

**AN INTELLIGENT HYBRID MODEL FOR TOMATO LEAF
DISEASE DETECTION AND CONTROL RECOMMENDATIONS**

ESOMONU NKECHI FAUSTINA, B.Tech, M.Sc.

20154947558

**A Ph.D DISSERTATION SUBMITTED TO DEPARTMENT OF
INFORMATION TECHNOLOGY, FEDERAL UNIVERSITY OF
TECHNOLOGY, OWERRI**

SEPTEMBER, 2023

©Federal University of Technology, Owerri

CERTIFICATION

This is to certify that this thesis entitled “An Intelligent Hybrid Model for Tomato Leaf Disease Detection and Control Recommendations” was carried out by **Esomonu Nkechi Faustina** with the Registration number **20154947558** and submitted to the Department of Information Technology, School of Information and Communication Technology, Federal University of Technology, Owerri in partial fulfilment of the requirement for the award of Doctor of Philosophy (Ph.D.) in Information Management Technology.



Prof. (Mrs.) U. F. Eze
(Lead Supervisor)

06-12-23
Date



Dr. A. M. John-Otumu
(1st Co-supervisor)

06-12-23
Date



Dr. I. I. Ayogu
(2nd Co-supervisor)

06/12/2023
Date



Dr. A. I. Otuonye
(Head of Department)

6/12/23
Date



Prof. (Mrs.) U.F.Eze
(Dean, SICT)

06-12-23
Date

Prof. B. O. Esonu
(Dean, Post Graduate School)

Date



Prof. M.O. Onyesolu
External Examiner

6/12/23
Date

DEDICATION

This endeavor stands as the outcome of numerous and demanding sacrifices. It is with sincere enthusiasm and immense pride that this effort is devoted to those who have been a wellspring of inspiration. From the guiding presence of parents, guardians, classmates, and a close-knit circle of friends, whose support proved invaluable during the challenges faced while completing this work, to the dedicated faculty and staff of the Information Technology department at Federal University of Technology Owerri.

Above all, heartfelt gratitude is extended to the Almighty God, who has bestowed blessings upon every facet of our lives, particularly for granting the fortitude, valor, patience, sagacity, time, and guidance necessary for the fulfillment of this endeavor.

ACKNOWLEDGEMENTS

First and foremost, I wish to convey my profound and genuine appreciation to my esteemed research mentor, Prof. (Mrs.) U. F. Eze, who served as my lead supervisor. I am equally indebted to my co-supervisors, Dr. A. M. John-Otumu and Dr. I. I. Ayogu. Their constructive feedback and invaluable guidance have been instrumental throughout this research journey. Their visionary leadership, sincerity, and motivational influence have been a profound source of inspiration. They imparted to me the research methodology and skills necessary to present the work with utmost clarity. Working and studying under their guidance has been a privilege and an honor. I extend my thanks to the Head of Department, Dr. Anthony Otonye, and the entire staff of the IFT department for their unwavering support during my PhD pursuit. Special acknowledgment goes to Mr. Suaghter Ayahan for his invaluable assistance in analyzing my model. I deeply appreciate their camaraderie, empathy, and delightful sense of humor. My gratitude extends to my parents for their unwavering love, prayers, care, and sacrifices in providing me with an education and preparing me for the future. I am equally thankful to my current family, the Esomonu Family, for their enduring love, understanding, prayers, and ongoing support in bringing my research to completion. I would be remiss not to express my sincere appreciation to my siblings, Chikodi, Chinenye, Chinagorom, Onyenonachi, Chisom, and my brothers Mr. Bruno and Magnus Esomonu. Their contributions have been immeasurable. My friends, Okoronkwo Chinomso, Christine Okolie, and Onyema Mac-Anthony, also deserve my heartfelt thanks for their unwavering presence and genuine interest in seeing me through the successful completion of this thesis. In all, I extend my heartfelt gratitude to Almighty God for bestowing His blessings upon my research endeavors, allowing me to successfully conclude this study.

TABLE OF CONTENTS

TITLE PAGE	i
DEDICATION	iv
ACKNOWLEDGEMENT	v
ABSTRACT	xiv
TABLE OF CONTENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi

CHAPTER ONE

INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Statement	5
1.3 Objectives	Error! Bookmark not defined.
1.4 Research Questions of the Study	7
1.5 Justification of the study	7
1.6 Scope of the study	9

CHAPTER TWO

LITERATURE REVIEW	1
2.1 Conceptual Framework	11
2.1.1 Historical Overview in the Area of Study	11
2.1.2 Origin of Tomato	12
2.1.3 Planting Season, Soil Type, Vegetation, and Climate that Support Tomato Growth	14
2.1.4 Nutritional Value of the Tomato	15
2.1.5 The Tomato's Economic and Social Significance	15
2.1.6 Diseases in Tomatoes	17
2.2 Theoretical Framework	30
2.2.1 Mathematical Theory of Communication	30

2.2.2 Matrix theory	31
2.2.3 Graph theory	33
2.2.4 Prediction theory	34
2.2.5 Theory of reasoning	35
2.2.6 Diagnostic theory	36
2.2.7 Network Theory	37
2.2.8 Deep learning theory	38
2.3 Empirical Framework	40
2.4 Research Gaps	72
 CHAPTER THREE	 73
METHODOLOGY	73
3.1 Research Methodology Overview	73
3.1.1 Software Methodology	73
3.2 Software methodology adopted	75
3.2.1 Data Gathering Repository from Online and Fieldwork	80
3.2.2 Mobilenetv2 Plus SVM Techniques Usage to Determine the Best Suited Model	80
3.2.3 CNN and SVM Analysis with Varying Filter Sizes	81
3.2.4 Building a Recommendation System for Tomato Disease-Related Content	81
3.2.5 Deployment of the Proposed Model	82
3.3 Data Collection	83
3.3.1 Description of the dataset	84
3.4 System Analysis Procedures	85
3.4.1 Analysis of the existing system	85
3.4.2The proposed system architecture	90
3.4.3 Mathematical Model: Tomato Disease Classification and Recommender System	94
3.4.3.1 Proposed code segment adopted	95
3.4.3.2 Coding using python program:	96
3.5 Data Modeling	100
3.5.1 Data flow diagram (DFD)	101

3.5.2 Entity relationship diagram (ERD)	102
3.5.3 Class diagram	103
3.6 System Modeling	104
3.6.2 Activity diagram	107
3.6.3 Sequence Diagram	108
3.6.4 Use Case Diagram	109
3.7 System requirement specification	110
3.7.1 Functional requirements	111
3.7.2 Non-functional requirements	112
3.7.3 User requirements	113
3.8 System Implementation	114
3.8.1 Minimum hardware requirements	114
3.8.2 Minimum software requirements.	114
3.8.3 Choice of programming language(s)/development environment	115
CHAPTER FOUR	116
RESULTS AND DISCUSSION	116
4.1 Testing	116
4.2 Model Testing	116
4.2.1 Test for non-tomato leaves using the proposed mobile application system developed	117
4.2.2 Test for tomato leaves using the proposed mobile application system developed	121
4.3 Model's Evaluation	124
4.3.1 Feature Extraction Process	124
4.3.2 SVM classification head	124
4.3.3 Confusion Matrix and Accuracy	127
4.3.4 Mosaic/Blind spot	128
4.3.5 Blind/Bacteria spot	130
4.3.6 Bacteria/Septoria leaf	132
4.3.7 Septoria leaf/Yellow_Leaf_Virus	134
4.3.8 Yellow_Leaf_Virus/late_blight	135

4.3.9Late_blight/Leaf_mold	137
4.3.10Leaf_mold/Early_blight	138
4.4Performance Evaluation	143
4.5 System Interface	144
 CHAPTER FIVE	
CONCLUSION AND RECOMMENDATIONS	148
5.1Conclusion	148
5.2Contribution to knowledge	149
5.3Recommendations	149
5.4 Limitations	150
5.5 Areas of future work	151
REFERENCES	152
APPENDICES	171

LIST OF TABLES

Table	Title	Page
2.1:	Summary of related works on tomato leaf diseases using AI approach	53
3.1:	Advantages and disadvantages Agile Methodology	76
3.2:	Nine classes of tomato leaf disease and also a healthy class considered.	83
3.3:	Basic system functional requirements specification of the proposed model	112
3.4:	Basic non-functional requirements of the proposed model	113
3.5:	User requirements specification of the proposed system	113
3.6:	Minimum hardware specification is for android version	114
3.7:	minimum software requirements	114
4.1:	Classification of non-tomato leave	120
4.2:	Classification of tomato leave	123
4.3:	A trade-off table between parameters used for cross validation	126
4.4	Mosaic/ Blind sport	130
4.5	Blind/Bacteria spot	131
4.6	Bacteria/Septoria leaf	133
4.7	Septoria leaf/Yellow_Leaf_Virus	135
4.8	Yellow_Leaf_Virus/late_blight	136
4.9	Late_Blight/Laef_Mold	138
4.10	Leaf_mold/Early_blight	139
4.11	Recall and Precision	140

LIST OF FIGURES

Figure	Title	Page
2.1:	Tomato Early Blight	21
2.2:	Bacterial canker tomato disease with marginal burning on tomato leaflets and bird's eye spots on tomato fruits	23
2.3:	speck bacterial infect tomatoes leaflet and petiole with brown lesions	24
2.4:	Sikora (2020), Lesions on tomato leaves with bacterial spot	25
2.5:	A stem infected with bacterial wilt	26
2.6:	A buckey rot infected tomatoes. Characteristic: Concentric rings appears in the lesions	27
2.7:	Tomato leaflet infected with early blight disease.	28
2.8:	Diagram showing the pooling layer operations	33
2.9:	Graph showing nodes connectivity	34
2.10:	Deductive reasoning flow process	36
2.11:	Diagram of diagnostic process flow	37
2.12:	Diagram showing network of nodes	38
2.13:	Diagram of deep neural Network	40
3.1:	Agile software methodology	76
3.2:	Architecture of the existing system	89
3.3:	Propose architectural design for tomato leaf disease predictive model	91
3.4:	DFD of the proposed system	101
3.5:	Entity relationship diagram of the proposed system	103
3.6:	Class diagram of the proposed system	104
3.7:	System flowchart of the proposed system	105
3.8:	End User flowchart	107
3.9:	Activity diagram of the proposed system	107
3.10:	Sequenced diagram of the proposed system	109
3.11:	Use case diagram of the proposed system	110
4.1	Bitter leaf	117
4.2	Cassava leaf	117
4.3	Cocoyam leaf	117

4.4 Lemon grass leaf	117
4.5 Palm kernel leaf	118
4.6 Pawpaw leaf	118
4.7 Pineapple leaf	118
4.8 Sugar cane leaf	118
4.9 Pumpkin leaf	119
4.10: Bar graph showing percentage of both correctly and wrongly classified non-tomato leaves	121
4.11: Tomato leaf with late blight disease	122
4.12: Tomato leaf with late blight disease	122
4.13: Tomato leaf with late blight disease	122
4.14: Tomato leaf with late blight disease	122
4.15: Tomato leaf with late blight disease	123
4.16: Tomato leaf with late blight disease	123
4.17: One vs One SVM	125
4.18: Graph of Train_Test_Split	127
4.19: Confusion Matrix	128
4.20 Graph of Mosaic/ Blind spot	130
4.21 : Graph of Blind/Bacteria spot	132
4.22 Graph of Bacteria/Septoria leaf	133
4.23 Graph of Septoria leaf/Yellow_Leaf_Virus	135
4.24 Graph of Yellow_Leaf_Virus/late_blight	136
4.25 Graph of Late_Blight/Leaf_Mold	138
4.26: Graph of Leaf_mold/Early_blight	140
4.27: Mosaic/Blind Spot	141
4.28: Late_blight/Leaf Mold	141
4.29: Leaf_Mold/Early blight	142
4.27, 4.28, 4.29: ROC Curve for three identified classifiers (small SVMs)	142
4.30: ROC Curve for the average 45 classifiers	143
4.31: Comparative evaluation between existing models and proposed model	144
4.32: Proposed system interface (Mobile view)	144

4.33: The Login page	146
4.34: No disease present	146
4.35: Confident level	147

ABSTRACT

Crop diseases pose a significant threat to global food security, with the negative impact on food production stemming from a lack of early detection and effective management. Despite technological advancements, especially in artificial intelligence (AI), smallholder farmers face challenges in accessing and utilizing these tools. Tomato, a crucial vegetable crop in the global food chain, is particularly susceptible to various debilitating diseases, leading to annual losses for farmers. This research aims to address the impact of disease-related losses on tomato farmers, specifically in Nigeria, by developing an AI-powered system for early detection, precise diagnosis, and continuous monitoring of tomato diseases. The motivation for this research stems from the recognition of the limited access smallholder farmers have to efficient technology tools for managing tomato diseases. The goal is to minimize the impact of these diseases on farmers' livelihoods and, by extension, society. Through early detection and effective management, the research seeks to empower farmers with the necessary tools to mitigate losses and enhance food production. The primary objective of this research is to develop a hybrid AI-powered system that aids smallholder farmers in early detection, effective management, and continuous monitoring of tomato plantation health. The approach involves the integration of deep and shallow learning models, with a focus on transfer learning. The functional unit comprises a convolutional neural network model (MobileNetv2), a support vector machine (SVM) model, and a natural language processing component for image processing, feature extraction, classification, and management counseling. To achieve the research objective, a hybrid architecture combining deep and shallow learning models was developed. The model utilized 11,000 tomato leaf images for training, ensuring a balanced representation of healthy and diseased types. The problem was formulated as a ten-class (multiclass) classification task. Evaluation of the proposed solution employed standard metrics, and the results demonstrated state-of-the-art performance. The hybridized MobileNetv2 and SVM achieved impressive metrics, including 93.68% precision, 93.84% F-1 score, 93.81% accuracy, and 92% recall. The evaluation of the proposed solution showcased its efficacy in achieving high precision and accuracy. The hybridized model, incorporating MobileNetv2 and SVM, demonstrated state-of-the-art performance in recognizing and diagnosing tomato diseases. The results indicate the potential of the developed AI-powered system for practical implementation in agriculture. In conclusion, this research contributes significantly to addressing the challenges faced by smallholder tomato farmers in Nigeria. The developed AI-powered system, combining deep and shallow learning models, demonstrates remarkable performance in early detection, precise diagnosis, and continuous monitoring of tomato diseases. The successful implementation of the model in an Android application suggests its potential for acceptability and adoption among the farming population. However, to enhance usability, further development of the natural language component, including speech processing capabilities, is recommended, particularly considering the literacy constraints of rural farmers. This research aligns with Sustainable Development Goals 1, 2, and 3, and it is recommended for further development and support by relevant government agencies to fulfill the promise of technology utilization in improving societal well-being through increased food availability.

Keywords: AI, CNN, Classification, Images, Deep Learning, USE

CHAPTER ONE

INTRODUCTION

1.1 Background Information

Agriculture from pre-historic times is the origin of all known human development efforts as well as the bane of economic growth for most societies. In agriculture, innovative technologies can help improve both the quality and quantities of agricultural products in the country. For instance, some technologies can be used by farmers for weather forecasting, disease prediction, identification of phenology and crop harvesting. Given the significantly elevated infant mortality rates, the global human population experienced sluggish growth until the year 1700. Approximately a century later, in about 1800, it reached the milestone of about one billion individuals, with the second billion being attained around 1928, and the third billion in 1960. Notably, the world's population reached the seven billion marks in 2017 (Domingues, Brando, & Ferreira, 2022). As forecasted by the United Nations, it is anticipated that the global population will reach 9.7 billion by the year 2050 and is projected to reach 10.9 billion by 2100 (Ahmad et al., 2020; Roser & Ortiz-Ospina, 2013).

In recent years, there has been a notable increase in the demand for and consumption of agricultural products, primarily driven by the global population's rapid growth. This has consequently resulted in a significant expansion of cultivation (Frona, Janos, & Harangi-Rakos, 2019). To meet the escalating demand for food, biofuels, and animal products, it is imperative to double crop yield production by 2050. Achieving this objective will necessitate a yearly enhancement in crop yield of about 2.4%, a substantial improvement compared to the current estimated annual increase of 1.3% (Ray, Mueller, West, & Foley, 2013).

Statistics have shown that agricultural sector is a very significant sector of Nigeria's economy based on its gross domestic product (GDP) with contribution of about 23% in the first half of the year 2020(Izuaka, 2020). Small scale farming system is predominately the most practiced agricultural system in Nigeria within an average farm size of 0.2 – 3.0 hectares. This subsistence farming is engaged in by about 70% of the Nigerian population whose produce struggles to provide about 75% of current food consumption in Nigeria (FAO, 2017). Based on this report, advancement and management of small-scale farming is critical to food security. As demand for food grows in Nigerian, urban populations are increasingly dependent on the limited production from small-scale farmers for their food supplies. Hence, the food availability for urban and peri-urban populations are highly dependent on small-scale farmers.

Tomatoes is one among the very common vegetable crops grown on small scale in Nigeria and globally. Tomato is one of the most extensively used vegetable among Nigerian families (Adekunle, Alarima, &Tolorunju, 2019). Tomato growing plays a critical role which has been linked to many health benefits, generating revenue and creating employment for rural and urban populations. Tomato, scientifically known as *Lycopersicon esculentum*, belongs to the Solanaceae family and is both an edible fruit and a vegetable. Its origins can be traced back to South America, likely originating from the variety *Lycopersicon esculentum*(cherry tomato). Early Mexican Indians brought it under domestication and began cultivating it throughout Central America. Based on Spanish colonization of the Americans, tomato cultivation spread throughout the world (Modi, Modi, & Hendriks, 2006). The tomato plant is an annual crop planted outdoors in temperate climates that normally develops to a height of 1-3 meters (3-10 feet). Tomatoes are loaded with a substance called lycopene. Lycopene is an antioxidant -- it fights molecules called free radicals that can damage your cells and affect your immune

system. Because of that, foods high in lycopene, like tomatoes, may make you less likely to have lung, stomach, or prostate cancer. Some research shows they might help prevent the disease in the pancreas, colon, throat, mouth, breast, and cervix as well. It gives them their bright red color and helps protect them from the ultraviolet rays of the sun. In much the same way, it can help protect your cells from damage. Tomatoes also have potassium, vitamins B and E, and other nutrients.

Based on annual crop production, tomatoes can be grouped into two: namely, determinant and indeterminant with more than 10,000 species. It has hairy leaves that vary in size and colour; like red, pink and yellow. The fruit is tomato's most important component. Botanically speaking, the fruit is a berry with two or more curved surfaces that soften as the fruit ripens as the seeds are fully formed (Nell, Wessels, Mokoka, & Machedi, 2006).

Mkhabela (2005) has it that a lot of novel tomato cultivars are frequently grown in greenhouses and in milder areas, like Roma VF, where they can be grown from winter through spring and summer. The maturation period ranges from 120 to 150 days. The quality of the fruit, adaptability, diseases, pests, plant growth habits, the market, and planting time all factor into the selection of the cultivar.

Again, Bhowmik, Kumar, Paswan and Srivastava (2012), believe that tomato fruits can be used in a variety of ways. Fresh tomatoes are eaten raw, processed into purees, pastes, powders, ketchup (tomato sauce), soup, or whole fruits canned. Tomato extracts have been utilized in traditional medicine to treat a variety of diseases according to research. According to the production guidelines for tomato as released in 2001, it is believed that tomato poultice has been used to treat edema in pregnant women in Japan, Greece, Peru and Guatemala while hot water extract of dried fruits has been used to heal ulcers, hemorrhoids, and burns. Report has also shown that fresh tomato fruits can improve

digestion and alleviate liver and renal problems. Additionally, the fruit is a good source of vitamin C and lycopene (an organic molecule that is a red carotenoid pigment present in tomato fruit).

As reported by Kheyrodin and Kheyrodin (2017), tomatoes have great nutritional value and are a good source of vitamins A and B which are essential for human diet. In many ways, the crop has significant economic worth especially in terms of disease-resistance which has positive attendant production output.

There is a profitable income from the sales of tomatoes, according to Oladejo and Laudia, (2011). Tomato economy has not only increased the agricultural sector's part of the national economy but also represents one of Nigeria's most promising sectors with a competitive advantage in the liberalized market. According to Mari (2009), tomato fruit is one agricultural produce that has an all-year-round demand for most homes globally.

However, tomato production poses numerous threats related to pests and diseases. There has been 80-100% depreciation in tomatoes production due to disease outbreak (Baudoin et al., 2017; Rovesti et al., 2019). Pest and diseases have negatively affected production success of small-scale farming. Pest and diseases are prominent factors that have contributed to substantial losses based on high mortality in crops and reduced productivity. For example, in accordance with Premium Times Newspapers of April 2020, *Tuta absoluta*, a highly destructive pest that impacts tomato cultivation significantly, resurfaced in Kaduna and Katsina States of Nigeria. According to Izuaka (2020), it resulted in approximately an 80 percent reduction in tomato production within the country leading to a substantial surge in the cost of this essential vegetable across the nation. Pests and diseases are major contributors to about 50 percent to total crop failure (Rovesti, Haile, & Saunyama, 2019). As a result, critical measures need to be taken to reduce this negative impact and improve the productivity of small-scale farmers.

In addition, early detection of plant diseases is a crucial management method in prevention of large-scale damage to crops. This proactive action of early treatment will prevent spread of the diseases or advancing of the diseases to untreatable levels (Buja et al., 2021). Most small-scale farmers depend on experience and human observation in identification of crop infestation. While effective, these methods are most of the time inaccurate or characterized by misinformation of the actual problem.

Advancement in technology has provided an opportunity for adoption of better crop diseases management practices. Among these methods is the integration of machine and deep learning models with many devices which are deployable in real-time. Early classification and detection of the various classes of diseases affecting tomato can be identified and can help farmers to have favourable yields thereby boosting sales.

1.2 Problem Statement

Crop diseases continue to invade and spread across the globe as the world becomes more interconnected. While diseases continue to pose serious threats to our biological community of interacting organisms and the physical environment, they may be an even bigger threat to our agriculture and food security (Sundström et al., 2014). The economic base of most peasant farmers is often in one way or another associated with vegetable production. However, the lack of necessary tools has had a negative impact, particularly in certain areas of the country's geopolitical regions where farmers are increasingly concerned about early tomato disease detection. Diseases attack tomato plants every year which reduce the number of tomatoes produced in Nigeria. Tomatoes are one of the most common vegetables used in every kitchen. If the diseases are not detected in a timely manner, they may have a detrimental impact on the overall production and lower farmers' profits. For instance, in late 2014, Nigeria declared an emergency due to the emergence of

a moth that had devastated more than 80% of the tomato crops in Kaduna State leading to scarcity and high cost of tomatoes across the country (Vanguard-Newspaper, 2016).

It has been observed that past research solutions on tomato disease prediction use computational techniques focused more on enhancing precision of the models based on detection while neglecting the need for possible recommendation after detection. The goal of plant disease management is to reduce the economic and aesthetic damage caused by plant diseases. An undocumented empirical study shows how out of 200 research papers on tomato, more than 80% of them did not offer management recommendations to farmers; they dwell only on leaf disease detection element.

In addition, certain new technology, like unmanned aerial vehicle (UAVs) machines, which have been found to provide near accurate results in terms of monitoring tomato plant, are exceedingly expensive for most farmers to afford (*Kim, Kim, Ju, Son, & Member, 2019*).

Therefore, it is crucial to explore quick, inexpensive, and accurate ways to identify tomato leaf disease situations with active recommendation measures incorporated. Tomato leaf disease detection using combination of deep learning techniques can offer hints for early disease identification and also recommend possible treatment. The current investigation is necessary because of the research gap in the previous studies.

1.3 Objectives of the Study

The main objective of this work is to develop an intelligent hybrid model for tomato leaf disease detection and control recommendations.

. The specific objectives are to:

- 1) gather large dataset from online repository and from field work for training and testing the proposed model.

- 2) extract features, detect, classify and multiclass tomato leaf diseases using Machine learning techniques.
- 3) develop a content-based recommendation system for providing control measures to tomato disease.
- 4) Conduct a comprehensive experiment that will evaluate the performance of the proposed tomato disease classification system.
- 5) Deploy the prototype recommender system on mobile devices and to evaluate its usability in real-life farming scenario.

1.4 Research Questions of the Study

- 1) What are the datasets from both online repositories and fieldwork which can be integrated and pre-processed for training, testing, and evaluation purposes?
- 2) Can the combination of MobileNetV2 and Support Vector Machine (SVM) techniques be effectively utilized to classify and predict multiclass tomato leaf diseases?
- 3) How can a content-based recommendation system be developed to provide control measures for tomato diseases?
- 4) How does the proposed tomato disease classification system compare to existing methods in terms of accuracy, precision and recall?
- 5) How does the usability of the prototype recommender system on mobile devices compare to traditional farming decision-making methods in real-life farming scenarios?

1.5 Justification of the Study

- i. **Rising Concerns in Tomato Cultivation:** Tomato cultivation plays a significant role in global food production and consumption. However, tomato plants are

susceptible to various diseases that can result in significant yield losses and quality deterioration. In recent years, there has been a growing concern about the impact of tomato leaf diseases on agricultural productivity. Therefore, the development of an intelligent hybrid model for tomato leaf disease detection and control recommendations is justified due to its potential to mitigate these concerns.

- ii. **Accurate and Timely Disease Detection:** Traditional methods of disease detection in tomato plants are often manual and time-consuming. Visual inspection by human experts may lead to subjective judgments and can be limited by human error. By utilizing an intelligent hybrid model, incorporating machine learning and image processing techniques, there is enhanced accuracy and efficiency of disease detection. This model can rapidly analyze large volumes of leaf images, accurately identifying the presence and severity of diseases, thereby enabling timely intervention.
- iii. **Improved Disease Control Recommendations:** Once a disease is detected, effective control measures need to be implemented promptly to prevent its spread and minimize crop damage. The intelligent hybrid model can not only detect diseases but also provide tailored control recommendations based on the identified disease type and severity. By integrating disease databases, historical data, and expert knowledge, the model can suggest appropriate treatment methods, such as specific fungicides, cultural practices, or integrated pest management strategies. This ensures that farmers receive accurate and targeted advice, leading to better disease management outcomes.
- iv. **Enhanced Efficiency and Resource Optimization:** The utilization of an intelligent hybrid model presents a noteworthy benefit concerning resource efficiency, through its capability to identify diseases in their early stages and offer precise control

suggestions, it diminishes the need for widespread chemical treatments. This targeted approach minimizes the unnecessary use of pesticides, thereby reducing environmental pollution and promoting sustainable agriculture. Additionally, by enabling timely intervention, the model helps prevent disease outbreaks and mitigate yield losses, resulting in improved overall crop productivity.

- v. **Potential for Scalability and Accessibility:** With the advancements in technology, intelligent hybrid models can be deployed on various platforms, including mobile devices and web applications. This enables easy accessibility for farmers, agronomists, and extension workers, even in remote areas with limited resources. By providing a user-friendly interface and real-time disease detection capabilities, the model empowers stakeholders at different levels of expertise to make informed decisions and take appropriate actions, ultimately benefiting the tomato farming community.

1.6 Scope of the study

The scope of study encompasses a comprehensive investigation into developing an integrated solution for identifying and categorizing diseases in tomato plants. The study aims to leverage the MobileNetV2 convolutional neural network architecture combined with Support Vector Machines (SVM) to build an efficient and accurate model for automated tomato disease detection. By employing deep learning techniques, the research intends to analyze and process images of diseased tomato plants got from online repository (Kaggle), enabling precise identification of various diseases such as blight, wilts, spots, and viruses etc.

Additionally, the study involves the implementation of a recommender system as a control measure to provide personalized and targeted recommendations for disease management. This system will offer farmers and agricultural experts tailored advice on appropriate

control measures and treatments based on the specific disease detected in their tomato crops. The ultimate goal of this research is to create an application and user-friendly solution that can aid in early disease detection, leading to timely interventions and effective control strategies, thereby enhancing tomato crop yield and reducing losses caused by diseases. The use of Unmanned Aerial Vehicles and checking soil texture before planting tomatoes are not included in this work. Different varieties of tomato exhibited varying levels of susceptibility to diseases and displayed different symptoms. Lastly, from some established literature, the types and prevalence of tomato diseases fluctuate from year to year and region to region. These factors were not taken into consideration during the research planning process.

CHAPTER TWO

LITERATURE REVIEW

2.1 Conceptual Framework

2.1.1 Historical Overview in the Area of Study

Agriculture plays a crucial role in the economy of some nations, serving as a primary source of income and food production for a continuously growing population. Plants hold a significant position in the society. In 2015, the United Nations (UN) introduced the 17 Sustainable Development Goals (SDGs), aiming to achieve zero hunger (SDG). Despite numerous efforts made over the past decades to formulate strategies and policies for food sustainability, the reality is that today, about one in ten people globally still face food scarcity issues (FAO et al., 2020). Agricultural productivity is a pivotal sector in many national economies, underscoring the vital importance of both food crops and cash crops for the well-being of both the environment and humanity. Each year, various crops, such as tomatoes, fall victim to numerous diseases. Plant diseases have perpetually posed a contentious challenge in agricultural production and stand as a primary impediment to the sustainable advancement of agriculture.

Due to incorrect diagnosis of crop diseases resulting from a lack of understanding of their symptoms and delayed treatments, numerous plants may perish. Plants hold significant importance as they serve as a vital energy source for humans. The growth of tomatoes is influenced by a range of environmental factors, leading to a heightened occurrence of diseases. According to current statistical data, there are over 200 different types of tomato diseases, which have had a severe impact on both the quantity and quality of tomato yields, resulting in substantial economic losses. Tomatoes are a widely cultivated vegetable and a significant cash crop in China, with cultivation spanning nearly 700

million square meters nationwide. Consequently, the prevention and treatment of tomato diseases play an exceedingly critical role in tomato production (Wang, Qi, Sun, Qu, & Xue, 2019)."Tomato stands out as one of the most extensively grown vegetables worldwide. In tropical nations, cash crops serve as the primary source of sustenance for many impoverished farmers (Cocaliadis, Fernández-Muoz, Pons, Orzaez, & Granell, 2014). Although it naturally thrives as a perennial plant, it is commonly cultivated as an annual crop in temperate regions. Tomato, scientifically known as *Lycopersicon esculentum* Mill., belongs to the Solanaceae family, the same botanical family as eggplant, pepper, and potato. This versatile foodstuff is believed to have originated in Central and South America and was first used as a culinary ingredient in Mexico. Its popularity spread worldwide following the Spanish colonization of the Americas (Smith, 1994).

The tomato offers a versatile culinary experience, as it can be enjoyed in a variety of ways, whether consumed fresh, used as a component in numerous recipes, incorporated into sauces, tossed into salads, or blended into beverages. This fruit is notably abundant in essential vitamins and minerals, making it a prized ingredient in a wide range of dishes. Economically speaking, tomatoes hold significant importance for many countries, providing valuable resources. According to the Linnaean taxonomic classification, the botanical name for the tomato is *Solanum lycopersicum*. Alternatively, in a study by Gulzar et al. in 2021, an alternative genus name, *Lycopersicon* (meaning "Latin-Wolf Peach"), was proposed, along with the suggestion of using *Lycopersicon esculentum* for cultivated tomatoes and *Lycopersicon pimpinellifolium* for native varieties.

2.1.2 Origin of Tomato

The Aztecs are credited with the tomato's initial cultivation around 700 A.D., a fact highlighted by Mehta (2019), which suggests its indigenous presence in the Americas. However, it wasn't until the 16th century that Europeans were introduced to this fruit

during their early explorations of uncharted lands. While Southern Europe readily adopted the tomato, it encountered resistance as it moved northward. The British, for instance, admired its appearance but harbored suspicions of its toxicity due to its resemblance to the wolf peach. This skepticism largely stemmed from the use of high-lead pewter utensils among the affluent, which resulted in lead poisoning when used with acidic foods like tomatoes. Conversely, those less fortunate who consumed meals from wooden dishes did not experience this issue and thus didn't share the aversion to tomatoes.

During the 18th century, there was a notable shift in how people perceived tomatoes. This transformation in attitude can be attributed to several factors, such as the mass migration of Europeans to America, where various cultures converged and embraced the tomato, particularly among Italian-Americans. Additionally, the emergence of pizza, a dish reliant on tomato sauce, played a pivotal role in popularizing the tomato. The birth of Pizza Margherita in Naples during the late 1880s further celebrated this shift and paid homage to Queen Margherite's visit by incorporating the colors of the new Italian flag into its ingredients: utilizing red tomato sauce, white mozzarella cheese, and green basil.

The tomato has established itself as a common ingredient in kitchens worldwide, and different regions have their unique culinary traditions associated with it. In the United States, more than 12 million tons of tomatoes are consumed annually. There has been an ongoing debate about whether tomatoes should be classified as fruits or vegetables. Botanists categorize them as fruits because they possess a fleshy structure that encases seeds, while horticulturists consider them to be plants with edible parts used in cooking. Interestingly, until the late 1800s, tomatoes were officially recognized as fruits to avoid certain taxes, but a Supreme Court ruling eventually declared them as vegetables (Mehta, 2017).

2.1.3 Planting Season, Soil Type, Vegetation, and Climate that Support Tomato Growth

Tomato is part of the Solanaceae family. The cultivated tomato is classified as *Solanum lycopersicum*, and its closest wild relative is *Solanum pimpinellifolium*, with about a nucleotide pair variance value of 0.6% (as reported in "The Tomato Genome" publication from 2012). Although the tomato has the capacity to flourish under diverse environmental conditions, it experiences optimal growth in warm temperatures, typically falling within the temperature range of 15°C to 25°C. It's important to note that excessive heat and humidity can lead to a reduction in fruit set and overall yields. Conversely, very low temperatures can delay the development of color and ripening, whereas temperatures exceeding 30°C can hinder not only fruit set but also the formation of lycopene and the development of flavor. To ensure the thriving of tomatoes, it is advisable to cultivate them in regions characterized by low to moderate rainfall, supplemented with irrigation during the off-season. Excessive moisture resulting from wet conditions can increase susceptibility to diseases and negatively impact fruit ripening. Tomatoes exhibit robust growth across a variety of soil types, provided they are rich in organic matter, well-drained, and maintain a pH level within the range of 5 to 7.5 (Waiganjo et al., 2006).

Tomato plants thrive in soil that is well-drained and rich in organic matter, with a good capacity to retain moisture. Ideal elevation for their growth falls within the range of 1000 to 2000 meters above sea level (Robert, 2005). These plants typically reach heights of 1 to 3 meters and feature a delicate stem that often sprawls across the ground and intertwines with other plants. While they are native perennials, they are commonly cultivated in temperate climates as annuals. The time required for tomato harvest varies, ranging from 60 to over 100 days, depending on the specific variety. In comparison to domesticated varieties, wild tomatoes are generally smaller in size (Bergougnoux, 2014).

2.1.4 Nutritional Value of the Tomato

Tomato a well-known global vegetable food has the largest production and consumption (FAO, 2016). According to USDA(2016), reported that a person consumes about 6kg of tomatoes, which makes it the second largest consumable vegetable, reports from the United States show tomato as the second most consumed fresh vegetable with 6 kg/person in 2017. The vegetable known as tomatoes is a highly nutritious food source rich in essential vitamins and minerals, including Vitamin A, Vitamin C, β carotene, and folate. Notably, tomatoes are abundant in key minerals such as potassium, lycopene, along with various phytonutrients like flavonoids, phytosterols, and polyphenols (Villanueva, 2018; Luthria, Mukhopadhyay, & Krizek, 2006). It is estimated that a 100 g of fresh tomato constitutes a 46%, 8% and 3.4% of the daily rations of vitamin A (being 900 UE), vitamin C (being 82.5 mg) and potassium (3500 mg), respectively (Gebhardt., 2002), (Kirstie et al., 2005). Tomatoes are further processed by industries into products like juice, paste, etc., that are also nutritious to human health (Bergougnoux, 2014).

2.1.5 The Tomato's Economic and Social Significance

Tomatoes are cultivated in various regions across the globe, including temperate, subtropical, and tropical areas (Blanca, Caizares, Cordero, Pascual, Diez, & Nuez, 2012). In terms of yield, it ranks as the second most prominent horticultural crop worldwide (FAO, 2016). According to data from 2016, the countries with the largest tomato production areas were the United States, China, India, and Turkey (FAO, 2016; USDA, 2016). In 2013, global fresh tomato production reached 163.719.357 tons, with approximately 4.5% of the tomatoes being traded. Mexico, the Netherlands, and Jordan were the leading tomato exporters during that year, accounting for 20%, 13%, and 8% of their respective tomato production. The revenue generated from tomato exports in these countries in the same year was \$1195, \$1675, and \$517 per ton (FAOSTAT, 2013).

Meanwhile, Russia, Germany, and France were the primary importers of tomatoes, accounting for 12%, 10.3%, and 7.8% of total imports, respectively (FAOSTAT, 2013).

Tomatoes are grown both in open fields and under controlled conditions during the autumn and winter seasons. Even in colder climates such as the Scandinavian countries, tomatoes are successfully cultivated. Controlled conditions involve the use of artificial lighting, heating systems to elevate temperatures, and fertilization techniques to ensure the production of high-quality tomatoes (Oda & Saito, 2006).

Tomato cultivation comes with significant costs (United States Department of Agriculture [USDA], 2016). These expenses kick off with the utilization of dependable seedlings (Tüzel & Öztekin, 2017). Additionally, farmers have the option to selectively breed crops, focusing on desirable attributes that can enhance production yields and bolster resistance against diseases. Breeding and seed production efforts are geared towards meeting growers' preferences, encompassing traits like high yield and disease resistance. These choices are also influenced by the intended use and consumer preferences (Carbonell et al., 2018; Yamamoto et al., 2016). Throughout the production process, modern technologies are employed to optimize both yield and product quality, while simultaneously reducing the reliance on pesticides. Techniques such as using seedbeds, mulch, branch management, pruning, drip irrigation, and the application of specific macro and micronutrients at various growth stages of tomatoes all play a role in realizing the maximum potential yield achievable with certified seeds (Tüzel & Öztekin, 2017). In the case of open-field production, mechanical harvesting is employed for tomatoes intended for industrial purposes, while handpicking is the method of choice for those destined for fresh consumption.

Notably, fresh tomato prices are observed to be more susceptible to fluctuations compared to processed tomatoes, with their pricing closely tied to the shipping-point price. It has

been estimated that a quarter of the final tomato price is attributable to the costs incurred in the shipping chain (USDA, 2016).

2.1.6 Diseases in Tomatoes

Tomatoes, scientifically known as *Solanum lycopersicum*, have the capability to thrive in a wide range of moderately well-drained soil types. The presence of an ample quantity of organic matter can lead to an increase in yield while simultaneously mitigating production-related issues. It is advisable to avoid planting tomatoes and related vegetables like potatoes, peppers, and eggplants in the same soil for at least three years consecutively. Ideally, any cover crop or the crop preceding the cultivation of tomatoes should belong to the grass family. Corn, which serves as an excellent rotational crop when paired with tomatoes, provides substantial organic matter without fostering the growth of disease-causing organisms that can harm tomatoes. In the context of sourcing seeds and plants, it is highly recommended to opt for certified varieties whenever feasible (Zachary, 2015).

Diseases pose a significant constraint on tomato production, and they can be categorized into two distinct groups. The first category encompasses diseases induced by infectious microorganisms, encompassing fungi, bacteria, viruses, and nematodes. These diseases are communicable and have the capacity to rapidly transmit from one plant to another within a field, particularly under favorable environmental conditions. On the other hand, the second group comprises diseases triggered by non-infectious factors of a physical or chemical nature, such as adverse environmental conditions, nutritional or physiological imbalances, and herbicide-induced injuries. Noninfectious diseases do not possess the ability to propagate from one plant to another, but they can exhibit uniform and extensive distribution if an entire planting is exposed to the adverse factor. It is of paramount importance for effective disease management to distinguish between infectious and

noninfectious diseases and to ascertain the specific type of microorganism responsible for infectious diseases.

Numerous tomato diseases and disorders result in the development of leaf spots and fruit rots. In general, these ailments do not result in plant fatalities, but they can result in substantial reductions in crop yield and quality. Numerous microorganisms responsible for these diseases can persist in plant remnants, on seeds, or within the soil. Limited knowledge among growers regarding tomato diseases has a detrimental effect on production. Consequently, the potential for disease outbreaks exists at any given time. These diseases encompass:

(A) Damping Off Tomato Disease (*Pythium aphanidermatum*):

This is one of the worst diseases of tomato occurring in the nursery. Damping off is a major disease of young tomato seedlings and losses can reach 80 - 100%. The disease can be caused by several different pathogens (*Pythium*, *Phytophthora* and *Rhizoctonia* species) but the same symptoms occur. The seedlings either fail to emerge from the soil or fall over or die immediately after emergence.

(i) Symptoms of Damping Off (*Pythium aphanidermatum*)

Damping-off symptoms can be detected from the time of sowing until approximately the fourth to sixth week after planting (Horst, 2013). The manifestations of this disease can be categorized into two phases based on when they become apparent. Pre-emergence symptoms occur when seeds deteriorate before seedlings emerge. This can happen either (i) before the seeds even germinate or (ii) when germinating seeds face biotic stresses while their shoot tissues are still beneath the soil surface (Horst, 2013). In the first scenario, the seeds become soft, decayed, and fail to undergo germination. In the second scenario, the stems of germinating seeds exhibit characteristic water-soaked lesions, which form at or below the soil line. On the other hand, post-emergence damping-off symptoms

manifest when seedlings decay, wilt, and ultimately perish after emerging from the soil (Horst, 2013). In most instances, these symptoms lead to the collapse and death of at least some seedlings within a given population. In cases involving soil-borne pathogens, seedling death can occur in clustered patches with a roughly circular pattern, and the affected seedlings may exhibit stem lesions at ground level.

Control of Damping Off (*Pythium aphanidermatum*)

Thoroughly preparing the field and seedbed, along with effective water management, can greatly minimize damping-off losses. Employing sprinklers during germination provides enhanced water control and reduces the risk of infection. Whenever feasible, refrain from planting when the soil is cold, as warmer soil promotes faster seed germination and more robust seedlings, reducing vulnerability to damage. The application of fungicidal treatments to the seeds can also be beneficial in preventing damping-off. In cases of substantial seedling loss, it may be essential to consider replanting. Seed treatment with fungal culture *Trichoderma viride* (4 g/kg of seed) or Thiram (3 g/kg of seed) is the only preventive measure to control the pre-emergence damping off. Soil drenching of the affected seedlings with Dithane M45 (3 g/litre of water) helps to reduce the disease incidence.

(B) Early Blight Tomato Disease (*Alternaria solani*)

The fungal pathogen *Alternaria solani* is responsible for the development of early blight disease in tomatoes, both in open fields and greenhouse environments. One prominent symptom of this disease is the visible damage to the tomato plants' leaves, which occurs regardless of weather conditions and leads to a reduction in both the quantity and quality of the harvested produce. This fungus primarily targets the plant's foliage, resulting in distinctive leaf spots and blight. Initially, early blight manifests as small, dark lesions, primarily appearing on the older leaves. As these spots grow larger, concentric rings

resembling a bull's eye pattern emerge at the center of the affected area, often accompanied by yellowing of the surrounding tissue.

In conditions of high temperature and humidity, a significant portion of the foliage can be killed by the fungus. Stem lesions exhibit similar characteristics to those found on leaves and can sometimes encircle the plant's stem, potentially causing damage near the soil line. Transplants that are infected by the late blight fungus often fail to thrive when planted in the field. Furthermore, this fungus can also infiltrate the fruit, typically through the calyx or stem attachment points. The resulting lesions can grow to considerable sizes, often affecting nearly the entire fruit, and they too display concentric rings as a characteristic feature.

(i) Symptoms of Early Blight (*Alternaria solani*)

Tomato plants can manifest signs of early blight at any stage of their growth. Typically, the initial infection occurs on older leaves closer to the ground, showing small lesions that measure between one-fourth to one-half inch in diameter, appearing brown or black at their centers. These lesions often develop distinctive "bullseye" rings, and you may notice yellowing of the surrounding leaf tissue. In the case of tomato stems, infected areas display lesions with a light-colored center that is dark on the outer edges. These stem lesions can sometimes encircle the stem entirely, leading to collar rot and even damping off in young seedlings.

As early blight progresses, leaf yellowing and leaf loss move upward toward the newer growth of the plant. On tomato fruits, the lesions are dark brown and have a leathery texture. Over time, these lesions may encompass the entire fruit and exhibit signs of velvety spores. You might also observe fruit lesions near the stem attachment and on the fruit's shoulder, which could be mistaken for or associated with fruit cracking. This fungus

can attack all parts of the plant, including stems, leaves, and fruit, and it rapidly forms yellow-colored lesions that initially appear as tiny brown dots, covering about 1/16 inch of the leaf's surface in the early stages. These symptoms progress from circular brown lesions to circular black lesions with a diameter ranging from 1/3 to 1/4 inch. Lesions on the leaves can merge, leading to leaf defoliation, and stem infections may also spread and worsen over time.



Figure 2.1: Tomato early blight (Melanson , 2017)

(ii) Control of Early Blight (*Alternaria solani*)

Devastating effect of early blight disease can be managed when crop rotation is done at interval of 2 to 3 years. The plantation of variates of crop that grow well in their best seasons is also a recommend measure of early blight. The farmer should ensure all weeds are discarded from the farmland in early stage. Quality air circulation is a factor that avoids spread of early blight, whereby the staking of tomatoes will assist to expose the plant to air. Application of fungicides on plants also mitigates the fast spread or maturity.

(C) The Bacterial canker (*Clavibacter michiganensis* subsp. *michiganensis*)

Bacterial canker disease is a prominent diseases outbreak commonly found in greenhouse agriculture. The pathogen mostly attacks the seed and spreads rapidly in plantation of dead leaves and weeds surroundings. Thera is also the possibility that pathogens can transferred to plants based on contact with farm workers to plants, especially in a moist conditioned during pruning activities.

(i) Symptoms of Bacterial Canker Tomato Disease

Some tomatoes infected by the bacterial canker pathogen exhibit varieties of symptoms, whereby wilting is one of the identifiable symptoms. Wilting disease mostly attacks plants that have cuts, as an entry point for pathogens which transit gradually to all parts of the plant. Stems tend to develop dark streaks which split and exposed to brown canker disease that is normally shows a dry necrotic lesion. The colour variation starts from light yellow to brown streaks, which eventually turn reddish-brown. The colouration develops in vascular tissues and are more obviously visible at plant nodes. In some plants, noticeable roots develop from the stems. The margins of the leaflets will likely change colour to brow to yellow, which forms margins as represented in Figure 2.2.



Figure 2.2: Bacterial canker tomato disease with marginal burning on tomato leaflets and bird's eye spots on tomato fruits (Melanson, 2017) Hansen., 1942 Bugwood.org)

(ii) Control of Bacterial canker

The control or management of Bacterial canker involves steps such as purifying the plants used, treating the seeds, maintaining cleanliness, controlling weed growth, and alternating crops.

(D) Bacterial speck tomato disease (bacterium: *Pseudomonas syringae* pv *tomato*)

Under favourable condition, the bacterial speck thrives very well with respect to disease infestations which becomes difficult to control when noticed late. A moist and cool environment favors the growth of the disease. The cause of the disease is present in the seeds and can also be dispersed through the splash of water, the use of contaminated tools and machinery, and by individuals. The pathogen survives all year round in crop debris, whenever conditions are favourable. Pathogen resistance to copper has been reported; however, there is no certainty whether the resistant pathogen populations are present in Mississippi.

(i) Symptoms of bacterial speck

This is a round like symptoms characterized by dark brown to black lesions. that appears on the tomato leaflets. Afterwards, a yellow halo may develop around lesions (See Figure 2.3).

Large portion areas of the leaflets may perish whenever lesions coalesce (grow together). Lesions may also form on stems, petioles, peduncles, pedicels, and sepals; these lesions are often elongated. Dark lesions also develop on fruit which can be characterized have a dark green halo around the leaf. Notably there are similarities in symptoms of bacterial speck and bacterial spot but are different diseases related to different bacteria infestations.



Figure 2.3: speck bacterial infect tomatoes leaflet and petiole with brown lesions (Holmes, Ekman, and Lovatt, 2015, Bugwood.org.)

(ii) Management and control of bacterial speck

The management and control of bacterial speck tomatoes diseases includes planting of a seed, not infested by pathogen, proper treatment of seed, sanitation, cultural practices, agricultural crop rotation and application of fungicides.

(E) Bacterial spot tomato disease (bacteria: *Xanthomonas* spp)

Bacterial spot outbreak of disease can extremely spread in a tomatoes farm in short time, whereby positive results from control measures under favourable weather condition can be challenging. Very importantly, precipitation and warm temperatures are factors that foster bacterial disease development in tomatoes. Human contact with infected plants and weather factors like wind and rain aids the spread of pathogen diseases in plants. These pathogens can affect the entire plant parts from bottom to the top.

(i) Symptoms of bacterial spot

The bacterial spot disease in tomato has a feature of circular brown lesions on its leaf tissues as shown in Figure 2.4. The lesions are characterized by a joined dark streak that sticks to the affected tissues. Leaves with many lesions may develop a yellow appearance. Leaves on which lesions coalesce may become blighted (rapid, extensive damage or

destruction). Lifeless foliage may also be present on the plant. Lesions on leaflets may also grow a shoot-hole appearance in which the centre of the lesion falls out of the leaflet. On fruit, lesions appear as raised blisters that become brown and scabby.



Figure 2.4: Sikora (2020), Lesions on tomato leaves with bacterial spot (Melanson et al., 2020)

(ii) Management and control of bacterial spot

The management and control of bacterial spot tomatoes disease includes pathogen-free seed, seed treatment, cultural practices, sanitation, crop rotation, fungicides.

(F) Bacterial wilt tomato disease (bacterium: *Ralstonia solanacearum*)

Tomato bacterial wilt is a severe disease that affects both outdoor and indoor tomato plants. The disease thrives in warm soil conditions and high humidity levels. The bacteria, which can infect multiple crops besides tomatoes, can persist in the soil for an extended period even when no susceptible crops are present. Difficulty to control bacterial wilt can be experienced whenever the pathogen has been a long habitant in the field. Additionally, the bacterium can spread through polluted tools, usage of farm equipment and human contact. It's worth mentioning that the bacterium *R. solanacearum* is categorised into five races, and the race that usually infects tomatoes is Early blight, a common disease caused

by the fungi *Alternaria solani* and *tomatophila*. Early blight affects the leaves, stems, and fruits of the plant, starting at the bottom and moving upwards. The pathogen thrives in moist and mild temperatures and is seed-borne, able to survive on infected plant debris.

(i) **Symptoms of bacterial wilt**

Bacterial wilt is characterized by circular or oblong brown lesions that form strong concentric rings that grow on infected plant tissue and broadens with time (see Figure 2.5). A chlorotic feature manifests on the body of the leaf tissue, fruits, stem and plant.



Figure 2.5: A stem infected with bacterialwilt (Melansonet al., 2020)

(ii) **Management and control of bacterial wilt**

The management of bacterial wilt involves crop rotational practice, plantation of uncontaminated soil, mulching, fungicide application and staking.

(G) Buckeyerot(oomycetes: *Phytophthora nicotianae* var. *parasitica*, *P. capsici*, and *P. drechsleri*)

Buckeye rot is a common disease of tomatoes in the South-eastern United States. High humidity and warm temperatures favor the disease development. The disease is most severe in soils with plenty of moisture. The pathogen can be spread by contaminated or splashing water.

(i) **Symptoms of buckeye rot disease**

The infected fruit develops a brownish-oil lesion as represented in Figure 2.6. The portion area of the lesion's spreads to other area of the fruit. A white cottony fungal grows on lesion fruit at quick rate in a moisture conditioned environment. Most fruit that are closer to ground soil are easily infected. These low-hanging fruit may also get infected due to infested soil that have been contaminated with pathogen, splashed on the fruit plant.

(ii) **Management of buckeye rot disease**

The management of buckeye rot disease can be controlled by growing plant in clean environment, rotational farming, mulching, spraying fungicides and staking.



Figure 2.6: A buckeye rot infected tomatoes. Characteristic: Concentric rings appear in the lesions(Melanson et al., 2020)

(H) The early blight disease (fungus: *Alternaria linariae* [= *A. tomatophila*])

Early blight is a common disease of tomatoes grown in the field. Leaves, stems, and fruits may be affected. Disease typically starts at the bottom of the plant and moves up. High humidity and mild temperatures favor the pathogen. The pathogen is seed borne and can survive on infected plant debris.

(i) Symptoms of Early blight

The symptoms of Early blight tomato disease include Infected plant tissues exhibit circular or elongated brown lesions with circular patterns that form on the surface, which enlarges over time. The body of the leaf develops chlorotic lesions as shown in Figure 2.7. These symptoms of early blight tomato disease can be found on its stem, leaves and fruits.

(ii) Management of Early blight

The management of early blight tomato disease includes proper environmental hygiene, application of fungicides on crop rotation, and mulching,



Figure 2.7: Tomato leaflet infected with early blight disease. (Melanson *et al.*, 2020)

(I) Fusarium wilt (fungus: *Fusarium oxysporum* f. sp. *lycopersici*)

Fusarium wilt diseasespreads speedily during warm-weather seasons. The pathogen related to fusarium wilt is habitant of soil, hence it's a soil borne disease. The disease outbreak is commonly transmitted from infected soul and farm equipment usage. Infection is facilitated by root wounds, which may be caused by root-knot nematodes.Cultivars resistant to one or more races of the fungus are commonly used to manage disease. Other varieties of *Fusarium* are Fusarium crown and root rot, in tomatoes.

(i) Symptoms of fusarium wilt

The wilting and chlorosis of leaves are notable symptoms of *Fusarium* wilt in plants. *Fusarium* wilt disease manifests symptoms at hottest period of the day, which can recuperate at night. Chlorosis colouration of leaves begins from small portion and gradually spread to the entire portion of leaves with time. The vascular tissue in the leaflet's changes colour from dark red to brown coloration.

(iii) Management of fusarium wilt

Crop rotation and sanitation of plantation area are cultural practices that could resist the fusarium wilt disease in tomatoes. Measures of control like alternating crops, proper agricultural techniques, and keeping environments clean.

(J) Gray leaf spot (fungi: *Stemphylium* spp.)

In areas of tomatoes plantation that have numerous species, can be overwhelmed by the gray leaf spot diseases. Wind is an environmental factor that facilitates the spread of pathogens on plants.

(i) Symptoms of gray leaf spot

The symptoms of grey leaf spot disease appear at the surface of leaflets have shape variations of round to oblong lesions at the upper and lower portions. Whenever this disease is not treated, with time these lesions can spread and merge on the leaflets. It is also expected that further cracks can occur on the leaflets which becomes and yellow, brown, which denotes death of plant.

(ii) Management of Gray leaf spot

The management and control of gray leaf spots includes disease-free plants, resistance, and fungicides.

2.2 Theoretical Framework

This section describes a foundational examination of accepted theories that acts as a guide for constructing this researched arguments. It validates the connection between various well known accepted theories and this study.

2.2.1 Mathematical Theory of Communication

Claude Shannon first proposed this hypothesis in the early 20th century. Entropy, a concept introduced by the theory, profoundly altered our environment, and ushered in the "Digital Information Age." The Mathematical Theory of Communication is one of the most well-known and frequently applied information theoretic ideas in deep learning and data sciences. According to Shannon, the "semantic characteristics of data are useless," and the nature and meaning of the data have no bearing on the information's content. Instead, he expressed information's quantification in terms of "uncertainty" and probability distribution.

Additionally, Shannon popularized the term "bit," which he modestly attributed to his colleague John Tukey. This ground-breaking concept not only served as the basis for information theory, but it also created new opportunities for advancement in areas like artificial intelligence. Finally, while Claude Shannon's information theory is not a core component of deep learning, it provides a theoretical framework and mathematical tools that can be applied to various aspects of deep learning, including model optimization, feature selection, regularization, and understanding how neural networks process and transmit information. Researchers have explored these connections to gain insights into the behavior and performance of deep neural networks.

$$H(X) = \sum_{i=1}^n p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) = - \sum_{i=1}^n p(x_i) \log_2 \left(\frac{1}{p(x_i)} \right) \quad (2.1)$$

Shannon Entropy, denoted as $H(X)$, and equation (2.1) it is used in information theory to quantify the amount of uncertainty or surprise associated with a random variable X .

Equation 2.1 denoted by:

$H(X)$: This is the symbol for Shannon Entropy, which measures the entropy or uncertainty of a random variable X . It quantifies how much information is contained in the variable X . Higher entropy values indicate more uncertainty, while lower values indicate less uncertainty.

$\sum_{i=1}^n$: This is the summation symbol, which means you will sum up the following expression for each value of i from 1 to n . In this context, it's used to sum over all possible outcomes or values of the random variable X .

$p(x_i)$: This represents the probability of the i th outcome or value of the random variable X . In other words, $p(x_i)$ is the probability that X takes on the value x_i . You'll sum over these probabilities for all possible values of X .

$\log_2$: This is the logarithm function with base 2. It is used to compute the logarithm of a number to the base 2.

$(1/p(x_i))$: This is the reciprocal of the probability $p(x_i)$. It's used inside the logarithm function.

2.2.2 Matrix theory

In 1850, Arthur Cayley proposed the matrix theory. A large area of mathematics that focuses on the study of matrices is called Linear Algebra, and it has a subset known as matrix theory. It started out as a branch of linear algebra but quickly expanded to cover topics in graph theory, algebra, combinatorics, and statistics. When considered as a matrix, linear equations can be used for a variety of purposes. Numerous fields of engineering and science use matrices and linear algebra, including machine learning, statistics, electrical

circuits, cryptography, probability, operations research, and any area where operations and transformations of a set of data are required (Crilly, 1999).

Convolutional neural networks (CNN), a key approach used in this study endeavor, are built on matrices. The concept of matrices is utilized in both the convolutional and pooling layers. The Conv2D class is employed to execute the convolution operation within a two-dimensional space, contributing to the formation of the convolutional layer in a Convolutional Neural Network (CNN). Put differently, the pooling layer plays a crucial role in diminishing the dimensions of the feature maps, as the kernel (filter) traverses the input image in a two-dimensional space. This process serves to decrease the computational workload of the network and reduces the number of parameters that need to be trained (Wu, 2017).

Matrix theory and deep learning are closely related because matrices are fundamental to the mathematical foundations of deep learning. Deep learning is a subfield of machine learning that focuses on training artificial neural networks to perform tasks like image recognition, natural language processing, and more.

In conclusion matrix theory is fundamental to understanding the mathematical underpinnings of deep learning. Matrices are used to represent and manipulate data, weights, activations, and gradients in neural networks, making them an essential tool in the development and optimization of deep learning models.

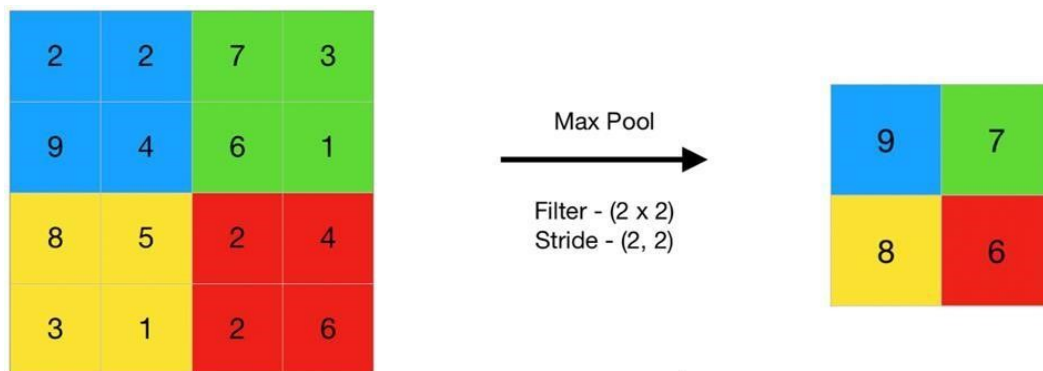


Figure 2.8: Diagram showing the pooling layer operations (Patel, 2019).

2.2.3 Graph theory

In the year 1735, Leonard Euler solved the Seven Bridges of Königsberg problem, laying the groundwork for the development of graph theory. The study of graphs, which are mathematical constructions used to represent pairwise relationships between things, is known as graph theory in mathematics. Vertices (also known as nodes or points) and edges connect the vertices of this context (also called links or lines). Graph theory is used to discover the shortest path between two nodes, which aids in finding the shortest path in a road or network. Graphs are the ideal representation for complex yet discrete phenomena and interactions because of their extensive combinatorial structure.

However, artificial intelligence has overcome some of the classic applications of graph's limitations. Deep learning models excel at handling data that is unidirectional and occupies a fixed amount of space. Graphs are not continuous and do not occupy a constant amount of memory, regardless of how they are represented; rather, each node maintains a reference to the nodes to which it is directly related (Ma & Tang, 2021). Numerous studies that apply graph theory in both machine learning and deep learning have had great success in their research.

Graph theory and deep learning are two separate fields of study, but they are increasingly intertwined as researchers discover ways to leverage the principles and techniques from one field to enhance the other, therefore, the relationship between graph theory and deep learning involves using graph representations to model structured data and developing specialized neural network architectures (e.g., GNNs) to process and learn from this graph-structured data. This integration has opened new possibilities for solving real-world problems where relationships and connections play a crucial role.

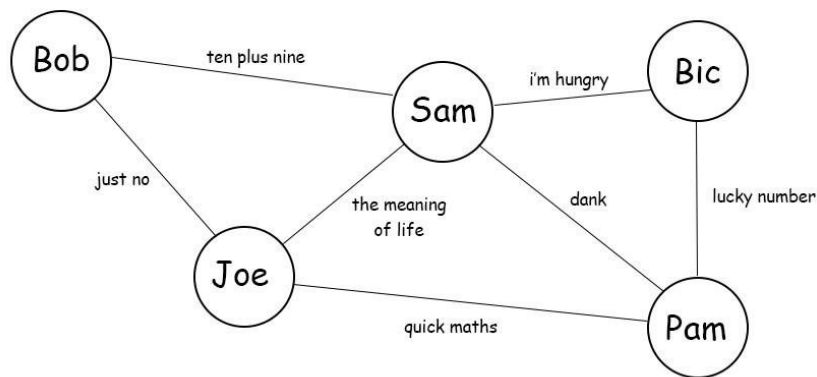


Figure 2.9: Graph showing nodes connectivity (Flawnson, 2019))

2.2.4 Prediction theory

Prediction theory and deep learning are closely related in the context of machine learning and artificial intelligence. Prediction theory, also known as predictive modeling or predictive analytics, is a fundamental concept in machine learning that involves using data to make predictions about future or unknown events. Deep learning, on the other hand, is a subfield of machine learning that focuses on using neural networks with multiple layers (deep neural networks) to automatically learn and represent complex patterns and features from data. It is possible to divide functions dependent on space or time into functions dependent on spatial frequency or temporal frequency according to a technique developed by a French mathematician named Jean-Baptiste Joseph Fourier in the 18th century. The Fourier transform is viewed as a function that can aid in the approximate computation of other functions (Zadeh& Ragazzini, 2012). The layers' calculation can be converted into element-wise products in the frequency domain using the Fourier transform, and the network's task will remain the same. As a result of the foregoing, we can conclude that the Fourier transform, and the convolutional layer are connected. In the time domain, convolutional layers are primarily equivalent to multiplication in the frequency domain. Since polynomial multiplication makes the convolutional layer simple to understand, it

can infer that the convolutional layer itself or its method is connected to the Fourier transform (Cave, 2020).

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(2\pi kt) + b_k \sin(2\pi kt)) \quad (2.2)$$

2.2.5 Theory of reasoning

In 1991, Byrne and Johnson-Laird were the first to introduce the theory of reasoning. According to the argumentative theory of reasoning, deductive inference, reason did not develop to assist us reason alone, but rather to help us reason collectively, or socially. The process of reasoning is typically understood as the act of thinking through a situation to improve our views and choices.

As part of the prioritizing function employed in a traditional reasoning system, theory of reasoning and deep learning can be merged. In other words, by acting on the existing "cognitive state," the consequence operator creates new cognitive states (i.e., logical entailment, or deduction). Like how a deep neural network acts on its own output, a deep neural network would become "recurrent" in the sense that its output would be fed back into its input layer. On the other hand, reasoning is accomplished by creating a loop by connecting the front and end of the deep network. One important concept is to abandon the idea of a "fixed" symbolic structure or knowledge representation (Byrne & Johnson-Laird, 2009).

In the same vein, the relationship between the Theory of Reasoning and Deep Learning is one of complementarity and integration. Researchers and practitioners often seek to combine these approaches to harness the advantages of both symbolic reasoning and deep learning, with the goal of creating more capable and explainable AI systems that can perform complex reasoning tasks while learning from large datasets. This intersection between symbolic AI and neural networks is an exciting area of ongoing research in the field of artificial intelligence.

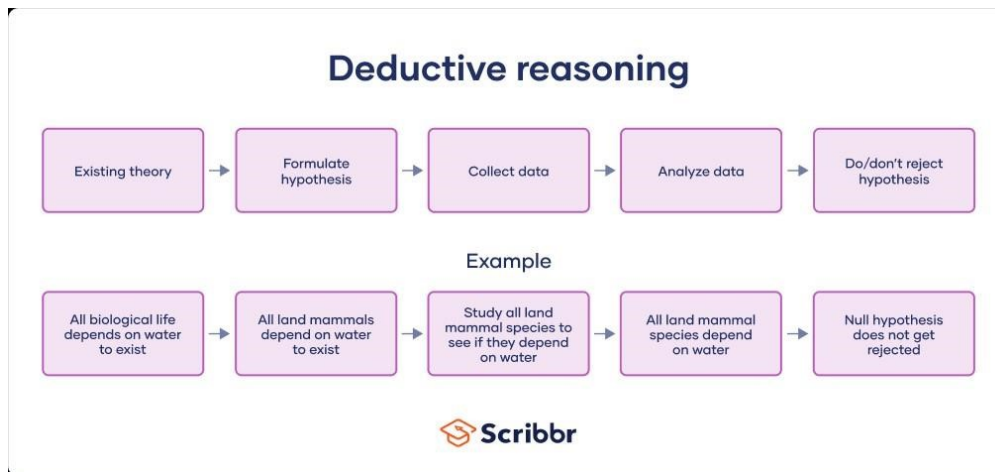


Figure 2.10: Deductive reasoning flow process (Bhandari., 2022)

2.2.6 Diagnostic theory

Diagnostic theory refers to the body of knowledge and principles that underlie the process of diagnosis. It is often used in fields like medicine, psychology, engineering, and various other disciplines where professionals need to identify and understand the causes of problems or conditions. The notion of diagnosis was put forth by (De Kleer, 1993). His strategy is focused on labelling model-artifact discrepancies as violation of assumptions. To avoid wasting valuable time on the incorrect course of treatment, a precise diagnosis is essential. Since analyzing medical data is a difficult cognitive endeavor that forms the basis of every treatment you might receive, from medications to surgery, human error makes correct diagnosis more difficult. The foundation for confirming or disproving symptoms and diseases are diagnostic theories. Your doctor requires specific data from a diagnostic test to make a proper diagnosis before developing a treatment plan. The theory leads to an algorithm for computing all diagnoses, as well as many discoveries about measurement methods for differentiating between competing diagnoses. It also reveals strong linkages between diagnostic reasoning and nonmonotonic reasoning (Reiter, 1986). In summary, the relationship between diagnostic theory and deep learning lies in the application of deep learning techniques to improve the diagnostic process across various

domains. Deep learning can enhance pattern recognition, automate tasks, and assist experts in making more accurate and efficient diagnoses. However, it is essential to carefully consider the challenges and ethical implications associated with the use of deep learning in diagnostics.

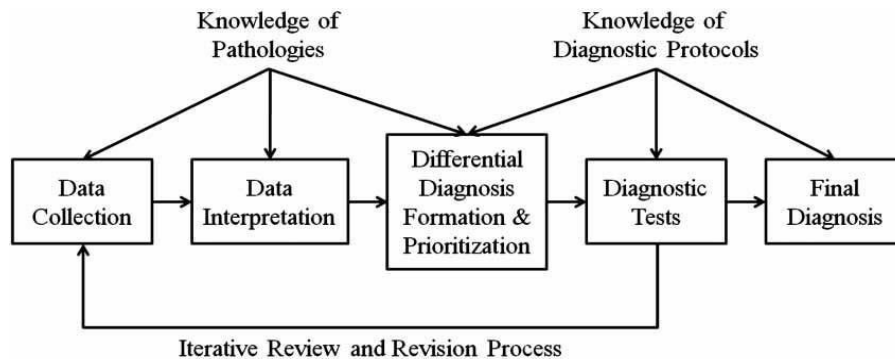


Figure 2.11: Diagram of diagnostic process flow (McFillen et al., 2013)

2.2.7 Network Theory

According to Gordon Bower, who developed network theory in 1981, emotions cause congruent memory fragments to be activated, which are then used to interpret new information as it comes in. Participants experience their environment more optimistically when they are in a good mood. He tries to explain the mood-congruency effect by introducing "The Network Theory of Affect." The significance of emotional state during the learning process is one of this theory's key tenets. Anderson and Bower (1973) hypothesized that thoughts, experiences, and emotions can all be represented as nodes within a network based on the earlier Human Associative Memory (HAM) model. Anderson and Bower's original network type only contained labelled linkages; the nodes themselves lacked semantic labels.

A theory known as network theory is used to analyze graphs that show systems with discrete objects and the relationships between those things. These types of graphs are frequently used in the finance industry, for instance, to comprehend financial flows and the

interconnection of central counterparties. It is possible to use genetic data to infer links between samples by using the study of graphs for linked discrete items. Multi-dimensional data can be seen in depth via network analysis (Singer & Salovey, 1988). In population genetics, when using analytical tools, we frequently rely on a set of presumptions, one of which is that populations develop in the shape of a tree and come from a specific ancestral population, and that we can refer to any other genetic influences as admixture. By incorporating the concepts of reticulate development into data, network theory can help in the investigation of historical demography.

Deep neural networks (DNNs) are frequently referred to as the best models of biological vision because of their extraordinary success in classifying photographic images of objects. DNNs also perform exceptionally well in predicting the pattern of human errors in classifying objects from various behavioral benchmark datasets and in predicting brain signals in response to images from various brain benchmark datasets (Kriegeskorte, 2015).

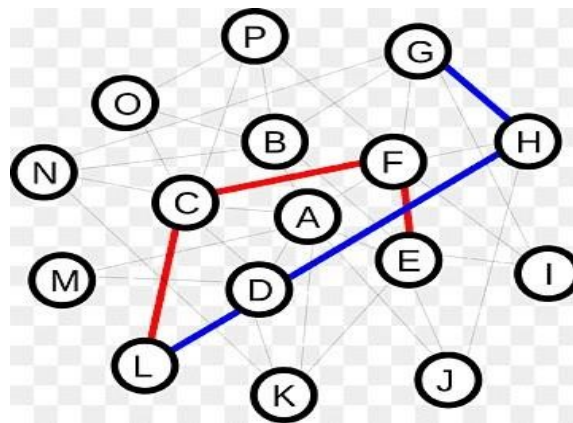


Figure 2.12: Diagram showing network of nodes (John, 1990)

2.2.8 Deep learning theory

Deep Learning (DL) was first mentioned in a work by LeCun, Bengio, &Hinton (2006) that appeared in the journal Science. The fundamental concept of deep learning is the use of neural networks for data analysis and feature learning. Data features are extracted by

multiple hidden layers, each of which can be thought of as a perceptron. Perceptrons are used to extract low-level features, which are then combined to obtain abstract high-level features, which can significantly alleviate the problem of local minimum.

Deep learning has captured the interest of an increasing number of academics because it overcomes the drawback that traditional algorithms rely on artificially created features. It is presently being successfully used in recommendation systems, computer vision, pattern recognition, speech recognition, and natural language processing (Liu et al., 2017). The underlying features of an image can only be extracted using traditional manual image classification and identification methods. In addition to the challenges that extract features from complicated images (Fergus et al., 2012). It is noted that this bottleneck can be eliminated via deep learning. It may do unsupervised learning directly on the original image to gather data on several image feature levels, including low-level, middle, and high-level semantic features. Traditional algorithms for detecting plant diseases and pests primarily use manually constructed features for picture recognition, which is challenging and relies on luck and experience. These algorithms are deficient and incapable of learning and extracting features of image dataset. Instead, deep learning can automatically extract features from enormous amounts of data without manual editing.

The model which has several layers and high feature expression and autonomous learning capabilities, can autonomously extract image features for image recognition and classification. As a result, deep learning has enormous potential in the field of picture recognition of plant diseases and pests. Stacked denoising autoencoder (SDAE), deep belief network (DBN), deep Boltzmann machine (DBM), and deep convolutional neural network (CNN) among other up-to-date deep neural networks (Bengio et al., 2013). Feature extraction functionalities are offered by application of various deep neural networks for image recognition application. The characterization capability of deep neural networks is

also getting better as more training samples are available and as computing power develops. Deep learning is currently experiencing a boom across industry and research, and all deep neural network models perform much better than conventional models (Bayraci & Susuz, 2019). Deep convolutional neural network is the most widely used deep learning framework in recent years.

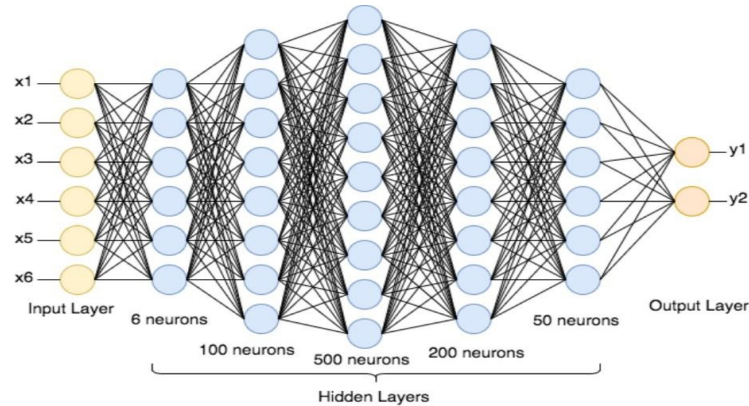


Figure 2.13: Diagram of deep neural Network (Meriem, 2018)

2.3 Empirical Framework

An empirical framework refers to a systematic approach or structure used in conducting empirical research or analysis. It encompasses the overall design, methodology, and procedures employed to collect and analyze data to generate empirical evidence. This section is based on personal observations and assessments of numerous research scholars' works in artificial intelligence for the classification and prediction of tomato leaf diseases.

Chowdhury et al.(2021a) developed a model based on predictive algorithm for spotting diseases on tomato leaves using computer vision approach. The data pre-processing stage employed the U-net segmentation model and data augmentation. The model was also strengthened using the pre-trained EfficientNet model. On Kaggle.com, a repository comprising 18,161 datasets was employed for this study. To mitigate overfitting, the training phase utilized 80% of the dataset, with the remaining 20% reserved for testing and evaluation. The experiment was conducted on a desktop computer equipped with a Xeon®

CPU E52697 v4 @ 2.30GHz, 64 GB of RAM, and a 16 GB NVIDIA GeForce GTX 1080 GPU. The simulation and output display were implemented using Python 3.7 and the PyTorch package. The obtained results revealed a remarkable 99.89% accuracy rate for the model. Further observations highlighted the suboptimal performance of the EfficientNet pre-trained model on hardware accelerators and underscored the absence of effective management measures.

Altuntaş and Kocamaz (2021) built a deep convolutional neural network for the purpose of forecasting tomato leaf diseases with the aid of 18835 datasets from the plant village on Kaggle.com. According to the study, 10% of the dataset was used for testing, and 40% of it was used to train the model. According to the results, ResNet50 with 50 convolution layers had 96.99% accuracy, followed by Alexnet with 92.18% accuracy, and GoogleNet with 89.31% accuracy. Important strategies were adopted that will optimize efficiency of the model, the authors suggested the fusion of several methodologies or methods. But additional observations showed that just 50% of the dataset was utilized, and overfitting could have an impact on the model's overall performance. Once more, no documented procedure for managing diseases and pests was found.

Chowdhury et al., (2021b) created a model for diagnosing tomato leaf diseases using convolutional neural network based on deep learning technique. 18,162 datasets from the Plant Village dataset on Kaggle.com were downloaded and utilized. Based on deep learning's need for huge datasets, which was achievable by application of data augmentation procedure that increases data sets. ResNet, MobileNet, DenseNet201, and InceptionV3 were the other four pre-trained models used in the trials. To improve the model's performance, some of the parameters were tuned using 70% of the datasets for training, 10% for testing, and 10% for evaluation. The PC has 64 GB of RAM, and a 16 GB NVIDIA GeForce ran under the Python 3.7 and the PyTorch library. Results showed

that the proposed model was 98.05% accurate. Additional observations also showed that the proposed model included a feedback system that offers the proper understanding, therapies, disease prevention, and control approaches to increase agricultural yields, but no preventive measures were noted.

Liu and Wang, (2020) used Ubuntu 16.04, Python, OpenCV, CUDA, and Mainboard to create a CNN-based model for forecasting tomato leaf diseases. WS X299 SAGE by Asus 1. Intel Core i7-9800X processor, 16 GB of DDR4 memory, GeForce GTX 1080 Ti graphics card, and 256 GB solid state drive is the configuration of the PC used in conducting the experiment. SSD, Faster R-CNN, and You Only Look Once (YOLO) version 3 were the three pre-trained models used in the proposed model. A total of 1,952 tomato leaves were collected from the field as the primary datasets, but the images from the leaves were of poor quality. This can be because of the capturing tools employed. Additional research revealed that while the Yolo v3 model is capable of learning from data to anticipate bounding boxes, it is less successful at generalizing objects with novel or odd aspect ratios or configurations. Additionally, it suffers with clusters of little things. Results showed that YOLO had a detection time of 20.39ms and an accuracy of 92.39%.

Vinutha and Sandarsh (2020) created a diagnostic model that could categorize 10 tomato diseases based on categorization of leaves images using a CNN-based technique. From the Plant village dataset on Kaggle.com, about 5,000 digital photos were used. The datasets were augmented with data to the extent that was reasonable. For improved outcomes, some of the dataset was additionally subjected to image segmentation. To accomplish the desired outcome and graphical representation or visualization, the python libraries TensorFlow version 2.8.0, NumPy version 1.21.5, and matplotlib version 3.2.2 were utilized. The NVIDIA DGX v100 system was used to run the suggested CNN model. The computer has 128 GB of RAM, 1000 TFLOPS speed, 5120 tensor cores, and 40600

CUDA cores. Three convolutional layers, three maximum pooling, two fully connected layers, RELU as the activation function, and batch normalization were all used in the CNN-based model. By utilizing three separate pre-trained models, VGG16, InceptionV3, and MobileNet, the model used some transferred learning. Results showed that the model had a 91.2% accuracy rate. However, there is no documented control method for the tomato plant disease.

Damicone and Brandenberger (2021) used both convolutional neural network (CNN) and recurrent neural network (RNN) techniques to develop a deep neural network-based model for detecting tomato leaf diseases. 1,500 datasets were used in total, but their sources were not noted. The model was training dataset was captured at 87% and a test aggregate of 13%. The model had an 86.18% accuracy rate, according to the results. Further analysis revealed that the suggested model lacked any management or control mechanisms. Once more, the datasets used are insufficient to implement a deep neural network application.

Based on tomato fruit disease, Wang et al.(2019) also developed a model for the prognosis of tomato diseases. The detection architecture was constructed using a Deep Recurrent and Convolutional Neural Network (D-RCNN). The experiment was conducted on a desktop computer with an Intel Core i7, 3.70 GHz Processor, and NVidia GeForce GTX 1080 Ti GPU, Python was used to create the model. In the convolutional layer of the proposed model, feature extraction was carried out using pre-trained networks including VGG-16, ResNet-50, ResNet-101, and MobileNet. Although 1,430 images were gathered, the sources were not noted. 20% of the dataset was utilized for testing, 20% for model validation, and 60% for training. The proposed model employing Mask R-CNN (ResNet-101) had an accuracy of 99.64% according to the results. Faster R-CNN (ResNet-101) had an accuracy of 88.53% but required a lengthy training period. MobileNet had the quickest detection time but had lower accuracy. The dataset is too small, further investigation

revealed, for the deployment of deep neural network applications. Consequently, the outcomes could not accurately reflect accuracy. Once more, no management actions were noted.

Vijay, (2021) created a hybrid model that combines Deep learning-CNN and Machine learningKNN to predict tomato diseases from the leaf. The Kaggle.com plant village dataset's 10,000 images were downloaded and used. 25% went toward testing the model, and 75% went toward training. In comparison to the CNN model, the KNN model had an accuracy of 83.6%, precision value of 90%, recall value of 84%, and f1-score value of 86%; while the CNN model also had precision value of 93%, recall value of 93%, f1-score of 93%, and accuracy of 98.5%. When the model was further examined, it became clear that crop management was not included. The CNN-based model outperformed the machine learning KNN model in terms of the entire performance index.

Islam et al.(2022) used CNN deep learning technique to forecast tomato leaf disease. The network was trained on a powerful computer with two NVIDIA GeForce RTX 3060 PCI-E GPUs, an Intel Core i9-10850K CPU running at 3.60 GHz (Cores 10), 32GB of DDR4 RAM, and 3200MHz random access memory (12GB, CUDA DeepD381v1). A total of 28,000 tomato leaf digital photos were gathered from the plant factory at Ehime University in Japan and the plant village dataset on Kaggle.com. 20% of the dataset was utilized to test the model, while the remaining 80% was used for training. The model had a 98.7% accuracy rate, a 98.8% sensitivity rate, a 99.8% specificity rate, a 99.8% precision rate, a f1-score of 98.8%, a 98.7% MCC rate, and a 98.67% j-score, according to the results. Additionally, further investigation showed that no management strategy was documented.

Mkonyi et al. (2020) developed a deep learning-CNN model using Python and the Keras toolkit to forecast tomato diseases from a leaf image. The test was performed on a desktop.

PC with Ubuntu 18.04 already loaded on it, an Intel Core i9-9900 3.6 GHz CPU, 16 GB of RAM, and a GeForce RTX 2080Ti GPU (12 GB memory). Also utilized was the Keras deep learning library with the Tensorflow backend. For efficiency, the model used the pre-trained network models ResNet50, VGG19, and VGG16. 2,145 datasets in all were collected from the internet, though the origins were kept secret. 15% of the dataset was used for testing, while 85% was used to train the model. Because of the small dataset used, further research shows that the model needs more data to improve classification performance. With the Visual Geometric Group (VGG) model deployed, the results showed that the model had the maximum accuracy rate of 91.9%. The study considered 10 tomato diseases, but no control measure techniques were recorded.

Gunarathna, Rathnayaka, and Kandegama (2020) created a deep learning-CNN model utilizing the Keras framework to identify 10 tomato diseases from digital images of the leaves. An ACER Aspire3 laptop with an Intel Core i5-8265U 1.66GHz processor, 8GB of DDR4 memory, and an NVIDIA GeForce MX230 graphics card was used to conduct the experiment. Three pre-trained models, including VGG16, MobileNet, and Inceptionv3, were implemented alongside the proposed CNN. The sources of the 22,930 datasets that were gathered via the Internet were not noted. Results showed that the accuracy of the suggested CNN model was 90%, VGG16 model was 89%, MobileNet was 91%, and that of Inceptionv3 was 87%. Again, the suggested model did not include crop management.

Trivedi et al. (2021) created a CNN based model for the prediction of 10 tomato plant diseases. For increased effectiveness, three pre-trained models, including VGG16, MobileNet, and Inceptionv3, were also used. For rapid computation, Google Colab was used to install the system. We used a total of 3000 plant village datasets from Kaggle.com. 20% of the dataset was used for testing, while the remaining 80% was used to train the model. The CNN model had an accuracy rate of 98.49%, VGG16 had a rate of 66.75%,

MobileNet had a rate of 79.52%, and Inceptionv3 had a rate of 64.25%, according to the results. Even though the limited amount data was insufficient to support the deep learning project. Further study showed that the proposed model did incorporate some level of disease management approach for various abiotic diseases caused by a lack of nutrients in tomato plant leaves. Once more, there is no comprehensive plan in place to control the tomato plant diseases and pests.

Ramakrishna (2019) predicted five kinds of tomato plant diseases from leaf data using both the deep learning and machine learning techniques. The implementation made use of the Python 3.7.3 version. The approaches VGG19, VGG16, and Random Forest XGBoost were applied. From the ImageNet dataset, a total of 6,500 photos were gathered. For training, 80% of the dataset was used, and for testing, 20%. Results were only reported using the VGG19 accuracy rate, which was 96%. There was no management process mentioned.

Khatoon, Hasan, Asif, Alshamari, and Yap(2020) built a CNN-based model for the purpose of predicting 24 kinds of diseases in tomato leaves and fruits. The Keras library and Python programming language were implemented. On a GPU-based workstation with 16 GB of RAM and an NVIDIA GTX 1080ti graphics card of a memory capacity of 8GB and CUDA of 1920 cores, the experiments were conducted. For better comparison, three pre-trained network models, including DenseNet, ResNet, and VGG16, were also used. In all, 23,716 photographs from nearby farms and publicly accessible datasets were used. In order to avoid overfitting, 90% of the dataset was used for training and 20% for testing. The results showed that only the DenseNet model had a recorded accuracy rate of 95.31%. Based on further investigation, it recommends that the training data sets should be retrieved from diverse sources. Once more, no management strategy was mentioned.

Sreelatha et al. (2021) created a machine learning predictive model to identify 14 kinds of tomato diseases from tomato leaves using KNN technique. 1,500 digital images in total were gathered, however the sources were not cited. For training, 80% of the dataset was used, and for testing, 20%. The model had an accuracy rate of 86.18%, according to the results. The KNN is computationally expensive, and no management technique was mentioned, according to further investigation.

Al-gaashani, Shang, Muthanna, Khayyat, and Abd El-Latif (2022) used the multinomial logistic regression machine learning technique to construct a model for the prediction of tomato diseases across 38 classes and 14 species. The model's implementation was done with Python and the Keras package. It used a computer with the following specs: CPU i7-3537, 2.50 GHz, and 8 GB RAM. Additionally used were 1,152 digitized photos from the Plant Village dataset on Kaggle.com. 25% of the dataset for testing and 75% of the dataset for training were used. Results showed that multinomial logistic regression had a 97% accuracy rate. Further research recommends that more efficiency will be achieved when a pre-trained deep learning technique is adopted in the prediction of tomatoes disease. There was no management process indicated.

Ahmed, Hasan, Ahmed, Sony, and Kabir (2022) employed various CNN pre-trained models to predict tomato diseases with 10 classes. An Intel Xeon CPU with a base clock speed of 2.3 GHz and an NVIDIA Tesla T4 GPU with a VRAM of 15 GB were used for the research. The computer had 13 GB of total useable memory. Packages for Python, Keras, and Tensorflow were implemented. NASNetMobile, ResNet50, ResNet152V2, DenseNet121, DenseNet201, EfficientNet-B0, MobileNet, MobileNetV2, and VGG19. Images created using CLAHE (Contrast Limited Adaptive Histogram Equalization) were employed. A data set of 20% was assigned for testing, 20% for validation, and 60% for training. Although the size was not specified, the images were taken from the Plant Village

dataset on kaggle.com. While the accuracy rate of the results was 99.30%, it is highly debatable, and further investigation also revealed that no management procedure was suggested.

Ozbilge, Ulukok, Toygar, and Ozbilge (2022) used CNN to categorize 10 kinds of diseases affecting tomato leaves.

On kaggle.com, a dataset of 18,160 images from the plant village was used. 10% of the dataset was used for testing, while 90% was used to train the model. The experimental research work produced an accuracy of 98.77%. Further research revealed that there was no documented procedure for managing diseases and pests.

In Bangladesh, 8 classes of tomato leaf diseases were studied using CNN by Mamun, Karim, Pinku, & Bushra (2020). 11,800 tomato plant images were used to validate the research. The model was trained on 80% of the dataset, whereby 10% was utilized for testing and validate on 10% image data. Utilizing two NVidia GeForce TitanXP GPUs, the system was trained. The research activity generated an accuracy of 98.77% and no control measures were observed.

Rubini and Kavitha (2021) predicted 9 types of diseases and 5 kinds of tomato plants using the CNN model. On Kaggle.com, uploaded a total of 17,200 plant datasets, which used 80%, 10% and 10% for training, testing and validation purpose respectively. The experimental result of the research showed an accuracy of 98.25%, and no control measures were observed.

In order to conduct an experimental study for tomato leaf disease identification in Palestine, *Qasrawi*, Amro, Zaghal, Sawafteh, & Polo (2021) adopted six ML algorithms. These algorithms include random forests, neural networks, naive bayes, SVM, logistic regression and decision trees techniques in the research work. A total of 3,000 images from the field were gathered; 30% of the dataset was used for testing, and 70% were used

for training. No control measure was found, however the results showed that neural networks and logistic regression had 70% accuracy.

Aversano, Bernardi, Cimitile, Iammarino and Rondinella (2020) created a Deep learning-CNN based model with 10 classes for identifying diseases in tomato leaves. Transfer learning techniques from the VGG-19, Xception, and ResNet-50 were applied in the suggested model. The plant village dataset (available on Kaggle.com) contained 16,000 photos. The model's accuracy was 97% and its precision was increased. There was no mention of managerial controls.

A hybrid model employing Fuzzy-SVM and R-CNN was presented by Nagamani & Sarojadevi (2022) for the prediction of 75 types of tomato leaf diseases. The plant village dataset (Kaggle.com) has 735 digital images that were used. The model was trained on 70% of the dataset, and then tested on the remaining 30%. No management control techniques were recorded, and the model had a 96.7% accuracy rate.

Gonzalez-Huitron et al. (2021) incorporated a Deep Learning and Machine Learning approaches to create a model for predicting tomato leaf diseases of 10 classes. Spyder IDE 4.1.3, OpenCV 3.4.2, a Raspberry Pi 4 for hardware implementation of ram capacity in Gigabyte, and an NVIDIA GTX 1070 GPU with 1920 CUDA cores and an 8 Gb GDDR5 memory interface were used to implement the procedure. Additionally, the suggested model utilized Xception, MobileNetV2, NasNetMobile, and MobileNetV3 to apply transfer learning techniques. Plant Village provided a dataset of 18,215 images in total (Kaggle.com). The dataset's size was increased by data augmentation. 30% of the dataset was utilized to test the model, while 70% was used for training. Although there was no discussion of a pest and soil management strategy, the results showed an accuracy of 94.5%.

In order to predict tomato leaf diseases, Chowdhury et al. (2021) suggested a CNN-based Deep learning model with 10 classes. Densenet120, residual networks such as ResNet101, ResNet50, ResNet30, ResNet18, Squeezenet, and VGGnet were some of the transfer learning models employed in the model. They gathered a dataset of roughly 18,160 images from plant village on Kaggle.com web site. Despite the proposed model's questionable 99.81% accuracy output, no management control mechanism was stated.

A Deep learning-CNN predictive model was created by Sumalatha, Singothu, & Rao, in 2021 for 10 classes of tomato leaf diseases. To achieve some level of transfer learning, several pretrained models that comprises of Xception, VGG16, InceptionV3, MobileNet, Resnet50, and DenseNet121 were employed. A total of 11,333 digital photos of tomato diseases that were impacted were downloaded from plant village in total (Kaggle.com). Eighty percent of the dataset was used to train the model, ten percent for testing, and ten percent for evaluation. The accuracy of DenseNet121 was reported to be 95.48%. Notably, no management and control tactics were addressed.

A Deep learning-CNN model was also created by Rajendran, & Thiruvengadam (2020) to identify ten kinds of tomato leaf diseases. 12,206 photos in all were gathered from the plant village (Kaggle.com). 80% of the dataset was devoted into training, while 20% was deployed testing. The Resnet50 pretrained model was also used in the model. The following system specifications were used to test the model: an Intel Core i7-8750 running at 2.2GHz, 16GB of RAM, 250GB of SSD, and PyTorch. 97% accuracy was revealed by Resnet50. No mention was made on managing pests and diseases.

To detect tomato leaf disease in 10 different classes, Brahimi, Boukhalfa, & Moussaoui (2017) employed a deep learning-CNN & shallow model. 14,828 digital photos from plant village on Kaggle were collected in total. The model made use of 80% dataset for its training, then testing data sets was allocated 20%. A desktop workstation with 128 GB of

RAM, two Intel Xeon processors running at 2.00 GHz, and 24 cores was used to conduct the trial experiment. Results obtained afterwards indicates that the CNN model had an accuracy of about 99.18%. No mention was made of managing pests and diseases.

Hong, Lin, and Huang (2020) used a variety of pretrained models, including ShuffleNet, Xception, Resnet50, MobileNet, and DenseNet121, to construct a Deep learning-CNN to predict 10 classes of Tomato Leaf diseases. From plant village on Kaggle.com, 41,263 photos in total were gathered. A training dataset was employed with 80% of the data, and a testing dataset with 20%. The model had a 91.10% accuracy even though there were imbalanced datasets that could have had an impact. There were no documented instances of controlling pests and diseases.

Agarwal, Singh, Arjaria, Sinha and Gupta (2020) used Deep learning-CNN to create a model to forecast tomato leaf diseases in 10 classes. On an NVIDIA DGXv100 computer, the suggested CNN model was run. The computer has 128GB of RAM, 1000 TFLOPS speed, 5120 tensor cores, 40600 CUDA cores, and. Also used were pre-trained models like VGG16, MobileNet, and InceptionV3. Plant Village provided a total of 17,500 photos (Kaggle.com). An estimated dataset of 70% were utilized for training, 10% deployed for testing, and 20% for validation. A 91.2% accuracy rate was found in the results. There was no discussion about managing pests and diseases.

To predict tomato leaf diseases, Shrestha, Deepsikha, and Dey (2020) created a model based on CNN. 3,000 photos from plant village on Kaggle.com were collected in total. In which 20% of the dataset was used for testing, and an outstanding 80% was used for training. The machine used for training and testing was equipped with a 16 GB DDR4 RAM module, at a 3.60 GHz Intel of a Core i7-9600k processor, and a 64-bit Windows 10 operating system. No mention of pest and disease management was made, and the proposed model recorded 88.80% accuracy.

A predictive model for tomato leaf disease with several classes using CNN and LVQ was created by Sardogan, Tuncer, & Ozen (2018). 500 images from a plant community (Kaggle.com) were utilized. An 80/20 split between training and testing was used for the dataset. Nobody brought up disease management.

A model to forecast tomato leaf disease was created by Ashok et al. (2022) utilizing CNN and AlexNet pre-trained models. On the small datasets acquired, data segmentation was performed. The model has a 98.12% accuracy rate, but no safety precautions or control measures were recommended.

Chohan, Khan, Chohan, Katpar and Mahar (2020) used the Deep learning CNN technique to create a model with 38 different classes to predict tomato leaf diseases. From plant village on Kaggle.com, a total of 70,295 digital images were gathered. 20% was utilized to test the suggested model, while the remaining 80% was for training. To expand the dataset's size, data augmentation was also used. Without including case diseases management, the model has an accuracy of 98.3%.

Sunidhi and Jalaja (2021) used pre-trained models like AlexNet, InceptionV3, and ResNet152V2 that develops a deep learning model that recognizes tomato leaf diseases. A sizable dataset was acquired from plant village on Kaggle.com. The proposed model has a 99% accuracy rate.

In order to predict tomato leaf diseases, Deng, Chang, Li, & Yang (2021) used machine learning and deep learning methods. Deep learning models that have already been trained are used, including AlexNet, VGGNet, GoogLeNet, and ResNet. There was also use of RAHC GAN. Dataset used were collected from the Plant Village (Kaggle.com). Even though training was not efficient in time of execution and utilisation of computing resources. The RAHC GAN performed better than the competition. There was no discussion of disease management.

Jadhav, Riswadkar, Jadhav and Gogawale (2022) proposed a model for predicting tomato leaf diseases using several machine learning techniques like Decision Tree, Logistic Regression, Random Forest, and XGBoost & content-based filtering. Datasets were gathered from an online repository (kaggle.com). Random forest classifier yielded an accuracy rate of 98.9 %.

A model to forecast tomato leaf diseases was created by De Luna, Dadios, & Bandala, (2018) using deep learning methodology. 4,923 images in all were collected. Data annotation and augmentation operations were conducted as a technique to improve size of data. 80% and 20% of the data sets were used for both training and testing, respectively. The model produced a 95.75% accuracy rate.

Table 2.1: Summary of related works on tomato leaf diseases using AI approach (Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
1	Chowdhury et al., (2021a)	Tomato Leaf	DL- CNN PyTorch library with Python	Densenet120, ResNet101, ResNet 50, ResNet 18, Squeezenet and VGGNet.	18,161 Plant Village dataset (Kaggle.com)	Segmentation Augmentation Training=80% Testing=20%	Accuracy = 99.89%	No provision of or control tips
2	Altuntaş & Kocamaz, (2021)	Tomato Leaf	Deep learning - CNN MATLAB	3 Pre-trained CNN models; Alexnet, GoogleNet ResNet50	18,835 from Plant Village (Kaggle.com)	Data segmentation Training=40% Testing=10%	Accuracy Alexnet = 92.18 GoogleNet = 89.31 ResNet50 = 96.99%.	No provision for control tips
3	Chowdhury et al., (2021b)	Tomato Leaf	Deep learning- CNN Python 3.7.	4 pre-trained models; ResNet, MobileNet, DenseNet201, InceptionV3	18,162 from plant village dataset (Kaggle.com)	Data Augmentation, Training=70% Testing=20% Evaluation=10%	Accuracy enseNet201 = 98.05%,	No control tips provided
4	Liu & Wang, (2020)	Tomato Leaf	Deep learning – CNN python,	3 pre-trained models; SSD, Faster R-CNN and YOLO v3	1,952 from Field collection	NA	Accuracy YOLO V3 = 92.39% time =20.39 ms	No provision for control tips

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
5	Hiremath & Gowda, (2022)	Tomato leaf	Deep learning CNN	VGG16, InceptionV3 and MobileNet.	5,000 images from (Kaggle.com)	Data augmentation Image segmentation	Accuracy CNN = 91.2%	No provision for control tips
6	Damicone & Brandenberger, (2021)	Tomato Leaf	Deep Learning MAT- LAB®	DNN, CNN RNN ,FNN BPNN	1,500 images from field collection	Training=1300 Testing =200	Overall Accuracy: DNN=86.18%	No provision for control tips
7	Wanget al., (2019)	Tomato fruit	DL DCNN & Object detection architecture, Python,	(VGG-16 , ResNet-50 ResNet-101 & MobileNet	1,430 from the internet	Segmentation, Training = 60% Test =20% Validation=20%	(ResNet101 = 99.64%) Faster R-CNN ResNet101 accuracy = 88.53%),	No provision for control tips
8	Mkonyi et al., (2020)	Tomato leaf	Deep learning- CNN	VGG16, VGG19, ResNet50,	2,145 from internet	Data augmentation, Training=85% Testing=15%	Accuracy VGG16=91.9%	No provision for control tips

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
9	Vijay, (2021)	Tomato leaf	Deep learning-CNN &Machine learning-KNN	CNN &KNN	10,000 from Plant village dataset (Kaggle.com)	Training=75% Testing=25%	CNN AC = 98.5% , P = 93% R = 93% F1-Score = 93% KNN AC = 83.6% P = 90% Recall = 84% F1-Score= 86%	No provision for control tips
10	Islam et al., (2022)	Tomato leaf	Deep learning-CNN, MAT-LAB(R2021a,	DeepD381v1	28,000 from Plant village dataset (Kaggle.com) & plant factory of Ehime University (Japan)	Data augmentation, Training=80% Testing=20%	AC= 0.9886 Sen= 0.9886 Speci= 0.9981 P = 0.9986 F1S= 0.9886 MCC= 0.987 j-score= 0.9867	No provision for control tips
11	Gunarathna, Rathnayaka, & Kandegama (2020)	Tomato leaf	Deep learning-CNN	VGG16, MobileNet, Inceptionv3	22,930 from internet	Data augmentation Training=75% Validation=25% Testing=5%	accuracy of CNN=90% VGG16 =89% MobileNet=91% Inceptionv387%	No provision for control tips

Table 2.1: Summary of related works on tomato leaf diseases using AI approach (Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
12	Trivedi et al., (2021)	Tomato Leaf	Deep learning-CNN Google Colab	VGG16, MobileNet, Inceptionv3	3,000 plant village datasets (Kaggle.com)	Training=80% Testing=20%	Accuracy CNN =98.49% VGG16=66.75% MobileNet79.52% Inceptionv3=64.25	No provision for control tips
13	Ramakrishna, (2020)	Tomato Leaf	Deep learning Machine learning Python,	VGG19, VGG16 SVM, XGBoost DenseNet , RestNet, VGGNet	6,500 from Image dataset	Data augmentation Training=80% Testing=20%	Accuracy VGG19=96%	No provision for control tips
14	Khatoun et al., (2021)	Tomato leaves & fruits	Deep learning-CNN Python,	NA	23,716 images	Training=90% Testing=10% raining=80% Testing=20%	Accuracy DenseNet = 95.31%	No provision for control tips
15	Sreelatha et al., (2021)	Tomato leaves	Machine learning-Matlab-based	NA	1,500 images	Training=80% Testing=20%	Accuracy, 86.18%	No provision for control tips

Table 2.1: Summary of related works on tomato leaf diseases using AI approach (Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
16	Al-gaashani <i>et al.</i> , (2021)	Tomato leaves	DL –CNN, ML, Python	MobileNetV2 & NASNetMobile; PCA, SVM, RF LR	1,152 images from plant village dataset	Training=75% Testing=25%	Accuracy, multinomial LR = 97%	No control tips provided
17	Ahmedet al., (2022)	Tomato leaves	Deep learning-CNN ML Python,	DenseNet121, DenseNet201, EfficientNet- B0, MobileNet, MobileNetV2, NASNet-Mobile, ResNet50, ResNet152V2, and VGG19 (CLAHE)	18,160 images	Data augmentation Training=60% Testing=20% Validation = 20%	Accuracy 99.30%	No Provision for control tips
18	Ozbilgeet al., (2022)	Tomato leaves	Machine learning		18,160 images. from plant village (Kaggle.com)	Data augmentation Training=90% Testing=10%	Accuracy healthy ones = 98.77%	No provision for control tips

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
19	Mamun <i>et al.</i> , (2020)	Tomato leaves	Deep learning-CNN GeForce TitanXP GPUs.	TLNET	11,800 images from plant village dataset (Kaggle.com)	Training=80% Testing=10% Validation=10%	Accuracy = 98.77%	Tomato leaves
20	Rubini & Kavitha, (2021)	Tomato leaves	Deep learning-CNN	VGG16 and DenseNet	17,200 images from plant village dataset (Kaggle.com)	Training=80% Testing=10% Validation = 10%	Accuracy DenseNet = 98.25 %	No control tips provided
21	Qasrawi et al., (2021)	Tomato leaves	ML Clustering techniques, NN,RFNB, SVM, DT,LR		3,000 images from field collection	Training=70% Testing=30%	Accuracy Logistic regression and neural network, = 70%	No control tips provided
22	Aversano et al., (2020)	Tomato leaves	Deep learning-CNN MATLAB 2019B.	VGG-19, Xception and ResNet-50	16,000 images from plant village dataset (Kaggle.com)	Training= NA Testing=NA	Accuracy VGG-19 = 97%.	No control tips provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd)

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
23	Nagamani & Sarojadevi, (2022)	Tomato leaves	Machine learning & Deep learning MATLAB	CNN, Fuzzy-SVM & R-CNN	735 images from plant village dataset (Kaggle.com)	Training=80% Testing=10% Validation = 10%	Accuracy DenseNet = 98.25 %	No control tips provided
24	Gonzalez-Huitron et al., (2021)	Tomato Leaf	DL&ML	MobileNetV2 Xception MobileNetV3	18,215 images from plant village (Kaggle.com)	Data augmentation Training=70% Testing=30%	Accuracy Xception = 94.5%	No control tips provided
25	Mohammad et al., (2021)	Tomato Leaf	Deep learning-CNN System specification (NVIDIA DGXv100)	Densenet120, ResNet101, ResNet 50, ResNet 18, Squeezenet and VGGNet.	18,160 images from plant village (Kaggle.com)	Data augmentation Training= NA Testing=NA	Accuracy Densenet120 = 99.81%	No control tips provided
26	Sumalatha et al., (2021)	Tomato Leaf	Deep learning-CNN	VGG16, InceptionV3, Xception, Resnet50, MobileNet, DenseNet121	11,333 images from plant village (Kaggle.com)	Training=80% Testing=10% Evaluation=10%	Accuracy DenseNet121=95.48%	No control tips provided

Tale 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd)

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
27	Khurana et al (2019)	Tomato leaves	Deep learning, Machine learning	VGG-19, RF, SVM, ANN	3242 images	Training = 85%, testing=15%,	98.6%.	No control measures.
28	Altalak et al (2022)	Tomato Leaf	Deep learning, machine learning	ResNet50-CBAM + SVM	11,000 images from plant village (Kaggle.com)	NA	97.2%	No control tips provided
29	Zaki et al., (2020)	Tomato leaf	Deep learning, Python	MobileNet V2	4,671 images from PlantVillage dataset	Training = 80%, testing=20%,	Accuracy = 90%	No control tips provided
30	Zhang et al., (2018)	Tomato leaf	Deep learning, Matlab, memory	AlexNet, GoogLeNet, and ResNet	5550 images from plant village	Training = 80%, testing=20%,	Accuracy = 97.28%	No control tips provided
31	Nithishet al., (2020)	Tomato Leaf	Deep learning-CNN PyTorch	Resnet50	12,206 images from plant village (Kaggle.com)	Data augmentation Training=80% Testing=20%	Accuracy Resnet50 = 97%	No control tips provided

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
32	Brahimi et al., (2017)	Tomato Leaf	Deep learning-CNN & ML	CNN, SVM, Random Forrest Preprocessing	14,828 images from plant village (Kaggle.com)	Training=80% Testing=20%	Accuracy CNN=99.18%	No control tips provided
33	Hong et al., (2020)	Tomato Leaf	Deep learning-CNN	ShuffleNet, Xception, Resnet50, MobileNet, DenseNet121 Xception	41263 images from plant village (Kaggle.com)	Data augmentation Training=80% Testing=20%	Accuracy DenseNet121_Xception=91.10%	No control tips provided
34	Agarwal et al., (2020)	Tomato Leaf	Deep learning-CNN.	VGG16 MobileNet, and InceptionV3	17,500 images from plant village (Kaggle.com)	Training=70% Testing=10% Validation=20%	Accuracy = 91.2%	No control tips provided
35	Shrestha et al., (2020)	Tomato Leaf	Deep learning CNN python	CNN	3,000 images from plant village (Kaggle.com)	Training=80% Testing=20%	Accuracy = 88.80%	No control tips provided
36	Sardogan et al., (2018)	Tomato Leaf	Deep learning python software	CNN & LVQ	500 images from plant village (Kaggle.com)	Training=80% Testing=20%	Accuracy = NA	No control tips provided

Table 2.1 Summary of related works on tomato leaf diseases using AI approach.

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis	
37	S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
38									
39	42	De Lunaetal., (2018)	Tomato Leaf	Deep learning, MATLAB	AlexNet,, F-RCNN, GUI,	4,923 images	Augmentation, Training=80% Testing=20%	Accuracy AlexNet = 95.75 %	No control strategy given
40	43	Karthiket al., (2020)	Tomato leaf	Deep learning Python	CNN, KNN, SVM	82,161 Plant Village (Kaggle.com)	Data augmentation Training=80% Testing=20%	Accuracy CNN= 98%	No control strategy given
41	44	Kirange & Gadade, (2020)	Tomato leaf	ML,	LR Analysis (LDA), KNN Classification, SVM, DT, Classification, Naïve Bayes Classification	3,000 Plant Village (Kaggle.com)	Data segmentation Training=70% Testing=30%	Accuracy SVM = 98%	No control measure given
42	45	Verma et al., (2018)	Tomato leaf	CNN Raspberry Pi 4	InceptionV3, ResNet50, AlexNet, MobileNetV1, MobileNetV2, MobileNetV3	16,004 images from Plant Village (Kaggle.com)	Training=90% Testing=10%	Accuracy MobileNetV3 & AdaGrad = 99.81%	No control tips provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
46	Rangarajan et al., (2018)	Tomato leaf	Deep learning & Object detection, Matlab	AlexNet & VGG16	13,262 images from Plant Village (Kaggle.com)	Augmented & images segmentation Training=90% Testing=10%	Accuracy AlexNet = 97.49%	No control tips provided
47	Ahmad et al., (2020)	Tomato leaf	Deep learning	VGG-16, VGG-19, Residual Neural network ResNet, Inception V3.	15,216 images from field collection	Data augmentation Training=70% Testing=10%, Validation=20%	Accuracy Inception V3 = 98%	No control tips provided
48	Abbas et al., (2021)	Tomato leaf	Deep learning	C-GAN DenseNet121	16,012 images from field collection	Data augmentation Training=60% Testing=30%, Validation=10%	Accuracy Inception V3 = 99.51%,	No control tips provided
49	Wspanialy & Moussa, (2020)	Tomato leaf	Deep learning	ResNet U-Net	18,005 images from Plant Village (Kaggle.com)	Data augmentation Segmentation Training=60% Testing=25%, Validation=5%	Accuracy = 98.7%,	No control tips provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
50	Shoaibet al., (2022)	Tomato leaf	Deep learning, python.	Inception Net model, U-Net and Modified U-Net,	18,161 images from Plant Village (Kaggle.com)	Data augmentation Training=90% Validation=10%	Accuracy InceptionNet3 = 99.89%	Only alert system is provided
51	Hammou & Boubaker, (2022)	Tomato leaf	Deep learning- CNN Python	DenseNet169 & InceptionV3	18,162 images from Plant Village (Kaggle.com)	Data Augmentation Training=80% Testing=20%	Accuracy = 100%	No Control tips provided
52	Geetharamani & J., (2019)	Tomato leaf	Machine learning Python	SVM, logistic regression, decision tree and KNN	61,486 images from Plant Village (Kaggle.com)	Training=80% Testing=20	Accuracy = 96.46%	No Control tips provided
53	Ikeda et al., (2021)	Different crops	DL-CNN Edge GHz, RAM of 16GB.	CNN	635 images from Plant Village, Kaggle & field collection	Training = NA Testing = NA	Accuracy = 100%	No control tips provided
54	Singh et al., (2019)	Tomato leaf	Deep Learning- RCNN	MobileNet	2,598 public dataset and field work	Training = NA Testing = NA	Accuracy = 70.53%	No control procedure

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
55	Ferentinos, (2018)	Tomato leaf	Deep Learning CNN	VGG	87,848 images taken in both laboratory conditions and fields	Training = NA Testing = NA	Accuracy = 99.53%	No control tips provided
56	Tulshan & Raul, (2019)	Tomato leaf	Machine Learning MATLAB	KNN, SVM	75 public dataset and field work	Training=70% Testing=30	Accuracy = 98.56%	No control measure used
57	Durmus et al., (2017)	Tomato leaf	Deep Learning CNN Python	AlexNet & SqueezeNet.	54,309 images from Plant Village (Kaggle.com)	Training=80% Testing=20	Accuracy AlexNet = 97.22%	No control tips provided
58	Zhao et al., (2021)	Tomato leaf	Deep learning CNN OpenCV under the Python	SE-ResNet50 GoogleNet, ResNet101, Xception, VGG-19	22,925 images from Plant Village (Kaggle.com)	Data augmentation Training=80% Testing=20%	Accuracy SE-ResNet50 = 96.81%	No control tips provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
59	Kaur & Bhatia, (2019)	Tomato leaf	Deep Learning MATLAB	ResNent 101	22925 images from Plant Village (Kaggle.com)	Training=60% Testing=40%	Accuracy = 98.8%	No control tips provided
60	Habiba & Islam, 2021)	Tomato leaf	Deep Learning-CNN	VGG16	Above 15000 images from Plant Village (Kaggle.com)	Augmentation. Training=60% Testing=20% Validation = 20%	Accuracy = 95.5%	No control tips provided
61	Militanteetal., (2019)	Tomato leaf	Deep Learning-CNN Python,.	CNN	35,000 images	Data augmentation Training=NA Testing=NA	Accuracy = 96.5%	No control tips provided
62	Hidayatuloh etal., (2018)	Tomato leaf	Deep learning	SqeezNet	1,400 images	Training=NA Testing=NA	Accuracy = 86.92%	No control tips provided
63	Gehlot & Saini, (2020)	Tomato leaf	Deep learning-CNN	AlexNet, VGG-16, Google Net, DenseNet-121, and ResNet-101	14,529 images from Plant Village dataset (Kaggle.com)	Training=80% Testing=20%	Accuracy DenseNet121 = 99.69 %	No control measures provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
64	Annabel & Muthulakshmi, (2019)	Tomato leaf	Machine learning	GLCM and random forest	2175 images (Kaggle.com)	Training=70% Testing=30%	Accuracy = 94.1 %	No control measures provided
65	Govardhan & Veena, (2019)	Tomato leaf	Machine learning Python	K Nearest Neighbors, LR, DT, RF, SVM	140 images From field collection	Training=90% Testing=10%	Accuracy Random Forest =95%	No control measures provided
66	Francis & Deisy, (2019)	Tomato & Apple leaf	Deep learning- CNN	VGG	Images from Plant Village dataset (Kaggle.com)	Training=NA Testing=NA	Accuracy = 88.7 %	No control measures provided
67	Ajra et al., (2020)	Tomato & Potato leaf	Deep learning MATLAB2018a.	AlexNet and ResNet-50	6,000 images from (Kaggle.com)	Training=70% Testing=30%	Accuracy ResNet-50= 96.1%	No control measures provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach(Cont'd).

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
67	Ajra et al., (2020)	Tomato & Potato leaf	Deep learning MATLAB2018a.	AlexNet and ResNet-50	6,000 images from (Kaggle.com)	Training=70% Testing=30%	Accuracy ResNet-50= 96.1%	No control measures provided
68	Kumar & Vani, (2019)	Tomato leaf	Deep learning- CNN. Python	LeNet, VGGNet, ResNet and Xception	14,903 images from (Kaggle.com)	Training=80% Testing=10% Validation = 10	Accuracy VGGNet = 96.1%	No control measures
69	Angel et al., (2021)	Tomato leaf	Deep learning- CNN Python.	CNN	1,500 images from (Kaggle.com)	Training=75% Testing=25%	Accuracy= 97.1%	No control measures provided
70	Sadik Tasrif Anubhoveetal., (2020)	Tomato leaf	Computer vision and machine learning Viola-Jones algorithm and Haar MATLAB,	Viola-Jones algorithm and Haar	377 images	Training=NA Testing=NA	Accuracy = NA	No control tips provided

Table 2.1: Summary of related works on tomato leaf diseases using AI approach.

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
71	Bir et al., (2020)	Tomato leaf	Deep Learning- CNN	EfficientNetB0, MobileNetV2 and VGG 19	15,000 images	Segmentation Training=90% Testing=10%	Accuracy EfficientNetB0 = 98.6%	No control measures mentioned
72	Jiang <i>et al.</i> , (2020)	Tomato leaf	Deep Learning- CNN	Resnet-50	6,794 images	Data augmentation Training=90% Testing=10%	Accuracy = 98.3%	No control measures
73	Chakravarthy & Raman, (2020)	Tomato leaf	Deep Learning- CNN	▪ResNet and Xception	4141 images	NA	99.952%.	No control measures
74	Akshai & Anitha, (2021)	Tomato leaf	Deep learning, Machine learning, Image processing,	CNN	Both public and private	NA	NA	No control measures
75	Lu et al.,(2018)	Tomato leaves	Machine Learning	KNN, SVIs and PCs	445 images	▪ Training=70% ▪ Testing=30%	Accuracy kNN = 100%	No control measures.

Table 2.1: Summary of related works on tomato leaf diseases using AI approach.

S/N	Author(s)/year	Image domain	Technique(s)/ Experimental environment used	Pre-trained Models	Size/source of dataset	Preprocessing of dataset	Performance Metric	Comments for Gap Analysis
76	Fuentes et al., (2017)	Tomato leaves	Deep learning,	Faster R-CNN(VGG-16 ResNet-50 ResNet-101 ResNet-152 ResNeXt-50) R-FCN (ResNet-50) SSD(ResNet-50)	5000 images collected from farms, Korean Peninsula.	Training = 80%, testing=10%, validation=10%	R-FCN (ResNet-50) with accuracy= 85.98%,	No control tips.
77	Vetal & R.S., (2017)	Tomato leaves	Machine learning. MATLAB	SVM	320 images from field collection	Training = 80%, testing=5%, validation=15%	93.75%.	No control measures.
78	Khurana et al (2019)	Tomato leaves	Deep learning Machine learning	VGG-19, RF, SVM, ANN	3242 images	Training = 85%, testing=15%,	98.6%.	No control measures.

2.4 Summary of the Literature Review

After examinations of several classification models for tomato leaf diseases by considering factors such as the type of model, size of datasets, techniques used, programming language, integration with recommender systems, and mobile compatibility. The issues with the current models can be summarized as follows:

- (a) Nearly all the tomato leaf disease prediction models focused on improving the accuracy of the model based on detection while neglecting the need for management in the literature.
- (b) Also, adoption of the model without checking the parameters such as computational time makes it difficult to implement on small devices which are easily available to farmers.
- (c) In addition, certain new technology which have been found to be more accurate in monitoring tomato plants appear to be expensive for most farmers to afford and require specialized training.
- (d) Using a fully connected layer in a model could significantly increase the size of the model and also increase the inference time, which will definitely increase the overall computational time
- (e) All existing tomato leaf disease prediction models reviewed only stopped at the classification phase. They did not provide a recommender system which can help to inform the farmers of the symptoms and as well as the preventive tips they should apply to curb or mitigate these diseases at their early stage.
- (f) Most tomato leaf disease models are not integrated into a mobile platform for use by the farmers or horticulturist. A model may have a high accuracy, good recall/precision tradeoff but perform poorly during model integration because of several factors inherited by the choice of model.

CHAPTER THREE

METHODOLOGY

3.1 Research Methodology Overview

This section highlights the research design and methodology that will be utilized in the investigation. Research methodology refers to the steps or methods employed to gather, choose, handle, and interpret data related to this research work. The research adopted an applied approach that utilized deep learning and optimization techniques addressed the research questions. Both primary and secondary data acquisition methods were employed to gather information for the study. Additionally, guidance and expert opinions were sought from notable organizations such as Imo River Basin, National Horticultural Research Institute, Ibadan. The data obtained from a public database provided by Plant Village organization and collected in the field were prepared for training through preprocessing. To train the model, a deep learning architecture based on residual networks was employed. The TensorFlow Lite framework was used to optimize and convert the model. The most effective model was saved for future use in predicting tomato leaf diseases.

3.1.1 Software Methodology

Software methodology refers to the process of planning, creating, testing and then deploying a project. This section of the research discusses the software methodology adopted based on its strength in achieving the research desired goals. Software development methodologies have come a long way since British computer scientist Tom Kilburn wrote the world's very first piece of software in 1948 using eight words of working storage plus 17 words of instructions, rendering a program size of 25 words.

A reliable methodology is an essential component of any software project, as it guides the team to cover edge-to-edge clients' business requirements and promises a feature-

packed application. It also aids both parties to communicate and complete the software creation economically and timely effectively. According to Taley and Pathak (2013), set of procedures are encompassed within a software development process, which is designed to obtain and ensure the software's development process and related products.

In addition, Software development life cycle model can be seen as a process by which a software project is completed or developed through the well-defined processes or stages. In present days there are a number of software development models to develop certain software project, so it is very difficult to decide which model is to use for certain type of software project, because every model has its pros and cons, example models take long time to complete, some required less time, some models are for non-technical clients some require well skilled employee, some need crystal clear knowledge about project, one for only developing software from scratch. Whatever model is chosen for developing software projects some mandatory stages are required. Software development methodology can be seen as a set of guidelines for creating software and is usually more than just a method, and it is frequently backed up by some underlying philosophy and an era from which it arose (Samar & Hiba., 2020). There are two software development methodologies namely, Structured System Analysis Design Methodology and the Object-Oriented Analysis and Design Methodology.

Structured Systems Analysis and Design Methodology (SSADM) is a systems approach to the analysis and design of information systems. SSADM was produced for the CCTA, a UK government office concerned with the use of technology in government, from 1980 onwards. The Structured Systems Analysis and Design Method, or SSADM, takes a systematic method during the analysis and design of information systems, according to (Waltz 2019). It uses text and diagrams throughout the system design life cycle, from the initial design concept to the actual physical design of the application. The analyst and users can decide whether to move

on to the next stage, abandon the project, or modify some of the stages at the conclusion of each stage in the life cycle of a project using SSADM. The Analysis and Design stages of the Systems Development Life Cycle are the emphasis of SSADM (SDLC). Within a systems development cycle, SSADM incorporates three methodologies that complement one another: logical data modelling, data flow modelling, and entity event modelling. Spiral models, waterfall models, agile methodologies, extreme programming, and prototype software development methodologies are only a few examples. Agile methodology will be used in this research work because of its speed and efficiency which perfectly suited to the rapidly advancing fields of AI and ML.

3.2 Software Methodology adopted.

Having rigorously analyzed the software techniques such as waterfall, prototyping, iterative spiral, and v-model, their functions, roles, advantages, and disadvantages were carefully analyzed. As a result, it became evident that these approaches could not serve as the desired software development life cycle for the proposed intelligent hybrid model due to their operational flaws and limitations. Consequently, the agile approach was selected for the research due to its numerous benefits.

In the agile paradigm, teams that are self-organized and possess a range of skills collaborate closely with their clients or end users to identify requirements and develop iterative solutions. This methodology promotes adaptive design, evolutionary progress, timely delivery, continuous improvement, and flexibility in accommodating changes.

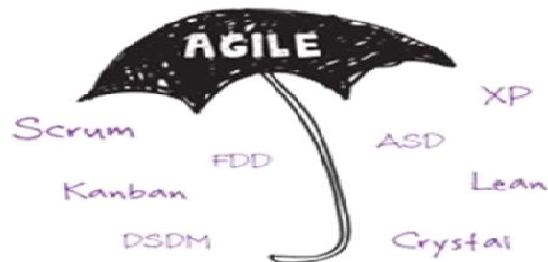


Figure 3.1: Agile software methodology

Figure 3.1 illustrates the agile development life cycle, comprising six methodologies: specifically, scrum, kanban, feature-driven design, extreme programming, lean methodology, crystal, and adaptive software design.

Table 3.1 shows some of the advantages of agile methodology and the disadvantages it has as a methodology.

Table 3.1: Advantages and disadvantages Agile Methodology

S/N	Advantages	Disadvantages
1	Superior quality product	Poor resource planning
2	Customer satisfaction	Limited documentation
3	Better control	Fragmented output
4	Improved project predictability	No finite end
5	Reduced risks	Difficult measurement
6	Increased flexibility	
7	Continuous improvement	
8	Improved team morale	

Extreme programming methodology which is a methodology under the agile methodology frameworks was used to design and manage the proposed model. The extreme programming (XP) phases begin with planning, managing, designing, coding, testing, deployment, and finally, the review phase.

The following activities were carried out in the different stages or phases under the extreme programming methodology.

(i) Planning stage

Under this phase all the user requirements, functional and non-functional requirements as represented in Tables 3.3, 3.4 and 3.5 below were gathered. One thousand digital images of the nine different classes of tomato leaf diseases each under the study area were downloaded from an online research community referred to as kaggle.com.

More so, over 120 research articles on existing AI based predictive models for tomato leaves diseases were also collected from different online research databases such as IEEE Xplore, ACM and google to get first-hand knowledge about the different techniques that has been used and an insight on their results.

(ii) Managing stage

In this phase, there were several meetings, interviews and discussion sessions with different agricultural experts at different research centers in both within and outside Owerri, Imo state. Several meetings were heard with AI based developers/researchers in the field of computer vision and deep learning to ascertain the right components to use in our designs and integrating the various components into a good blueprint design.

(iii) Designing stage

This phase is where the architectural framework of the model comes in. Here the system is modelled using high-level modelling tools like the activity diagram, use case diagram and class diagram. The aim of this stage is to create a plan that can be converted into a

functional system during the coding phase. A paid Lucidchart platform to construct the architectural framework, create a use case diagram, and develop a flowchart for the future system was used. Additionally, we designed a data flow diagram to illustrate the process flow, which will help coders gain a deeper understanding and facilitate the effective implementation of the classifier model.

(iv) Coding stage

In this phase, the transformation of all diagrams developed in the design stage into a functional system takes place. The dataset, collected during the planning phase of the system life cycle, undergoes cleaning using a Python IDE named Jupyter Notebook. Throughout the coding process, Google Colab was employed for the task. In the Google Colab environment, the GPU runtime was selected to expedite the task. Before the initial dataset split, a meticulous check ensures proper labeling. The entire dataset is utilized for training the MobileNet V2 model.

Prior to applying the base model to the dataset, preprocessing is conducted, mirroring the MobileNetV2 preprocessing task. Seeking rich features with deep semantic meaning, we employ weights like MobileNet V2 on ImageNet, while freezing the top layer typically used for prediction. The global average pooling method is applied to average the features, resulting in a uniform length and shape of the feature map sized at 1024. Tensorflow Keras serves as the backend, and the numpy library is utilized to store the features.

For the SVM aspect, the sklearn library is invoked to obtain the SVM model. Initial splitting of the features is performed for training and testing, along with their corresponding labels. Various kernel methods are incorporated to achieve optimal results. Parameter tuning on C is carried out to mitigate overfitting and ensure the model does not generalize excessively.

The Universal Sentence Encoder (USE), a deep learning transformer model, is employed along with a package imported from the seaborn library. A list of recommendations is

created and used as input. Each sentence or recommendation undergoes embedding using the embed function. During the linking or representation process, values ranging from 0 to 9 are assigned to link classes to recommendation vectors.

(v) Testing stage

Testing procedures involved calling the SVM's predict function on test data samples, with weights saved post-prediction for subsequent use. Both system and field tests were conducted. Initially, approximately 1000 leaf samples from various plants were gathered to validate the classifier's ability to distinguish tomato plant leaves from others. Subsequently, tomato leaves with one or more diseases were collected, photographed, and uploaded to the system, which successfully identified all disease types and provided tailored recommendations for farmers. In addition to field tests, system performance was evaluated using the confusion matrix from the scikit-learn metrics library in Python. The performance test results, outlining the classifier's accuracy based on correctly and incorrectly identified leaves, were detailed in Chapter Four. Acceptance testing involved distributing a questionnaire alongside the developed system to gauge user responses. The questionnaire, utilizing a 5-point Likert scale, focused on three aspects: ease of usage, speed of prediction, and efficiency of results. This multifaceted approach to testing ensured a comprehensive evaluation of the proposed system's effectiveness and user acceptance.

(vi) Deployment stage

The deployment phase is the transformation of the proposed system into use. The model is embedded to run on a mobile platform which was the motivation for using the MobileNetV2. The proposed model follows a 2-tier architecture having a front end and middleware. The front end of the proposed system was developed using the flutter wave and also was used to unanimously interact with the saved model. The enhanced model was saved in the json.h5 format using the json library so that it could be called up

anytime a prediction is to be made. The flutter framework works like a server interacting directly with a front end to accept input from the user, handles the user request by calling on the weights of the saved model on the input image. Finally, providing a richly interactive and user-friendly result to the end user.

3.2.1 Data Gathering Repository from Online and Fieldwork

To create a model, start by defining its scope and objectives, clarifying the problem you aim to solve and identifying the necessary data. Next, search for online repositories like Kaggle, UCI Machine Learning Repository, and OpenML that provide relevant datasets. Download the datasets, ensuring you understand their limitations. If fieldwork data is needed, identify suitable sources such as surveys, interviews, or observations. Collect the data, adhering to ethical guidelines. Once you have the data, clean and preprocess it to ensure it's suitable for analysis. If the data is unlabeled, label it for training purposes. Split the data into training, testing, and evaluation datasets in a representative manner. Proceed to train your model using the training dataset and evaluate its performance using the testing dataset. Assess the model's generalization using the evaluation dataset. If necessary, repeat the process, gathering additional data and improving the model's performance.

3.2.2 Mobilenetv2 Plus SVM Techniques Usage to Determine the Best Suited Model

To develop and evaluate a model for a specific task, follow these steps. Firstly, gather relevant data by collecting a dataset of images. Secondly, preprocess the images by resizing them to a standard size and normalizing the pixel values. Next, train and evaluate the MobileNetV2 model on the dataset. Then, train the SVM model using features extracted from the MobileNetV2 model and evaluate its performance. Compare the performance of both models on the same dataset to determine the most suitable one for your task. Fine-tune the chosen model by adjusting its hyperparameters to achieve better performance. Test the final model on a separate dataset to evaluate its

generalization performance. Analyze the results and draw conclusions about the effectiveness of using MobileNetV2 and SVM techniques for your specific task. Finally, write a report detailing the methodology, results, and conclusions of your experiments.

3.2.3 CNN and SVM Analysis with Varying Filter Sizes

To perform a CNN and SVM analysis with varying filter sizes, follow these steps. First, gather a suitable dataset that includes labelled data for training and evaluation. Ensure the dataset aligns with your problem. Next, preprocess the data by cleaning and applying necessary transformations like normalization, resizing, and splitting it into training and test sets.

Then, implement a CNN architecture with multiple convolutional layers using different filter sizes. Train and evaluate the CNN model using the prepared training and test datasets. Extract features from the trained CNN model and feed them into an SVM classifier. Train and evaluate the SVM model using the same datasets. Evaluate the performance of both models by comparing their results using appropriate evaluation metrics such as accuracy, precision, recall, or F1-score. Analyze the performance of different filter sizes on both the CNN and SVM models. Select the best model based on the analysis, considering factors like overall accuracy, computational efficiency, and interpretability. Fine-tune the chosen model by adjusting hyperparameters, exploring different architectures, or employing techniques like cross-validation for robustness. If the results are inconclusive or unsatisfactory, repeat the process with different variations or additional experiments to gain further insights.

3.2.4 Building a Recommendation System for Tomato Disease-Related Content

To build a recommendation system for tomato disease-related content, start by gathering a dataset consisting of articles, research papers, and other relevant textual information. Then, preprocess the data by cleaning it, converting text to lowercase, and removing stop

words. Next, use the Universal Sentence Encoder (USE) to embed the preprocessed text into fixed-length vector representations that capture semantic information.

Afterward, construct a similarity matrix by computing pairwise similarity between the embedded text vectors using cosine similarity or other similarity metrics. Implement a recommendation algorithm that takes user queries or inputs and finds similar tomato disease-related content based on the similarity matrix. To ensure the system's effectiveness, evaluate its performance metrics such as precision, recall, or F1 score using a validation set. Refine the system based on the evaluation results. Finally, deploy the recommendation system into an application or platform for easy access and continuously monitor and collect feedback to enhance its effectiveness and accuracy over time.

3.2.5 Deployment of the Proposed Model

To develop a mobile application for early tomato leaf disease detection, the following steps will be taken.

- a) First, create a mobile application compatible with smartphones and tablets that incorporates the proposed disease detection model.
- b) Second, optimize the model for efficient operation on portable smart devices by employing techniques such as model compression and quantization.
- c) Third, design a user-friendly interface within the application to enable easy interaction with the disease detection system, providing clear instructions and visualizations of results.
- d) Fourth, integrate data acquisition capabilities to capture tomato leaf images securely for analysis by the model. Next, the model will be trained and validated using a diverse dataset of healthy and diseased tomato leaves to ensure reliable disease detection. The trained and optimized model, along with the mobile application, will be deployed on

portable smart devices, making it available through app stores and ensuring compatibility with different operating systems and devices. The mobile application will offer real-time assistance by promptly analyzing captured images and providing disease detection results, along with management recommendations.

Continuously updating and improving the system is crucial. Regular updates will be made based on user feedback and advancements in disease detection techniques, incorporating new features, bug fixes, and performance enhancements. User support channels will be provided to address queries and issues, and educational resources will be offered to farmers to help them make informed decisions regarding disease management. Collaboration with farmers, agricultural experts, and stakeholders will be fostered to gather feedback and continuously enhance the system's functionality and effectiveness in assisting farmers.

3.3 Data Collection

This section discusses the data gathering method adopted in this research work. The researcher combines both secondary and primary sources of data in order to understand the study. The primary data collection was done by collecting live images of both clean and infected tomato which was used to perform model testing and to evaluate the system performance when tested with a live image. The secondary source of data was curled from Kaggle community which is a portal where datasets are made available for AI and intelligent system analysis. The dataset was created by Kaustubh (2019) and was uploaded to Kaggle. The size of the dataset is 188MB which contains 11,000 files separated by two folders namely training set and the validation set. The classes of tomatoleaves diseases are listed in Table 3.2.

Table 3.2: Nine classes of tomato leaf disease and a healthy class considered.

S/N	Disease Name	Family Name	Symptoms
-----	--------------	-------------	----------

1	Mosaic Virus	<i>Tobamoviridae</i>	Tiles, wrinkles, reduction and curvature of leaflets, and irregular ripening of fruits.
2	Tomato Blind Spot	<i>Corynesporascaceae</i>	Pinpoint-sized, water-soaked spots on the upper leaf surface, Initial symptoms on stems and petioles are lesions that are slightly sunken, brown flecks.
3	Bacteria Spot	<i>Xanthomonadaceae</i>	Small, round, water-soaked spots that gradually turn dark-brown or black and are surrounded by yellow halo. The size of lesions is variable, but rarely develops to more than 1/10 inch in diameter.
4	Tomato yellow leaf curl virus	<i>Geminiviridae</i>	Small leaves that become yellow between the veins. The leaves also curl upwards and towards the middle of the leaf.
5	Tomato late blight	<i>Peronosporaceae</i>	Tomato leaves are irregularly shaped, water-soaked lesions, often with a lighter halo or ring around them.
6	Leaf Mold	<i>Mycosphaerellaceae</i>	The oldest leaves are infected first. Pale greenish-yellow spots, usually less than 1/4 inch, with no definite margins, form on the upper sides of leaves.
7	Early Blight	<i>Pleosporaceae</i>	Small dark spots form on older foliage near the ground. Leaf spots are round, brown and can grow up to 1/2 inch in diameter
8	Tomato red spider mite	<i>Tetranychida</i>	Tomato red spider mite feeding causes whitening or yellowing of leaves, which then dry out and eventually fall off.
9	Septoria leaf spot	<i>Mycosphaerellaceae</i>	Appears on the lower leaves after the first fruit sets. Spots are circular, about 1/16 to 1/4 inch in diameter with dark brown margins and tan to gray centers with small black fruiting structures
10	Tomato Healthy	<i>Solanaceae</i>	Healthy

3.3.1 Description of the dataset

For the training set, we have 1,000 records in each of folders of the 10 classes which sum up to 10,000 records for the training set. Also, the validation set contains 100 files each in each folder and a total of 1,000 files belonging to the 10 classes.

We first trained the model using the default split as contained in the dataset repository and we compared the result with our splitting strategy. Also, to avoid problems of over-fitting due to errors and outliers, we devise a splitting strategy where we merge the entire folders and use 8,800 (80%) for training, 20% of the datasets for validation. Augmentation has remained the best way to increase and improve model performance without an additional cost.

To benefit from such a golden statement, we perform some dataset re-scaling, shearing and flipping. We applied horizontal flip of 50%, shear range of 20% and finally the datasets were re-scaled by multiplying by 1 and dividing by 255 in order to keep the pixel values uniformly distributed.

3.4 System Analysis Procedures

This section describes the entire designs that will help in achieving both the major and the desired specific objectives in the research work.

3.4.1 Analysis of the existing system

Before the advancements in technology based on deep learning, traditional or conventional techniques for predicting and managing tomato leaf diseases relied on manual observation and analysis. Here are some of the common techniques used:

1. **Visual Inspection:** Farmers and experts would visually examine the tomato plants for symptoms of leaf diseases. They would look for visual cues such as discoloration, wilting, spots, lesions, deformities, or abnormal growth patterns on the leaves.
2. **Field Surveys:** Agricultural experts would conduct field surveys to monitor the prevalence and severity of tomato leaf diseases. They would visit farms and collect data on the occurrence and distribution of diseases, noting the affected plants and the symptoms observed.
3. **Symptomatology:** Symptoms displayed by the infected tomato leaves were observed, documented, and analyzed. Specific patterns and characteristics of

diseases, such as the shape, size, color, or texture of lesions, would be studied to identify the causal agents or pathogens.

4. **Manual Identification:** Agricultural experts and plant pathologists would rely on their knowledge and experience to identify the specific disease affecting the tomato leaves. They would compare the observed symptoms with their understanding of various pathogens and diseases, using field guides, manuals, and reference books.
5. **Laboratory Analysis:** In cases where visual inspection alone was not sufficient, laboratory analysis was employed. Samples of diseased tomato leaves would be collected and taken to the lab for further investigation. Microscopic examination, culturing of pathogens, and biochemical tests were performed to identify the specific disease-causing agents.
6. **Disease Risk Assessment:** Based on historical data, weather conditions, and other factors, experts would assess the risk of disease outbreaks. This involved analyzing environmental factors, such as temperature, humidity, rainfall, and the presence of potential vectors or carriers of diseases.
7. **Crop Rotation and Sanitation:** Crop rotation, where tomato plants were not grown in the same field consecutively, was a common practice to reduce the build-up of diseases in the soil. Sanitation measures like removal and destruction of infected plants, pruning, and cleaning of equipment were employed to minimize the spread of diseases.
8. **Chemical Sprays and Fungicides:** To manage or control the spread of diseases, farmers would often use chemical sprays and fungicides. These treatments were applied preventively or curatively, based on the knowledge of disease cycles and the specific pathogens involved.

These traditional techniques relied heavily on human expertise and experience. While they provided valuable insights and guidance, they were limited by the subjectivity of visual observations, the need for manual analysis, and the inherent complexity of diagnosing and managing diseases. The advent of deep learning and advanced technology has since revolutionized disease prediction and management in agriculture, offering more accurate and efficient solutions.

It is true that tomatoes are one of the most consumable and essential crops in our world. They vary in size based on the soil topology and the kind of fertilization they are exposed to. Farmers sometimes experience low tomato harvest because of tomato plant diseases. There are numerous literatures proposed by researchers to tackle these identified tomato leaf diseases.

We analyzed the existing tomato leaf diseases classification models based on the model type, datasets, recommender system and mobile integration. Many of the research papers that were consulted, used the Convolution Neural Network (CNN) model to extract features and also perform classification. They used a custom-built CNN which in their findings showed a better performance.

However, their results were based on hyper-parameter tuning which is a very rigorous step. Data is like food for a CNN model. They tend to produce excellent results when the amount of training data is so large.

Furthermore, there are still hidden features inherent in the input data. The weaknesses of the existing models can be summarized as:

- (a) Most of the tomato leaf disease prediction models used custom CNN for feature extraction as well as classification.
- (b) All existing tomato leaf disease prediction models only stop at the classification phase. They do not provide a recommender system which can help to inform the

farmers of the symptoms and as well as the preventive tips they should apply to curb or mitigate these diseases at their early stage.

- (c) Most tomato leaf disease models are not integrated into a mobile platform for use by the farmers or horticulturist. A model may have high accuracy, good recall/precision tradeoff but perform poorly during model integration because of several factors inherited from the choice of model.
- (d) Using a fully connected layer in a model could significantly increase the size of the model and increase the inference time, which will definitely increase the overall computational time.

Architecture of the existing system

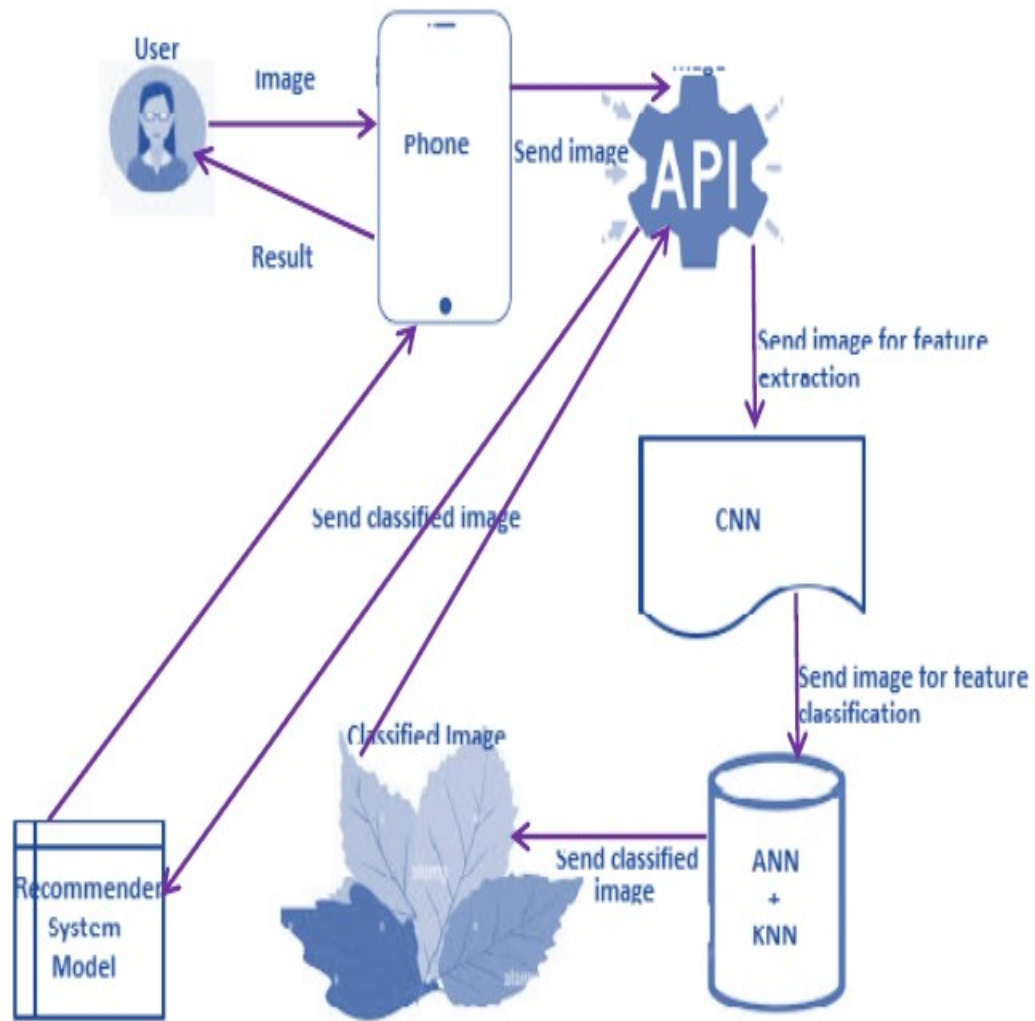


Figure 3.2: Architecture of the existing system

In figure 3.2 we observed that developing a mobile system architecture for tomato leaf disease detection and classification without incorporating recommender systems and hybridized techniques poses several limitations. First and foremost, the absence of a recommender system means that users may not receive personalized recommendations or guidance on how to effectively manage and mitigate tomato leaf diseases. A recommender system could analyze historical data, user preferences, and real-time environmental conditions to provide tailored advice, such as suggesting specific treatments, fungicides, or preventive measures, which can significantly enhance the system's usability and value to farmers. Without this feature, users may miss out on crucial information and strategies for disease control, potentially leading to suboptimal outcomes in their crop management efforts. Secondly, the exclusion of hybridized techniques in disease detection and classification may result in lower accuracy and reliability. Hybridized techniques often combine multiple approaches, such as machine learning algorithms, image processing, and sensor data fusion, to improve the overall performance of disease detection systems. Without these techniques, the mobile system may struggle to handle the complexity and variability of tomato leaf diseases, leading to higher rates of false positives or false negatives. Moreover, hybridized techniques can enhance the system's adaptability to changing environmental conditions and disease strains, which is crucial in agriculture where disease dynamics can evolve rapidly. In summary, the absence of recommender systems and hybridized techniques can limit the effectiveness and robustness of the mobile system, hindering its ability to provide valuable support to farmers in managing tomato leaf diseases.

3.4.2 The proposed system architecture

The proposed system architecture reveals the true blueprint of the internal working mechanism of the proposed model.

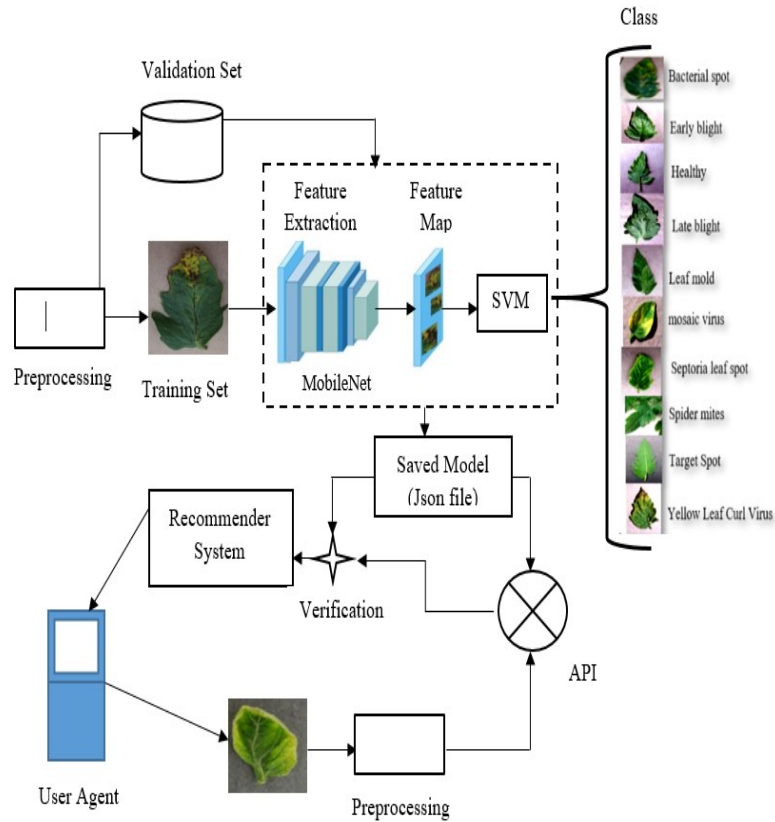


Figure 3.3: Propose architectural design for tomato leaf disease predictive model.

In Figure 3.3, we employed a transfer learning approach to extract deep enriched features from our training samples of tomato images. During the preprocessing phase, the datasets are examined for correct name labels as well as masking was done around the region of disease in order to clearly distinguish the 10 different classes (including a healthy tomato leaf).

The datasets were divided into different ratios based on different splitting strategies and the results are presented in chapter 4 of this research report. The base neural network model was MobileNet V2 which is a lightweight State of The Art (SOTA) deep learning model deployable in mobile application. The input dimension into this model was $224 \times 224 \times 3$ where 3 signifies the channel (RGB), 224 signifies the height and width of the training set.

The mobileNet preprocessing function was used to normalize the training samples to a scale range from 0 to 1. We fixed the feature extraction part of the network by setting base_model trainable option to false. Also, the weights are being loaded from a state dictionary which was saved by training mobileNet with ImageNet of about 22k images. ImageNet is a very large dataset which contains rich information suitable to train any deep learning model. The total number of trainable and non-training parameters were 284, 860 and 640 respectively.

The size of each extracted feature was 1024 which was the size of the global average 2D. The extracted features were saved to a Numpy array which serves as input to train the Support Vector Machine (SVM) in order to predict the class which, the extracted feature belongs. SVM was used as the classification technique because they are well suited for classifying complex small to medium datasets and good for the classification of multi-class data as compared to the fully connected layer.

SVM was imported from the sklearn python library. For the loss function, we used the hinge loss type which finds the maximum point between 0 and (t-1). Hinge loss is equal to zero when $t \geq 1$ and it is equal to -1 when $t < 1$.

The imported features in the numpy file are read and divided into X_train, Y_train and X_test ,Y_test following the splitting strategies we earlier recommended. To control overfitting problem in the multiclass SVM, we applied the kernel trick approach. Different kernel options were also investigated while performing different hyper-parameter tuning. For instance, using polynomial kernel, we tuned the degrees of the polynomial kernel from 3 to 10 to see how the model performance in terms of instance range of influence Vs decision boundary. Also, using Gaussian RBF kernel, the gamma options were tuned in the range of 0 to 5. The best combination result was adopted in the design of our prototype. The accuracy graphs, as well as the recall and precision scores were recorded and reported in chapter four.

Also, we performed a precision/recall tradeoff comparison and output the result using the ROC graph. The weights from the feature extraction phase and the regressor head were saved to a json file which we call savedmodel as identified in Figure 3.3. This savedmodel was called each time a prediction needs to be made by end users uploading image of tomato leaf. After prediction of the class of disease, it is necessary to provide farmers or end users with a recommender tip on how to tackle the identified disease following collected information gotten from Horticulture and botanist experts.

The recommender system was designed using Universal Sentence Encoder (USE) model, which is also a Deep Learning Transformer Network Model for Natural Language Processing (NLP) and was implemented using seaborn package. The USE is suitable for less supervised training data. It is easy to get sentence level embedding. It is very compatible in transfer learning as each sentence embedding was used to ascertain the sentence level similarity meaning.

It is used to encode feature vectors used for text classification. We first downloaded the pre-trained USE from Tensorflow-hub. Again, we selected 3 best recommended advice provided by experts for each class of tomato leaf disease. The output of the USE is always a 5 x 512 matrix dimension, where each expert tip is a vector of size 512. When a class of disease is predicted, the identified class is tied to this matrix and thus, the matrix values are then converted to text using similarity sentence level meaning.

Again, from Figure 3.3, the input response sent to the end user is a predicted class name as well as the useful recommender tips to help get rid of such disease as early as possible. The API calls the savedmodel which is in **.h5 format**. This **savedmodel** predicts the class of the disease and sends the result to the Application Programming Interface (API) which then forwards the received results to the recommender system.

However, the prediction from the saved model is still forwarded to the recommender system as a way to ensure that the predicted class from the API is correct, which is useful for data validation/verification in order to reduce high false positive rate.

3.4.3 Mathematical Model: Tomato Disease Classification and Recommender System

To create a mathematical model for tomato disease detection using MobileNetV2 + SVM with a recommender system, we need to outline the various components and steps involved in the process. Let's break it down into steps:

Step 1: Data Collection and Preprocessing: We Collected dataset of tomato images that contains both healthy and diseased tomatoes. Preprocess the images to resize them to a consistent size and normalize pixel values.

Step 2: Feature Extraction with MobileNetV2: MobileNetV2 a pre-trained deep learning model was used to extract features from the pre-processed tomato images. This step converts the images into a compact representation in a feature space.

Step 3: SVM Classifier: We trained a Support Vector Machine (SVM) classifier on the extracted features. The SVM will learn to separate the features corresponding to healthy and diseased tomatoes in the feature space.

Step 4: Recommender System: A recommender system was developed that takes input images of tomatoes and output recommendations based on the SVM classifier's predictions.

Step 5: Mathematical Model: Let's denote the following:

1. **Dataset and Labels:** Let X be a dataset of tomato images, where each image x_i is represented by its pixel values. The dataset consists of K samples: $X = \{x_1, x_2, \dots, x_K\}$.

Let Y be the set of possible tomato disease categories, where each element y_i represents a label:

$$Y = \{0 \text{ (healthy)}, 1 \text{ (diseased - 1)}, \dots, N \text{ (diseased - N)}\}.$$

2. **Preprocessing:** Define the function $f(X)$ that takes in the tomato images and returns the preprocessed data in the correct format: $f(X) = X^{\wedge}$, where $X^{\wedge} = \{x_{_1}^{\wedge}, x_{_2}^{\wedge}, \dots, x_{_K}^{\wedge}\}$ is the preprocessed data.
3. **Feature Extraction:** Apply the MobileNetV2 model M to the preprocessed images $X^{\wedge}$ to extract features: For each $x_{_i}^{\wedge}$ in $X^{\wedge}$, $M(x_{_i}^{\wedge}) = g(x_{_i}^{\wedge})$, where $g(x_{_i}^{\wedge})$ represents the function that processes the output of MobileNetV2 to obtain the extracted feature $Z_{_i}$.
4. **SVM Classifier:** Train an SVM classifier using the extracted features $Z_{_i}$ and the corresponding labels $y_{_i}$: For each i , $SVM(Z_{_i}) = y_{_i}$, where $SVM(Z_{_i})$ is the prediction of the SVM classifier for the i th image.
5. **Recommender System:** Design the recommender system P to take the predicted labels $y_{_i}$ as input and output control measures for each image: For each $y_{_i}$, $P(y_{_i}) = P(x_{_i})$ represents the control measures recommended for the corresponding image $x_{_i}$.

General Equation: The SVM classifier predicts the labels for the extracted features:

$$SVM(M(X^{\wedge})) = \{y_{_1}, y_{_2}, \dots, y_{_K}\}.$$

Then, the recommender system provides control measures based on the predicted labels:

$$P(SVM(M(X^{\wedge}))) = \{P(y_{_1}), P(y_{_2}), \dots, P(y_{_K})\}.$$

3.4.3.1 Proposed code segment adopted.

1. **Tomato Leaf Disease Detection using MobileNetV2 as Feature Extraction:** The first step is to use MobileNetV2, a pre-trained convolutional neural network, to extract features from tomato leaf images. We will freeze the convolutional layers and only train the classification layers on top of it.

3.4.3.2 Coding using python program:

1. Load and preprocess the tomato leaf image dataset.
2. Load the pre-trained MobileNetV2 model without the top classification layers.
3. Add custom classification layers on top of MobileNetV2.
4. Compile the model with an appropriate loss function and optimizer.
5. Train the model on the dataset.
6. Save the trained model for future use.
2. Step 1: Import the necessary libraries.
3. `import os`
4. `import numpy as np`
5. `import tensorflow as tf`
6. `import tensorflow_hub as hub`
7. `from tensorflow.keras.applications import MobileNetV2`
8. `from tensorflow.keras.preprocessing.image import ImageDataGenerator`
9. `from sklearn.svm import SVC`
10. `from sklearn.metrics import accuracy_score`
11. Step 2: Load and preprocess the dataset
12. Make sure your dataset is organized into subfolders with each class name, containing the corresponding images. For example, the directory structure could be like this:
 13. `dataset/`
 14. `|- diseased_class_1/`
 15. `|- diseased_class_2/`
 16. `|- ...`
 17. `|- diseased_class_9/`

```

18.     |- healthy_class/
19.     dataset_path = 'path/to/dataset/'
20.     image_size = (224, 224) # Image size required for MobileNetV2
21.     # Data augmentation to prevent overfitting
22.     datagen = ImageDataGenerator(
23.         rescale=1./255,
24.         shear_range=0.2,
25.         zoom_range=0.2,
26.         horizontal_flip=True,
27.         validation_split=0.2 # 80% training, 20% validation
28.     )
29. train_generator = datagen.flow_from_directory(
30.     dataset_path,
31.     target_size=image_size,
32.     batch_size=32,
33.     class_mode='categorical',
34.     subset='training'
35. )
36. validation_generator = datagen.flow_from_directory(
37.     dataset_path,
38.     target_size=image_size,
39.     batch_size=32,
40.     class_mode='categorical',
41.     subset='validation'

```

```

42. )

43. Step 3: Feature extraction using MobileNetV2

44. base_model = MobileNetV2(input_shape=(224, 224, 3),
45. include_top=False,
46. weights='imagenet')
47. # Freeze the base model layers
48. base_model.trainable = False
49. # Extract features from MobileNetV2 for training and validation datasets
50. train_features = base_model.predict(train_generator)
51. validation_features = base_model.predict(validation_generator)

52. Step 4: Train SVM classifier

53. # Flatten the extracted features
54. train_features_flatten = train_features.reshape(train_features.shape[0], -1)
55. validation_features_flatten =
    validation_features.reshape(validation_features.shape[0], -1)

56. # Get labels for training and validation datasets
57. train_labels = train_generator.classes
58. validation_labels = validation_generator.classes

59. # Train SVM classifier

60. svm_classifier= SVC(kernel='linear', C=1.0, random_state=42)

61. svm_classifier.fit(train_features_flatten, train_labels)

# Make predictions on validation set

62. predictions = svm_classifier.predict(validation_features_flatten)

# Calculate accuracy

```

```

63. svm_accuracy = accuracy_score(validation_labels, predictions)

64. print("SVM classifier accuracy:", svm_accuracy)

65. Step 5: Universal Sentence Encoder for a recommender system

# Load Universal Sentence Encoder

66. embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")

70.# Example text descriptions for each disease class

71. disease_descriptions = {
72.     "diseased_class_1": "Description for diseased class 1.",
73.     "diseased_class_2": "Description for diseased class 2.",
74.     # Add descriptions for other classes
75.     "diseased_class_9": "Description for diseased class 9.",
76.     "healthy_class": "Description for healthy class."
77. }

# Function to get the recommendation for an input query

78. def get_recommendation(input_text):
79.     # Get embeddings for input query and disease descriptions
80.     query_embedding = embed([input_text])[0]
81.     description_embeddings = embed(list(disease_descriptions.values()))
82.     # Calculate similarity scores using dot product of embeddings
83.     similarity_scores = np.inner(query_embedding, description_embeddings)
84.     # Get index of the most similar description
85.     most_similar_index = np.argmax(similarity_scores)
86.     # Get the corresponding disease class
87.     recommended_class = list(disease_descriptions.keys())[most_similar_index]

```

88. return recommended_class

A comprehensive pipeline for the tomato leaf disease detection project have established. Inference is conducted using the SVM classifier, while the Universal Sentence Encoder-based recommender system suggests disease classes based on text descriptions. This section provides an overview of the algorithms employed in designing the proposed system. Feature extraction relies on the MobileNetV2 algorithm, classification utilizes SVM, and the recommender system is built upon the Universal Sentence Encoder (USE). The objective is to provide recommendations for controlling various classes of tomato leaf diseases. The MobileNetV2 algorithm is employed for feature extraction, involving a global average pooling with a feature size of 1024. This process is iterated until the batch size of training samples is exhausted, and each output from the feature extraction step is saved as an .npy file in a local directory. Simultaneously, the SVM algorithm is used for multi-class prediction, considering the distance of each support vector to a hyperplane. Various kernel sizes, including linear, Gaussian, and polynomial functions, are tested. Hyperparameter tuning involves adjusting different C values to achieve a well-defined hyperplane. The Universal Sentence Encoder (USE) algorithm, implemented in the latter part of the provided code, is designed based on recommendations from experts in botany and horticulture. Expert recommendations are embedded into vectors, normalized between 0 and 1, and assigned unique keys. These keys serve as identifiers, linking each class of tomato disease prediction to the corresponding vectors.

3.5 Data Modeling

This section covers every activity which shows how the process of the proposed system, and the data elements are used to model the proposed system. In a holistic view, data modeling refers to the graphical representation of data model of any information system using formal techniques. Data modeling can also be seen as the process of creating a

visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes. We modeled the data flow of the proposed system using data-flow diagrams, class diagram and entity-relationship diagrams.

3.5.1 Data flow diagram (DFD)

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. There are several notations for displaying data-flow diagrams. We used the notation styles where lozenges (rounded rectangles) represent processes, squares or rectangles denote external entities, open-ended rectangles symbolize data storage, and arrows indicate data flow.

The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. A data-flow diagram has no control flow which implies that they are devoid of decision rules and has no loops.

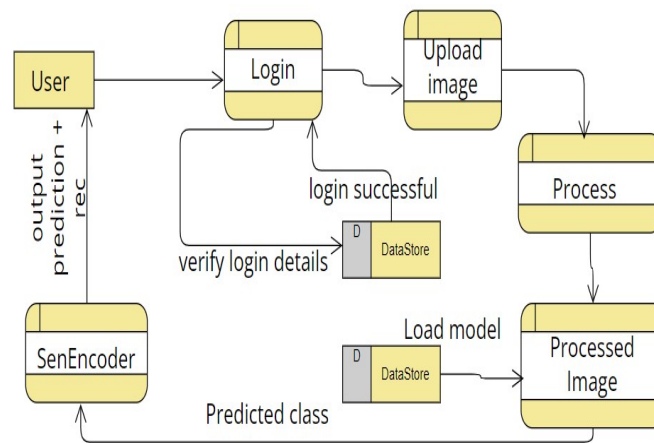


Figure 3.4: DFD of the proposed system

Figure 3.4 shows the DFD of the proposed system, in which the flow from the external entity called User down to the final output going back to the user. End users first collect different tomato leaves then create an account if they are first timer to the application or login with existing details. The system carries a verification step to check if password entered has been captured during enrolment stage. Then the database sent a response through the server which enables the user to upload the image and perform prediction. The end users have no connection about how the process phase is done. From the process phase down to the SenEncoder phase are kept hidden from the end users as our end users are farmers or botanists who are interested to know the class of the tomato leaf diseases and possible tips of controlling the diseases as recommended by the proposed model.

3.5.2 Entity relationship diagram (ERD)

An ERD is a form of flowchart that shows the relationships between "entities" like people, things, or concepts inside a system. ERDs are frequently used in the disciplines of software engineering, corporate information systems, education, and research to build or troubleshoot relational databases. The ERDs or ER Models employ a predetermined collection of symbols to represent the interconnectivity of entities, connections, and their qualities. These symbols include rectangles, diamonds, ovals, and connecting lines. They have verbs for connections and nouns for entities, mirroring the grammatical framework.

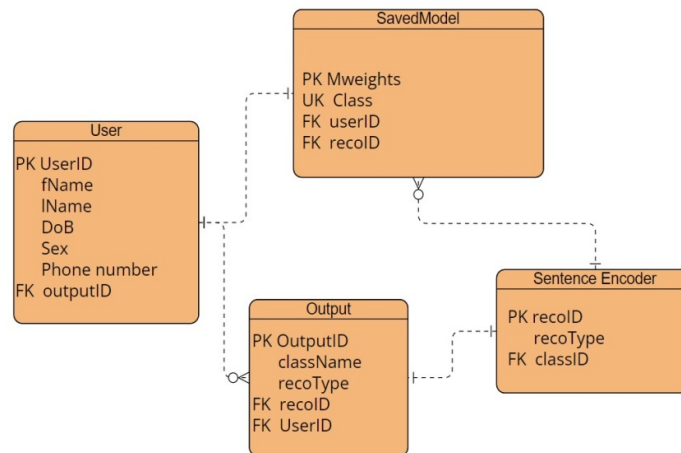


Figure 3.5: Entity relationship diagram of the proposed system

Figure 3.5 shows the ERD of the proposed system, where the end user table contains information peculiar to him/her. In the end user table, we have the outputID which provides an avenue for end users to recall an old prediction if they forgot some of the recommendations provided by the system. Both the sentence encoder and the model are linked with one vs zero to many relationships. Similarly, comparing the model table with the user table, a user can call the model as much as the user wants but only one prediction will be sent at end to the user.

3.5.3 Class diagram

One of the most helpful forms of diagrams in UML is a class diagram, which accurately depicts a system's structure by modeling its classes, properties, operations, and connections among objects. A visual notation used to build and visualize object-oriented systems is the UML Class diagram. An example of a static structural diagram is a class diagram in the Unified Modeling Language (UML), which demonstrates a system's:

- i. Classes
- ii. Their attributes
- iii. Operations (or methods)
- iv. The relationships among objects

A Class is an object's blueprint. Classes and objects are mutually exclusive. Without discussing the other, we cannot discuss the first. And since we utilize classes to generate objects, the entire goal of object-oriented design is not about objects. A class is not the actual thing; but rather, it is a description of what an object would be.

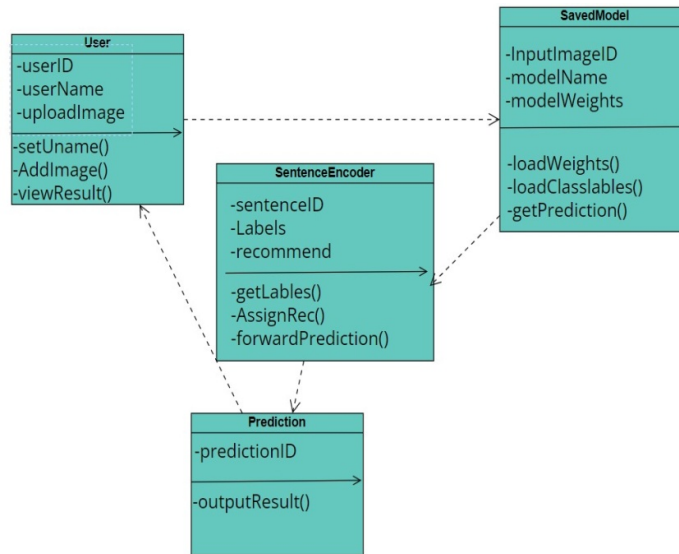


Figure 3.6: Class diagram of the proposed system

Figure 3.6 illustrates the class diagram of the proposed model, having four objects and the relationship between the objects.

3.6 System Modeling

System modeling is the process of creating abstract representations of a system, with each model offering a distinctive viewpoint. It has to do with graphical notation, which is now nearly invariably based on notations in the Unified Modeling Language, to depict a system (UML). Models are used to interact with consumers and to assist analysts in comprehending how the system works. Different viewpoints on the system can be explained through models:

- i. An external viewpoint through which you simulate the system's context or environment.
- ii. An approach to modeling interactions where you consider how a system interacts with its surroundings or with its individual parts.

- iii. A structural viewpoint, in which you represent a system's structure or the structure of the data the system processes.
- iv. A behavioral approach through which you simulate the system's dynamic behavior and how it reacts to events.

In this section, we used the five (5) important and most useful UML diagrams such as activity, use case, sequenced and system flowchart to model the proposed system. **3.6.1**

System flowchart

A system flowchart can be seen as a diagram that shows a process' individual phases in their proper order. It is a general tool that may be customized for a wide range of uses and can also be used to define several processes, including a project plan, an administrative or service process, or a manufacturing process. It is one of the seven fundamental quality tools and a typical tool for process analysis.

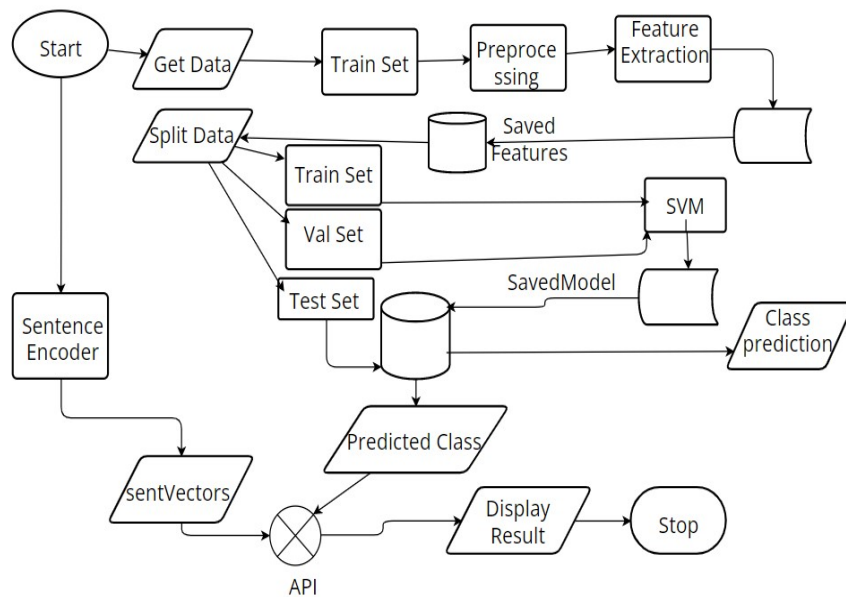


Figure 3.7: System flowchart of the proposed system

Figure 3.7 represents the proposed system flowchart, which has to do with the model training phase and the construction of the sentence encoder. The training phase starts by getting tomato datasets then using the training portion of the datasets in training a light

weight deep learning model called the MobileNetV2. We then get the features from the last layer of the deep learning model. We later designed a support vector machine (SVM) to replace the fully connected layer (FC) to make accurate prediction of each disease class using the saved features.

We save the weights of both the feature extraction and the SVM part to a json file which will be called up anytime a prediction is made. This action was carried out because of the training time and the urgent need for end users to make predictions. The split data divides the saved features into train, test and validate. The test samples were used to get the true accuracy of the entire model. For each class prediction made, a corresponding recommendation is provided by the sentVectors which harbours weights of each sentence and provide a recommendation based on a particular class of prediction yielded by the model. The recommendations were ascertained from the field of experts in plant diseases. To model the activity of a user, we designed another flowchart in Figure 3.8 which shows the process flow of the end user and the level of interaction he/she has with the proposed system.

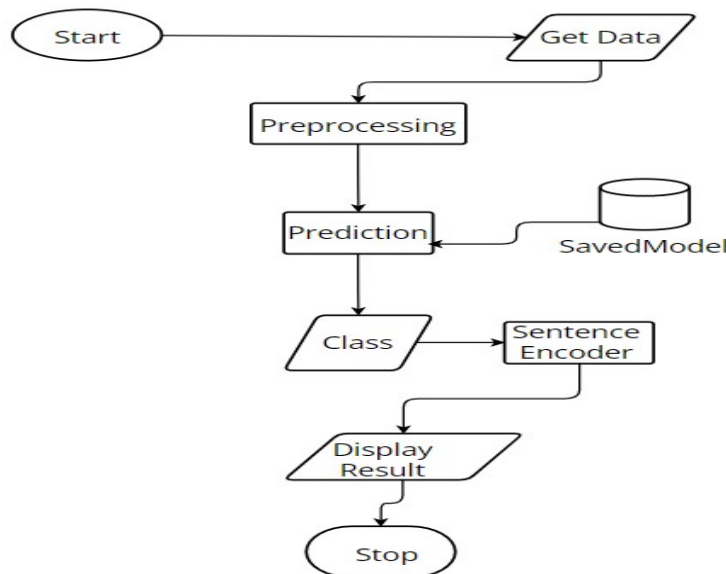


Figure 3.8: End User flowchart

3.6.2 Activity diagram

Activity diagrams are visual depictions of processes with choice, iteration, and concurrency supported by activities and actions. Activity diagrams are used in the Unified Modeling Language to represent organizational and computational processes (i.e., workflows), as well as the data flows that interact with the linked activities. Activity diagrams can incorporate features that represent the movement of data between activities through one or more data stores, even if their primary purpose is to depict the overall flow of control. Activity diagram is a more sophisticated form of a flow chart that models the transition from one activity to another.

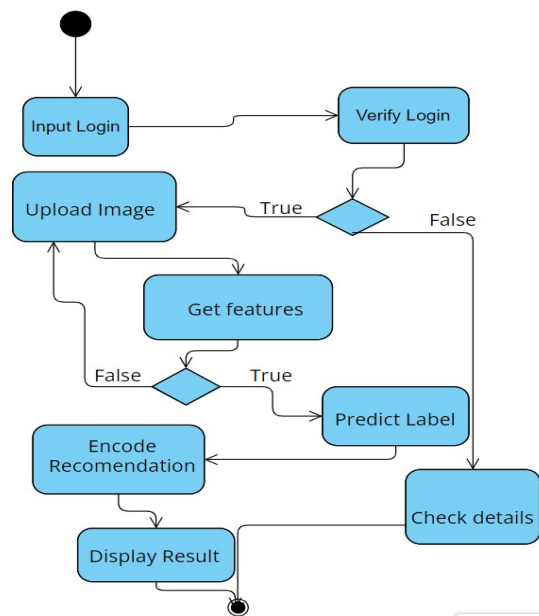


Figure 3.9: Activity diagram of the proposed system

Figure 3.9 depicts the activity diagram of the proposed system in which the end users will be profiled at the early stage of the system. The system will always request them to put correct password as a measure to ensure security. Ordinarily, a prediction is very much okay to a data analyst or a scientist working in plant disease centers. This study incorporated a recommender system in order to bridge the gap between the prediction

made and the need to seek expert judgment or get recommendation. The system always checks for a tomato leaf because uploading leaf different from tomato will force the system to produce a false positive result which is detrimental to the plant leaf uploaded. Hence there is a feature similarity check between every leaf uploaded or checked.

3.6.3 Sequence Diagram

A sequence diagram's major objective is to specify the event sequences that lead to a particular intended conclusion. Most sequence diagrams will explain what messages are exchanged between a system's objects as well as the order in which they occur. The emphasis is less on the messages themselves and more on the order in which messages occur.

This information is depicted in the picture along both the horizontal and vertical axes: the horizontal axis lists the object instances that the messages are delivered to, while the vertical axis illustrates the temporal sequence of calls and messages from top to bottom. Sequence Diagrams are interaction diagrams that describe the steps used to complete a task. They depict how items interact within the framework of a cooperation.

Sequence Diagrams are time-focused, and they use the vertical axis of the diagram to represent time in order to highlight the sequence of the interaction and when messages are conveyed. At the initial stage, the system captures the personal information of end users of the system. Next is to ask users to login their details and it verifies the authenticity of their passwords and grants them an entry to predict tomato leaf disease. If the entered password is not correct, they are being asked to go back and re-enter their password hence we use an iterator to define such activity in the activity diagram.

Whenever a leaf which is not tomato is uploaded, the model checks its feature space and if it has no categories of features like the feature from the leaf uploaded, the system prompts the user that uploaded image is not related to tomato and request for a tomato leaf to be

uploaded. Hence the system performs some structural similarity check to first verify that what it needs to predict is a tomato leaf disease. Next after prediction is to check if the predicted class has a corresponding recommendation by calling the sentence encoder. If recommendation is present as seen in Figure 3.10 below, then the prediction is successful, and the end user sees the result or the output of the prediction. It is worthy of note that tick arrows represent messages while the dotted arrows are replies sent back.

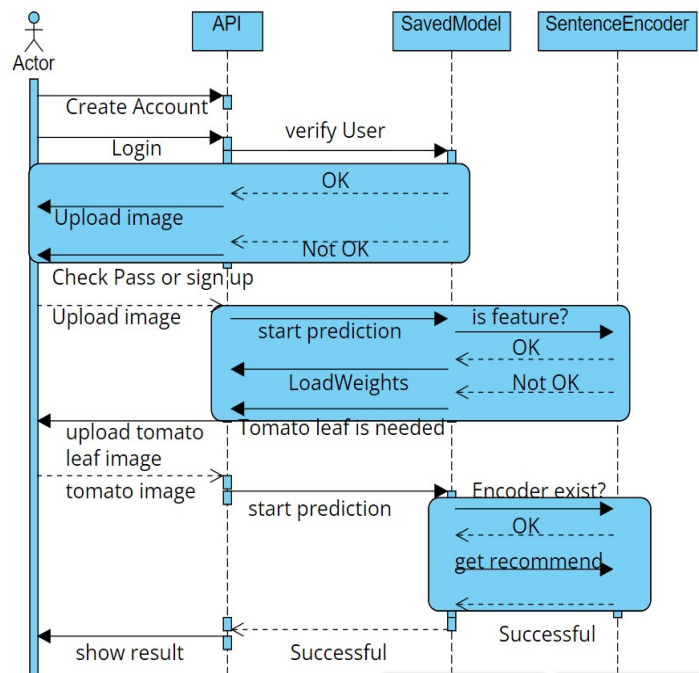


Figure 3.10: Sequenced diagram of the proposed system

3.6.4 Use Case Diagram

Use case diagrams, also known as behavior diagrams, are used to illustrate a series of tasks that a system or systems (the topic) should or might do in coordination with one or more external users of the system (actors). Each use case should deliver a tangible benefit to the system's actors or other stakeholders. A graphical representation of a user's potential interactions with a system is called a use case diagram. A use case diagram, which is frequently complemented by other types of diagrams, displays the numerous use

cases and user types of the system has. Either circles or ellipses are used to depict the use cases. Stick representations of the actors are frequently used.

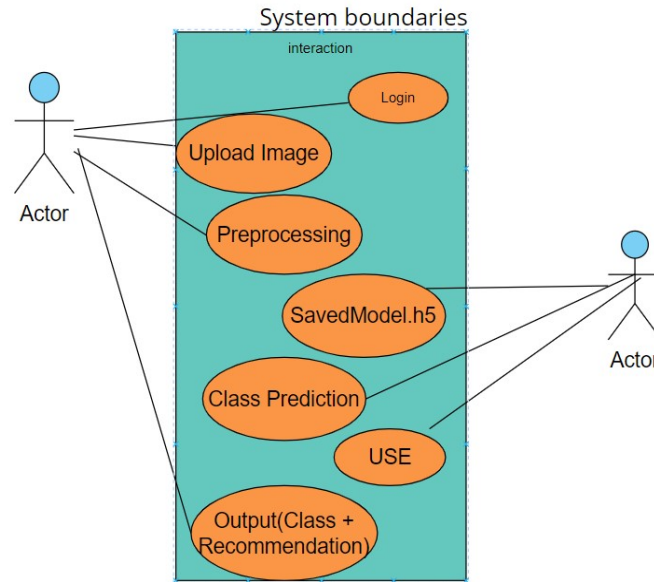


Figure 3.11: Use case diagram of the proposed system.

Figure 3.11 represents a use case diagram of the proposed system in which the end user to the system is depicted by an actor. He/she has some levels of interaction with the system. Similarly, on the right-hand side, the system administrators who have some levels of interaction with the system such as to maintain updates and fixes possible issues within the system.

3.7 System requirement specification

Requirements analysis is a crucial step in determining if a system or software project will be successful. Functional and non-functional requirements are the two main categories of requirements.

3.7.1 Functional requirements

These are the specifications that the system must meet to satisfy the end user's fundamental needs.

Table 3.3: Basic system functional requirements specification of the proposed model

Functional Requirement Number	Functional requirements
FR1	Snap leaves with camera
FR2	Pick leaves pictures from gallery
FR3	Predict tomato disease with its class
FR4	Save predictions to database
FR5	Retrieve and view predictions from database
FR6	Make another prediction

This proposed system is a comprehensive agricultural tool designed to facilitate the diagnosis and management of tomato diseases. It offers users the capability to capture leaf images using the camera or select images from the gallery (i) and subsequently employs a predictive algorithm to identify tomato diseases and classify them (iii). The system not only provides real-time predictions but also allows users to store these predictions in a database (iv) for future reference. Users can easily access and view previous predictions from the database (v), fostering data-driven decision-making in agriculture. Additionally, the system supports making new predictions without explicitly referencing the functional requirements (vi), offering a seamless and efficient user experience for disease detection and crop health management.

3.7.2 Non-functional requirements

This is a requirement that specifies the criteria that is used to assess or judge the operations of the proposed.

Table 3.4: Basic non-functional requirements of the proposed model

Non-functional requirement no	Non-functional requirements
NFR1	Only pictures of tomato leaves should be properly classified and predicted by the proposed mobile application
NFR2	All pictures should be snapped or captured in a properly lit environment

3.7.3 User requirements

The User Requirements Specification outlines the business requirements for the system that users expect from it. Typically, before the system is built, user requirements specifications are produced early in the validation phase. In essence, it involves the demand that the user has the capacity or desire to employ a system capability or activity. As a result, it describes the tasks a user may carry out using the system. Users with minimal technical skills can nonetheless grasp user requirements, which include both functional and non-functional criteria.

Table 3.5: User requirements specification of the proposed system

Requirement No	Description of requirements
UR1	The system shall allow the profiling of users on first use
UR2	The system shall allow users make tomato disease prediction
UR3	The system shall allow users also get expert-wise recommendation on the possible steps they can go to combat possible reoccurrence of such disease
UR4	The system shall display detailed information for different class of disease predicted
UR5	The system shall also allow users to browse previous predictions in order to reduce inference time
UR6	The system shall notify the user of possible conflicts whenever an image different from tomato is uploaded

3.8 System Implementation

Here, implementation is considered as the realization of A methodological design involves coding and deploying software components. This section describes the minimum hardware / software requirements and programming languages / development environment needed to successfully achieve this.

3.8.1 Minimum hardware requirements

Table 3.6: Minimum hardware specification is for android version

S/N	Description
1	ARM64, X86_64
2	2GB RAM
3	Screen size: 5 inches
4	Resolution: 1024-pixel width

The proposed system is a versatile computing device designed to support both ARM64 and X86_64 architectures, ensuring compatibility with a wide range of software applications. It boasts 2GB of RAM for efficient multitasking and smooth performance. With a compact form factor featuring a 5-inch screen, it offers a portable and ergonomic user experience. The display provides a resolution of 1024 pixels in width, delivering crisp visuals for various tasks and applications.

3.8.2 Minimum software requirements.

Table 3.7: minimum software requirements

S/N	Description
1	Android version >10.0.0
2	Google Chrome (use latest version)
3	Firefox (use latest version)
4	Microsoft Edge (use latest version)

The proposed system is designed to be highly compatible with modern web browsers and Android devices running version 10.0.0 or higher. Users can access the system seamlessly through Google Chrome, Firefox, and Microsoft Edge, using the latest versions of these browsers. This ensures optimal performance, security, and accessibility for a wide range of users across various platforms, offering a robust and user-friendly experience.

3.8.3 Choice of programming language(s)/development environment

The application was built in visual Studio IDE using FLUTTER, a dart framework for building cross platform Mobile applications. While running the experiments, we used the Google colab simulation environment which is suitable to run and save experiments. Each result from the experiment was saved to the Google drive which was imported at the earlier stage of the simulation.

The python programming language was used for the simulation. Tensorflow as the backend framework for all the tasks during simulation. The packages we used within the python language range from numpy, matplotlib, seaborn, Keras, pandas, etc.

CHAPTER FOUR

RESULTS AND DISCUSSION

This chapter focuses on the various results obtained from the implementation of an intelligent tomato disease prediction model. Also, the integration of the built model to a mobile based platform to help farmers as well as botanist have first-hand result on the class of diseases affecting their tomato leaf and the necessary steps that should be taken to slowdown or eliminate the disease. To allow readability, this chapter four is sectioned to have the following as its main focused; mobile-based application system interface, model testing and system testing, model evaluation and performance evaluation with existing tomato disease prediction models.

4.1 Testing

This section describes all testing activities carried out to ensure that the model and system meets expected requirements as stipulated in the objective of this study. Testing is done during model development to ensure the classification head used to classify each class of tomato disease does so at a certain level of confidence.

4.2 Model Testing

As illustrated in the flowchart diagram of the proposed system. The total number of features gotten from the feature extraction phase follows a train_test_split where 20% was set aside for testing. The selected features were first normalized by calling the standardScaler() method to convert each values to be between 0 and 1. The labels were scaled and converted to different categories. The trained SVM model was used by calling the svm_model.predict() function on these normalized features. Result shows that out of 2200 test samples, 90% were classified correctly in their respective classes. To visualize the actual classes and the predicted classes, we have clearly shown the steps in section 4.3.3.

4.2.1 Test for non-tomato leaves using the proposed mobile application system developed.

In this section, we examined the mobile application developed using our proposed special hybridization model for classification and detection of tomato leaves utilizing nine (9) different non-tomato leaves. We needed to determine whether the mobile application will accurately or incorrectly predict leaves other than tomato leaves as tomato leaves. Figure 4.1

- Figure 4.9 displays the proposed mobile application interface of the predicted results for the nine non-tomato leaves.

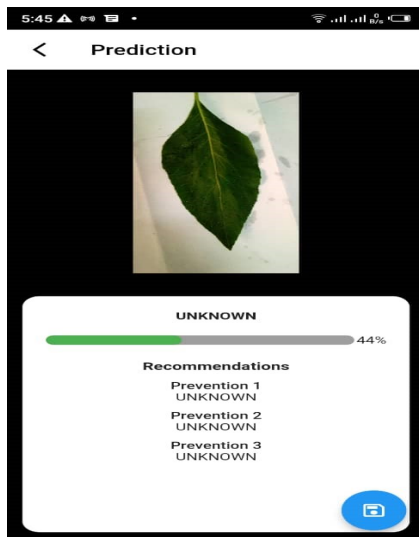


Figure 4.1 Bitter leaf

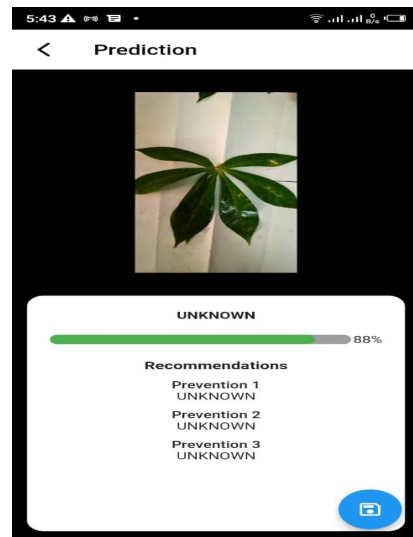


Figure 4.2 Cassava leaf

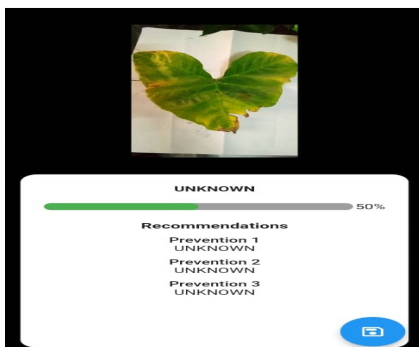


Figure 4.3 Cocoyam leaf

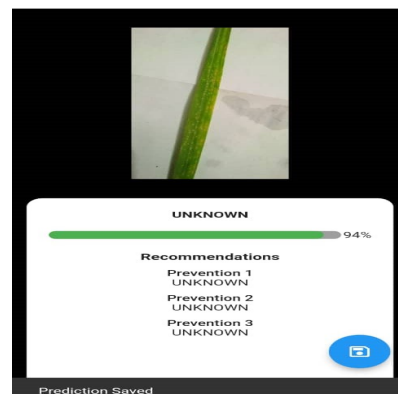


Figure 4.4 Lemon grass leaf

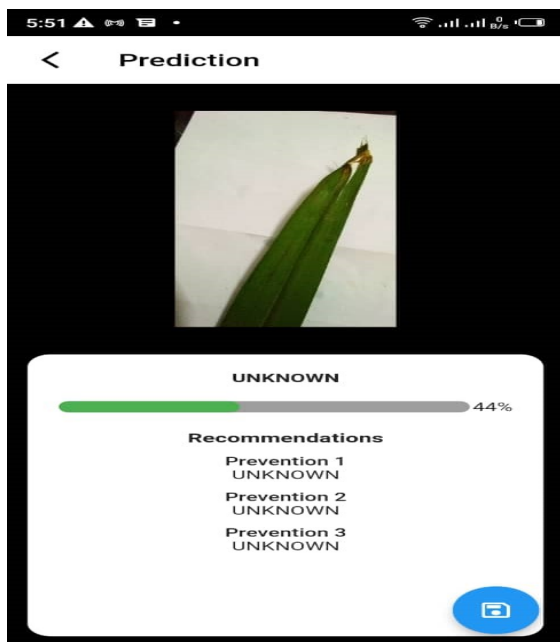


Figure 4.5 Palm kernel leaf

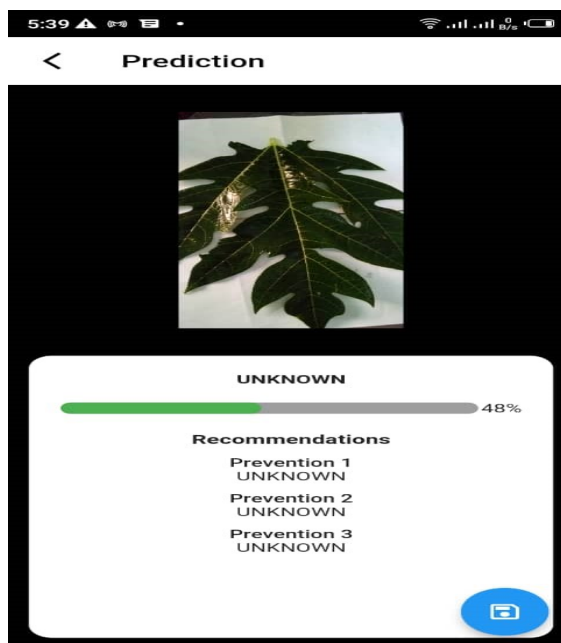


Figure 4.6 Pawpaw leaf

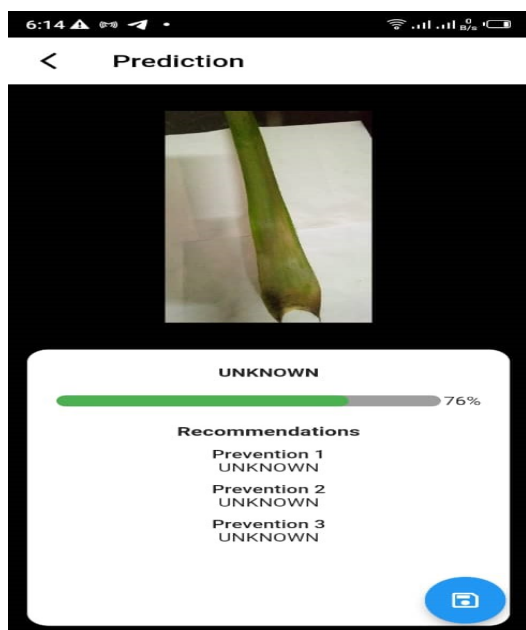


Figure 4.7 Pineapple leaf

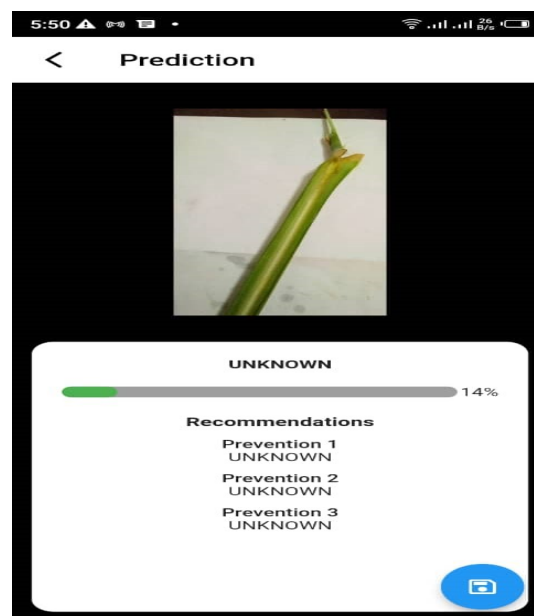


Figure 4.8 Sugar cane leaf

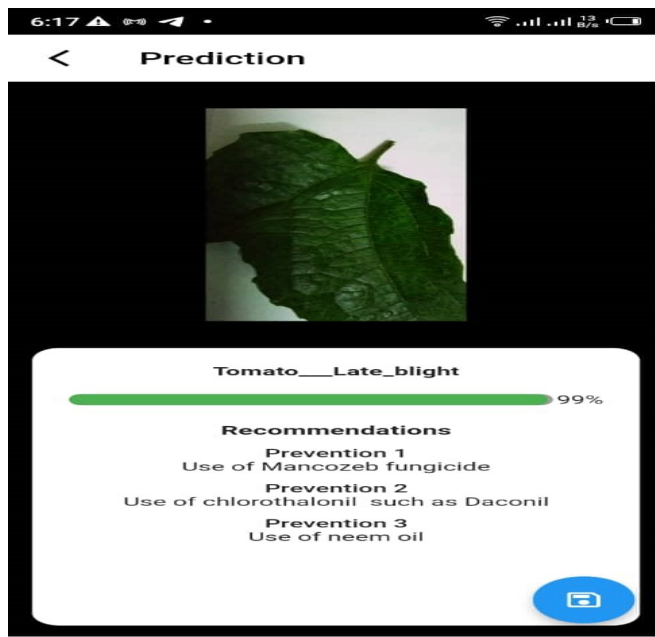


Figure 4.9 Pumpkin leaf

Table 4.1: Classification of non-tomato leave

S/N	Non-tomato leave	Model's prediction result	Human expert opinion assessment
1	Bitter leaf	UNKWOWN	Correctly classified
2	Cassava leaf	UNKWOWN	Correctly classified
3	Cocoyam leaf	UNKWOWN	Correctly classified
4	Lemon grass leaf	UNKWOWN	Correctly classified
5	Palm kernel leaf	UNKWOWN	Correctly classified
6	Pawpaw leaf	UNKWOWN	Correctly classified
7	Pineapple leaf	UNKWOWN	Correctly classified
8	Sugar cane leaf	UNKWOWN	Correctly classified
9	Pumpkin leaf	Tomato leaf with late blight disease	Wrongly classified as Tomato leaf with late blight disease

Eight out of the nine non-tomato leaves that were tested using the mobile application developed because of the research work were correctly classified as unknown (non-tomato), according to an analysis of the entries in Table 4.1, while only one, a pumpkin leaf, was incorrectly classified as a tomato leaf with late blight disease.

Again, the proposed model's detection rate of the non-tomato leaves can be simply computed as:

$$\begin{aligned}\text{Detection rate} &= \frac{8}{9} \times \frac{100}{1} \\ &= 88.88\%\end{aligned}$$

While the wrongly detected (classified) leaf can be simply computed as:

$$\begin{aligned}\text{Detection rate} &= \frac{1}{9} \times \frac{100}{1} \\ &= 11.11\% \text{ (False alarm)}\end{aligned}$$

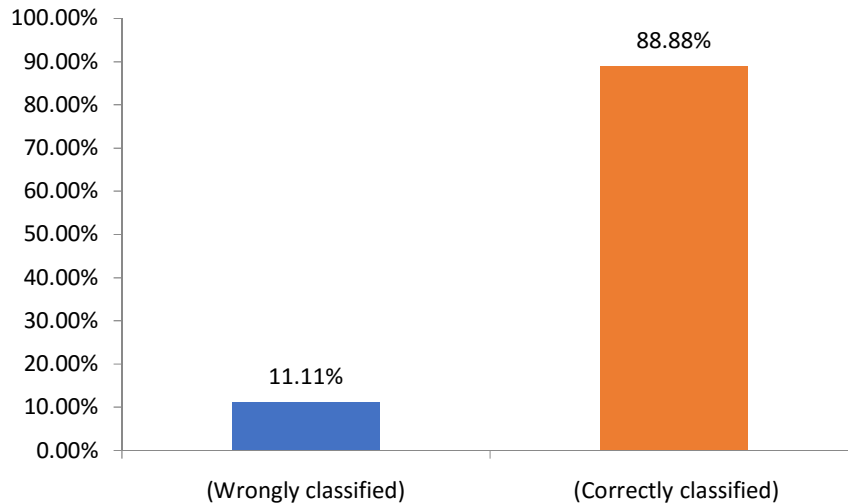


Figure 4.10: Bar graph showing percentage of both correctly and wrongly classified non-tomato leaves.

The graph of figure 4.10 shows that the proposed system achieved an 88.88% accuracy rate in correctly classifying non-tomato leaves, while 11.11% of instances were misclassified.

4.2.2 Test for tomato leaves using the proposed mobile application system developed.

In this section, we equally examined the mobile application developed utilizing our suggested special hybridization model for classification and detection of tomato leaves using six (6) tomato leaves with late blight disease. Again, we needed to determine whether the mobile application will accurately or incorrectly detect the leaves as tomato leaves with late blight disease or not. Figure 4.11 - Figure 4.16 displays the proposed mobile application view of the predicted results for the six tomato leaves with late blight disease.

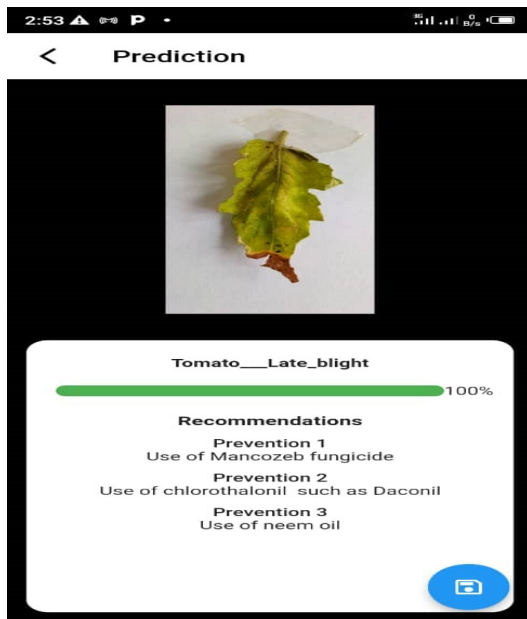


Figure 4.11: Tomato leaf with late blight disease

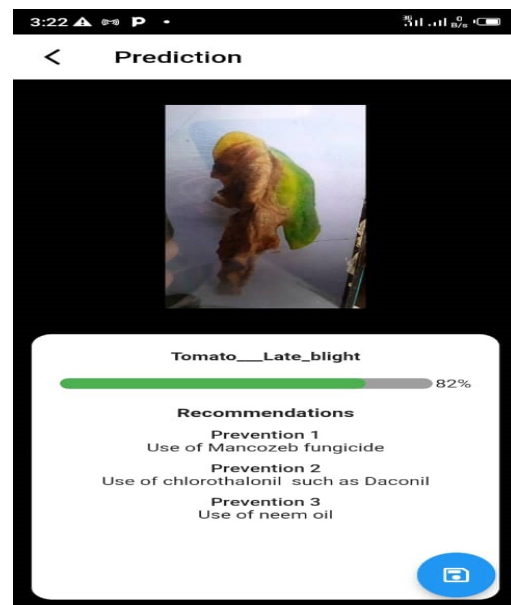


Figure 4.12: Tomato leaf with late blight disease

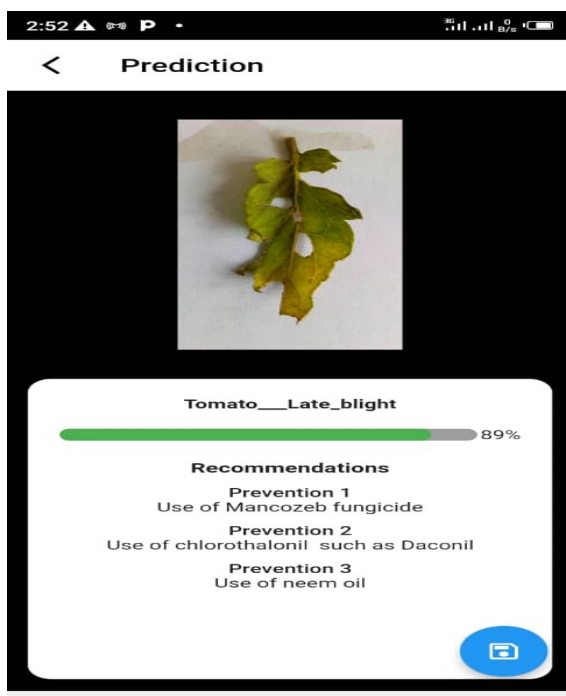


Figure 4.13: Tomato leaf with late blight disease

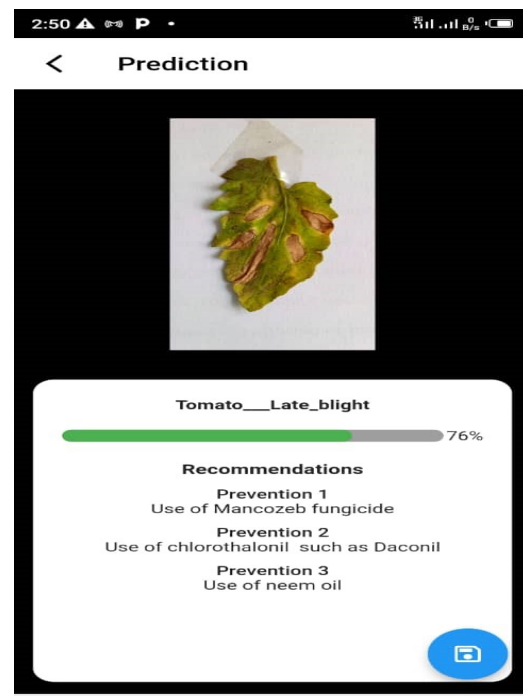


Figure 4.14: Tomato leaf with late blight disease

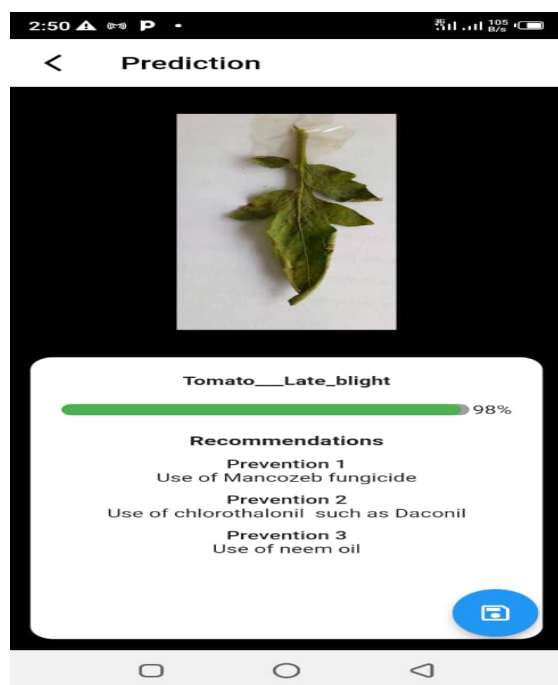


Figure 4.15: Tomato leaf with late blight disease

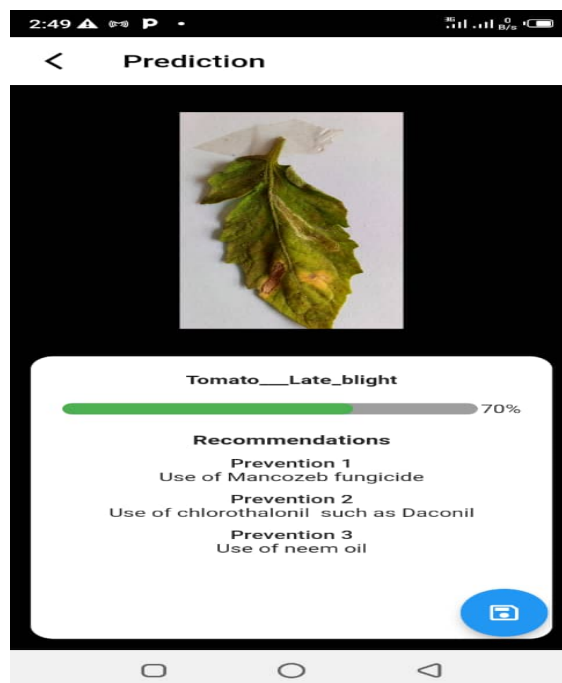


Figure 4.16: Tomato leaf with late blight disease

Table 4.2: Classification of tomato leave

S/N	Tomato leave	Model's prediction result	Human expert opinion assessment
1	Tomato leaf	Tomato_late blight	Correctly classified
2	Tomato leaf	Tomato_late blight	Correctly classified
3	Tomato leaf	Tomato_late blight	Correctly classified
4	Tomato leaf	Tomato_late blight	Correctly classified
5	Tomato leaf	Tomato_late blight	Correctly classified
6	Tomato leaf	Tomato_late blight	Correctly classified

Six tomato leaves already classified by human expert as tomato leaf with late blight disease were tested using the proposed mobile application developed because of the research work and they were all correctly classified as tomato leaves with late blight disease. Table 4.2 shows an analysis of the entries.

Again, the proposed model's detection rate of the tomato leaves with late blight disease can be simply computed as:

$$\begin{aligned}\text{Detection rate} &= \frac{6}{6} \times \frac{100}{1} \\ &= 99.99\%\end{aligned}$$

4.3 Model's Evaluation

4.3.1 Feature Extraction Process

To extract features, we used the total training datasets (10, 000) with the additional 1000 set aside for validation. We used 11, 000 datasets during the feature extraction phase because data has been tagging a rich source of information for a deep neural model. The mobileNetV2 was used as its input is 224 x 224 x 3. So, we scaled and resized our datasets by reducing the size from 256 to 224 in order to avoid errors from standard input shapes which Neural network suffers from. To get features from each class, we used the last global pooling layer of the MobileNetV2 with the classification head chopped off by setting trainable equals false. Each feature shape was [1000, 1280] where 1280 is the output size of the global pooling layer of the MobileNetV2 while 1000 corresponds to the number of datasets contain in one class. I carried out some augmentation process where each tomato disease images were rotated at range of 10%, randomly flipped horizontally and carefully zoomed at 9% because too much zooming is invariants to blur cues in digital images and spatial varying effects where the lines of the edges of digital images looks irregular. The feature extraction process took approximately 12hrs to complete because I used a batch size of 4 in other to ensure fine-grained features were grabbed from each image.

4.3.2 SVM classification head

To utilize the stored features stored in numpy array files, I initially set the 'pickle' parameter to true, granting me the ability to manipulate the saved numpy features. The collective count of feature shapes stands at 1280, while the total number of training samples employed for SVM classification amounts to 8800, with the remaining samples allocated for model

testing. The task of predicting tomato disease leaves falls into the realm of multiclass classification. However, support vector machines are primarily designed for binary classification tasks. To adapt SVMs for multiclass classification, one can employ either a one-vs-all or a one-vs-one strategy. In this study, we opted for a one-versus-one multiclass classification method. Various multiclassification approaches, including one-vs-one, one-vs-all, binary classification tree, and binary coding, are available. In our one-vs-one approach, we developed a classifier to predict one class as positive and another class as negative. Consequently, we utilized a total of 45 classifiers, as determined by the following formula.

$$No. of Classifier = R(R - 1)/2$$

Where R is the number of classes involved in the classification task. Hence, substituting our class as 10.

$$No. of Classifier = 10(10 - 1)/2$$

$$\frac{10 \times 9}{2} = 90/2$$

$$No. of Classifier = 45$$

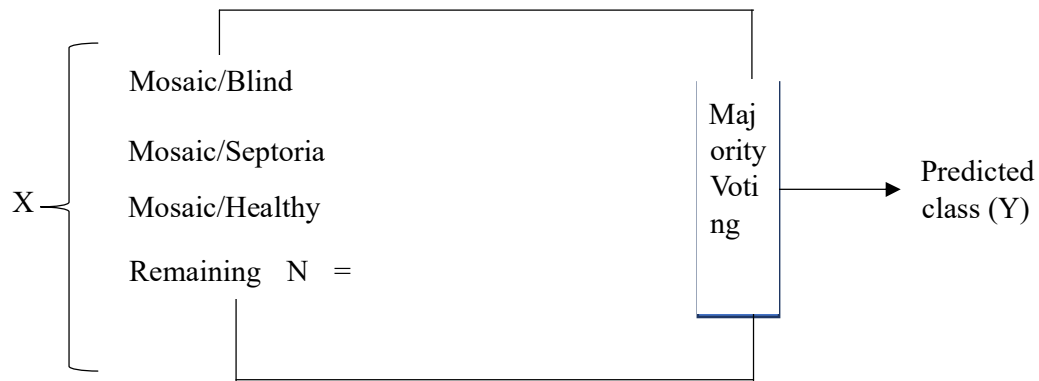


Figure 4.17: One vs One SVM

Figure 4.17 shows 45 smaller SVMs which are used to perform a binary classification. All the results obtained are processed using a majority voting to vote the highest number of prediction accuracy as that will inform us the exact class X belongs. The goal of this approach is that the training process is done at once and no repeatition. To use this approach in python, we import SVC from the scikit learn library. To ensure fairness, we equally conducted different splitting as well as different kernel types ranging from RBF, Linear or Gaussian. Also, we set our C which is a hyperparameter in SVM to random values because we want to know which is okay for our mobile application development. We run a cross validation for CV=5 and perform a grid search to get the result best for the mobile application development. Table 4.3 shows the different combination carried out as well as the respective output in terms of training accuracy.

Table 4.3: A trade-off table between parameters used for cross validation.

Train_Test_Split	Gamma	Kernel	C value	Training accuracy
60% Training & 40% testing	0.001, 0.0001	RBF, Linear	1, 10, 100, 1000	0.9211, 0.9308 0.9451, 0.9366, 0.9422
Training 80% & testing 20%	0.001 , 0.0001	RBF , Linear	1, 10, 100 , 1000	0.9121, 0.9200, 0.9255, 0.9477, 0.9550

Splitting the number of features based on the identified train_test_split of 20% testing samples gave the best result on a five (5) validation set. Each validation gives five results based on the number of different combinations. RBF and Linear kernel type were tested with similar or different settings. The C value was used to fine tune the accuracy of the support vector machine fit function. The Best kernel was RBF, an 80% and 20% train-test split, were selected for the building of the mobile application. Also, the C value 100 with a combination

of the RBF kernel and gamma was set to 0.001. Based on these identified combinations, the best training accuracy was 0.95%

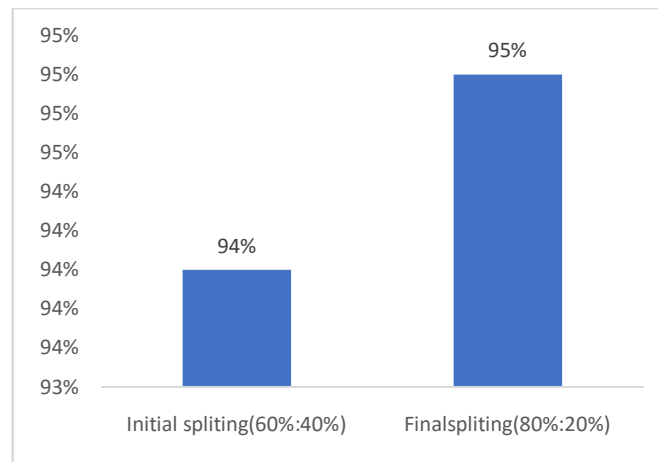


Figure 4.18: Graph of Train_Test_Split

The best split ratio method of training worth optional result of 95% training accuracy.

4.3.3 Confusion Matrix and Accuracy

We tested the built model using 2200 samples with 1280 feature size. After performing training, we used the prediction function on the test generator to get the predicted classes of a tomato disease. For clarity, we used the confusion matrix from sklearn.metrics to generate the confusion report of how the model truly classified our test datasets. Additionally, a confusion matrix gives an idea of how many predictions made were correct and how many percentages of predicted classes were correct. It also shows how many numbers of times is a prediction correct. The generator classes were set as `Y_test_label`, that represents the original classes of the test samples.

Figure 4.19 illustrates the confusion matrix generated through a comparison of the output, denoted as Y_{pred_label} , obtained from predictions. To accurately visualize and construct this confusion matrix, a specific definition for plotting and coloring has been devised.

Mosaic Viru	359	0	0	0	0	0	0	0	0
Blind Sp	5	175	10	0	0	0	0	0	1
Bacteria Spo	0	10	150	5	0	0	0	0	0
Septoria leaf sp	0	0	5	268	8	0	0	0	0
Yellow Leaf Vir	0	0	0	8	143	10	0	0	0
Tomato late blig	0	0	0	0	20	159	5	0	0
Leave mol	0	0	0	0	0	0	300	0	0
Early Blig	0	0	0	0	0	0	0	130	4
Spider mite	0	0	0	0	0	0	0	5	228
Healthy clas	0	0	0	0	0	0	0	0	0
	Mosaic Viru	Blind Sp	Bacteria Spo	Septoria leaf sp	Yellow Leaf Vir	Tomato late blig	Leave mol	Early Blig	Spider mite

Figure 4.19: Confusion Matrix

To analyze the result gotten from Figure 4.6, we need to estimate the recall, accuracy and precision values from the matrix. Since the strategy selected is one versus one, we will estimate each class against one from the rest of the class. Example, mosaic_virus vs blind_spot class, blind_spot vs bacteria_spot, septoria_leaf vs yellow_leaf, etc.

4.3.4 Mosaic/Blind spot

(a) **Precision:** as a metric, provides us with an understanding of the accuracy in correctly identifying positive cases.

$$Precision(Mosai /Blind spot) = \frac{TP}{TP + F}$$

$$Precision(Mosaic/Blind spot) = \frac{359}{359 + 5}$$

$$Precision(Mosaic/Blind\ spot) = 0.99$$

This represents the proportion of correctly predicted True classes relative to the total number of positive class instances. The computed precision results, obtained through the utilization of the Python library scikit-learn metrics, are presented in Table 4.2.

(b) Recall: refers to the ratio of accurately classified 0s to the total number of actual 0s (where 0 represents our positive class). Recall aims to address the query of how accurately actual positives were detected.

$$Recall\ (Mosaic/Blind\ spot) = \frac{TP}{TP + FN}$$

$$Recall\ (Mosaic/Blind\ spot) = \frac{359}{359 + 0} = 1.00$$

(c) F1 score: This represents the relationship between precision and recall. The F1 score employs the Harmonic Mean instead of the Arithmetic Mean because it penalizes extreme values to a greater extent.

$$F1 - score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

$$F1 - score = 2 * \frac{(1.00 * 0.99)}{(1 + 0.99)}$$

$$F1 - score = 0.99 = 99\%$$

(d) Accuracy:

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

$$Accuracy = \frac{359 + 175}{(359 + 175 + 5 + 0)}$$

$$Accuracy = 99\%$$

Table 4.4 Mosaic/ Blind sport

Precision	Recall	Accuracy	F ₁ -score
99%	100%	99%	100%

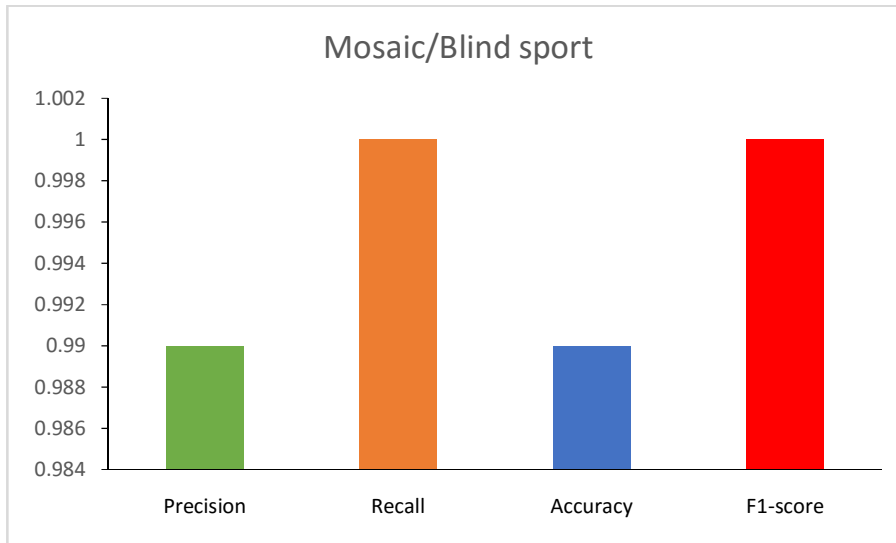


Figure 4.20 Graph of Mosaic/ Blind sport

4.3.5 Blind/Bacteria spot

(a) **Precision:** this metric gives us an insight of what proportion of positive identifications was actually correct.

$$Precision(Blind/Bacteria\ spot) = \frac{TP}{TP + FP}$$

$$Precision(Blind/Bacteria\ spot) = \frac{175}{175 + 15}$$

$$Precision(Blind/Bacteria\ spot) = 0.92$$

This is a ratio of True classes predicted correctly and total number of positive classes. The output of the computation of precision is shown in Table 4.4 which was done by using the python library scikit-learn metrics.

(b) Recall: recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$\text{Recall (Blind/Bacteria spot)} = \frac{TP}{TP + FN}$$

$$\text{Recall (Blind/Bacteria spot)} = \frac{175}{175 + 16} = 0.916$$

(c) F1 score: The F1 score is determined by the ratio of precision to recall, and it employs the Harmonic Mean instead of the Arithmetic Mean because it places greater emphasis on penalizing extreme values.

$$\text{F1 - score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

$$\text{F1 - score} = 2 * \frac{(0.916 * 0.92)}{(0.916 + 0.92)}$$

$$\text{F1 - score} = 0.92$$

(d) Accuracy:

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

$$\text{Accuracy} = \frac{175 + 150}{(175 + 150 + 15 + 16)}$$

$$\text{Accuracy} = 91\%$$

Table 4.5 Blind/Bacteria spot

Precision	Recall	Accuracy	F ₁ -score
0.945	0.916	91%	0.92

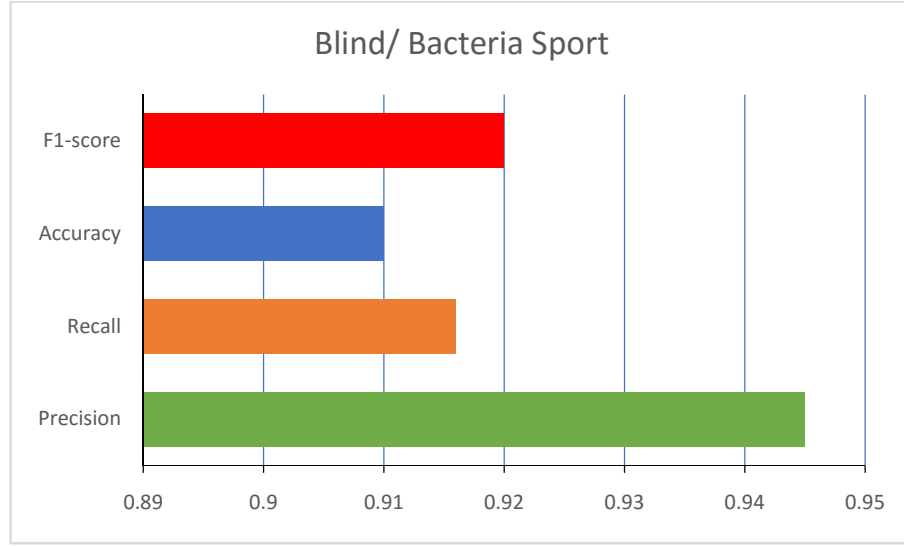


Figure 21: Graph of Blind/Bacteria spot

4.3.6: Bacteria/Septoria leaf

(a) Precision: This measure provides us with an understanding of the accuracy of positive identifications.

$$Precision(Bacteria/Septoria\ leaf) = \frac{TP}{TP + FP}$$

$$Precision(Bacteria/Septoria\ leaf) = \frac{150}{150 + 23}$$

$$Precision(Bacteria/Septoria\ leaf) = 0.87$$

(b) Recall: Recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$Recall\ (Bacteria/Septoria\ leaf) = \frac{TP}{TP + FN}$$

$$Recall\ (Bacteria/Septoria\ leaf) = \frac{150}{150 + 15} = 0.909$$

(c) F1 score: This is the ratio of precision and recall. The F1 score uses the Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

$$F1 - score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

$$F1 - score = 2 * \frac{(0.909 * 0.87)}{(0.909 + 0.87)}$$

$$F1 - score = 0.890$$

(d) Accuracy:

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

$$Accuracy = \frac{150 + 268}{(150 + 268 + 15 + 23)}$$

$$Accuracy = 92\%$$

Table 4.6 Bacteria/Septoria leaf

Precision	Recall	Accuracy	F ₁ -score
0.87	0.91	92%	0.89

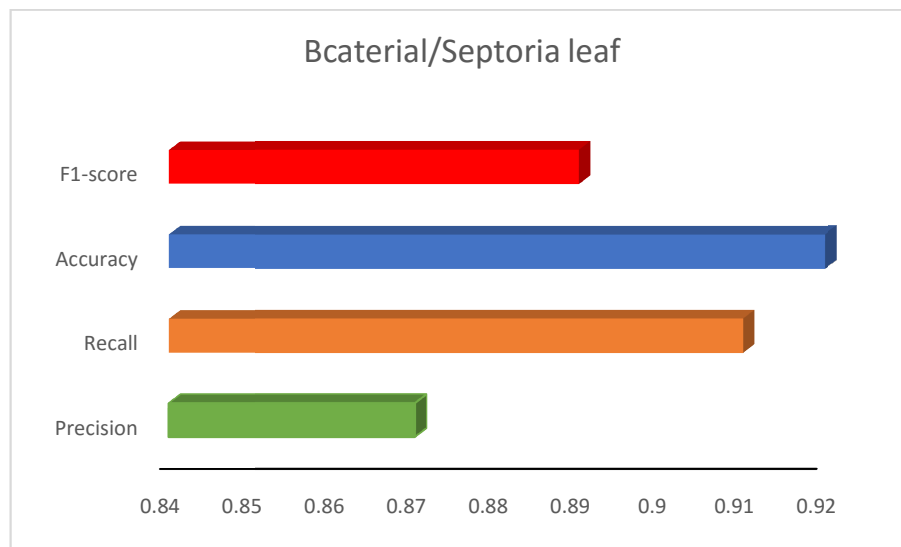


Figure 22 Graph of Bacteria/Septoria leaf

4.3.7 Septoria leaf/Yellow_Leaf_Virus

(a) **Precision:** this metric gives us an insight of what proportion of positive identifications was correct.

$$Precision(Septoria\ leaf/Yellow_Leaf_Virus) = \frac{TP}{TP + FP}$$

$$Precision(Septoria\ leaf/Yellow_Leaf_Virus) = \frac{268}{268 + 18}$$

$$Precision(Septoria\ leaf/Yellow_Leaf_Virus) = 0.94$$

(b) **Recall:** Recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$Recall\ (Septoria\ leaf/Yellow_Leaf_Virus) = \frac{TP}{TP + FN}$$

$$Recall\ (Septoria\ leaf/Yellow_Leaf_Virus) = \frac{268}{268 + 33} = 0.89$$

(c) **F1 score:** This is the ratio of precision and recall. The F1 score uses the Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

$$F1 - score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

$$F1 - score = 2 * \frac{(0.94 * 0.89)}{(0.94 + 0.89)}$$

$$F1 - score = 0.915$$

(d) **Accuracy:**

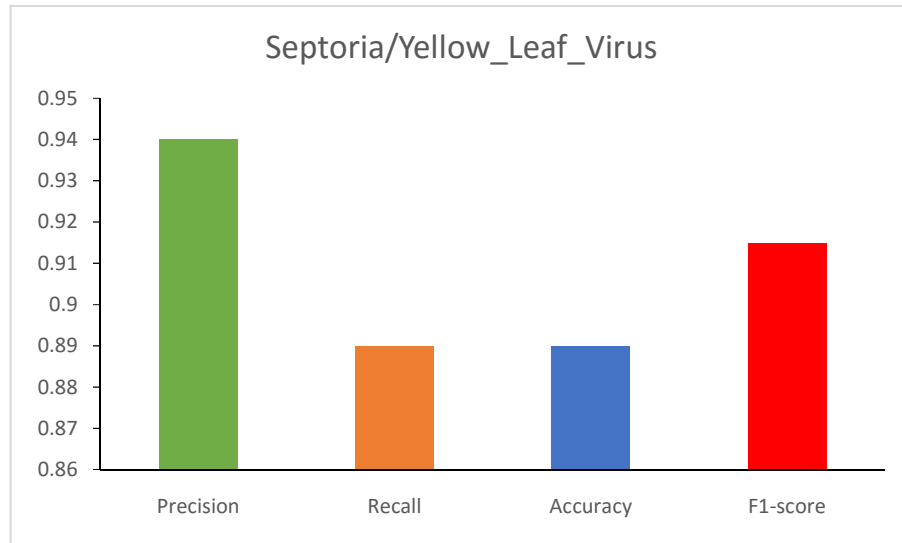
$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)}$$

$$Accuracy = \frac{268 + 143}{(268 + 143 + 33 + 18)}$$

$$Accuracy = 89\%$$

Table 4.7 Septoria leaf/Yellow_Leaf_Virus

Precision	Recall	Accuracy	F ₁ -score
0.94	0.89	89%	0.915

**Figure23 Graph of Septoria leaf/Yellow_Leaf_Virus****4.3.8 Yellow_Leaf_Virus/late_blight****(a) Precision:**

This metric gives us an insight of what proportion of positive identifications was actually correct.

$$\text{Precision}(\text{Yellow_Leaf_Virus/late_blight}) = \frac{TP}{TP + FP}$$

$$\text{Precision}(\text{Yellow_Leaf_Virus/late_blight}) = \frac{143}{143 + 25}$$

$$\text{Precision}(\text{Yellow_Leaf_Virus/late_blight}) = 0.85$$

(b) Recall: Recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$\text{Recall}(\text{Yellow_Leaf_Virus/late_blight}) = \frac{TP}{TP + FN}$$

$$\text{Recall (Yellow_Leaf_Virus/late_blight)} = \frac{143}{143 + 18} = 0.89$$

(c) **F1 score:** This is the ratio of precision and recall. The F1 score uses the Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

$$\text{F1 - score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

$$\text{F1 - score} = 2 * \frac{(0.89 * 0.85)}{(0.89 + 0.85)}$$

$$\text{F1 - score} = 0.88$$

(d) **Accuracy:**

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

$$\text{Accuracy} = \frac{143 + 159}{(143 + 159 + 18 + 25)}$$

$$\text{Accuracy} = 88\%$$

Table 4.8 Yellow_Leaf_Virus/late_blight

Precision	Recall	Accuracy	F ₁ -score
0.85	0.89	98%	0.88

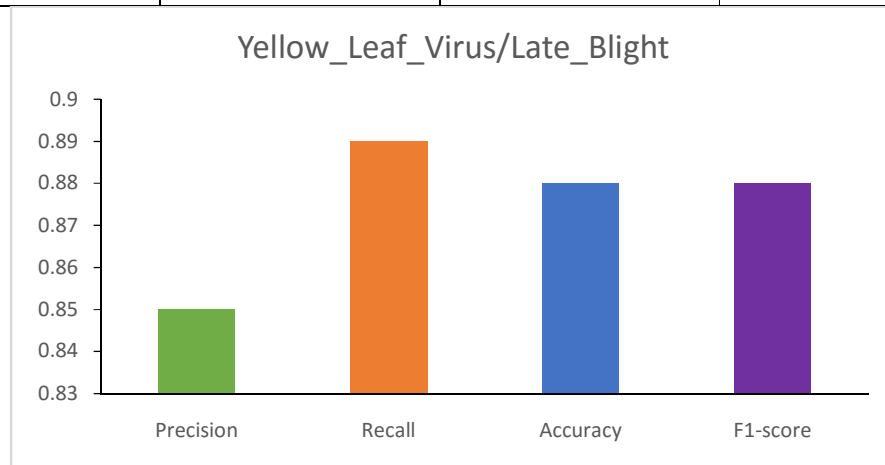


Figure 24 Graph of Yellow_Leaf_Virus/late_blight

4.3.9 Late_blight/Leaf_mold

(a) Precision:

This metric gives us an insight of what proportion of positive identifications was actually correct.

$$Precision(late_blight/Leave_mold) = \frac{TP}{TP + FP}$$

$$Precision(late_blight/Leave_mold) = \frac{159}{159 + 10}$$

$$Precision(late_blight/Leave_mold) = 0.94$$

(b) Recall: Recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$Recall(late_blight/Leave_mold) = \frac{TP}{TP + FN}$$

$$Recall(late_blight/Leave_mold) = \frac{159}{159 + 25} = 0.86$$

(c) F1 score: This is the ratio of precision and recall. The F1 score uses the Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

$$F1 - score = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

$$F1 - score = 2 * \frac{(0.94 * 0.86)}{(0.94 + 0.86)}$$

$$F1 - score = 0.898$$

(d) Accuracy:

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

$$Accuracy = \frac{159 + 300}{(300 + 159 + 10 + 25)}$$

$$Accuracy = 93\%$$

Table 4.9 Late_Blight/Laef_Mold

Precision	Recall	Accuracy	F ₁ -score
0.94	0.86	93%	0.90

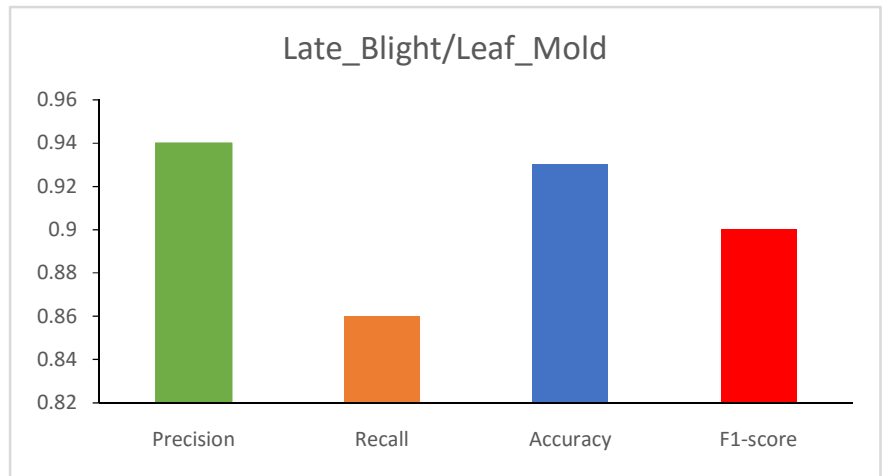


Figure 25 Graph of Late_Blight/Laef_Mold

4.3.10 Leaf_mold/Early_blight

(a) Precision:

This metric gives us an insight of what proportion of positive identifications was correct.

$$Precision(Leave_mold/Early_blight) = \frac{TP}{TP + FP}$$

$$Precision(Leave_mold/Early_blight) = \frac{300}{300 + 4}$$

$$Precision(Leave_mold/Early_blight) = 0.986$$

(b) Recall: Recall is the fraction of correctly classified 0s among all the real 0s (0 is our positive class). Recall attempt to answer the question of what proportion of actual positives was identified correctly?

$$Recall(Leave_mold/Early_blight) = \frac{TP}{TP + FN}$$

$$\text{Recall (Leave_mold/Early_blight)} = \frac{300}{300 + 0} = 1.00$$

(c) **F1 score:** This is the ratio of precision and recall. The F1 score uses the Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more.

$$\text{F1 - score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})}$$

$$\text{F1 - score} = 2 * \frac{(0.986 * 1)}{(0.986 + 1)}$$

$$\text{F1 - score} = 0.99$$

(d) **Accuracy:**

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

$$\text{Accuracy} = \frac{300 + 130}{(300 + 130 + 4 + 0)}$$

$$\text{Accuracy} = 99\%$$

Table 4.10 Leaf_mold/Early_blight

Precision	Recall	Accuracy	F ₁ -score
0.99	1.00	99%	0.99

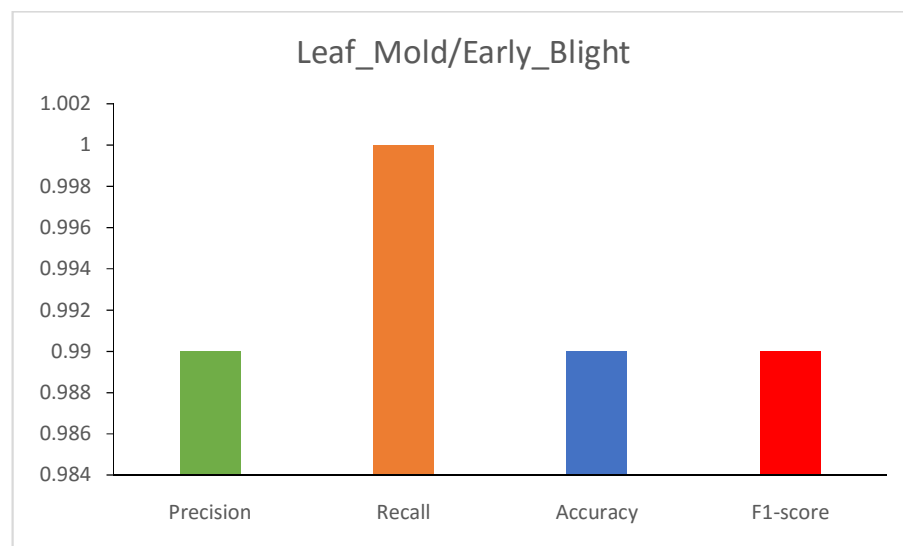


Figure 26: Graph of Leaf_mold/Early_blight

Table 4.11 Recall and Precision

SVMs	Precision	Recall	Accuracy	F1-score
Mosaic/Blind spot	99%	100%	99%	100%
Blind/Bacteria spot	92%	91.6%	91%	92%
Bacteria/Septoria leaf	87%	91%	92%	89%
Septoria leaf /Yellow_Leaf_Virus	94%	89%	89%	91.5%
Yellow_Leaf_Virus/late_blight	85%	89%	88%	88%
late_blight/Leaf_mold	94%	86%	93%	90%
Leave_mold/Early_Blight	99%	100%	99%	99%

Table 4.4 shows the combination of each smaller SVMs for binary classification. The total number of classifiers is 45. For readability and non-verbose comprehension, we decided to show the calculation for only 7 binary classifiers without a backward iteration. Back iteration means comparing the negative classes against one another. To illustrate the relationship between true positive rate and the false positive rate, we use the ROC curve. The ROC curve illustrates the relationship between the true positive rate, also known as recall, and the false positive rate. The false positive rate represents the proportion of negative cases that are inaccurately labeled as positive. This rate is a complement to the true negative rate, which indicates the proportion of negative cases correctly identified as negative.

The True Negative Rate (TNR) is also referred to as specificity. Consequently, the Receiver Operating Characteristic (ROC) curve illustrates the trade-off between sensitivity (recall) and 1 – specificity. To generate this diagram, we utilized the roc_curve function, passing the

False Positive Rate (FPR) and True Positive Rate (TPR) as inputs. The resulting output was then visualized using Matplotlib.

The ROC curve provides insight into the balance between recall and precision. Notably, as recall increases, so does the number of false positives (FPR) generated by the classifier. In Figure 4.26-4.29, the effective classifier strives to position itself as far away from this line as possible, ideally in the top-left corner.

For evaluating our tomato disease classifier, we calculate the Area Under the Curve (AUC) of the ROC curve, which serves as a perfect metric for comparison. A well-performing classifier will yield an ROC_AUC value close to or equal to 1, while a random classifier will have an ROC_AUC value of 0.5, facilitating a meaningful comparison.

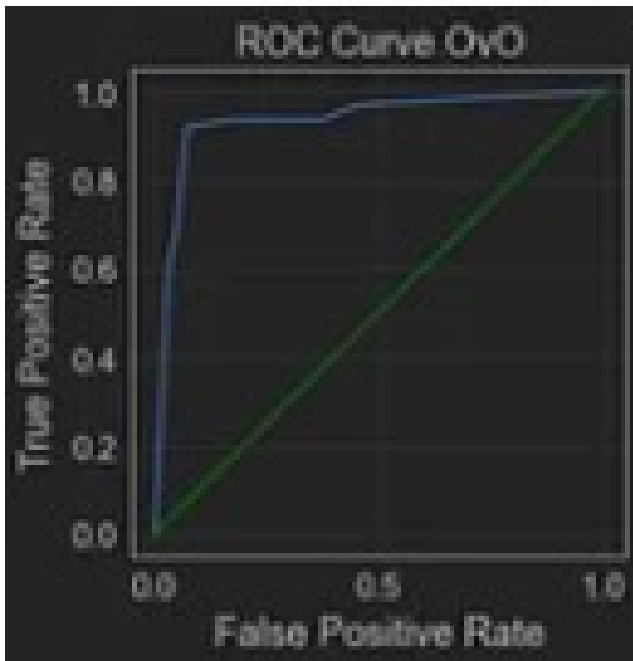


Figure 4.27: Mosaic/Blind Spot

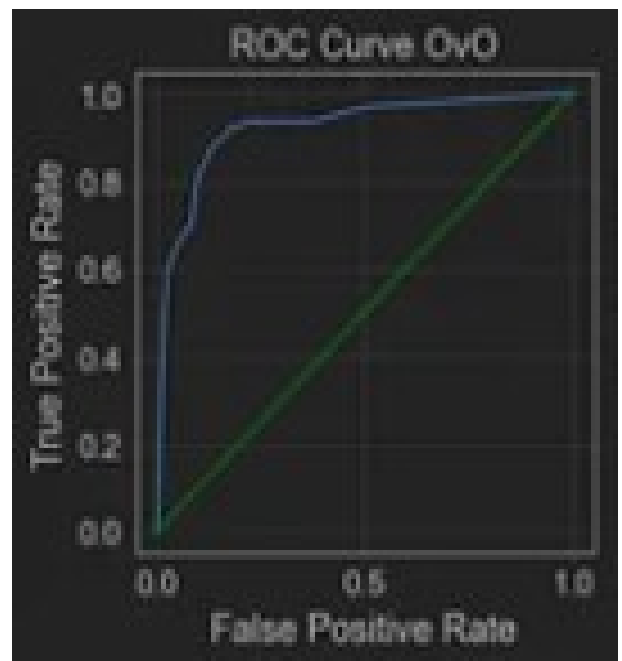


Figure 4.28: Late_blight/Leave Mold

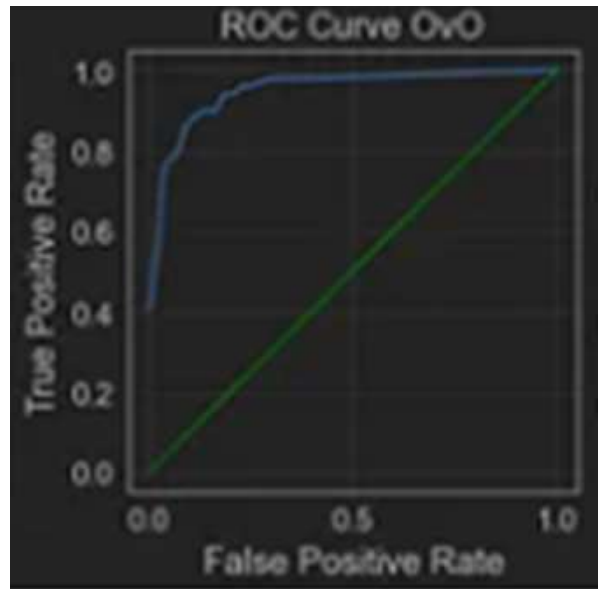


Figure4.29: Leave_ Mold/Early blight

Figure 4.27, 4.28,4.29: ROC Curve for three identified classifiers (small SVMs)

Figure 4.29 represents some identified ROC values with accuracy equal to 99%, 93% and 99% respectively.

The average ROC curve for the entire SVM put together is given in Figure 4.7. The average accuracy of the whole mini or smaller classifiers totals .946% which is equal to 95% average accuracy. The relationship and behavior between the true positive classes and the false positive classes, we output the average accuracy ROC shown in Figure 4.30.

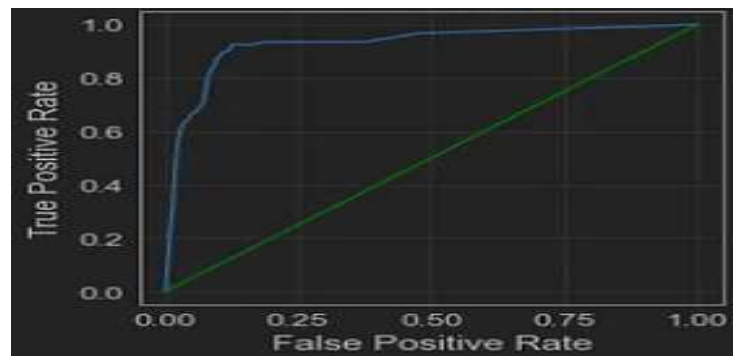


Figure 4.30: ROC Curve for the average 45 classifiers

4.4 Performance Evaluation

To compare the result obtained from this experiment with the result in published literature, we followed a particular order. Firstly, we carefully examined and compared only works available on Scopus indexed journals for fairness and transparency. For the years of publication, we only gather journal papers between 2017 and 2022 as shown in Table 2.1 (chapter two) due to the high volume of literature available in this domain of tomato disease detection.

Since the model's accuracy values was not the major bone of contention in this research work, but rather the integration of a control measure for recommending possible treatment tips. Hence a comparative evaluation was done between the existing models and the proposed model. We only evaluated models that incorporated a recommender system, which off course none out of the 78-research works reviewed has a recommender system as depicted in Figure 4.31



Figure 4.31: Comparative evaluation between existing models and proposed model

4.5 System Interface

(a)

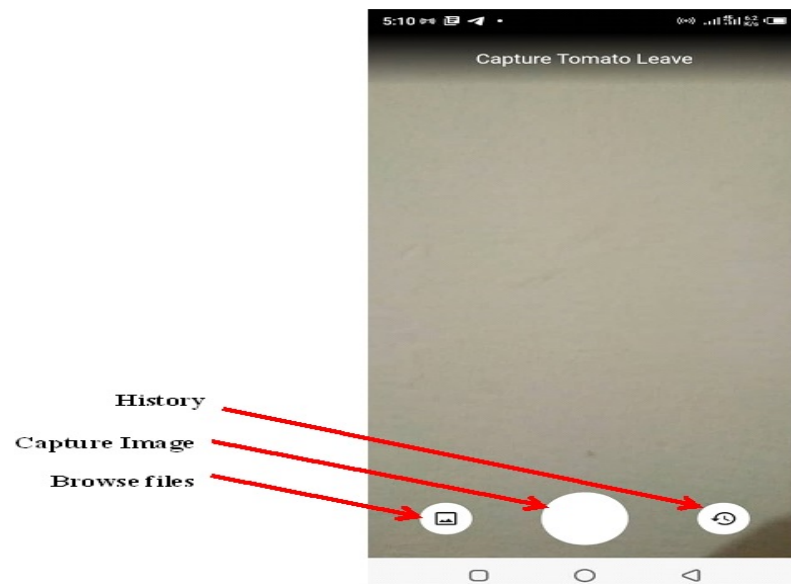


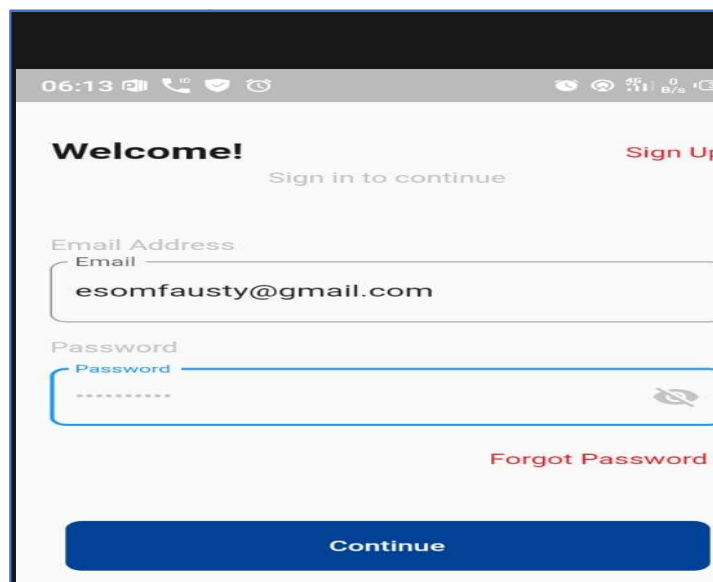
Figure 4.32: Proposed system interface (Mobile view)

1. **History:** The history component in the system interface displays a log or a chronological list of past interactions, actions, or events that have occurred within the system. It can be presented as a text-based log, a timeline, or any other visual representation that allows

users to review their past activities or the system's past states. The history feature is useful for various purposes, such as tracking progress, reviewing previous actions, or referring to past information.

2. **Capture:** The capture button is a feature that enables you to capture or record information, typically in the form of screenshots, images, or data. When you click the capture button, the system may prompt you to select an area of the screen or an element you want to capture. Once you've made the selection, the system will save or generate a representation of the selected content for later use. Capture functionality is often useful when you want to preserve a specific moment or extract relevant data from the system's interface.
3. **Browse:** The browse button allows you to navigate through different files, documents, or directories within the system. By clicking the browse button, you can access and explore the system's file structure or content hierarchy. It provides a way to search, open, and browse through various resources or locations within the system's interface. The browse feature is particularly helpful when you need to locate specific files, access different sections of the system, or organize and manage your data effectively.

- (a) The link <https://tomtoesdiseases.indexcracktools.com/>



The image shows a mobile application interface for a login screen. At the top, there is a status bar with the time 06:13 and various icons. Below the status bar, the screen has a white background with a dark blue header. The header contains the text "Welcome!" in bold black font, followed by "Sign in to continue" in a smaller grey font. In the top right corner of the header, there is a red "Sign Up" link. Below the header, there are two input fields. The first is labeled "Email Address" and "Email" in grey text, with the email address "esomfausty@gmail.com" entered. The second is labeled "Password" and "Password" in grey text, with a masked password "*****" and a toggle icon on the right. Below the password field, there is a red "Forgot Password" link. At the bottom of the screen, there is a large blue button with the text "Continue" in white.

Figure 4.33: The Login page.

This page serves as the initial gateway for users to access the app's functionalities. It functions as both a point of entry and authentication, allowing registered users to sign in securely while providing newcomers with a way to create an account if necessary. Once users are authenticated, they can seamlessly continue to the tomato disease detection page, where they can capture images of tomato plants, submit them for analysis, and receive information about potential diseases affecting their crops. In essence, this home/login page is the starting point that ensures user identity and access to the app's specialized features, fostering a personalized and efficient user experience.

(b)

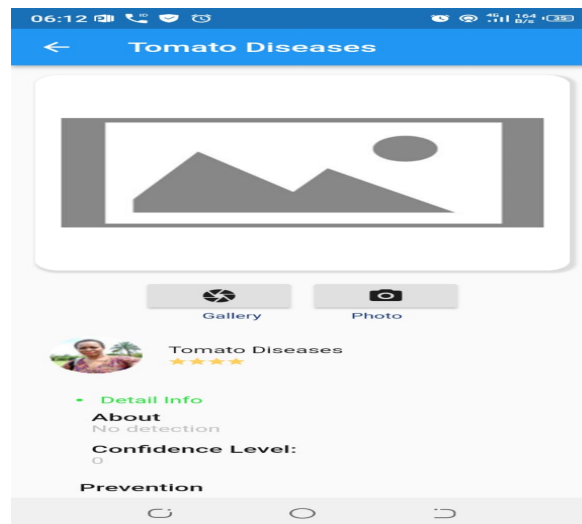


Figure 4.34: No disease present.

Figure 4.34 typically means that the app's algorithm was unable to identify any specific disease or issue on the tomato leaf. This outcome could result from various factors, including the image quality, the absence of any noticeable disease symptoms on the leaf.

(c)

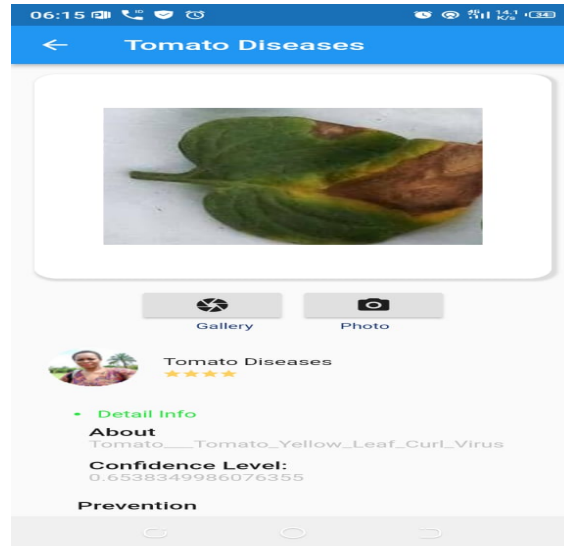


Figure 4.35: Confident level

This figure 4.35 displayed detailed information of a leaf capture and its associated confidence level within a mobile app for tomato disease detection serves as a crucial component of the application's user interface. This page offers users a comprehensive breakdown of the analysis results for a captured tomato leaf, presenting a detailed report on the detected disease, its severity, and the confidence level of the diagnosis. By providing this granular information, users can make informed decisions regarding the health of their tomato plants, enabling timely interventions and preventive measures to mitigate disease outbreaks. This user-friendly presentation of data enhances the app's usability, empowering users to better manage their crops and contribute to the overall health of their tomato plants.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

Indeed, there is a great deal to ensure adequate utilization of machine learning in the agricultural sector. You can clearly see the number of diseases affecting tomato alone is more and they contribute a lot to the annual yield from the cultivation of tomato plants. AI will continue to better human life in numerous amounts of ways. Farmers in some of these rural areas always complain about the damage they receive yearly from diseases which affect their crops. Yes, there are lot of tomato disease prediction models available but none of the articles understudied has been set into production and none has a built-in recommendation system.

The farmers need to get accurate and fast results which are not just reliable but can be successfully implemented by the farmer based on his level of education. Again, following the pervasiveness of smart devices, it is necessary to take this advantage in order to deploy a smart phone application which can help them break even in their various establishments throughout the country. For country like Nigeria where her main source of revenue is coming from agriculture. A computer-assisted tool is required that can quickly generate results with a reasonable level of accuracy. Not just for tomato disease leaf prediction alone but for as many crops as possible.

A tomato leaf disease prediction was developed to mitigate the problems, which are different from designs available in literature. The available datasets contain 10 classes of diseases including a healthy class for tomato. One may decide to use deep learning model with a softmax activation layer at the top-layer. Deep learning is reliable for image processing as it performs a moving average on the entire image made possible by the convolution kernel. However, this method is okay, but the method suffers from over-fitting and under-fitting and

we wish to use a supervised approach. For these reasons, we borrowed the rich feature extraction of deep learning model (MobileNetV2) to extract features from our image datasets. We designed an SVM classifier to learn and help classify the extract features.

The SVM is tested with different combinations of parameters in order to find an optimal parameter setting which will be kept constant during the model deployment phase. The accuracy of our SVM classifier is 94.9% and we also utilized the Universal Sentence Encoder to build the recommender model for giving out control measures tips to users. We then integrated the trained model with a flutter backend for mobile application development. For comparison, our tomato prediction model has a higher accuracy as against some of the results of some published models developed.

5.2 Contribution to knowledge

This research work on multi-class tomato diseases classification and detection model contributes to knowledge as follows:

- (a) The unique hybridization of CNN pre-trained model (MobileNetv2), Support Vector Machine (SVM) and Universal Sentence Encoder (USE) gave better accuracy in the classification and detection of multi-class tomato diseases.
- (b) In addition, the integration of a recommender system by using the Universal Sentence Encoder technique to give users quick control measure advice on potential agents or chemicals to employ on the tomato plant to remove the specific tomato diseases identified which is not in any other model reviewed.

5.3 Recommendations

Based on the experimental activities covered in this work as well as the satisfactory result obtained. We recommend the following:

- (i) This work should be adopted and be marketed to the Federal, State and Local Government agricultural sectors in Nigeria as it will be a very useful in reducing tomato leaf diseases, thereby increasing its crop productivity.

- (ii) Another recommendation is to develop sound minds in AI whose duties shall be working closely with these farmers to get datasets which is coming from our farmers directly because each country has its own soil structure based on differences in environment, climate and weather conditions.

5.4 Limitations

From the period of the research conception, through the design process, gathering primary data for training, evaluation and write up, some challenges were encountered.

1. Access to cognate/relevant materials for the study was not readily available. The institutional database where the researcher operated from (FUTO), there was very scanty literature on the research. More so, indigenous authors seemed not to have conducted accessible studies on the topic of the study.
2. Again, the availability of computational resources presented a strong limitation. Training deep learning models like MobileNetV2 required substantial computing power, which potentially hampered the scalability and efficiency of the proposed system.
3. Furthermore, erratic power supply, as is always the case in most research laboratories in Third World Countries, disrupted the standard workflow, leading to some repeats of analytical processes. This led to extended research hours, man-hour losses and unnecessary duplications. This was particularly evident when selecting appropriate hyperparameters and fine-tuning them for optimal performance.
- 4 It's important to note that current models have solely been trained on individual leaf data. This limitation may result in less-than-optimal performance when these models are applied to images featuring multiple leaves. However, this limitation presents an opportunity for enhancement.

5.5 Areas of future work

1. In future research endeavors, it is advisable to explore the utilization of a Quantum Support Vector Machine (QuSVM). The Quantum package can be accessed through Google TensorFlow Quantum.
2. An interesting advantage of implementing QuSVM is its alignment with the ongoing transition towards the Near Intermediate Quantum era, a movement embraced by tech giants like IBM and Microsoft in the realm of quantum computing.
3. To further enrich ongoing research efforts, it is beneficial to extend investigations into additional transfer learning models.
4. Looking ahead of the potential exists to diversify these models by integrating object detection techniques, which would enable them to identify pests or insects capable of causing harm to plants.
5. Furthermore, it is essential for the model's development to give equal consideration to the incorporation of Internet of Things (IoT) technology. This technology can serve the purpose of alerting farmers when their farms are threatened by diseases, empowering them to respond promptly and effectively.

REFERENCES

- Abbas, A., Jain, S., Gour, M., & Vankudothu, S. (2021). Tomato plant disease detection using transfer learning with C-GAN synthetic images *Computers and Electronics in Agriculture*, 187. <https://doi.org/10.1016/j.compag.2021.106279>
- Adekunle, C. P., Alarima, C. I., & Tolorunju, E. T. (2019): Differentials in willingness to pay for biofortified tomato fruits in Abeokuta metropolis, Ogun state, Nigeria. *Nigeria Agricultural Journal*, 50(2):1-9.
- Agarwal, M., Singh, A., Arjaria, S., Sinha, A., & Gupta, S. (2020). ToLeD: Tomato Leaf Disease Detection using Convolutional Neural Network *Procedia Computer Science*, 167 (2019):293– 301. <https://doi.org/10.1016/j.procs.2020.03.225>
- Ahmad, I., Hamid, M., Yousaf, S., Shah, S. T., & Ahmad, M. O. (2020). Optimizing pretrained convolutional neural networks for tomato leaf disease detection. *Complexity*. 1-6 <https://doi.org/10.1155/2020/8812019>
- Ahmed, S., Hasan, M. B., Ahmed, T., Sony, M. R. K., & Kabir, M. H. (2022). Less is more: Lighter and Faster Deep Neural Architecture for Tomato Leaf Disease Classification. *IEEE Access*, 10. <https://doi.org/10.1109/ACCESS.2022.3187203>
- Ajra, H., Nahar, M. K., Sarkar, L., & Islam, M. S. (2020). Disease Detection of Plant Leaves Using Image Processing and CNN with Preventive Measures. *International Conference on Emerging Technology in Computing, Communication, and Electronics* <https://doi.org/10.1109/ETCCE51779.2020.9350890>
- Akshai, K. P., & Anitha, J. (2021). Plant disease classification using deep learning. In the proceedings of *3rd International Conference on Signal Processing and Communication, ICPSC*, 407–411. <https://doi.org/10.1109/ICSPC51351.2021.9451696>

- Al-gaashani, M. S. A. M., Shang, F., Muthanna, M. S. A., Khayyat, M., & Abd El-Latif, A. A. (2022). Tomato leaf disease classification by exploiting transfer learning and feature concatenation. *16(3)*:913–925. <https://doi.org/10.1049/ipr2.12397>
- Altalak, M., Uddin, M. A., Alajmi, A., & Rizg, A. (2022). A hybrid approach for the detection and classification of tomato leaf diseases. *Applied Sciences*, *12*(16), 8182.
- Altunta, Y., & Kocamaz, F. (2021). Deep Feature Extraction for the Detection of Tomato Plant Diseases and Pests Based on Leaf Images. *Celal Bayar Üniversitesi Fen Bilimleri Dergisi*, *17*(2), 145–152. <https://doi.org/10.18466/cbayarfb.812375>
- Anderson, J. R., & Bower, G. H. (1974). A propositional theory of recognition memory. *Memory & Cognition*, *2*(3), 406–412.
- Annabel, L. S. P., & Muthulakshmi, V. (2019). AI-Powered Image-Based Tomato Leaf Disease Detection. *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics, and Cloud, I-SMAC*, 506–511. <https://doi.org/10.1109/ISMAC47947.2019.9032621>
- Ashok, S., Kishore, G., Rajesh, V., Suchitra, S., Sophia, S. G., & Pavithra, B. (2020, June). Tomato leaf disease detection using deep learning techniques. *Proceedings of 5th International Conference on Communication and Electronics Systems (ICCES)* (pp. 979–983).
- Aversano, L., Bernardi, M. L., Cimitile, M., Iammarino, M., & Rondinella, S. (2020). Tomato disease classification is based on VGG and transfer learning. *In the proceedings of International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor* 129–133. <https://doi.org/10.1109/MetroAgriFor50201.2020.9277626>
- Baudoin, W., Nersisyan, A., Shamilov, A., Hodder, A., Gutierrez, D., PASCALE S, D. E., ... & Tanny, J. (2017). *Good Agricultural Practices for greenhouse vegetable production*

- in the Southeast European countries-Principles for sustainable intensification of smallholder farms* (Vol. 230, pp. 1-449). FAO. <http://www.fao.org/3/a-i6787e.pdf>
- Bayraci, S., & Susuz, O. (2019). A Deep Neural Network (DNN) based classification model in application to loan default prediction. *Theoretical and Applied Economics*, 4(621), 75-84.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: a review and new perspectives. In the proceedings of *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828. <https://doi.org/10.1109/TPAMI.2013.50>
- Bergougnoux, V. (2014). The history of the tomato: from domestication to biopharming. *Proceedings of Biotechnology Advances*, 32(1), 170–189. <https://doi.org/10.1016/j.biotechadv.2013.11.003>
- Bhandari, P. (2022). What Is Deductive Reasoning? | Explanation and Examples *Scribbr*, 1–11. <https://www.scribbr.com/methodology/deductive-reasoning/>
- Bhowmik, D., Kumar, K. P. S., Paswan, S., & Srivastava, S. (2012). Tomato: A Natural Medicine and Its Health Benefits. *In the proceedings of Phytojournal*, 1(1), 33–43.
- Bir, P., Kumar, R., & Singh, G. (2020). Transfer learning-based tomato leaf disease detection for mobile applications. *Proceeding of IEEE International Conference on Computing, Power, and Communication Technologies*, 34–39 <https://doi.org/10.1109/GUCON48875.2020.9231174>
- Blanca, J., Caizares, J., Cordero, L., Pascual, L., Diez, M. J., & Nuez, F. (2012). Variation revealed by SNP genotyping and morphology provides insight into the origin of the tomato. *PLoS ONE*, 7(10) <https://doi.org/10.1371/journal.pone.0048198>
- Brahimi, M., Boukhalifa, K., & Moussaoui, A. (2017). Deep Learning for Tomato Diseases: Classification and Symptom Visualization. *Applied Artificial Intelligence*, 31(4), 299–315. <https://doi.org/10.1080/08839514.2017.1315516>

- Buja, I., Sabella, E., Monteduro, A. G., Chiriaco, M. S., De Bellis, L., Luvisi, A., & Maruccio, G. (2021). Advances in plant disease detection and monitoring: From traditional assays to in-field diagnostics. *Sensors*, 21(6), 2129.
- Byrne, R. M. J., & Johnson-Laird, P. N. (2009). "If" and the problems of conditional reasoning *Trends in Cognitive Sciences*, 13(7), 282-287. <https://doi.org/10.1016/j.tics.2009.04.003>
- Carbonell, P., Alonso, A., Grau, A., Salinas, J. F., Garca-Martnez, S., & Ruiz, J. J. (2018). Twenty years of tomato breeding at EPSO-UMH: Transfer resistance from wild types to local landraces—from the first molecular markers to genotyping by sequencing (GBS) *Diversity*, 10(1). <https://doi.org/10.3390/D10010012>
- Cave, W. C. (2020). *Prediction_Theory_4Spp.Pdf*. 07762(732).
- Chakravarthy, A. S., & Raman, S. (2020). Early Blight Identification in Tomato Leaves Using Deep Learning. In the proceedings of 2020 *International Conference on Contemporary Computing and Applications*, IC3A, 154–158 <https://doi.org/10.1109/IC3A48958.2020.233288>
- Chohan, M., Khan, A., Chohan, R., Katpar, S. H., & Mahar, M. S. (2020). Plant disease detection using deep learning. *International Journal of Recent Technology and Engineering*, 9(1), 909-914.
- Chowdhury, M. E. H., Rahman, T., Khandakar, A., Ayari, M. A., Khan, A. U., Khan, M. S., Al-Emadi, N., Reaz, M. B. I., Islam, M. T., & Ali, S. H. M. (2021). Automatic and Reliable Leaf Disease Detection Using Deep Learning Techniques. *AgriEngineering*, 3(2), 294–312. <https://doi.org/10.3390/agriengineering3020020>
- Cocaliadis, M. F., Fernández-Muoz, R., Pons, C., Orzaez, D., & Granell, A. (2014). Increasing tomato fruit quality by enhancing fruit chloroplast function A double-

- edged sword. *Journal of Experimental Botany*, 65(16), 4589–4598.
<https://doi.org/10.1093/jxb/eru165>
- Crilly, T. (1999). Arthur Cayley as Sadleirian Professor: A Glimpse of Mathematics Teaching at 19th-century Cambridge. *Historia Mathematica*, 26(2), 125–160.
<https://doi.org/10.1006/hmat.1999.2233>
- Damicone, J. P., & Brandenberger, L. (2021). Common Diseases of Tomatoes. 1–6.
<http://pods.dasn.okstate.edu/docushare/dsweb/Get/Document-1323/EPP-7625web.pdf>
- De Kleer, J. (1993). A view on qualitative physics. *Artificial Intelligence*, 59(1–2), 105–114.
[https://doi.org/10.1016/0004-3702\(93\)90176-C](https://doi.org/10.1016/0004-3702(93)90176-C)
- De Luna, R. G., Dadios, E. P., & Bandala, A. A. (2018) Automated Image Capturing System for Deep Learning-Based Tomato Plant Leaf Disease Detection and Recognition *IEEE Region 10 Annual International Conference, Proceedings/TENCON, 2018-October (October)*, 1414–1419. <https://doi.org/10.1109/TENCON.2018.8650088>
- Deng, H., D. L., Chang, Z., Li, H., & Yang, X. (2021). *SS symmetry RAHC_GAN: A Data Augmentation Method for Tomato Leaf*
- Domingues, T., Brando, T., & Ferreira, J. C. (2022) Machine Learning for Detection and Prediction of Crop Diseases and Pests: A Comprehensive Survey *Agriculture*, 12(9), 1350. <https://doi.org/10.3390/agriculture12091350>
- Durmus, H., Gunes, E. O., & Kirci, M. (2017). Disease detection on the leaves of the tomato plants by using deep learning. *The Proceedings of 6th International Conference on AgroGeoinformatics*. <https://doi.org/10.1109/Agro-Geoinformatics.2017.8047016>
- FAO (2013). Food and Agricultural Organization of the United Nations. *Agricultural Sciences*, 06(08), 5. <https://doi.org/10.4236/health.2014.68084>

- FAO (2016). *USDA, 2016. Foreign Agricultural Service*. 12–13.
- FAO, IFAD, UNICEF, WFP, and WHO (2020). Transforming Food Systems for Affordable, Healthy Diets *the State of Food Security and Nutrition in the World 2020*.
<https://www.fao.org/documents/card/en/c/ca9692en>
- FAO. (2017). The future of food and agriculture: trends and challenges: *Sustainability*. 4(4),
www.fao.org/publications%0Ahttp://www.fao.org/3/a-i6583e.pdf%0A
- FAO. (2019). Integrated Pest Management in Tomatoes in Eritrea In *Sustainable Management of Arthropod Pests of Tomato*, 2(1). <https://doi.org/10.1016/B978-0-12-802441-6.00012-7%0A>
- Ferentinos, K. P. (2018) Deep learning models for plant disease detection and diagnosis
Computers and Electronics in Agriculture, 145, 311–318.
<https://doi.org/10.1016/j.compag.2018.01.009>
- Flawnson Tong (2019). Everything you need to know about Graph Theory for Deep Learning. *Data Science*, 1–22 <https://towardsdatascience.com/graph-theory-and-deep-learning-know-hows6556b0e9891b>
- Francis, M., & Deisy, C. (2019). Disease Detection and Classification in Agricultural Plants Using Convolutional Neural Networks: A Visual Understanding. In the Proceedings of 2019 6th International Conference on Signal Processing and Integrated Networks, 1063–1068 <https://doi.org/10.1109/SPIN.2019.8711701>
- Frona, D., Janos, S., & Harangi-Rakos, M. (2019). The challenge of feeding the poor. *MDPI Journal for Sustainability*, 3–4.
- Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). A robust deep-learning-based detector for real-time tomato plant diseases and pest recognition. In *Sensors (Switzerland)*, 17(9). <https://doi.org/10.3390/s17092022>

- Geetharamani, G., & J., A. P. (2019). Identification of plant leaf diseases using a nine-layer deep convolutional neural network *Computers and Electrical Engineering*, 76, 323–338. <https://doi.org/10.1016/j.compeleceng.2019.04.011>
- Gehlot, M., & Saini, M. L. (2020). Analysis of Different CNN Architectures for Tomato Leaf Disease Classification. *Proceedings of 5th IEEE International Conference on Recent Advances and Innovations in Engineering, (ICRAIE)*. <https://doi.org/10.1109/ICRAIE51050.2020.9358279>
- Gonzalez-Huitron, V., León-Borges, J. A., Rodriguez-Mata, A. E., Amabilis-Sosa, L. E., RamirezPereda, B., & Rodriguez, H. (2021). Disease detection in tomato leaves via CNN with lightweight architectures implemented in the Raspberry Pi. *Proceedings of Computers and Electronics in Agriculture*, 181. <https://doi.org/10.1016/j.compag.2020.105951>
- Govardhan, M., & Veena, M. B. (2019). Diagnosis of Tomato Plant Diseases Using Random Forest. *In the proceedings of 2019 Global Conference for Advancement in Technology, GCAT 2019*, 1–5. <https://doi.org/10.1109/GCAT47503.2019.8978431>
- Gulzar, N., Kamili, A. N., & Shah, M. A. (2021). Tomato early blight (*alternaria solani*), pathogen, disease development, and defense response phytohormone signaling. *International Research Journal of Plant Science*, 12(4), 1–10.
- Gunarathna, M. M., Rathnayaka, R. M. K. T., & Kandegama, W. M. W. (2020). Identification of an Efficient Deep Learning Architecture for Tomato Disease Classification Using Leaf Images. *Journal of Food and Agriculture*, 13(1), 33. <https://doi.org/10.4038/jfa.v13i1.5230>
- Habiba, S. U., & Islam, M. K. (2021). Tomato Plant Disease Classification Using a Deep LearningBased Classifier from Leaf Images. *Proceedings of International*

Conference on Information and Communication Technology for Sustainable Development, (ICICT4SD), 82–86.

<https://doi.org/10.1109/ICICT4SD50815.2021.9396883>

Hammou, D. R., & Boubaker, M. (2022). Tomato Plant Disease Detection and Classification Using Convolutional Neural Network Architectures and Technologies. *Proceedings of ProceedingsSmart Innovation, Systems, and Technologies*, 237, 33–44.
https://doi.org/10.1007/978-98116-3637-0_3

Hidayatuloh, A., Nursalman, M., & Nugraha, E. (2018). Identification of Tomato Plant Diseases by Leaf Image Using the Squeezenet Model. *Proceedings of International Conference on Information Technology Systems and Innovation (ICITSI)*. 199–204.
<https://doi.org/10.1109/ICITSI.2018.8696087>

Hiremath, V. B., & Gowda, S. (n.d.) DISEASE PREDICTION OF TOMATO LEAF USING CNN AND DEEP LEARNING TECHNIQUES in the *International Research Journal of Modernization in Engineering, Technology, and Science*.
www.irjmets.com

Hong, H., Lin, J., & Huang, F. (2020). Tomato Disease Detection and Classification by Deep Learning *Proceedings of the 2020 International Conference on Big Data, Artificial Intelligence, and Internet of Things Engineering, ICBAIE*, 25–29.
<https://doi.org/10.1109/ICBAIE49996.2020.00012>

Horst, R. K. (Ed.). (2013). *Westcott's plant disease handbook*. Springer Science & Business Media.

Ikeda, M., Ruedeeniraman, N., & Barolli, L. (2021). An intelligent VegeCareAI tool for next generation plant growth management. *Internet of Things*, 14, 100381.

Islam, M. P., Hatou, K., Aihara, T., Seno, S., Kirino, S., & Okamoto, S. (2022). Performance prediction of tomato leaf disease by a series of parallel convolutional neural

- networks. *Proceedings of Smart Agricultural Technology*, 2, 100054.
<https://doi.org/10.1016/j.atech.2022.100054>
- Izuaka, M. (2020). Traders report a tomato shortage as a pest reappears in Kano. In *premiumtimesng Federal Ministry of Agriculture*.
<https://www.premiumtimesng.com/agriculture/agricnews/394536-traders-report-tomato-shortage-as-pest-reappears-in-kano.html>.
- Jadhav, A., Riswadkar, N., Jadhav, P., & Gogawale, Y. (2022) Crop Recommendation System Using Machine Learning Algorithms. *International Research Journal of Engineering and Technology (IRJET)*, 09(04), 1227-1230.
- Jiang, Di., Li, F., Yang, Y., & Yu, S. (2020). A Tomato Leaf Disease Classification Method Based on Deep Learning. *Proceedings of the 32nd Chinese Control and Decision Conference, CCDC*, 1446-1450 <https://doi.org/10.1109/CCDC49329.2020.9164457>
- Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A., & Menaka, R. (2020). Attentionembedded residual CNN for disease detection in tomato leaves. *Applied Soft Computing Journal*, 86, 105933. <https://doi.org/10.1016/j.asoc.2019.105933>
- Kaur, M., & Bhatia, R. (2019). Development of an improved tomato leaf disease detection and classification method. *In the proceedings of 2019 IEEE Conference on Information and Communication Technology, CICT*, 1–5.
<https://doi.org/10.1109/CICT48419.2019.9066230>
- Khatoon, S., Hasan, M. M., Asif, A., Alshmari, M., & Yap, Y. K. (2021). Image-based automatic diagnostic system for tomato plants using deep learning. *Computers, Materials, and Continua*, 67(1), 595–612. <https://doi.org/10.32604/cmc.2021.014580>

- Kheyroodin, H., & Kheyroodin, S. (2017). Importance of the Tomato as such as medical plant. *International Journal of Advanced Research in Biological Sciences (IJARBS)*, 4(4), 106-115
- Khurana, J., Sharma, A., Chhabra, H. S., & Nijhawan, R. (2019). An integrated deep learning framework of tomato leaf disease detection. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(11S).
- Kim, J., Kim, S., Ju, C., Son, H. I. L., & Member, S. (2019). *Unmanned Aerial Vehicles in Agriculture: A Review of the Perspective of Platform, Control, and Applications*
- Kirange, G. (2020). Machine Learning Approach towards Tomato Leaf Disease Classification. 231– 237.
- Kirstie Canene-Adams, Jessica K. Campbell, Susan Zaripheh, Elizabeth H. Jeffery, and John W. Erdman, J. (2005). The Tomato as a Functional Food
- Kriegeskorte, N. (2015). Deep Neural Networks: A New Framework for Modeling Biological Vision and Brain Information Processing. *Annual Review of Vision Science*, 1(1), 417–446, <https://doi.org/10.1146/annurev-vision-082114-035447>
- Kumar, A., & Vani, M. (2019). Image-Based Tomato Leaf Disease Detection. *In the proceedings of 2019 10th International Conference on Computing, Communication, and Networking Technologies, ICCCNT 2019*, 1–6.
<https://doi.org/10.1109/ICCCNT45670.2019.8944692>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553):436-444.
- Liu, J., & Wang, X. (2020). Tomato Diseases and Pest Detection Based on an Improved Yolo V3 Convolutional Neural Network. *In the proceedings of Frontiers in Plant Science*, 1–12. <https://doi.org/10.3389/fpls.2020.00898>
- Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications *Neurocomputing*, 234:11–26.

<https://doi.org/10.1016/j.neucom.2016.12.038>

Lu, J., Ehsani, R., Shi, Y., de Castro, A. I., & Wang, S. (2018). Detection of multi-tomato leaf diseases (late blight, target, and bacterial spots) in different stages by using a spectral-based sensor.. *In the proceedings of Scientific Reports*,8(1).
<https://doi.org/10.1038/s41598-018-21191-6>

Luthria, D. L., Mukhopadhyay, S., & Krizek, D. T. (2006). Content of total phenolics and phenolic acids in tomato (*Lycopersicon esculentum* Mill.) fruits as influenced by cultivar and solar UV radiation. *In the proceedings of Journal of Food Composition and Analysis*, 19(8), 771–777. <https://doi.org/10.1016/j.jfca.2006.04.005>

Ma, Y., & Tang, J. (2021). Deep learning on graphs. *Cambridge University Press*.
[://books.google.com.tr/books?hl=tr&lr=&id=Xfw7EAAAQBAJ&oi=fnd&pg=PR13&dq=dee p+learning+on+graphs&ots=-H9tuB9Kn-&sig=46Q_77QWVlHrMfgT8lWyb-7yy9Y&redir_esc=y#v=onepage&q=deep learning on graphs&f=f=f=f](https://books.google.com.tr/books?hl=tr&lr=&id=Xfw7EAAAQBAJ&oi=fnd&pg=PR13&dq=dee+p+learning+on+graphs&ots=-H9tuB9Kn-&sig=46Q_77QWVlHrMfgT8lWyb-7yy9Y&redir_esc=y#v=onepage&q=deep+learning+on+graphs&f=f=f=f)

Mamun, M. A. Al, Karim, D. Z., Pinku, S. N., & Bushra, T. A. (2020). TLNet: A Deep CNN Model for Prediction of Tomato Leaf Diseases *ICCIT. In the proceedings of 23rd International Conference on Computer and Information Technology, Proceedings*, 19–21 <https://doi.org/10.1109/ICCIT51783.2020.9392664>

Mari, F. M. (2009). Structure and Efficiency Analysis of Vegetable Production and Marketing in By Fateh Mohammad Mari Registration No. 99-165, *Department of Agricultural Economics, Faculty of Agricultural Social Sciences, Sindh Agriculture University, Tando Jam*. 99, 223.

- McFillen, J. M., O'Neil, D. A., Balzer, W. K., & Varney, G. H. (2013). Organizational Diagnosis: An Evidence-Based Approach *Journal of Change Management*, 13(2), 223–246. <https://doi.org/10.1080/14697017.2012.679290>
- Mehta, R. (2017). History of the Tomato (Poor Man's Apple (Poor Man's Apple). *In the proceedings of Journal of Humanities and Social Science*, 22(8), 31–34. <https://doi.org/10.9790/08372208033134>
- Melanson, R. A. (2017). Common Diseases of Toads. *Int. J. Adv. Res. Biol. Sci*, 4(4), 1–10. <https://doi.org/10.22192/ijarbs>
- Melanson, R., Clarissa, B., Rick, S., & Lamar, A. (2020). Common Diseases of Tomatoes
- Militante, S. V., Gerardo, B. D., & Dionisio, N. V. (2019). Plant leaf detection and disease recognition using deep learning. *In the proceedings of 2019 IEEE Eurasia Conference on IOT, Communication, and Engineering, ECICE 2019*, 579–582. <https://doi.org/10.1109/ECICE47484.2019.8942686>
- Mkhabela, T. (2005). Technical efficiency in a vegetable-based mixed cropping sector in Tugela Ferry, Msinga District, Kwazulu-Natal, *Agrekon*, 44(2), 187–204. <https://doi.org/10.1080/03031853.2005.9523709>
- Mkonyi, L., Rubanga, D., Richard, M., Zekeya, N., Sawahiko, S., Maiseli, B., & Machuve, D. (2020). Early identification of *Tuta absoluta* in tomato plants using deep learning *Scientific African*, 10. <https://doi.org/10.1016/j.sciaf.2020.e00590>
- Modi, M., Modi, A., & Hendriks, S. (2006). POTENTIAL ROLE FOR WILD VEGETABLES IN HOUSEHOLD FOOD SECURITY. *African Journal of Food, Agriculture, Nutrition, and Development*, 6(1), 1–13.

- Nagamani, H. S., & Sarojadevi, H. (2022). Tomato Leaf Disease Detection Using Deep Learning Techniques *International Journal of Advanced Computer Science and Applications*, 13(1), 305–311. <https://doi.org/10.14569/IJACSA.2022.0130138>
- Nell, W., Wessels, B., Mokoka, J., & Machedi, S. (2006). A creative multidisciplinary approach towards the development of food gardening. *Development Southern Africa*, 17(5), 807–819. <https://doi.org/10.1080/03768350020013642>
- Nithish, E. K., Kaushik, M., Prakash, P., Ajay, R., & Veni, S. (2020). Tomato leaf disease detection using a convolutional neural network with data augmentation. *Proceedings of the 5th International Conference on Communication and Electronics Systems, ICCES*, 1125–1132. <https://doi.org/10.1109/ICCES48766.2020.09138030>
- Nyu, R., Yu, K., Lee, H., Salakhutdinov, R., & Taylor, G. (2012). Deep Learning and Feature Learning Methods for Vision. *In the proceedings of Ipam.Ucla.Edu*. https://www.ipam.ucla.edu/publications/gss2012/gss2012_10551.pdf
- ODA, M. & SAITO, T. 2006. Tomato. *Horticulture in Japan*. Kyoto Japan: The Japanese Society for Horticulture Science
- Oladejo, J. A., & Laudia, O. (2011). *Technical efficiency of tomato production in Oyo State*.
- Ozbilge, E., Ulukok, M. K., Toygar, O., & Ozbilge, E. (2022). Tomato Disease Recognition Using a Compact Convolutional Neural Network *IEEE*, 77213–77224. <https://doi.org/10.1109/ACCESS.2022.3192428>
- Patel, K. (2019). *Convolutional Neural Networks: A Beginner's Guide*, 1–13
- Prajwala TM, Alla Pranathi, Kandiraju Sai Ashritha, Nagaratna B. Chittaragi, and S. G. K. (2018). Tomato Leaf Disease Detection and Classification Using Convolutional Neural Networks. *Proceedings of the 2020 Innovations in Intelligent Systems and Applications Conference, ASYU*, 2–4. <https://doi.org/10.1109/ASYU50>

717.2020.9259832. *Proceedings of the 3rd International Conference on Intelligent Communication Technologies*

Qasrawi, R., Amro, M., Zaghal, R., Sawafteh, M., & Polo, S. V. (2021). Machine Learning Techniques for Tomato Plant Disease Clustering, Prediction, and Classification *Proceedings of the 2021 International Conference on Promising Electronic Technologies, ICPET*, 40–45 <https://doi.org/10.1109/ICPET53277.2021.00014>

Rajendran, R. K., & Thiruvengadam, M. P. (2023). *ORPN Algorithm to Diagnosis and Detect Plant Diseases Based on Image Segmentation Using Machine Learning Techniques* (No. 9760). EasyChair.

Ramakrishna, R. (2020). *A Machine Learning-Based Approach to the Detection and Classification of Tomato Plant Leaf Diseases* <http://trap.ncirl.ie/4275/>

Rangarajan, A. K., Purushothaman, R., & Ramesh, A. (2018). Tomato crop disease classification using a pre-trained deep learning algorithm. *Procedia Computer Science*, 133, 1040–1047. <https://doi.org/10.1016/j.procs.2018.07.070>

Ray, D. K., Mueller, N. D., West, P. C., & Foley, J. A. (2013). Yield Trends Are Insufficient to Double Global Crop Production by 2050. *PLoS ONE*, 8(6) <https://doi.org/10.1371/journal.pone.0066428>

Reiter, R. (1986). A theory of diagnosis from first principles *Lecture Notes in Computer Science. (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 230 LNCS (1987), 153. https://doi.org/10.1007/3-540-16780-3_87

Robert, R. (2005) Growing tomatoes University of Georgia College of Agricultural and Environmental Sciences, "Cite." 28.

- Roser, M., & Ortiz-Ospina, E. (2013). *World Population Growth for Semester 2 (H) Geography Students Paper-CC3 (TH) Topic-7. 2*, 1–21
- Rovesti, L., Haile, A., & Saunyama, I. (2019). Integrated Pest Management in tomato in Eritrea: a facilitator's field guide.
- Rubini, P. E., & Kavitha, P. (2021). Deep learning model for early prediction of plant disease.
- Sadik Tasrif Anubhove, M., Ashrafi, N., Saleque, A. M., Akter, M., & Saif, S. U. (2020). Machine Learning Algorithm-Based Disease Detection in Tomato with Automated Image Telemetry for Vertical Farming. *In the proceedings of 2020 International Conference on Computational Performance Evaluation, ComPE*, 250–254. <https://doi.org/10.1109/ComPE49325.2020.9200129>
- Sardogan, M., Tuncer, A., & Ozen, Y. (2018). Plant Leaf Disease Detection and Classification Based on CNN with the LVQ Algorithm. *In the proceedings of 3rd International Conference on Computer Science and Engineering*, 382–385. <https://doi.org/10.1109/UBMK.2018.8566635>
- SE Gebhardt, R. T. (2002). Nutritive Value of Foods. *Bmj*, 1(4511), 888–888
- Sheril Angel, J., Mary, E. J., Dikshna, U., Malar, B. A., Diana Andrushia, A., & Neebha, T. M. (2021). Deep learning-based disease detection in tomatoes 2021 3rd International Conference on Signal Processing and Communication, ICPSC 2021, May, 388–92. <https://doi.org/10.1109/ICSPC51351.2021.9451731>
- Shoaib, M., Hussain, T., Shah, B., Ullah, I., Shah, S. M., Ali, F., & Park, S. H. (2022). Deep learningbased segmentation and classification of leaf images for the detection of tomato plant disease. *Frontiers in Plant Science*, 1–18. <https://doi.org/10.3389/fpls.2022.1031748>

- Shrestha, G., Deepshikha, Das, M., & Dey, N. (2020). Plant Disease Detection Using CNN. *Proceedings of the 2020 IEEE Applied Signal Processing Conference, ASPCON*, 109–113 <https://doi.org/10.1109/ASPCON49795.2020.9276722>
- Singer, J. A., & Salovey, P. (1988). Mood and memory: Evaluating the network theory of affect.. *In the proceedings of Clinical Psychology Review*, 8(2), 211–251. [https://doi.org/10.1016/02727358\(88\)90060-8](https://doi.org/10.1016/02727358(88)90060-8)
- Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., & Batra, N. (2020). PlantDoc: A dataset for visual plant disease detection. *In the proceedings of ACM International Conference Proceeding Series*, 249–253. <https://doi.org/10.1145/3371158.3371196>
- Smith, A. F. (1994). The Tomato in America: Early History, Culture, and Cookery, Columbia, SC, USA: *In the proceedings of University of South Carolina Press*, 3–5.
- Snipes, Z. B. (2015) Tomato Diseases and Disorders Clemson Extension Home and Garden Information. *Center Home & Garden Information Center*, 1–11.
- Sreelatha, P., Udayakumar, M. S., Karthick, S., Ch, S. C., Kavya, K. C. S., & Madijagan, M. (2021). Managing Tomato Leaf Disease Detection Accuracy Using Computer Vision-Based Deep Neural Network. *Journal of Contemporary Issues in Business and Government*, Vol. 27, No. 1, pp. 3425–3437
- Sumalatha, G., Singothu, J. R., & Rao, S. K. (2021). Transfer learning-based plant disease detection. *International Journal for Innovative Engineering and Management Research*, 10, 469–477. <https://doi.org/10.48047/ijiemr/v10/i03/99>
- Sundström, J. F., Albiñ, A., Boqvist, S., Ljungvall, K., Marstorp, H., Martiin, C., Nyberg, K., Vgsholm, I., Yuen, J., & Magnusson, U. (2014). Future threats to agricultural food production posed by environmental degradation, climate change, and animal and plant diseases: a risk analysis in three economic and climate settings. *In the*

proceedings of Food Security, 6(2), 201– 215. <https://doi.org/10.1007/s12571-014-0331-y>

Sunidhi, N., & Jalaja, S. (2021). AI-Based Automatic Crop Disease Detection System *Proceedings of CONECCT 2021: In the proceedings of 7th IEEE International Conference on Electronics, Computing, and Communication Technologies*, 1–6. <https://doi.org/10.1109/CONECCT52877.2021.962270>

Taley, D. S., & Pathak, B. (2020). Comprehensive study of software testing techniques and strategies: a review. *Int. J. Eng. Res*, 9(08), 817-822.

Trivedi, N. K., Gautam, V., Anand, A., Aljahdali, H. M., Villar, S. G., Anand, D., Goyal, N., & Kadry, S. (2021). Early detection and classification of tomato leaf disease using a high-performance deep neural network. *Sensors*, 21(23). <https://doi.org/10.3390/s21237987>

Tulshan, A. S., & Raul, N. (2019). Plant leaf disease detection using machine learning. *In the proceedings of 10th International Conference on Computing, Communication, and Networking Technologies, ICCCNT*, 1–6 <https://doi.org/10.1109/ICCCNT45670.2019.8944556>

Tüzel, Y., & Ztekin, G. (2017). Good Agricultural Practices for greenhouse vegetable production in the Southeast European countries *European Journal of Horticultural Science*, 7(5), 12. <https://doi.org/10.17660/eJHS.2018/83.5.3>

Vanguard-Newspaper. (2016). *Kaduna state government on Tuesday declared a state of emergency after months* 1–1.

Verma, S., Chug, A., & Singh, A. P. (2018). Prediction Models for Identification and Diagnosis of Tomato Plant Diseases *2018 International Conference on Advances in*

- Computing, Communications, and Informatics, ICACCI*, 1557–1563.
<https://doi.org/10.1109/ICACCI.2018.8554842>
- Vetal, S., & R.S., K. (2017). Tomato Plant Disease Detection using Image Processing *Ijarce*, 6(6), 293–297. <https://doi.org/10.17148/ijarce.2017.6651>
- Vijay, N. (2021). Detection of plant diseases in tomato leaves: *With focus on providing explainability and evaluating user trust*.
- Villanueva, E. (2018). An overview of recent studies of tomato (*Solanum lycopersicum* spp) from a social, biochemical and genetic perspective on quality parameters. *Sveriges Lantbruksuniversitet, December*.
- Virtual Mobile Networks, ICICV*, 1104–1107.
<https://doi.org/10.1109/ICICV50876.2021.9388538>
- Waiganjo, M. M., Wabule, N. M., Nyongesa, D., Kibaki, J. M., Onyango, I., Wepukhulu, S. B., & Muthoka, N. M. (2006). Tomato production in Kirinyaga district, Kenya: a baseline survey report. *In the proceedings of Kenya Agricultural Research Institute, Nairobi, Kenya*, 1–43.
- Wang, Q., Qi, F., Sun, M., Qu, J., & Xue, J. (2019). Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques. *Computational Intelligence and Neuroscience*.
<https://doi.org/10.1155/2019/9142753>
- Wspanialy, P., & Moussa, M. (2020). A detection and severity estimation system for generic diseases of tomato greenhouse plants. *Computers and Electronics in Agriculture*, 178, 105701. <https://doi.org/10.1016/j.compag.2020.105701>

- Wu, J. (2017). Introduction to Convolutional Neural Networks, 1–31
https://web.archive.org/web/20180928011532/https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf
- Yamamoto, E., Matsunaga, H., Onogi, A., Kajiya-Kanegae, H., Minamikawa, M., Suzuki, A., Shirasawa, K., Hirakawa, H., Nunome, T., Yamaguchi, H., Miyatake, K., Ohyama, A., Iwata, H., & Fukuoka, H. (2016). A simulation-based breeding design that uses whole-genome prediction in tomatoes. *Scientific Reports*, 1–11.
<https://doi.org/10.1038/srep19454>
- Zachary, B.S. (2015). Tomato Diseases & Disorders Clemson Extension Home and garden Information center.
- Zadeh, L. A., & Ragazzini, J. R. (1950). An extension of Wiener's theory of prediction. *Journal of Applied Physics*, 21(7), 645-655.
<https://doi.org/10.1063/1.1699725>
- Zaki, S. Z. M., Zulkifley, M. A., Mohd Stofa, M., Kamari, N. A. M., & Mohamed, N. A. (2020). Classification of tomato leaf diseases using MobileNetV2. *In the proceedings of International Journal of Artificial Intelligence*, 9(2), 290–296,
<https://doi.org/10.11591/ijai.v9.i2.pp290-296>
- Zhang, K., Wu, Q., Liu, A., & Meng, X. (2018). Can deep learning identify tomato leaf disease? *In the proceedings of Advances in Multimedia*.
<https://doi.org/10.1155/2018/6710865>
- Zhao, S., Peng, Y., Liu, J., & Wu, S. (2021). Tomato leaf disease diagnosis based on an improved convolutional neural network by the attention module. *In the proceedings of Agriculture (Switzerland)*, 11(7)
<https://doi.org/10.3390/agriculture11070651>

APPENDIX A

SOURCE CODE LISTING

```
# Miscellaneous
*.class
*.log
*.pyc
*.swp
.DS_Store
.atom/
.buildlog/
.history
.svn/
migrate_working_dir/
# IntelliJ related
*.iml
*.ipr
*.iws
.idea/
# The .vscode folder contains launch configuration and tasks you configure in
# VS Code which you may wish to be included in version control, so this line
# is commented out by default.
#.vscode/
# Flutter/Dart/Pub related
**/doc/api/
**/ios/Flutter/.last_build_id
.dart_tool/
.flutter-plugins
.flutter-plugins-dependencies
.packages
.pub-cache/
.pub/
/build/
# Symbolication related
```

```

app.*.symbols
#Obfuscation related
app.*.map.json
# Android Studio will place build artifacts here
/android/app/debug
/android/app/profile
/android/app/release
45 .metadata
@@ -0,0 +1,45 @@
# This file tracks properties of this Flutter project.
# Used by Flutter tool to assess capabilities and perform upgrades etc.
#
# This file should be version controlled.
Version:
revision: d9111f64021372856901a1fd5bfb386cade3318
channel: stable
project_type: app
# Tracks metadata for the flutter migrate command
migration:
platforms:
- platform: root
  create_revision: d9111f64021372856901a1fd5bfb386cade3318
  base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: android
  create_revision: d9111f64021372856901a1fd5bfb386cade3318
  base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: ios
  create_revision: d9111f64021372856901a1fd5bfb386cade3318
  base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: linux
  create_revision: d9111f64021372856901a1fd5bfb386cade3318
  base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: macos

```

```

create_revision: d9111f64021372856901a1fd5bfb386cade3318
base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: web
create_revision: d9111f64021372856901a1fd5bfb386cade3318
base_revision: d9111f64021372856901a1fd5bfb386cade3318
- platform: windows
create_revision: d9111f64021372856901a1fd5bfb386cade3318
base_revision: d9111f64021372856901a1fd5bfb386cade3318
# User provided section
# List of Local paths (relative to this file) that should be
# ignored by the migrate tool.
#
# Files that are not part of the templates will be ignored by default.
unmanaged_files:
- 'lib/main.dart'
- 'ios/Runner.xcodeproj/project.pbxproj'
16 README.md
@@ -0,0 +1,16 @@
# aiapp
A new Flutter project.
## Getting Started
This project is a starting point for a Flutter application.
A few resources to get you started if this is your first Flutter project:
- [Lab: Write your first Flutter app](https://docs.flutter.dev/get-started/codelab)
- [Cookbook: Useful Flutter samples](https://docs.flutter.dev/cookbook)
For help getting started with Flutter development, view the
[online documentation](https://docs.flutter.dev/), which offers tutorials,
samples, guidance on mobile development, and a full API reference.
29 analysis_options.yaml
@@ -0,0 +1,29 @@
# This file configures the analyzer, which statically analyzes Dart code to
# check for errors, warnings, and lints.
#

```

```

# The issues identified by the analyzer are surfaced in the UI of Dart-enabled
# IDEs (https://dart.dev/tools#ides-and-editors). The analyzer can also be
# invoked from the command line by running `flutter analyze`.
# The following line activates a set of recommended lints for Flutter apps,
# packages, and plugins designed to encourage good coding practices.
include: package:flutter_lints/flutter.yaml

linter:
  # The lint rules applied to this project can be customized in the
  # section below to disable rules from the `package:flutter_lints/flutter.yaml`
  # included above or to enable additional rules. A list of all available lints
  # and their documentation is published at
  # https://dart-lang.github.io/linter/lints/index.html.
  # Instead of disabling a lint rule for the entire project in the
  # section below, it can also be suppressed for a single line of code
  # or a specific dart file by using the `// ignore: name_of_lint` and
  # `// ignore_for_file: name_of_lint` syntax on the line or in the file
  # producing the lint.

rules:
  # avoid_print: false # Uncomment to disable the `avoid_print` rule
  # prefer_single_quotes: true # Uncomment to enable the `prefer_single_quotes` rule
  # Additional information about this file can be found at
  # https://dart.dev/guides/language/analysis-options
  13 android/.gitignore
  @@ -0,0 +1,13 @@
  gradle-wrapper.jar
  /.gradle
  /captures/
  /gradlew
  /gradlew.bat
  /local.properties
  GeneratedPluginRegistrant.java

# Remember to never publicly share your keystore.
# See https://flutter.dev/docs/deployment/android#reference-the-keystore-from-the-app

```

```

key.properties
**/*.keystore
**/*.jks
76 android/app/build.gradle
@@ -0,0 +1,76 @@
def localProperties = new Properties()
def localPropertiesFile = rootProject.file('local.properties')
if (localPropertiesFile.exists()) {
    localPropertiesFile.withReader('UTF-8') { reader ->
        localProperties.load(reader)
    }
}

def flutterRoot = localProperties.getProperty('flutter.sdk')
if (flutterRoot == null) {
    throw new GradleException("Flutter SDK not found. Define location with flutter.sdk in the
    local.properties file.")
}

def flutterVersionCode = localProperties.getProperty('flutter.versionCode')
if (flutterVersionCode == null) {
    flutterVersionCode = '1'
}

def flutterVersionName = localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
    flutterVersionName = '1.0'
}

apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
android {
    compileSdkVersion flutter.compileSdkVersion
    ndkVersion flutter.ndkVersion

    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
    }
}

```

```

targetCompatibility JavaVersion.VERSION_1_8
}
kotlinOptions {
jvmTarget = '1.8'
}
sourceSets {
main.java.srcDirs += 'src/main/kotlin'
}
defaultConfig {
//TODO:Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).
applicationId "com.example.aiapp"
// You can update the following values to match your application needs.
// For more information, see: https://docs.flutter.dev/deployment/android#reviewing-the-
build-configuration.
minSdkVersion 21
targetSdkVersion flutter.targetSdkVersion
versionCode flutterVersionCode.toInteger()
versionName flutterVersionName
}
buildTypes {
release {
// TODO: Add your own signing config for the release build.
// Signing with the debug keys for now, so `flutter run --release` works.
signingConfig signingConfigs.debug
}
}
aaptOptions {
noCompress 'tflite'
noCompress 'lite'
}
}
flutter {

```

```

source '../..'
}
dependencies {
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
}
8 android/app/src/debug/AndroidManifest.xml
@@ -0,0 +1,8 @@
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.aiapp">
<!-- The INTERNET permission is required for development. Specifically,
the Flutter tool needs it to communicate with the running application
to allow setting breakpoints, to provide hot reload, etc.
-->
<uses-permission android:name="android.permission.INTERNET"/>
</manifest>
34 android/app/src/main/AndroidManifest.xml
@@ -0,0 +1,34 @@
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.aiapp">
<application
android:label="aiapp"
android:name="${applicationName}"
android:icon="@mipmap/ic_launcher">
<activity
android:name=".MainActivity"
android:exported="true"
android:launchMode="singleTop"
android:theme="@style/LaunchTheme"
android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSi
ze|locale|layoutDirection|fontScale|screenLayout|density|uiMode"
android:hardwareAccelerated="true"
android:windowSoftInputMode="adjustResize">
<!-- Specifies an Android theme to apply to this Activity as soon as

```

the Android process has started. This theme is visible to the user while the Flutter UI initializes. After that, this theme continues to determine the Window background behind the Flutter UI. -->

```
<meta-data
```

```
  android:name="io.flutter.embedding.android.NormalTheme"
```

```
  android:resource="@style/NormalTheme"
```