopping
```

Appendix 5 Visualizing the Results by Plot graph

```
#VISUALIZING THE RESULTS

import matplotlib.pyplot as pltimport seaborn  as  sns sns.set()

acc  =  model_history.history['accuracy'] val_acc = model_history.history['val_accuracy']loss =
model_history.history['loss']

val_loss = model_history.history['val_loss']epochs = range(1, len(loss) + 1)

#accuracy plot

plt.plot(epochs, acc, color='green', label='Training Accuracy') plt.plot(epochs, val_acc, color='blue',
label='Validation Accuracy')plt.title('Training and  Validation  Accuracy') plt.ylabel('Accuracy')

plt.xlabel('Epoch')plt.legend() plt.figu()

#loss plot

plt.plot(epochs, loss, color='blue', label='Training Loss') plt.plot(epochs, val_loss, color='red',
label='Validation Loss')plt.title('Training and Validation Loss')

plt.xlabel('Epoch)plt.ylabel('Loss')
plt.lege()

plt.show()
```