SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

A

PROJECT STAGE-I REPORT ON

“Corn Leaves Disease Detection Using Deep Learning”

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY (COMPUTER ENGINEERING)

BY

MR. ADITYA MENDHAKAR	[UCS20M1080]
MR. ASHITOSH ROHOM	[UCS20M1110]
MR. CHINMAY SADAPHAL	[UCS20M1114]
MR. TEJAS SALVE	[UCS20M1118]

UNDER THE GUIDANCE OF

Dr. A. V. BRAHMANE

Department of Computer Engineering

SANJIVANI COLLEGE OF ENGINEERING

KOPARGAON-423603

(AN AUTONOMOUS INSTITUTE)

2023-2024

[10/2023-2024]

Sanjivani College of Engineering, Kopargaon-423603

(An Autonomous Institute)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the project entitled

“Corn Leaves Disease Detection Using Deep Learning”

Submitted by

MR. ADITYA MENDHAKAR	**[UCS20M1080]**
MR. ASHITOSH ROHOM	**[UCS20M1110]**
MR. CHINMAY SADAPHAL	**[UCS20M1114]**
MR. TEJAS SALVE	**[UCS20M1118]**

is a bonafide work carried out by students under the supervision of Dr. A. V. BRAHMANE and it is submitted towards the partial fulfillment of the requirement of Bachelor of Technology (Computer Engineering).

During the Academic Year 2023-24

Dr. A. V. BRAHMANE
[Internal Guide]

Dr. S.R. Deshmukh
[Project Co-Ordinator]

Dr. D. B. Kshirsagar
[Head of Dept.]

Dr. A. G. Thakur
[Director]

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A

PROJECT STAGE- I REPORT

ON

“Corn Leaves Disease Detection Using Deep Learning”

Is Successfully Completed By

MR. ADITYA MENDHAKAR	**[UCS20M1080]**
MR. ASHITOSH ROHOM	**[UCS20M1110]**
MR. CHINAMY SADAPHAL	**[UCS20M1114]**
MR. TEJAS SALVE	**[UCS20M1118]**

At

DEPARTMENT OF COMPUTER ENGINEERING

SANJIVANI COLLEGE OF ENGINEERING,KOPARGAON – 423603

(AN AUTONOMOUS INSTITUTE)

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

ACADEMIC YEAR 2023-24

Dr. A. V. BRAHMANE
[Internal Guide]

Dr. S.R. Deshmukh
[Project Co-Ordinator]

Dr. D. B. Kshirsagar
[Head of Dept.]

Dr. A. G. Thakur
[Director]

ACKNOWLEDGEMENT

"Corn Leaves Disease Detection Using Deep Learning" has been a subject with tremendous scopeto research, which leads one's mind to explore new heights in the field of Computer Engineering, and its miscellaneous applications. We dedicate all our project work to our esteemed guide **Prof. A. V. BRAHMANE** whose interest and guidance helped us to complete the work successfully as well as he has provided facilities to explore the subject with more enthusiasm.

This experience will always encourage us to do our work perfectly and professionally. We also extend our gratitude to **Dr. D. B. Kshirsagar (H.O.D. Computer Department).**

We express our immense pleasure and thankfulness to all the teachers and staff of the Department of Computer Engineering, Sanjivani College of Engineering, Kopargaon for their cooperation and support. Last but not least, we thank all others, and especially our parents and friends, who in one way or another, helped us in the successful completion of this project.

MR. ADITYA MENDHKAR
MR. ASHITOSH ROHOM
MR. CHINMAY SADAPHAL
MR. TEJAS SALVE

(B.TECH. COMPUTER)

ABSTRACT

Agriculture been the most important work or task which has no end, the work with such an endless entity needs automation and smartness.As population increasing day by day the number to people to be stomach filled are increasing. To fulfill this increasing demand for food increase in yield is major concern. This needs hybridization, each innovation has major profits and some losses. In this case the crops have been acquainted to various diseases to tackle such an huge proportion diseases their need to have an prior knowledge about them and also quick identification for this disease. The document mainly focuse on the priority solutions for disease detection in crops(majorly corn) using machine learning and deep learning techniques. To maintain such an interesting disease record using image comparison to train the data the large dataset with huge sample images to train the model is must. The paper focuses on methodology and future trends in this disease detection with the learning technologies and maintaing accuracy at the results.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Problem Definition

The project mainly focus on detecting leaf disease in corn crop. Corn has been extensively grown as it as multiple uses and has a great market, as the crop maize is profitable it also has large amount of diseases been prone to to tackle this diseases on leaves a system based on machine learning can help detecting diseases on the maize. With the help of huge amount of sample images in the dataset to detect the diseases in the maize leave works on training and testing model designed using machine learning algorithms.

1.2 Literature Review/Relevant Theory

Crop diseases are one of the major challenges in the agriculture sector. Traditionalmethods for detecting crop diseases are not time and cost-efficient, as well as mis-diagnosis rate is also high. Moreover, the misdiagnosis of the diseases reduces cropyield efficiency and increases economic loss. In the advent era of machine learning,the deep neural network show prominent outcomes and allowed researchers to im-prove the accuracy of object detections and recognition systems. The main objectiveof the study to find suitable deep-learning-based architectures to classify and local-ize corn leaf diseases. We propose a convolutional neural network-based systemwith data augmentation combined with transfer learning and hyperparameter tun-ing for disease classification and localization tasks.

Conference/Journal: Published in 2021 8th IEEE Conference on Data and Network Technologies.

Paper Title: Corn Leaf Diseases Diagnosis Based on K-Means Clustering and Deep Learning.

Author: Majdi Mafarja, Hamza Turabieh

Corn is currently the highest-yielding food crop around the world, an important food, and industrial raw material. The stable and healthy development of corn production plays a pivotal role in food security, farmers' income growth, and the national economy. Corn diseases directly affect its yield and quality. There are more than a dozen common diseases in corn, most of which occur in leaves, ears, and roots. Among them, leaf spots and rust are typical. Leaf spot, there are oval or rectangular, spindle-shaped lesions on the leaves, with yellow-brown halos around them, 5-10cm long and 1.2-1.5 cm wide. In severe cases, several lesions are connected, and the leaves die early. Rust disease mainly occurs in the middle and upper leaves of the plant. At first, small light-yellow dots scattered or clustered on the front of the leaf, then protruded and expanded to round to oblong, yellowish-brown, or brown, and the surrounding epidermis turned up. Gray leaf spot, also known as corn Cercospora leaf spot and corn mildew, is a more severe disease. The initial stage of the disease is light brown spots in the shape of water stains, which extend parallel to the veins and are often rectangular. However, the diagnosis of corn diseases has mainly relied on agricultural exports for field identification. This method has many shortcomings, such as subjective, high cost of time and energy, low efficiency, and so on . Therefore, it is vital to be able to accurately and quickly identify corn leaf diseases.

1.3 Scope

1. Disease Classification: Machine learning can be used to classify a wide range of corn diseases, including but not limited to rust, leaf blight, stalk rot, and viral infections. Developing models that can accurately distinguish between these diseases is a critical aspect of the scope.

2. Early Detection: The ability to detect diseases at their earliest stages, even before visible symptoms appear, can greatly improve the effectiveness of disease management. ML models can be trained to identify subtle changes in plant health, facilitating early intervention.

3. Image-Based Detection: Machine learning algorithms can analyze images of corn plants to identify disease symptoms, such as leaf discoloration, lesions, and unusual growth patterns. The scope includes the development of robust image recognition models for this purpose.

1.4 Objectives

- Detect diseases in corn plants at an early stage, allowing for prompt intervention and treatment.
- Develop machine learning models that can accurately identify and classify various corn diseases, ensuring reliable results to guide disease management decisions.
- Create a user-friendly interface or mobile application that allows farmers to easily access and interpret disease detection results in the field.
- Use machine learning models as a tool for educating farmers about corn diseases, their causes, and prevention strategies.
- Enable the model to distinguish between different corn diseases and potentially non-disease conditions.

Chapter 2

REQUIREMENT ANALYSIS

Requirements Analysis or requirement engineering is a process of determining user expectations for new software or providing updates for previous products. These core points must be measurable, relevant and detailed. In the software engineering field this term is also called functional specifications. Requirements analysis mainly deals with communication with users or customers to determine system feature expectations, requirements and reduce convicts as demanded by various software users. Energy should be directed towards ensuring that the system or product conforms to user needs rather than attempting to turn user expectations to the requirements.

2.1 Requirement Specifications

Requirement specification describes the function and performance of the computer based system and constraints which govern its development. It can be a written document, a set of graphical models, a collection of scenarios, or any combination of above. These are of 3 types:

1. NR: Normal Requirements
2. ER: Expected Requirements
3. XR: Exciting Requirements

2.1.1 Normal Requirements

These are the requirements which are clearly stated by the customer so all these requirements will be present in the project for user satisfaction.

- NR1: System should analysis the corn leaves disease.
- NR2: Maize leaves disease will also depend on the outcome of root.
- NR3: Prediction of result of diseases will also depend upon the previous outcomes of those diseases.

2.1.2 Expected Requirements

These requirements are expected by the customer but not clearly stated by the customer. These are implicit types of requirements.

- ER1: System should be fast and reliable.
- ER2: System should have a neat and clean user interface.
- ER3: System should provide instruction of usage to the user.

2.1.3 Exciting Requirements

These requirements are not stated by the customer but externally provided by the developer in order to maintain a good relationship with the customer.

- XR1: Develop a real-time detection system that can process and analyze corn leaf images on the fly, enabling immediate response to disease outbreaks.

2.2 Validation of Requirements

Requirements validation is the process of checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation. We usually use requirements validation to check error at the initial phase of development as the error may increase excessive rework when detected later in the development process.

In the requirements validation process, we perform a different type of test to check the requirements mentioned in the Software Requirements Specification (SRS), these checks include:

- Completeness checks

- Consistency checks
- Validity checks
- Realism checks
- Ambiguity checks
- Verifiability

The output of requirements validation is the list of problems and agreed on actions of detected problems. The lists of problems indicate the problem detected during the process of requirement validation. The list of agreed action states the corrective action that should be taken to fix the detected problem.

The development of software begins once the requirements document is ready. One of the objectives of this document is to check whether the delivered software system is acceptable. For this, it is necessary to ensure that the requirements specification contains no errors and that it specifies the users requirements correctly. Also, errors present in the SRS will adversely affect the cost if they are detected later in the development process or when the software is delivered to the user. Hence, it is necessary to detect errors in the requirements before the design and development of the software begins. To check all the issues related to requirements, requirements validation is performed. In the validation phase, the work products produced as a consequence of requirements engineering are examined for consistency, omissions, and ambiguity. The basic objective is to ensure that the SRS reflects the actual requirements accurately and clearly. The requirement checklist as follows:

1. Are all requirements consistent?
2. Are the requirements really necessary?
3. Is each requirement testable?
4. Does the requirement model properly reflect the information function and behaviour of the system to be built?

2.3 Functional Requirement

- System should take the input from the user.

- System should accept parameters like image etc.
- System should predict the result of corn diseases.
- System should predict the name of diseases.
- System should give proper information.

2.4 Non-Functional Requirement

- System shall provide guidelines for user.
- System shall live for 24 hours.
- System shall backup the data.

2.5 System Requirements

Requirements specify what features a product should include and how those features should work. They help to define the test criteria, which is vital for verification and validation.

2.5.1 Hardware Requirements

1. RAM: 4GB (min)
2. Hard Disk :20 GB
3. Processor: Intel core i3 8th Gen
4. 64-bit CPU

2.5.2 Software Requirements

- Operating system: Windows 7,10,11.
- Browser: Chrome, Firefox
- IDE: Visual Studio Code or Jupiter
- Language: Python 3.7

Chapter 3

SYSTEM MODEL

Software process model is an intellectual demonstration of a process. It presents an explanation of a process. Process models may contain activities that are part of the software process. All software process models work on the five generic framework activities such as communication, planning, modelling, construction, and deployment. Each and every activity has its own functionality. The goal of the process model is to provide guidance for systematically coordinating and monitoring the tasks that must be accomplished in order to achieve the end product and the project objective.

3.1 Process Model

Software process model is an abstract representation of a process. The goal of process model is to provide guidance for systematically coordinating and controlling the tasks that must be performed in order to achieve the end product and the project objective. Incremental model is used as the process model in our system.

3.1.1 Incremental Model

Incremental model in software engineering is a one which associates the elements of waterfall model in an iterative manner. It delivers a series of releases called increments which provide progressively more functionality for the user as each increment is delivered. In the incremental model of software engineering, the waterfall model is frequently applied in each increment. The incremental model applies linear sequences in a required pattern as calendar time passes. Each linear sequence produces an increment in the work.

A, B, C are modules of Software Product that are incrementally developedand delivered. Every subsequent release of a module adds function to the previous release. This process is

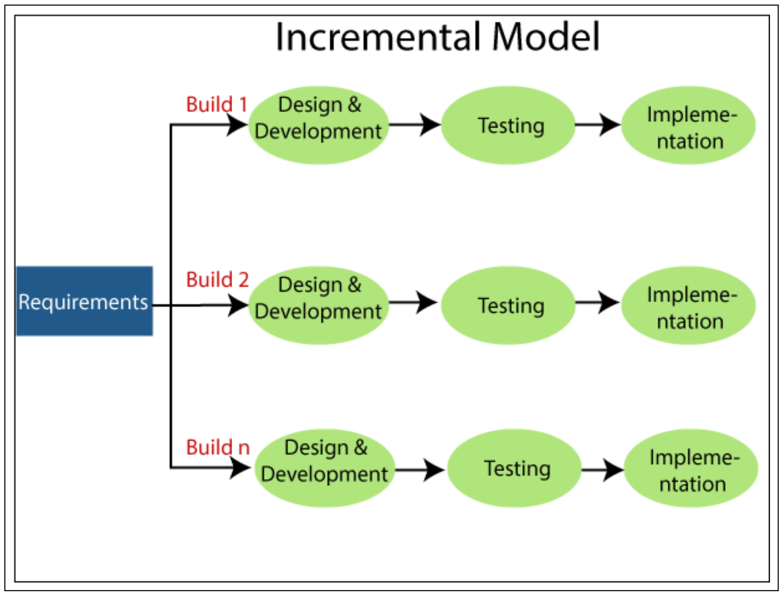


Figure 3.1: Incremental Model

continued until the complete system is achieved

Phases of Incremental Model

1. Requirement analysis: In the very first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design Development: In this phase of the development, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style anddevelopment phase.

3. Testing: In the incremental model, the testing phase checks the performance of each and every existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

4. Deployment: Once the product is tested, it is deployed in the production environment or first User Acceptance Testing (UAT) is done depending on the customer expectation. In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing.

5. Maintenance: After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

3.1.2 Why to use Incremental Model

We used Incremental model for our system design because of reasons mention below:

- Major requirements must be defined: however, some detail can evolve with time.
- There is a need to get a product to market early.
- The software will be produced quickly during the software life cycle.
- It is flexible and less luxurious to change requirements and scope.
- Though the development stages changes can be done.
- This model is less costly than others.
- A customer can answer back to each building.

3.1.3 Advantages

1. Initial product delivery is faster.
2. Lower initial delivery cost.
3. Core product is developed leading i.e., main functionality is added in the first increment.
4. After each iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly recognized because few changes are made within any single iteration.
5. It is generally easier to test and debug than other methods of software development because comparatively smaller changes are made during each iteration. This allows for more targeted and difficult testing of each element withinthe overall product.
6. With each release, a new article is added to the product.

7. Customers can reply to features and review the product.
8. Risk of changing requirements is reduced.
9. Workload is less.

3.1.4 Disadvantages

1. It requires good planning and designing.
2. Problems might be caused due to system architecture as such not all requirements are collected up front for the entire software lifecycle.
3. Each iteration phase is rigid and does not overlap each other.

Rectifying a problem in one unit requires correction in all the units and consumes a lot of time. It may arise affecting to system architecture because not all necessities are gathered up front for the entire software life cycle.

3.2 System Breakdown Structure

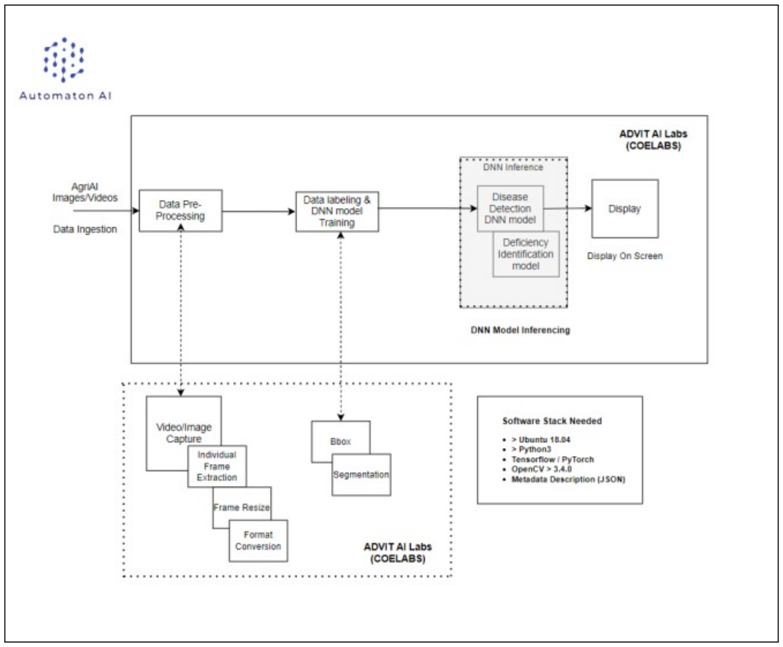


Figure 3.2: System Breakdown Structure

3.3 Module Details

These are the following modules which will be executed in this system.

1. Users
2. System

3.3.1 User

- User can give the input for particular parameter.
- User can view the predicted win or loss probability and score.

1. Parameter Input:

   In order to predict the leaves diseases of the corn the user has to enter some parameters so this module takes these parameters and sends them to the system for prediction and analysis.

2. Diseases Prediction:

   It predicts the diseases of the corn crops, the user has to enter some parameters so this module takes these parameters and sends them to the system for diseases of the corn.

3. Diseases in Percentage Prediction:

   For entered parameters the system will predict the diseases probability of the corn crops then this module will take that result and display it to the user.

3.3.2 Admin Module

- Admin module will store details in the database.
- Admin module will produce a message.
- Admin module will send a message to the user.

1. Data Collection:

   Data collection is the process of gathering and measuring information from countless different sources.

2. Pre-processing:

   Preprocessing involves adding the missing values, the correct set of data, and extracting the functionality. Data set form is important to the process of analysis.

3. Model Building:

   (a) Data Splitting: Data splitting is when data is divided into two or more subsets. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. Data splitting is an important aspect of data science, particularly for creating models based on data.

   (b) Training Prediction Model: Training data is the data you use to train an algorithm or machine learningmodel to predict the outcome you design your model to predict.

   (c) Testing Prediction Model: Test data is used to measure the performance, such as accuracy or efficiency,of the algorithm you are using to train the machine.

4. Performance analysis of machine learning algorithm:

   The performance of any ML algorithm may improve with the utilization of a distinct set of features in the same training dataset.

3.4 Project Estimation

In this section various calculation and estimations related to project has been calculated. The figure shows the system modules. The number of lines required for implementation of various modules can be estimated as follows

3.4.1 Total Time Required for Project Development

The total time required can be calculated as follows:

Table 3.1: Time Required for Project Development

Task	Time Required
Requirement Analysis and Design	2 months
Implementation and Testing	2.5 months
Total	4.5 months

Hence, total time required is nearly 4.5 months.

3.4.2 Number of Developers (N)

The project is assigned to a group of 4 people (developers). Hence, the number of developers is taken 4.

The developers are as follows:

- D1: Aditya Mendhkar.
- D2: Ashitosh Rohom.
- D3: Chinmay Sadaphal.
- D4: Tejas salve.

Chapter 4

SYSTEM ANALYSIS

4.1 Project Scheduling and Tracking

Project Preparation and Tracing is important because in order to build a complex system, many software engineering tasks occur in parallel, and the result of work performed during one task may have a reflective effect on work to be conducted in another task. The inter dependencies are very difficult to understand without a detailed schedule.

4.1.1 Project Work and Breakdown Structure (Analysis)

The project work is decomposed into the following work breakdown structure as a part of the analysis phase.

Various tasks have been mentioned in the above diagram.

- **T1 : Communication** Software development process starts with the communication between customer and developer. According to need of project, we gathered the requirements related to project. Requirement gathering is an important aspect as the developer will come to know what customer expects from the project and also he can help a customer to know more features that can be added to project as he is a technical person. The most important thing needed is that communication should be smooth and clear that means developer should easily understand the demands of customer.

- **T2 : Planning** It includes complete estimation and scheduling (complete time line chart for project development). Before starting the project tasks should be scheduled that means there should be starting and ending date assigned for each and every task and developer should work harder to complete the required task within time chosen at the time of scheduling.

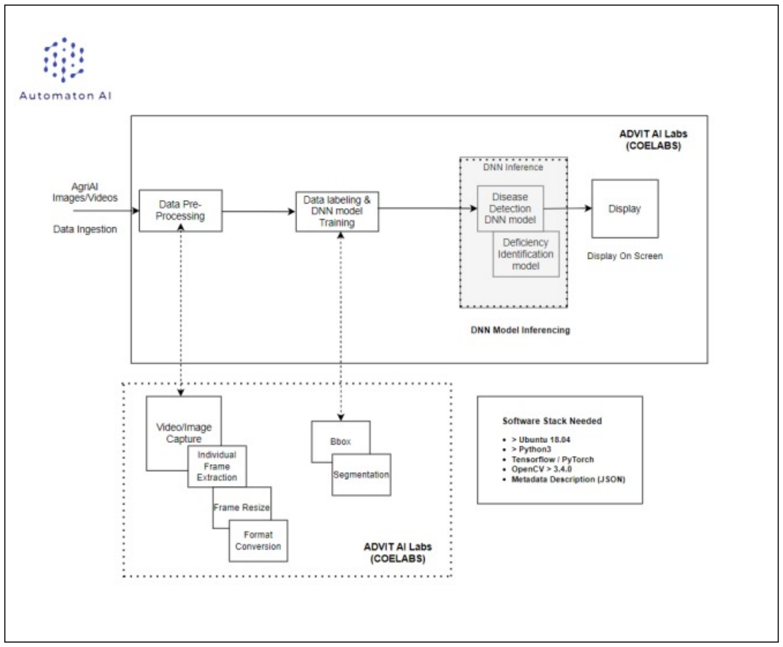


Figure 4.1: System Breakdown Structure

- **T3 : Modeling** It includes detailed requirements analysis and project design (algorithm, flowchart etc). Flowchart shows complete pictorial view of the project and algorithm is step by step solution of problem. Both flowchart and algorithm will be helpful in knowing the overall view of project and serve as a base for development of whole project.
- **T4 : Risk Management** It is a process of identifying, organizing, assessing and controlling threats to some organizations' capitals and earnings which assets overall or partial software product or performance. These threats, or risk, could stem from a wide variety of sources, including financial uncertainty, legal liabilities, strategies, management errors, accidents and natural disasters.

4.2 Project work breakdown structure (Implementation)

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

Implementation is the stage of the project when the theoretical design isturned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The modules included in the system are:

User

- Users can easily give input parameters.
- Users can view the Corn leaves diseases.

1. Parameter Input: In order to predict the leaves diseases the user has to enter some parameters so this module takes these parameters and sends them to the system for corn leaves diseases analysis.
2. Diseases Prediction: For entered parameters the system will predict the diseases of the corn leaves then this module will take that result and display it to the user.
3. Corn leaves diseases detection: For entered parameters the system will predict the diseases of the corn crops then this module will take that result and display it to the user.

Admin

- System will do the corn leaves diseases prediction.
- System will show the performance analysis of Machine Learning Algorithms.

1. Data Collection: Data collection is the process of gathering and mea- suring information from countless different sources. In order to use the data we collect to develop practical artificial intelligence (AI) and ma- chine learning solutions, it must be collected and stored in a way that makes sense for the business problem at hand.
2. Pre-processing: Preprocessing the data is considered as a significant step in the machine learning phase. Preprocessing involves adding the missing values, the correct set of data, and extracting the functionality. Data set form is important to the process of analysis. The data collected in this step will be induced in Google Colab platform

in the form of python programming in order to get the desired output. In data preprocessing we are going to use the different libraries like pan- das, Numpy, matplotlib to perform operations and analyze the data. Using these libraries, we are performing the different significant operation such as Extracting dependent and independent variables, handling the missing data, Feature scaling.

3. Feature Analysis: In this Module we are analyzing feature that how it affects the output on different values. We will analyse all the features.

4. Training Prediction Model: Training data is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict. If you are using supervised learning or some hybrid that includes that approach, your data will be enriched with data labeling or annotation. In order to train machine learning model, we have to split our data into two parts, splitting of data will be like 70% train set and 30% test set or 80% train set and 20% test set etc. Use the train test split() function in sklearn to split the sample set into a training set, which we will use to train the model, and a test set, to evaluate the model.

5. Testing Prediction Model: Test data is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine. Test data will help you see how well your model can predict new answers, based on its training. Both training and test data are important for improving and validating machine learning models.

6. Performance analysis of machine learning algorithm: In accurate prediction, machine learning (ML) algorithms and the selected features play a major role. The performance of any ML algorithm may improve with the utilization of a distinct set of features in the same training dataset. We are using following three Machine Learning Algorithms:

    (a) Linear regression: To predict the continuous values, Linear regression is used. Certain known parameters are given to the machine learning algorithms, it predicts the continuous values as output. It cannot used for the classification problems. The proposed model predicts the score using the Linear Regression.

    (b) Ridge regression: Ridge regression is also used to predict the continuous values. When the variables used for the prediction greater than the observations of when multicollinearity present in the data, ridge regression is used set has multicollinearity (correlations between predictor variables).

(c) Lasso regression: Lasso regression is a type of linear regression that used for predicting the continuous values. Shrinkage is used in the lasso regression. When data values focus towards central point shrinkage occurs. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models.

(d) Decision tree: It is supervised machine learning algorithm. It is used for classification problem. It splits the data based on the entropy and information gain. It looks like flow chart. It creates the various categories within the categories. It splits the data of the high information gain first. It created a "leaves diseases" as root node and split the data further.

(e) Random forest: Random Forest classifier creates multiple decision trees and find out the individual output. It combines all the results together and give the results with more accuracy. It can be used as both classification and regression.

(f) SVC: Classification is used, when the target variable represents particular category. Proposed system used classification for the crops diseases of the corn crops. diseases prediction is the binary classification.

4.3 Task Identification

Following analysis and design tasks are to be carried out in the process of analysis and design of the project. All project modules are divided into following tasks.

- T1: Project Definition Searching
- T2: Project Definition Preparation
- T3: Literature Collection
- T4: Project Definition Finalization
- T5: Dataset Collection
- T6: Synopsis Preparation

- T7: Requirement Analysis and Validation
- T8: Determine Process Model (Incremental Model)
- T9: System Breakdown Structure
- T10: Project Estimation
- T11: Project Scheduling and Tracking
- T12: Analysis Modelling using Behavioral, Functional and Architectural Modelling
- T13: Feasibility Management of Project using Mathematical Modelling
- T14: Risk Analysis, Project Management and Risk Management
- T15: Report Preparation
- T16: Data Preprocessing using Preprocessing Techniques
- T17: Extracting the Features from the Preprocessed Dataset
- T18: Creating Graphical User Interface
- T19: Model Training using Linear Regression, Ridge Regression, Lasso Regression, SVC, Decision Tree, Random Forest Algorithms
- T20: Predicting and analysis the corn diseases
- T21: Testing of the Generated Model
- T22: Deployment of Project

4.3.1 Project Schedule

Table 4.1 describes the schedule for project development and also highlight all the task to be carried out along with their duration, dependency and developer(s) assign to accomplish the task.

Table 4.1: Project Task Table

Task	**Days**	**Dependencies**	**Developer Assigned**
T1	5	-	D1,D2,D3,D4
T2	7	T1	D1,D2,D3,D4
T3	3	T2	D1,D2,D3,D4
T4	6	T1, T2,T3	D1,D2,D3,D4
T5	6	T2	D1,D2,D3,D4
T6	8	T3,T4	D1,D2,D3,D4
T7	9	T6	D1,D2,D3,D4
T8	5	-	D1,D2,D3,D4
T9	4	T5	D1,D2,D3,D4
T10	3	-	D1,D2,D3,D4
T11	5	-	D1,D2,D3,D4
T12	4	T7, T8, T9	D1,D2,D3,D4
T13	5	T11	D1,D2,D3,D4
T14	6	T8	D1,D2,D3,D4
T15	15	T7,T8	D1,D2,D3,D4
T16	15	T14	D1,D2,D3,D4
T17	12	T15	D1,D2,D3,D4
T18	20	T16	D1,D2,D3,D4
Total	**135**		

4.4 Project Table and Time-line Chart

4.4.1 Project Schedule Time

Table 4.2: Project Schedule Time Table

Test ID	Exp. Start Time	Act. Start Time	Exp. End Time	Act. End Time	Developers
T1	07/08/23	07/08/23	14/08/23	14/08/23	D1,D2,D3,D4
T2	14/08/23	14/08/23	21/08/23	21/08/23	D1,D2,D3,D4
T3	03/09/23	03/09/23	06/09/23	06/09/23	D1,D2,D3,D4
T4	06/09/23	08/09/23	11/09/23	12/09/23	D1,D2,D3,D4
T5	12/09/23	13/09/23	14/09/23	15/09/23	D1,D2,D3,D4
T6	15/09/23	18/09/23	25/09/23	27/09/23	D1,D2,D3,D4
T7	25/09/23	27/09/23	04/10/23	06/10/23	D1,D2,D3,D4
T8	05/10/23	09/10/23	11/10/23	13/10/23	D1,D2,D3,D4
T9	11/10/23	13/10/23	16/10/23	18/10/23	D1,D2,D3,D4
T10	16/10/23	18/10/23	18/10/23	20/10/22	D1,D2,D3,D4
T11	18/10/23	20/10/23	23/10/23	25/10/23	D1,D2,D3,D4
T12	30/10/23	30/10/23	03/11/23	03/11/23	D1,D2,D3,D4
T13	06/11/23	06/11/23	09/11/23	09/11/23	D1,D2,D3,D4
T14	10/11/23	10/11/23	14/11/23	14/11/23	D1,D2,D3,D4
T15	15/11/23	15/11/23	27/11/23	27/11/23	D1,D2,D3,D4
T16	28/11/23	28/11/23	01/12/23	01/12/23	D1,D2,D3,D4
T17	04/12/23	04/12/23	06/12/23	06/12/23	D1,D2,D3,D4
T18	06/12/23	06/12/23	06/12/23	06/12/23	D1,D2,D3,D4

4.4.2 Time-Line Chart

Timeline chart shows the progress of project development in various phases.

4.5 Analysis Modeling

The system analysis model is made up of class diagram, sequence or collaboration diagrams and state chart diagrams. Between them they constitute a logical, implementation free view of computer system that includes a detail definition of every aspect of functionality. Analysis model contains following modeling:

1. Behavioral modeling.
2. Functional modeling.
3. Architectural modeling.

Analysis modeling uses a combination of text and diagrammatic form to depict requirement for data, function and behaviour in a way that is relatively easy to understand and more important, straightforward to review for correctness, completeness and consistency.

4.5.1 Behavioral modeling

Use case diagram

A use case involves a sequence of interactions between the motivator and the system, possibly including other actors.

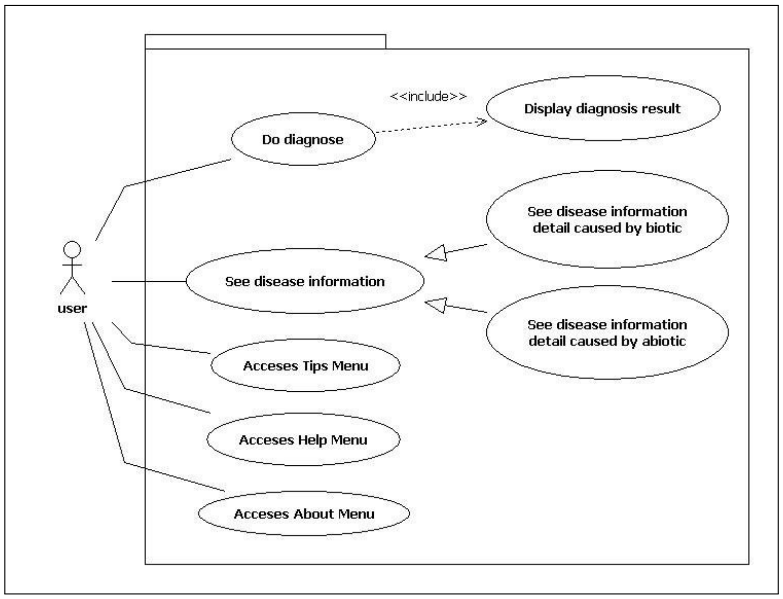


Figure 4.2: Use Case Diagram

Sequence Diagram

A sequence diagram is a graphical view of a state that shows object interface in a time-based sequence.

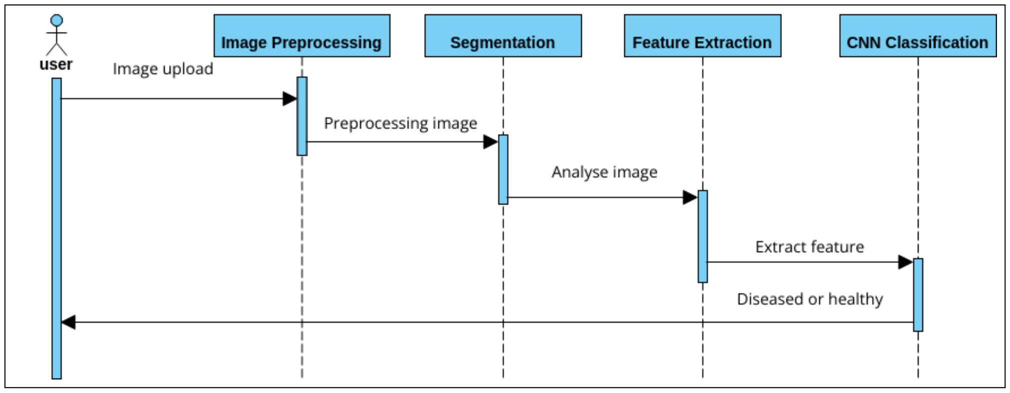


Figure 4.3: Sequence Diagram

Activity Diagram

Activity diagram is an important diagram to describe the dynamic aspects of the system. Activity diagram is essentially a flowchart to represent the flow from one activity to another activity.

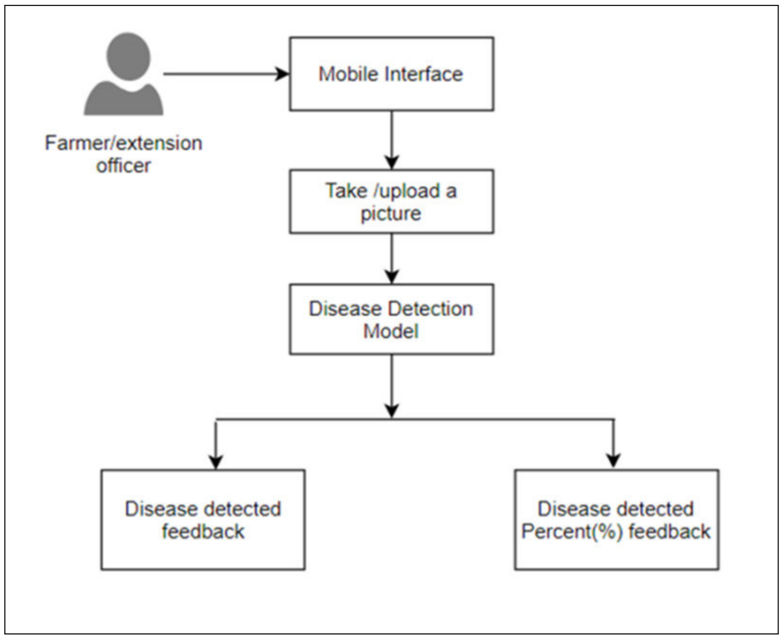


Figure 4.4: Activity Diagram

Chapter 5

RISK MANAGEMENT

Project risk management is the process of identifying, analyzing and then responding to any risk that arises over the life cycle of a project to help the project remain on track and meet its goal. Managing risk isn't reactive only, it should be part of the planning process to figure out risk that might happen in the project and how to control that risk if it in fact occurs. A risk is anything that could potentially impact your projects timeline, performance or budget. Risks are potentialities, and in a project management context, if they become realities, they then become classified as issues that must be addressed. So risk management, then, is the process of identifying, categorizing, prioritizing and planning for risks before they become issues .This article gives us ten golden rules to apply risk management successfully in our project.

5.1 Risk Identification

- **Product size risk**

  R1: Is the development team able to decompose the required program into smaller, highly cohesive modules.

  R2: Are there enough development team members available relative to the size of the project.

- **Business impact**

  R3: Delay in project delivery (violation in time constraints) can hamper the customer economically.

  R4: If system is not more efficient than the existing system, it will cause economic losses.

- **Customer related risk**

  R5: User or service provider is a non-technical person; if proper guide- lines were not mentioned then it will create ambiguity.

  R6: If a user wants any modifications that lead to changing the entire System.

- **Process risk** R7: The risk of technology errors or security incidents that disrupt or invalid processes.

  R8: Quality of a process itself that leads to failures. A low quality process may not properly work and may break down the system.

- **Technical Risk**

  R9: Lack of database stability and concurrency.

  R10: Module integration fails.

  R11: Wrong trained data may lead to wrong results.

- **Development Environment Related risk**

  R12: Lack of proper training and less knowledge of programming leads to a moderate risk. It will delay product development and deployment.

5.2 Strategies Used to Manage Risk

- S1: Formulation and follow up of the project plan on a regular basis.
- S2: Keep assigned work under certain deadlines.
- S3: Web Development cycle should be used.
- S4: Regular meetings with users reduce the risk to some extent, design systems with flexibility and maintain necessary documentation for the same.
- S5: Re-defined software process at higher degree.
- S6: Proper training on required technical tools for development of projects reduces risk.
- S7: Make certain rules that each one the members are taking part in the design.

- S8: Study and understanding of project definition, programming language.
- S9: Take a look at model development and all its associated used software.
- S10: Time constraints must be followed to avoid economical risks.
- S11: Each and every module must be tested for its functioning.
- S12: After unit testing, the system must be integrated and validated accordingly.
- S13: Integration testing for authentication hierarchy.
- S14: Use of standard database technology which supports concurrency more.

5.3 Risk Projection

5.3.1 Preparing Risk Table

The risk table shown below lists all possible risks which may occur at any stage during development of a project. Table also clearly shows the impact of the risks and RMMM (Risk Mitigation Monitoring and Management) planto deal with any such risks.

Table 5.1: Risk Management Table

Risk	Category	Probability	Impact	Plan
R1	Product Size Risk	More	High	S1,S3,S4
R2	Product Size Risk	Less	Less	S4,S6,S1
R3	Business Impact Risk	More	High	S2
R4	Business Impact Risk	More	High	S4
R5	Customer Related Risk	More	High	S11,S12,S13
R6	Customer Related Risk	More	Less	S11,S12,S13
R7	Process Risk	More	High	S14
R8	Process Risk	More	High	S14
R9	Technical Risk	Less	High	S1,S6
R10	Technical Risk	More	High	S7
R11	Development Environment Related Risk	Less	Less	S4,S5,S7,S8
R12	Product Size Risk	More	High	S9,S10

5.4 Feasibility

Feasibility is defined as an evaluation or analysis of the potential impact of a proposed project.

- Technical feasibility: - It is disturbed with specifying equipment and software that will successfully preserve the task required.
- SAT (Satisfiability): - Boolean formula is satisfiability if there exists at least one way of assigning value to its variable so as to make it true and we denote it by using SAT. The problem of deciding whether a given formula is satisfiability or not.
- Facility to produce output in given times.
- Response time under certain conditions.
- Operational Feasibility: -It is related to human organization.
- What changes will be brought in with the system.
- How organizational Structure will be distributed.
- What new skills are required.
- Economic Feasibility: -It is the most frequently used technique for evaluating the effectiveness of proposed systems. Most usually identified as cost/benefit analysis.

Chapter 6

TECHNICAL SPECIFICATION

6.1 Software requirement specification

6.1.1 Operating System (OS)

- Windows 10 or higher.

6.1.2 Integrated Development Environment (IDE)

- Visual Studio Code (VS Code) or any equivalent.

6.1.3 Programming Language

- Python 3.x

6.2 Hardware Requirement Specifications

The algorithms used in the project are computationally intensive and thus require high computing capabilities in terms of hardware. For the testing and demo purpose we will need good hardware usually found in desktop type of computers. Thus, to reduce the computation time of the project we recommend to use high performance system.

- Hard Disk: 120 GB
- Ram: 2GB
- Processor: Intel core i5 9th Gen
- 64-bit CPU

Chapter 7

IMPLEMENTATION DETAILS

In this chapter implementation details are explained as follow:

Code of data preprocessing for Corn leaves diseases detection using deep learning

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
from matplotlib.image import imread
import seaborn as sns
import matplotlib.pyplot as plt
import os
import pandas as pd

train_path = r'C:/Users/urmii/Downloads/Maize-Diseases-Detection-master'
valid_path=r'C:/Users/urmii/Downloads/Maize-Diseases-Detection-master'
```

```
    os.listdir(train_path)

import os
from skimage.io import imread
import seaborn as sns

# Assuming 'valid_path' is defined somewhere in your code
valid_path = r'C:/Users/urmii/Downloads/Maize-Diseases-
Detection-master - Copy/Traning/Dataset/test'

dim1 = []
dim2 = []

directory_path = 'C:/Users/urmii/Downloads/Maize-Diseases-Detection-
master - Copy/Traning/Dataset/test/Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot'

for image_filename in os.listdir(os.path.join
(valid_path, directory_path)):

img_path = os.path.join(valid_path, directory_path, image_filename)
img = imread(img_path)
d1, d2, colors = img.shape
dim1.append(d1)
dim2.append(d2)

sns.jointplot(dim1, dim2)

IMAGE_SIZE=[244,244]
folders=glob(train_path+'/*')

import tensorflow as tf
from keras.applications import VGG16

# Assuming you have already defined IMAGE_SIZE
```

```
vgg = VGG16(input_shape=IMAGE_SIZE + [3],
weights='imagenet', include_top=False)

for layer in vgg.layers:
    layer.trainable = False

# Print the updated TensorFlow version
print("TensorFlow version:", tf.__version__)

x = Flatten()(vgg.output)
# x = Dense(1000, activation='relu')(x)
prediction = Dense(len(folders), activation='softmax')(x)

# create a model object
model = Model(inputs=vgg.input, outputs=prediction)

# view the structure of the model
model.summary()


model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

traindata_gen=ImageDataGenerator(
                             rotation_range=10,
                             rescale=1./255,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             shear_range=0.1,
                             zoom_range=0.1,
                             fill_mode='nearest'
                              )
```

```
testdata_gen=ImageDataGenerator(rescale=1./255)

traning_set=traindata_gen.flow_from_directory(train_path,
                            target_size = (224, 224,),
                            batch_size = 32,
                            class_mode = 'categorical')

testing_set=testdata_gen.flow_from_directory(valid_path,
                        target_size = (224, 224),                         batch_si
                        class_mode = 'categorical',
                        shuffle=False)

result = model.fit_generator(
  traning_set,
  validation_data=testing_set,
  epochs=2,
  steps_per_epoch=len(traning_set),
  validation_steps=len(testing_set)
)

losses=pd.DataFrame(model.history.history)
losses[['loss','val_loss']].plot()

model.metrics_names

from tensorflow.keras.models import load_model
model.save('maize_disease_detection_new_model.h5')
```

7.0.1 Testing for single image

```
test_image='C:/Users/urmii/Downloads/Maize-Diseases-Detection-
master - Copy/Traning/Dataset/test/Corn_(maize)___Cercospora
_leaf_spot Gray_leaf_spot/065fe7da-dcaf-41be-9332-5ec5ebceb94b
___RS_GLSp 9337_270deg.JPG'


#test_image='C:\\Users\\1CYNOSA1\\Desktop\\vgg16\\valid\\Corn_(maize)
___Cercospora_leaf_spot Gray_leaf_spot\\065fe7da-dcaf-41be-9332
-5ec5ebceb94b___RS_GLSp 9337_270deg.JPG'

from tensorflow.keras.preprocessing import image

my_image=image.load_img(test_image,target_size=IMAGE_SIZE)
```

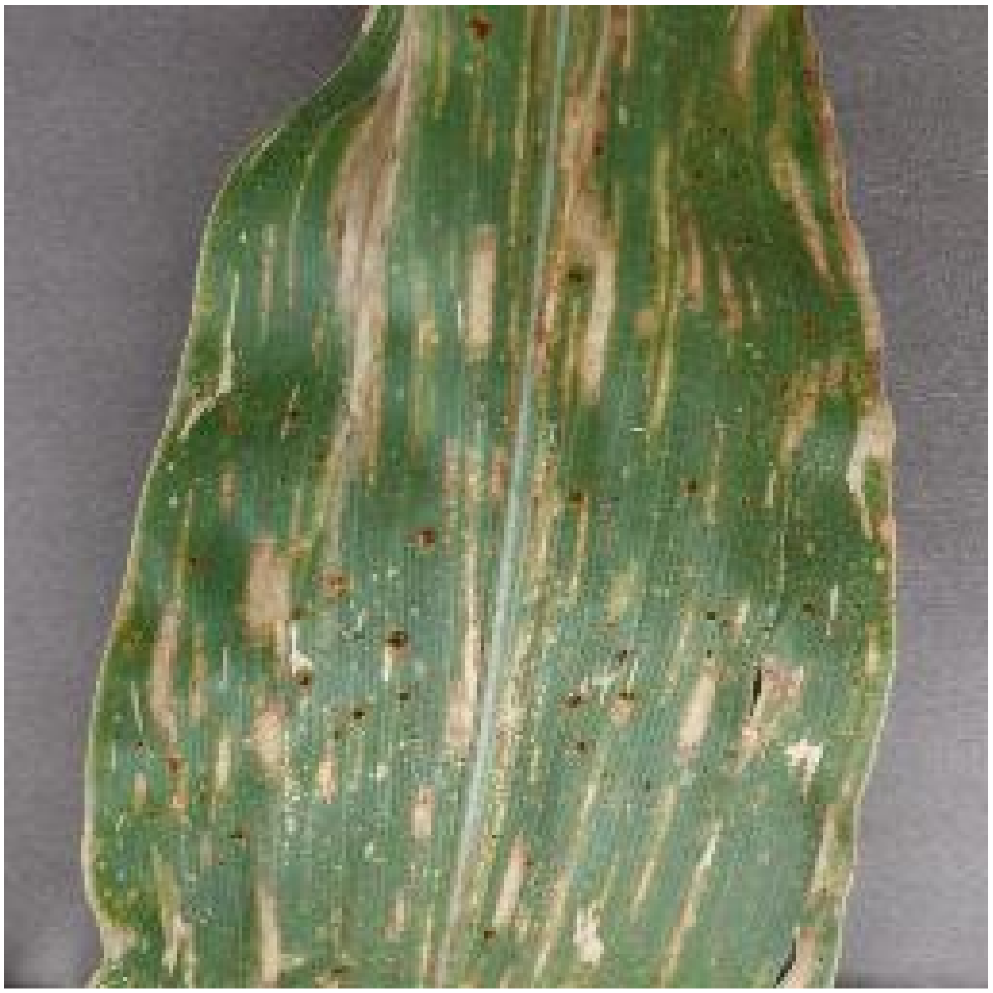


Figure 7.1: Affected corn leaf

```
my_image = image.img_to_array(my_image)
my_image=np.expand_dims(my_image,axis=0)
my_image.shape
x=model.predict(my_image)

Output:
Array([[1.0000000e+00, 3.0638152e-35, 0.0000000e+00, 0.0000000e+00]],
      dtype=float32)
```

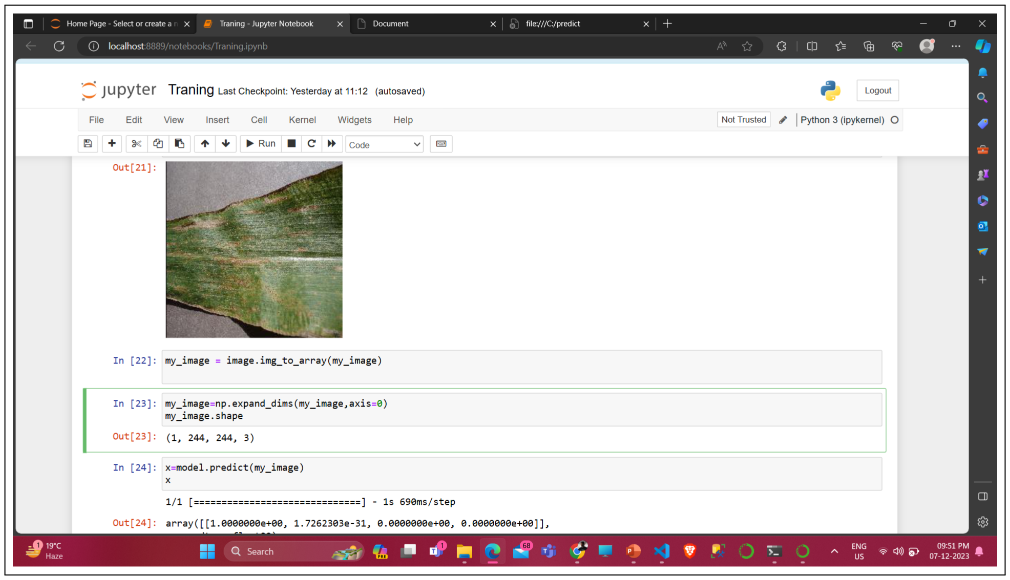
For Clean Image

```
y='C:/Users/urmii/Downloads/Maize-Diseases-Detection-master
- Copy/Traning/Dataset/test/corn_(maize)___healthy/d976fc74
-23c2-4c34-a068-db120a61fe0f___R.S_HL 5561 copy_flipLR.jpg'


#y='C:\\Users\\1CYNOSA1\\Desktop\\vgg16\\valid\\Corn_(maize)
___healthy\\d976fc74-23c2-4c34-a068-db120a61fe0f
___R.S_HL 5561 copy_flipLR.jpg'

test2=image.load_img(y,target_size=IMAGE_SIZE)
test2
```

Figure 7.2: Clean corn leaf

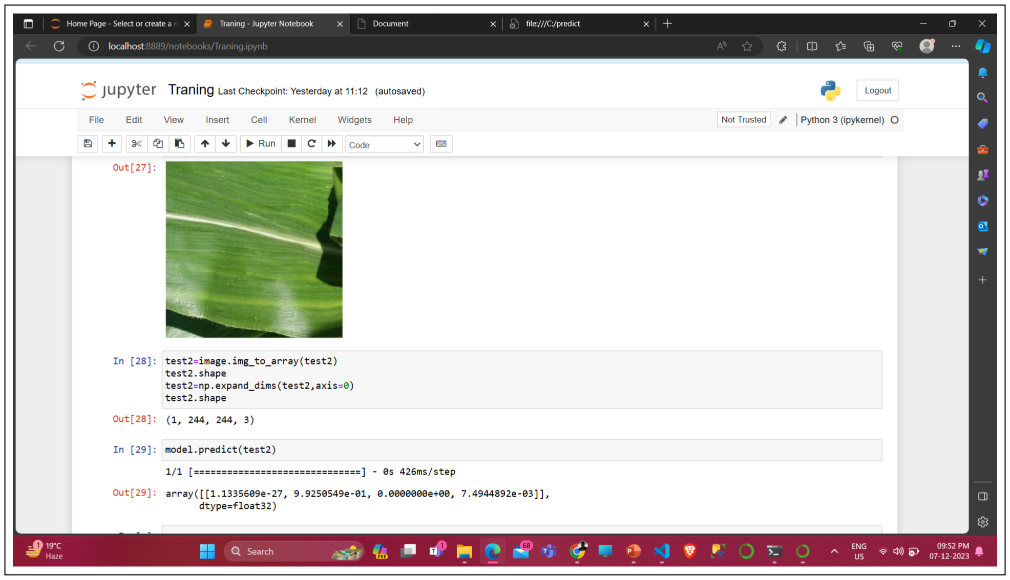
```
test2=image.img_to_array(test2)
test2.shape
test2=np.expand_dims(test2,axis=0)
test2.shape

(1, 244, 244, 3)

model.predict(test2)

Output:
array([[9.3376475e-23, 1.0000000e+00, 0.0000000e+00, 0.0000000e+00]],
      dtype=float32)
```

7.0.2 System Implementation


localhost:8889/notebooks/Traning.ipynb
jupyter Traning Last Checkpoint: Yesterday at 11:12 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
Out[21]:

In [22]: my_image = image.img_to_array(my_image)

In [23]: my_image=np.expand_dims(my_image,axis=0)
         my_image.shape

Out[23]: (1, 244, 244, 3)

In [24]: x=model.predict(my_image)
         x

         1/1 [==============================] - 1s 690ms/step

Out[24]: array([[1.0000000e+00, 1.7262303e-31, 0.0000000e+00, 0.0000000e+00]],
```

Figure 7.3: System Output Image 1


localhost:8889/notebooks/Traning.ipynb
jupyter Traning Last Checkpoint: Yesterday at 11:12 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
Out[27]:

In [28]: test2=image.img_to_array(test2)
         test2.shape
         test2=np.expand_dims(test2,axis=0)
         test2.shape

Out[28]: (1, 244, 244, 3)

In [29]: model.predict(test2)

         1/1 [==============================] - 0s 426ms/step

Out[29]: array([[1.1335609e-27, 9.9250549e-01, 0.0000000e+00, 7.4944892e-03]],
               dtype=float32)
```

Figure 7.4: System output Image 2

Chapter 8

APPLICATIONS OF THE PROJECT

- Early Disease Detection:Deep learning models can be trained to identify the early signs of diseases on corn leaves. Early detection allows farmers to take timely action, such as applying pesticides or implementing preventive measures, to control the spread of the disease and minimize crop damage.
- Precision Agriculture:By integrating deep learning models into precision agriculture systems, farmers can target specific areas of their fields that are affected by diseases. This enables more efficient use of resources, such as pesticides, as they can be applied only where needed, reducing costs and environmental impact.
- Crop Monitoring:Deep learning can be employed to continuously monitor crops, providing real-time information on the health of corn plants. This ongoing monitoring allows farmers to respond quickly to changes in disease prevalence or other environmental factors affecting crop health.
- Disease Classification:Deep learning models can classify different types of corn leaf diseases. This information is valuable for farmers to understand the specific threats to their crops, as different diseases may require different treatment strategies.
- Research and Development:Deep learning applications in corn disease detection can also support agricultural research efforts. Researchers can use the data generated by these models to study disease patterns, understand environmental factors contributing to outbreaks, and develop more resilient crop varieties.

Chapter 9

CONCLUSION & FUTURE SCOPE

9.1 Conclusion

The application of deep learning for corn leaf disease detection holds significant promise for revolutionizing modern agriculture. The integration of advanced technologies into the farming sector, specifically in the realm of disease detection, brings forth a range of benefits that contribute to sustainable and efficient crop management. Through the analysis of leaf images using deep learning models, farmers can achieve early detection of diseases, enabling timely intervention and mitigation strategies. The precision and accuracy offered by deep learning models empower farmers to adopt a targeted approach to crop protection. By focusing resources only on affected areas, they can optimize the use of pesticides and reduce the environmental impact associated with widespread application. This not only has economic advantages for farmers but also aligns with the principles of sustainable and environmentally conscious agricultural practices.

9.2 Future Scope

1. **Improved Accuracy:** Continued advancements in computer vision algorithms and deep learning techniques are likely to result in higher accuracy rates for Corn leaves diseases detection using deep learning.
2. **Enhanced Accuracy and Robustness:** Continuous efforts to improve the accuracy and robustness of deep learning models for disease detection will be crucial. This includes refining the algorithms, incorporating more diverse datasets, and addressing challenges such as variations in lighting conditions, different stages of plant growth, and diverse environmental factors.

3. **Advanced Machine Learning Models:** As technology and computing power continue to advance, more sophisticated machine learning models can be developed to improve the accuracy of Corn leaves diseases detection. Deep learning algorithms, such as recurrent neural networks (RNNs) and transformers, can be employed to capture complex patterns and dependencies in the data, leading to more precise predictions.

4. **Real-Time Monitoring Systems:** The development of real-time monitoring systems that provide instantaneous feedback to farmers will be essential. This could involve the integration of edge computing capabilities, enabling on-site analysis of images and prompt decision-making without the need for extensive data transfer to centralized servers.

5. **Disease Progression Modeling:** Developing models that can not only detect diseases but also predict their progression over time will be valuable. This can assist farmers in implementing proactive measures to manage diseases and minimize their impact on crop yield.

6. **User Customization and Personalization:** Future prediction systems can offer customization and personalization options, allowing users to tailor the predictions to their preferences and specific requirements. Users can input their own weightage to different factors or adjust the model's parameters based on their domain knowledge, thereby improving the predictions' relevance and usefulness to individual users.

7. **User-Friendly Interfaces:** The development of user-friendly interfaces for farmers that integrate seamlessly with existing farm management systems is essential. This will empower farmers to easily interpret model outputs and make informed decisions in their daily operations.

References

[1] Corn Leaf Diseases Diagnosis Based on K-Means Clustering and Deep Learning HELONG YU1, JIAWEN LIU1, CHENGCHENG CHEN 2, QIAN ZHANG4, HUILING CHEN 3 , (Associate Member, IEEE), MAJDI MAFARJA 5, AND HAMZA TURABIEH Received September 29, 2021, accepted October 11, 2021, date of publication October 15, 2021, date of current version October 27, 2021. Digital Object Identifier 10.1109/ACCESS.2021.3120379

[2] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, and J. Dong, “Identifying medical diagnoses and treatable diseases by image-based deep learning,” Cell, vol. 172, no. 5, pp. 1122–1131, Feb. 2018.

[3] S. S. Chouhan, U. P. Singh, and S. Jain, “Applications of computer vision in plant pathology: A survey,” Arch. Comput. Methods Eng., vol. 27, no. 2, pp. 611–632, Apr. 2020

[4] J. G. A. Barbedo, L. V. Koenigkan, and T. T. Santos, “Identifying multiple plant diseases using digital image processing,” Biosyst. Eng., vol. 147, pp. 104–116, Jul. 2016.