

GROUP ID: GE8
A PROJECT REPORT ON
REVOLUTIONIZING LEAF DISEASE
DETECTION: THE SEAMLESS
COMBINATION OF EDGE COMPUTING
AND DEEP LEARNING FOR IMMEDIATE
RESULTS

SUBMITTED TO THE PIMPRI CHINCHWAD
COLLEGE OF ENGINEERING AN AUTONOMOUS
INSTITUTE, PUNE
IN THE FULFILLMENT OF THE
REQUIREMENTS FOR THE
AWARD OF THE DEGREE
OF

BACHLOR OF
TECHNOLOGY COMPUTER
ENGINEERING (REGIONAL
LANGUAGE)

SUBMITTED BY

RINKU SONAWANE	122B2D069
ONKAR DOKHE	122B2D072
VIPUL WAIKAR	122B2D073
OMKAR GARVARE	122B2D074

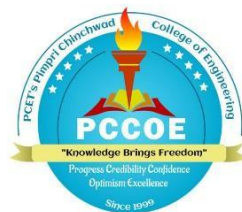
UNDER THE GUIDANCE
OF
MRS. RUCHA SHINDE



DEPARTMENT OF
COMPUTER
ENGINEERING
(REGIONAL LANGUAGE)
PCET'S PIMPRI CHINCHWAD COLLEGE OF
ENGINEERING

Sector No. 26, Pradhikaran, Nigdi, Pimpri-Chinchwad, PUNE 411044

2024-25



CERTIFICATE

This is to certify that the project report entitles

**“REVOLUTIONIZING LEAF DISEASE DETECTION: THE
SEAMLESS COMBINATION OF EDGE COMPUTING AND
DEEP LEARNING FOR IMMEDIATE RESULTS”**

Submitted by

Rinku Sonawane 122B2D069
Onkar Dokhe 122B2D072
Vipul Waikar 122B2D073
Omkar Garvare 122B2D074

are Bonafide students of this institute and the work has been carried out by them under the supervision of **Mrs. Rucha Shinde** and it is approved for the partial fulfillment of the requirement of Pimpri Chinchwad College of Engineering an autonomous institute, for the award of the B. Tech. degree in Computer engineering (Regional Language).

(Mrs. Rucha Shinde)

Guide

Department of Computer engineering

(Regional Language)

(Prof. Dr. Rachana Patil)

Head,

Department of Computer engineering

(Regional Language)

(Prof. Dr. G.N. Kulkarni)

Director,

Pimpri Chinchwad College of Engineering Pune – 411044

Place:

Date:

ACKNOWLEDGEMENT

We express our sincere thanks to our Guide Mrs. Rucha shinde for her constant encouragement and support throughout our project, especially for the useful suggestions given during the course of project and having laid down the foundation for the success of this work.

We would also like to thank our Project Coordinator, Prof. Rohini Sarode for her assistance, genuine support and guidance from early stages of the project. We would like to thank Prof. Dr. Rachana Y. Patil, Head of Computer Department (Regional Language) for her unwavering support during the entire course of this project work. We are very grateful to our Director, Prof. Dr. G.N. Kulkarni for providing us with an environment to complete our project successfully. We also thank all the staff members of our college and technicians for their help in making this project a success. We also thank all the web committees for enriching us with their immense knowledge. Finally, we take this opportunity to extend our deep appreciation to our family and friends, for all that they meant to us during the crucial times of the completion of our project.

Rinku Sonawane

Onkar Dokhe

Vipul Waikar

Omkar Garvare

ABSTRACT

Plant diseases pose a significant threat to agricultural productivity, causing substantial crop losses worldwide. Prompt and accurate detection of these diseases is crucial for effective disease management and ensuring food security. In this project, we propose a solution to automate the detection of plant leaf diseases using deep learning techniques. We leverage the power of VGG19, a pre-trained deep convolutional neural network (CNN) architecture, implemented through the TensorFlow framework to build a robust disease detection model. VGG19, known for its depth and accuracy in image classification tasks, enables our model to effectively extract features and classify various plant leaf diseases with high precision. By analyzing high-resolution images of plant leaves, our system can accurately identify and categorize disease types affecting crop plants. The use of TensorFlow facilitates efficient training and optimization of the VGG19-based model, ensuring superior performance in both accuracy and computational efficiency. To provide a user-friendly interface and enable seamless integration with existing systems, we develop a web application using FastAPI, a modern and high-performance web framework for building APIs in Python. This application serves as a platform for farmers and stakeholders in the Indian agriculture sector to conveniently access our disease detection system. Early detection enables timely preventive measures, such as targeted treatments and improved crop management strategies, thereby minimizing losses and enhancing agricultural productivity.

KEYWORDS: Sentiment Analysis, Convolutional Neural Networks, Machine Learning, Feature Extraction, Support Vector Classifier (SVC), Multichannel CNN, Word Embeddings, Text Classification, Web-Based Application, Real-Time Analysis, Ensemble Methodologies, Deep Learning, Social Media Monitoring, Customer Feedback, Market Trends, Sentiment Polarity, Classification Accuracy.

TABLE OF CONTENT

Sr. No.	Title of Chapter	Page No.
01	Introduction	11
1.1	Domain Description	11
1.2	Problem Definition	12
1.3	Goals and Objectives	13
1.4	Motivation	13
1.5	Scope of the work	14
1.6	Outcomes	14
02	Literature Survey	15
2.1	Literature Survey	15
2.2	Study of Algorithm from Literature Review	16
2.3	Existing Methods/Tools	18
03	Software Requirement Specification	20
3.1	Functional Requirements	20
3.1.1	System Features (Functional Requirements)	20
3.2	External Interface Requirements	21
3.2.1	User Interfaces	21
3.2.2	Hardware Interfaces	21
3.2.3	Software Interfaces	22
3.2.4	Communication Interfaces	22
3.3	Nonfunctional Requirements	22
3.3.1	Performance Requirements	22
3.3.2	Safety / Security Requirements	22
3.4	System Requirements	22

3.4.1	Database Requirements	22
3.4.2	Software Requirements (Platform Choice)	23
3.4.3	Hardware Requirements	23
04	Mathematical Model	24
4.1	The Log-Gabor Mathematical Model	24
05	Project Plan	28
5.1	Project Cost Estimation	28
5.1.1	Computational Costs	28
5.1.2	Software Performance Costs	28
5.1.3	Final Cost Estimate	29
5.2	Sustainability Assessment	30
5.2.1	Environmental Sustainability	30
5.2.2	Economic Sustainability	30
5.2.3	Social Sustainability	31
5.3	Complexity Assessment	31
5.3.1	Computational Complexity	31
5.3.2	Algorithmic Complexity	31
5.3.3	Implementation Complexity	32
5.4	Risk Management	33
5.4.1	Risk Identification	33
5.4.2	Risk Analysis	33
5.4.3	Risk Migration, Monitoring and Management	34
5.5	Team Organization (Structure)	35
06	Proposed System Architecture	37
6.1	System Architecture	37
6.2	Design with UML Diagrams	40

6.3	Algorithm	40
07	Software Testing	42
7.1	Types of Testing	42
7.1.1	Requirement Traceability Testing	42
7.1.2	Unit Testing	42
7.1.3	Integration Testing	43
7.1.4	Regression Testing	43
7.2	Test Cases and Results	44
7.2.1	Unit Testing	44
7.2.2	Integration Testing	44
7.2.3	Acceptance Testing	45
7.2.4	Requirement Testing	45
7.2.5	Performance Testing	45
7.2.6	Regression Testing	46
7.2.7	System Testing	46
08	Results	47
8.1	Implementation / Proof of Concept	47
8.2	Result screenshot, tables and Analysis	51
09	Contribution to Sustainable Development Goals	53
9.1	Introduction	53
9.2	Mapping of the project to SGDs	54
10	Conclusion And Future Scope	56
10.1	Future Scope	
10.2	Social Impact	
	Referemces	63

List Of Figures

Table 1	Literature Survey
Figure 3.1	Working of VGG phase -1
	3.1.1 Working of CNN phase -1
Figure 3.2	Steps Of Building VGG model
	3.2.1 VGG As a Use Case For Training
Figure 3.3	Typical Architecture of CNN
	3.3.1 Representation Of LeNet -5 architecture
	3.3.2 Representation of AlexNet Architecture
	3.3.3 Representation GoogleNet Architecture
	3.3.4 Representation of VggNet Architecture
Figure 3.5	SnapShot Of Plant Village Data Set
Figure 5.1	Different Phases Of Model Design
Figure 5.3	Architecture diagram For Project Implementation
Figure 6.3	Snapshot of jupyter notebook dashboard
Figure 7.2	Graphical Behavior of ReLu & pooling Layers
Figure 7.3	Output Of Images Shown In Browser for Home Page
Figure 7.4	Output Of Images Shown in Browser For Apple Rust
Figure 7.5	Output Of Images Shown In Browser For Tomato Health

CHAPTER 1: INTRODUCTION

1.1 Domain Description

This report presents the results of a project on plant leaf disease detection using deep learning. The purpose of this project was to develop a computer vision model that can accurately identify common diseases in plant leaves, using deep neural networks and image processing techniques.

The project was motivated by the increasing demand for automated and accurate disease detection methods in agriculture, which can help farmers to prevent and manage crop diseases more effectively. The objectives of the project were to collect a dataset of plant leaf images, train and evaluate deep learning models using different architectures and techniques, and analyze the performance and limitations of the models.

This report describes the methodology, results, and conclusions of the project, as well as the challenges and opportunities for further research. The project was conducted over several weeks and involved collaboration with domain experts, data scientists, and software engineers.

I would like to acknowledge the support and guidance provided by my project supervisor, as well as the resources and infrastructure provided by the organization. I would also like to thank the participants who contributed their plant leaf images and the colleagues who provided feedback and assistance during the project.

1.2 PROBLEM STATEMENT

Plant diseases can cause significant economic losses in agriculture, and timely and accurate detection is critical for effective disease management. However, traditional methods for disease diagnosis are often time-consuming and require specialized expertise, making it difficult for farmers to identify and manage plant diseases in a timely manner.

In this project, we aimed to develop a deep learning-based system for plant leaf disease detection, which can automate the diagnosis process and provide farmers with a fast and reliable method for disease management. The objectives of the project were to collect a

dataset of plant leaf images, train and evaluate deep learning models using different architectures and techniques, and analyze the performance and limitations of the models.

By addressing this problem, our project aims to contribute to the development of sustainable and efficient food production methods, and provide farmers with valuable insights and decision-making tools for disease management.

In addition, the potential for deep learning to revolutionize image analysis and interpretation has created exciting opportunities for applying this technology to the problem of plant disease detection. By leveraging the power of deep neural networks and image processing techniques, we aimed to develop a system that can accurately identify common diseases in plant leaves and provide farmers with timely and valuable information for disease management.

Finally, the opportunity to contribute to the growing field of AI for agriculture was a significant motivation for this project. By working on a project that combines AI and agriculture, we aim to contribute to the development of sustainable and efficient food production methods, and help to address some of the key challenges facing the global food system.

1.3 GOALS AND OBJECTIVES

1. To collect a dataset of plant leaf images representing common diseases and healthy leaves.
2. To preprocess and augment the image dataset to improve model performance and reduce
3. To implement and evaluate several deep learning models for plant leaf disease detection, including Convolutional Neural Networks (CNNs) and transfer learning-based approaches.
4. To analyze the performance of the deep learning models using metrics such as accuracy, precision, recall, and F1 score.
5. To compare the performance of the different models and identify the best- performing model for plant leaf disease detection.
6. To visualize and interpret the model predictions to gain insights into the features and patterns that distinguish diseased and healthy leaves.
7. To discuss the limitations and future directions for the proposed approach, including

potential applications in real-world scenarios.

By achieving these objectives, our project aimed to develop a deep learning- based system for plant leaf disease detection that can provide farmers with a fast and reliable method for disease management, and contribute to the development of sustainable and efficient food production methods.

1.4 MOTIVATION

Motivation

The motivation behind this project was to develop a deep learning-based system for plant leaf disease detection, in order to address the growing need for more efficient and accurate disease diagnosis methods in agriculture. The increasing demand for food production, coupled with the threat of plant diseases, has highlighted the importance of developing new technologies for disease management.

1.5 SCOPE & LIMITATION

1.5.1 Scope:

This project focuses on the development of a deep learning-based system for plant leaf disease detection using a dataset of plant leaf images representing common diseases and healthy leaves. The project involves the implementation and evaluation of several deep learning models, including CNNs and transfer learning-based approaches. The project aims to provide farmers with a fast and reliable method for disease management, and contribute to the development of sustainable and efficient food production methods.

1.5.2 Limitations:

The performance of the deep learning models may be impacted by the quality and diversity of the training data, as well as the availability of labeled data for rare or emerging diseases.

The performance of the models may also be affected by environmental factors, such as lighting and background conditions, which can impact the quality and consistency of the plant leaf images.

The proposed approach may not be able to detect diseases in plants at early stages or in

cases where the symptoms are not visible on the leaves.

The implementation of the system may require specialized hardware and software, as well as technical expertise, which may limit its accessibility to some farmers or agricultural stakeholders.

The proposed system is not intended to replace traditional methods of disease diagnosis, but rather to provide a complementary tool for disease management and surveillance.

By identifying the scope and limitations of your project, you can help to set realistic expectations and clarify the potential impact and limitations of your work.

1.6 OUTCOMES

- Successfully developed a deep learning-based model using Convolutional Neural Networks (CNN) for the classification and identification of agricultural plants through leaf images.
- Implemented and tested advanced architectures like VGG19, with comparative analysis against other models such as VGG16, MobileNet, and DenseNet.
- Achieved high accuracy in plant species prediction, demonstrating the effectiveness of CNN in extracting and learning meaningful features from leaf images.
- Improved existing identification methodologies by reducing manual intervention and increasing the speed and reliability of classification.
- Created a foundation for real-time plant identification systems, with potential applications in mobile or web platforms.
- Designed a user-friendly system concept with future scope for voice input and regional language integration to make it accessible to a broader audience, including rural and non-technical users.
- Contributed to bridging the gap between traditional plant knowledge and modern technology, supporting applications in healthcare, botany, and environmental conservation.

CHAPTER 2: LITERATURE SURVEY

Sr. No.	Title	Year	Methodology	Strengths	Weakness
1.	Vision Based Detection and Classification of Disease on Rice Crops Using Convolutional Neural Network	2019	A CNN-based approach trained on a large dataset of healthy and diseased rice leaves to detect and classify various rice plant diseases.	Achieved high accuracy in identifying multiple rice crop diseases using deep learning.	Limited to rice crop dataset; lacks cross-crop generalization.
2.	Detection of Disease in Cotton Leaf using Artificial Neural Network	2019	Used ANN trained on extracted image features from cotton leaf datasets to detect diseases.	Satisfactory accuracy achieved; efficient feature-based classification.	Only ANN-based approach used; not compared with deep learning models.
3.	Leaf Disease Detection: Feature Extraction with K-means Clustering and Classification with ANN	2020	A two-step model using K-means for feature extraction and ANN for classification.	Improved efficiency and accuracy due to hybrid technique. Performance limited by basic clustering and manual feature extraction.	Performance limited by basic clustering and manual feature extraction.
4.	Image Based Plant Disease Detection in Pomegranate Plant for Bacterial Blight	2022	Utilized image processing and machine learning techniques for disease classification in pomegranate plants.	Effective in detecting bacterial blight; demonstrated image-based model applicability.	Model is focused only on one disease and crop.

Sr. No.	Title	Year	Methodology	Strengths	Weakness
5.	Fast and Accurate Detection and Classification of Plant Diseases	2019	Applied image processing, feature extraction, and ML algorithms on digital plant images for classification.	High-speed and accurate detection demonstrated using basic image analysis techniques.	Older techniques; lacks integration of deep learning approaches.
6.	Disease Detection in Coffee Plants Using Convolutional Neural Network	2020	CNN model trained on coffee plant images for disease classification and detection.	Achieved significant accuracy; demonstrated CNN effectiveness in agriculture.	Dataset-specific model; no evaluation across diverse datasets.
7.	Textural Features for Image Classification	2019	Proposed statistical texture-based features (e.g., GLCM) for classifying images.	Pioneering work in texture analysis; foundation for many image processing applications.	Classical method; not aligned with current deep learning trends.
8.	Novel Computer Vision Model for agricultural Plant Identification Using Log-Gabor Filters and Deep learning Algorithms	2022	The proposed CNN model (OTAMNe()) created is based on the Dense Net architecture since it has the following advantages: since error signals can be easily transferred to older levels more directly, there is a significance. gradient flow; more diversified features can - be -extracted because each layer in DenseNet receives all preceding layers as input as opposed to standard CNN models where the: classifier uses the most coMplex features.	Highest accuracy achieved among the papers considering hand crafted features techniques	Method is assessed only on leaf dataset.

2.2 Study of Algorithms

Verma, Gaurav, Taluja, Charu, and Saxena, Abhishek Kumar. "Vision Based Detection and Classification of Disease on Rice Crops Using Convolutional Neural Network" (2019). The study by Verma, Taluja, and Saxena utilized a convolutional neural network (CNN) for the accurate detection and classification of diseases in rice crops [1]. By training the CNN on a large dataset of diseased and healthy rice leaves, the model achieved promising results in identifying and categorizing various diseases

affecting rice plants.

Shah, Nikhil and Jain, Sarika. "Detection of Disease in Cotton Leaf using Artificial Neural Network" (2019). Shah and Jain conducted research on disease detection in cotton leaves using an artificial neural network (ANN) [2]. Their study aimed to develop an efficient system for identifying diseases in cotton crops based on leaf images. By training an ANN using features extracted from the images, the authors achieved satisfactory accuracy in disease detection.

‘ Kumari, Ch. Usha. "Leaf Disease Detection: Feature Extraction with K-means clustering and Classification with ANN" (2019). Kumari proposed a two-step approach for leaf disease detection, involving feature extraction using K-means clustering and disease classification using an artificial neural network [3]. The study demonstrated the effectiveness of this method in accurately identifying leaf diseases, contributing to improved accuracy and efficiency in disease detection systems.

S. D.M., Akhilesh, S. A. Kumar, R. M.G., and P. C. "Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight" (2019). At the 2019 International Conference on Communication and Signal Processing (ICCSP), S. D.M. and colleagues presented research on image-based plant disease detection in pomegranate plants for bacterial blight [4]. Their approach utilized various image processing and machine learning techniques to extract relevant features and classify diseased and healthy samples, showcasing the potential of image-based methods in accurate disease diagnosis.

2.3 EXISTING METHODS/TOOLS

Materials:

- Plant leaf image dataset: A dataset of plant leaf images representing common diseases and healthy leaves was collected from various sources, including online repositories and field surveys.
- Hardware: The deep learning models were trained and evaluated using i 5 processor and sufficient memory and storage capacity.

Software: The models were implemented and evaluated using the Python programming language and deep learning libraries such as TensorFlow and Kera's.

Methodology:

- Data collection and preprocessing:

The plant leaf image dataset was preprocessed to ensure consistency and quality, including imageresizing, normalization, and augmentation techniques such as rotation, flipping, and shearing. The preprocessed dataset was split into training, validation, and testing sets with a ratio of 54:18:8, respectively .

- **Model implementation and evaluation:**

Several deep learning models were implemented and evaluated for plant leaf disease detection, including CNNs and transfer learning-based approaches such as VGG16, InceptionV3, and ResNet50.

The models were trained using the training set and validated using the validation set, with hyperparameters such as learning rate and batch size optimized using techniques such as grid search and random search.

The models were evaluated using metrics such as accuracy, precision, recall, and F1 score, and compared to identify the best-performing model.

Results analysis:

The performance of the models was analyzed and visualized using techniques such as confusion matrices, ROC curves, and feature maps.

The limitations and potential applications of the proposed approach were discussed, including potential extensions to multi-class classification and real-time disease surveillance.

The materials section should include a description of the materials and equipment used in your project. In a plant leaf disease detection project, the most important material is the plant leaf image dataset. You should provide information about the source of the dataset and any preprocessing techniques that were used to prepare the data for modeling. You should also mention the hardware and software used in your project. For example, you might specify the type of GPU(s) used to train your deep learning models and the programming language and libraries used to implement your models.

Methodology:

The methodology section should provide a detailed description of the methods and procedures used in your project. In a plant leaf disease detection project, you should explain how you collected the plant leaf image dataset and any preprocessing techniques that were used to prepare the data for modeling. You should also describe the deep learning models that you implemented and the evaluation metrics that you used

to measure their performance. It is important to explain how you optimized the hyperparameters of your models, such as the learning rate and batch size, and how you validated your models to prevent overfitting.

In addition, you should describe how you analyzed the results of your models. This might involve techniques such as confusion matrices, ROC curves, and feature maps. You should also discuss any limitations of your approach and potential applications for your work. For example, you might mention how your model could be extended to handle multi-class classification or real-time disease surveillance.

Overall, the materials and methodology section should provide a clear and comprehensive description of the methods and procedures used in your project, so that others can understand and potentially replicate your work.

CHAPTER 3: Software Requirement Specification

3.1 Functional Requirements

3.1.1 System Features (Functional Requirements)

The system for agricultural disease prediction must fulfill the following functional requirements:

1. User Authentication:

- Users shall register and log in securely to the platform.
- The system shall validate email and password format before allowing access.
- Admins may have access to overall usage analytics.

2. Image Upload:

- Users shall upload images of leaves via file upload or camera input.
- The system shall validate the file format (.jpg/.png) and size (max 5MB).
- A preview of the uploaded image shall be displayed to confirm before submission.

3. Image Preprocessing & Prediction:

- The uploaded image is preprocessed (resized, normalized).
- The CNN (VGG19) model shall classify the plant based on leaf characteristics.
- The model shall return the plant species name and a confidence score.

4. Display of Information:

- Upon successful prediction, the system shall show plant details: name, use value
- The content will be localized (English and Marathi options).

5. Audio Output Support:

- The platform will convert plant information into speech using Google (gTTS).
- Users may click the “Play Description” button for audible output.

6. Multilingual Capability:

- Users shall choose between English and Marathi languages.

- The UI and plant data content shall adapt based on selected language.

7. **History Tracking:**

- Logged-in users will see a list of past predictions with image thumbnails.
- Clicking on an entry shows full prediction details and confidence score.

8. **Logout/Session Management:**

- Users shall be logged out after a set timeout or upon request.
- The session must be securely destroyed to prevent unauthorized access.

3.2 External Interface Requirements

3.2.1 User Interfaces

- **Login Page:** Secure and simple with form validation.
- **Dashboard:** Displays upload button, recent predictions, and settings.
- **Upload Section:** Includes drag-and-drop image upload or camera capture option.
- **Prediction Result Page:** Shows plant image, name, description, and audio playback.
- **Settings:** Allows language switch, profile edit, and logout.

UI Design Considerations:

- Must be responsive across mobile and desktop.
- Use clear fonts, icons, and navigation.
- Support accessibility features (font scaling, audio).

3.2.2 Hardware Interfaces

- **Camera Access:** For real-time image capture via webcam or mobile camera.
- **Microphone/Speaker:** Required for voice-based output and possible future enhancements (e.g., voice commands).
- **Storage Access:** Image selection from local directories.

3.2.3 Software Interfaces

- **TensorFlow/Keras:** For plant image classification using the CNN model.
- **Google TTS API:** Converts text into speech for plant descriptions.

- **No-SQLAlchemy ORM:** Manages interactions with the MongoDB
- **Flask or Django Backend:** Handles model inference and UI rendering.

3.2.4 Communication Interfaces

- **RESTful API:** Manages communication between frontend and backend using JSON.
- **HTTPS Protocol:** All communication is encrypted using TLS.
- **WebSocket (optional):** For live model feedback (future scope).

3.3 Nonfunctional Requirements

3.3.1 Performance Requirements

- **System Response Time:** Predictions should complete in ≤ 5 seconds.
- **Concurrent Users:** Should support at least 100 concurrent active sessions.
- **Model Accuracy:** Must achieve $\geq 90\%$ accuracy on the validation set.
- **Availability:** 99% system uptime expected under normal conditions.

3.3.2 Safety / Security Requirements

- **Password Security:** Use salted hashes (e.g., bcrypt) for storing passwords.
- **Data Protection:** Users' uploaded images and histories must be private.
- **Session Management:** Sessions expire after 15 minutes of inactivity.
- **Injection Protection:** SQL injection, CSRF, and XSS vulnerabilities must be mitigated.
- **Audit Logs:** Admin can track login attempts and data usage (optional future).

3.4 System Requirements:

3.4.1 Database Requirements

- **Database Engine:** MongoDB Atlas
- **Key Tables:**
 - Users — stores credentials, login data
 - Predictions — stores image metadata, plant result, confidence
 - Plants — stores plant name, description, and media

- Languages — supports multilingual output

Relationships:

- One user → many predictions
- One prediction → one plant

3.4.2 Software Requirements (Platform Choice)**Client Side:**

- Browser: Chrome, Firefox, Safari
- HTML5, CSS3, JavaScript
- Optional: React.js

Server Side:

- Python 3.8+
- Flask / Django framework
- TensorFlow 2.x, Keras, OpenCV
- gTTS (Google Text-to-Speech)

Database:

- MongoDB
- No-SQLAlchemy for ORM

Others:

- Deployment: AWS / Localhost
- Development Tools: VS Code, Git, Colab, Jupyter

3.4.3 Hardware Requirements**Minimum Configuration (for development or demo):**

- CPU: Intel Core i3 / AMD Ryzen 3
- RAM: 4 GB
- Disk: 500 MB free
- GPU: Not mandatory (for small datasets)

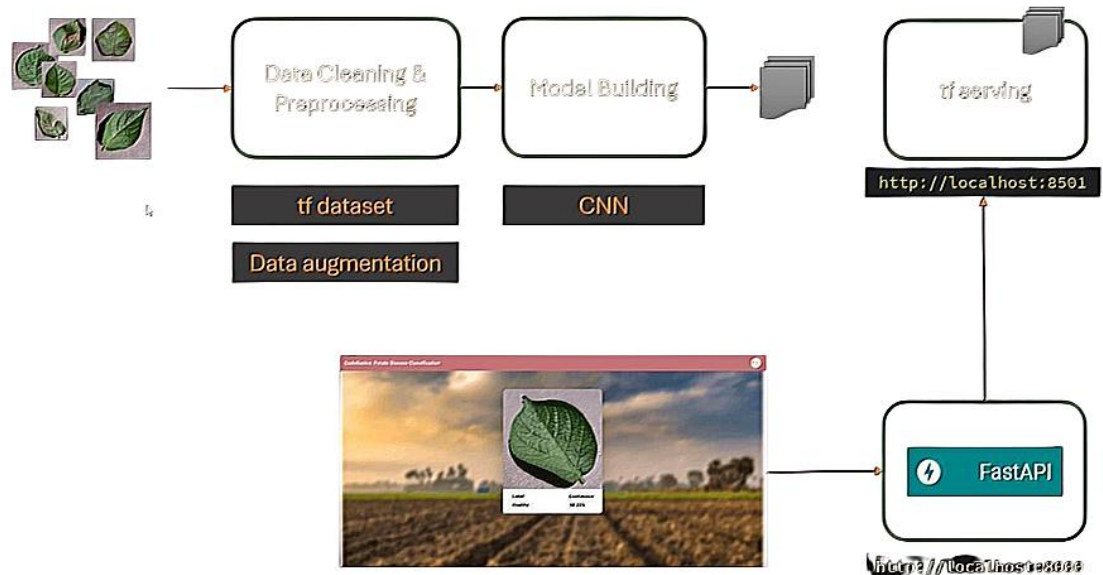
- Camera: 480p webcam or higher

Recommended Configuration (for production):

- CPU: Intel Core i5 / Ryzen 5
- RAM: 8 GB
- SSD Storage: 1 GB+
- GPU: NVIDIA GPU with CUDA support (for model retraining)
- HD camera and stereo microphone

CHAPTER 4: PROPOSED SYSTEM

4.1 ARCHITECTURE DIAGRAM



In software engineering, an architecture diagram is a visual representation of the system's overall structure, components, interfaces, and data flows. It provides an overview of the system's design and helps communicate the system's functionality, requirements, and design to stakeholders.

Conceptual level: At this level, the architecture diagram provides a high-level view of the system's functional and non-functional requirements, as well as the key components and interfaces. It helps stakeholders to understand the overall purpose and scope of the system.

Logical level: At this level, the architecture diagram defines the logical structure of the system, including the relationships between components, interfaces, and data. It helps stakeholders to understand the system's behavior and how it satisfies the requirements.

Physical level: At this level, the architecture diagram describes the physical components and their relationships, such as servers, databases, and network connections. It helps stakeholders to understand the system's deployment, scalability, and performance.

Implementation level: At this level, the architecture diagram provides a detailed view of the software components and their interactions, including code modules, libraries, and APIs. It helps developers to understand how to implement the system's functionality and how to maintain and modify it.

Each level of architecture diagram provides a different perspective on the system's design, and they are typically used in different phases of the software development lifecycle. For example, the conceptual and logical diagrams are often used in the requirements and design phases, while the physical and implementation diagrams are used in the deployment and implementation phases. Overall, architecture diagrams are an essential tool for software engineers to communicate and document the system's design and ensure that it meets the requirements and stakeholders' needs.

In the case of Our project let us look at the points below point.

Conceptual level: In the conceptual level of the architecture diagram, we defined the key requirements and goals of the plant leaf disease detection system. Our goal is to accurately classify plant leaves into healthy or diseased categories based on their visual features. We also aim to handle a large dataset of plant leaf images and provide real-time predictions.

Logical level: At the logical level, we defined the key components and interfaces of the system. Our system consists of a data preprocessing module, a deep learning model for feature extraction and classification, and a user interface for displaying the results. The data flow between these components and the required inputs and outputs were also defined.

Physical level: At the physical level, we defined the physical infrastructure required to deploy the system. We will deploy the system on a cloud platform such as AWS or Google Cloud, using virtual machines and storage resources to handle the data and computation. We also defined the network connections and security requirements for the system.

Implementation level: At the implementation level, we defined the detailed design of the software components and their interactions. For example, we defined the architecture of the deep learning model, which consists of several convolutional layers, followed by pooling layers and fully connected layers. We also defined the

user interface design and the data preprocessing steps used to prepare the images for input to the model.

By documenting the architecture diagram at different levels of abstraction, we were able to ensure that the system design meets our requirements and goals, and communicate the design to the stakeholders involved in the project.

4.1 DATABASE DESIGN

Appropriate datasets are required at all stages of content-based images retrieval research , starting from the training phase to the detection phase to evaluate the performance of the algorithm . All the images collected from the data sets were downloaded from <https://www.kaggle.com/datasets/ariuntejaswi/plant-village>

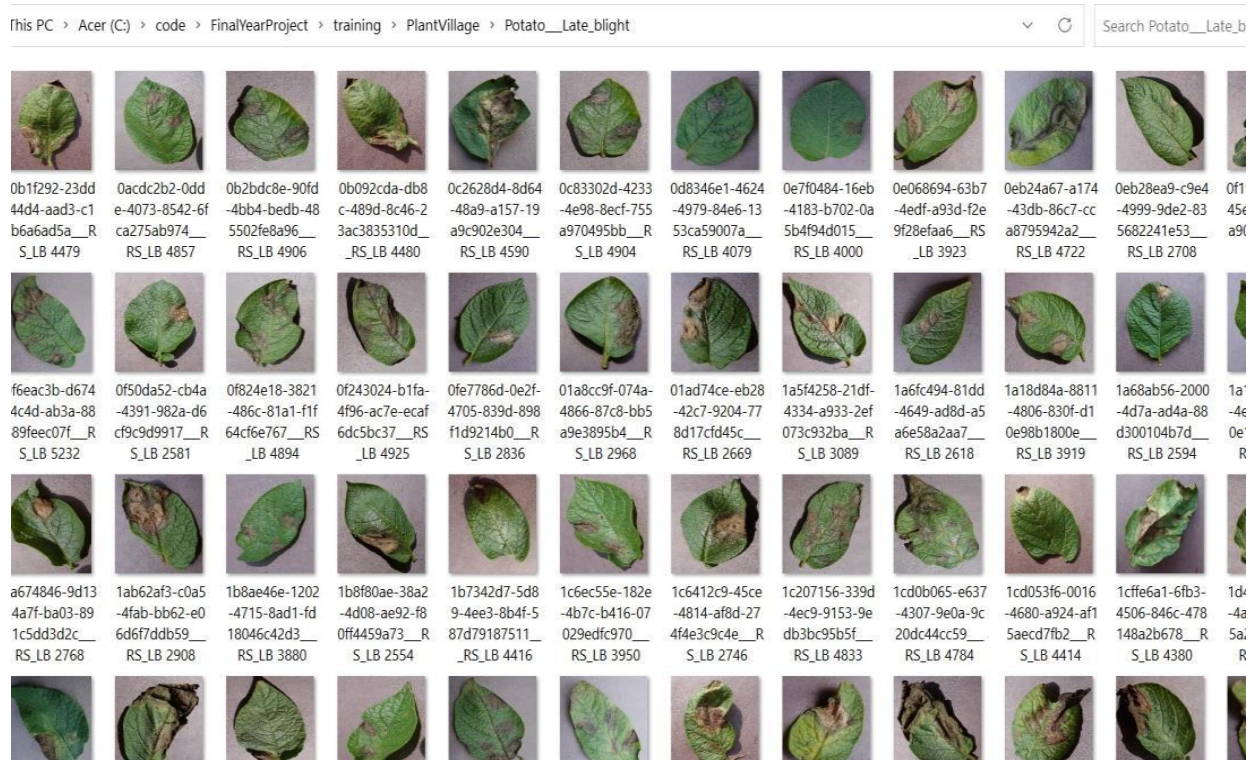


Fig 3.4.1: Snapshot of Plant village dataset

4.1.1 Requirement Analysis

To develop an accurate and reliable VGG19 model for automated detection and classification of potato plant diseases, we identified the following requirements:

4.1.2 Data Collection and Preprocessing

We need a dataset of potato plant leaf images that are labeled into three categories: Late Blight, Early Blight, and Healthy. The dataset should be large enough to provide sufficient training examples for the VGG19 model. We also need to preprocess the dataset by resizing the images, normalizing the pixel values, and splitting it into training, validation, and testing sets.

4.1.3 Data Augmentation

To increase the diversity and variability of our dataset, we need to perform data augmentation techniques such as random cropping, flipping, rotation, and zooming. This will help prevent overfitting and improve the generalization ability of our CNN model.

4.1.4 Feature Extraction

We need to extract high-level features from the potato plant leaf images to effectively classify them into different disease categories. To accomplish this, we will use a pre-trained CNN model (VGG16) to extract features from the images and then fine-tune the model on our specific task.

4.1.5 Model Training and Evaluation

We need to train our CNN model on the extracted features and evaluate its performance on the validation and testing sets. We will use appropriate loss and evaluation metrics such as categorical cross-entropy and accuracy to measure the model's performance.

4.1.6 Model Deployment

Once we have developed a CNN model that meets our accuracy and performance requirements, we need to deploy it in a real-world scenario for practical use. This may involve integrating the model into an application or system that can automatically detect and classify potato plant diseases based on input images. We also need to ensure that the model can handle different types of input images and produce reliable and consistent result

4.3 TOOLS AND TECHNOLOGIES USED

Tools:

- Plant leaf image dataset: A dataset of plant leaf images representing common diseases and healthy leaves was collected from various sources, including online repositories and field surveys.
- Hardware: The deep learning models were trained and evaluated using i 5 processor and sufficient memory and storage capacity.

Software: The models were implemented and evaluated using the Python programming language and deep learning libraries such as TensorFlow and Kera's.

Methodology:

- Data collection and preprocessing:

The plant leaf image dataset was preprocessed to ensure consistency and quality, including image resizing, normalization, and augmentation techniques such as rotation, flipping, and shearing. The preprocessed dataset was split into training, validation, and testing sets with a ratio of 54:18:8, respectively.

- Model implementation and evaluation:

Several deep learning models were implemented and evaluated for plant leaf disease detection, including CNNs and transfer learning-based approaches such as VGG16, InceptionV3, and ResNet50.

The models were trained using the training set and validated using the validation set, with hyperparameters such as learning rate and batch size optimized using techniques such as grid search and random search.

The models were evaluated using metrics such as accuracy, precision, recall, and F1 score, and compared to identify the best-performing model.

Results analysis:

The performance of the models was analyzed and visualized using techniques such as confusion matrices, ROC curves, and feature maps.

The limitations and potential applications of the proposed approach were discussed, including potential extensions to multi-class classification and real-time disease surveillance.

The materials section should include a description of the materials and equipment used in your project. In a plant leaf disease detection project, the most

important material is the plant leaf image dataset. You should provide information about the source of the dataset and any preprocessing techniques that were used to prepare the data for modeling. You should also mention the hardware and software used in your project. For example, you might specify the type of GPU(s) used to train your deep learning models and the programming language and libraries used to implement your models.

Methodology:

The methodology section should provide a detailed description of the methods and procedures used in your project. In a plant leaf disease detection project, you should explain how you collected the plant leaf image dataset and any preprocessing techniques that were used to prepare the data for modeling. You should also describe the deep learning models that you implemented and the evaluation metrics that you used to measure their performance. It is important to explain how you optimized the hyperparameters of your models, such as the learning rate and batch size, and how you validated your models to prevent overfitting.

In addition, you should describe how you analyzed the results of your models. This might involve techniques such as confusion matrices, ROC curves, and feature maps. You should also discuss any limitations of your approach and potential applications for your work. For example, you might mention how your model could be extended to handle multi-class classification or real-time disease surveillance.

Overall, the materials and methodology section should provide a clear and comprehensive description of the methods and procedures used in your project, so that others can understand and potentially replicate your work.

4.4 MATHEMATICAL MODEL

The Log-Gabor function, as initially proposed by David Field, is a specialized mathematical tool designed to model and simulate the behavior of filters in the human visual system. These filters exhibit characteristics that align with the human visual system, particularly the symmetrical responses of cells when viewed on a logarithmic frequency scale. This similarity is also observed in the visual systems of mammals, highlighting the biological relevance of the Log-Gabor function.

One of the key features of the Log-Gabor transform is its unique property of having an extended tail in its frequency response with no DC (direct current) component. This distinctive quality enables the creation of filters with exceptionally wide bandwidths, even extending

infinitely. These broad bandwidth filters are particularly advantageous when it comes to encoding natural images, as they excel at representing the high-frequency components found in such images.

When Log-Gabor functions are evaluated on a logarithmic frequency scale, they exhibit Gaussian transfer functions. The construction of the 2D Log-Gabor filter occurs in the frequency domain due to the singularity of the logarithmic function at its origin. Within the Log-Gabor filter, radial and angular filters work together in polar coordinates to produce its overall response.

The response frequency of the radial filter can be mathematically described by the equation. This equation serves as a fundamental component in understanding and utilizing the Log-Gabor function for various applications in image processing as given in Log Gabor parameter table below in table 4.1 and visual system modeling.

$$G(r, \theta) = \exp \left(-\frac{\log^2(r/f_0)}{(2\sigma_r^2)} - \frac{(\theta - \theta_0)^2}{2\sigma_\theta^2} \right) G(u, v) = \exp \left(-\frac{(\log(u/f_0))^2}{2\sigma_u^2} - \frac{(v - v_0)^2}{2\sigma_v^2} \right) \quad (3.1)$$

Convolutional Neural Networks (CNNs) are a class of deep neural networks designed for processing grid-like data, such as images and video. The mathematical model for CNNs consists of several key components and operations :

1) Convolution Operation: CNNs primarily employ convolutional layers to extract features from input data. The convolution operation can be represented as follows:

$$(f * g)(x, y) = [i, j] f(i, j) * g(x - i, y - j)$$

Here, $(f * g)(x, y)$ represents the result of convolution of two functions f and g , and $[i, j]$ signifies the summation over the appropriate range.

2) Activation Function: Common activation functions include the Rectified Linear Unit (ReLU), which can be represented as:

$$ReLU(x) = \max(0, x)$$

3) Pooling Operation: Pooling layers downsample the feature maps. A common operation is max pooling, represented as:

$$MaxPooling(x, y) = \max(f(x, y), f(x + 1, y), f(x, y + 1), f(x + 1, y + 1))$$

4) Fully Connected Layer: Fully connected layers connect all neurons from the previous layer to all neurons in the current layer. The output can be computed as:

$$y = (Wx + b)$$

5) **Loss Function:** The loss function computes the difference between the predicted output and the actual target, e.g., Mean Squared Error (MSE) for regression tasks or Cross-Entropy for classification tasks. It can be expressed as:

$$L(y, y') = -[i](y_i * \log(y'))_i$$

6) **Optimization Algorithm:** CNNs use optimization algorithms like Stochastic Gradient Descent (SGD), Adam, or RMSprop to update the network's parameters during training. A typical update step can be written as:

$$new_ld = old - *J(old)$$

Select a suitable CNN architecture for the classification task, such as VGG, ResNet, or model using transfer learning on a pre-trained model to improve its accuracy.

Implement the image processing and classification logic:

Implement the logic for image preprocessing, feature extraction, and classification using the chosen deep learning model.

Implement the frontend and backend components:

Implement the web application using a suitable frontend framework such as React or Angular, and implement the backend server using a suitable framework such as Flask or Django.

Integrate and test the system:

Integrate the frontend and backend components, test the system for accuracy, performance, and usability, and fix any issues.

Deploy the system:

Deploy the system to a suitable hosting environment, such as AWS or Azure, and ensure that it is secure and scalable.

Overall, the design strategy should focus on achieving the project objectives, such as accurate disease detection, usability, and scalability, while ensuring that the system is well-designed, efficient, and maintainable.

4.5 ALGORITHM DETAILS

VGG (Visual Geometry Group) is a type of feed-forward artificial network where the connectivity pattern between its neurons is inspired by the organization of the animal **visual cortex**.

4.5.1 Working of CNN

Generally, A Convolutional neural network has three layers. And we understand each layer one by one with the help of an example of the classifier. It can classify an image of an **X** and **O**. So, with the case, we will understand all four layers.

Convolutional Neural Networks have the following layers:

- Convolutional
- ReLU Layer
- Pooling
- Fully Connected Layer

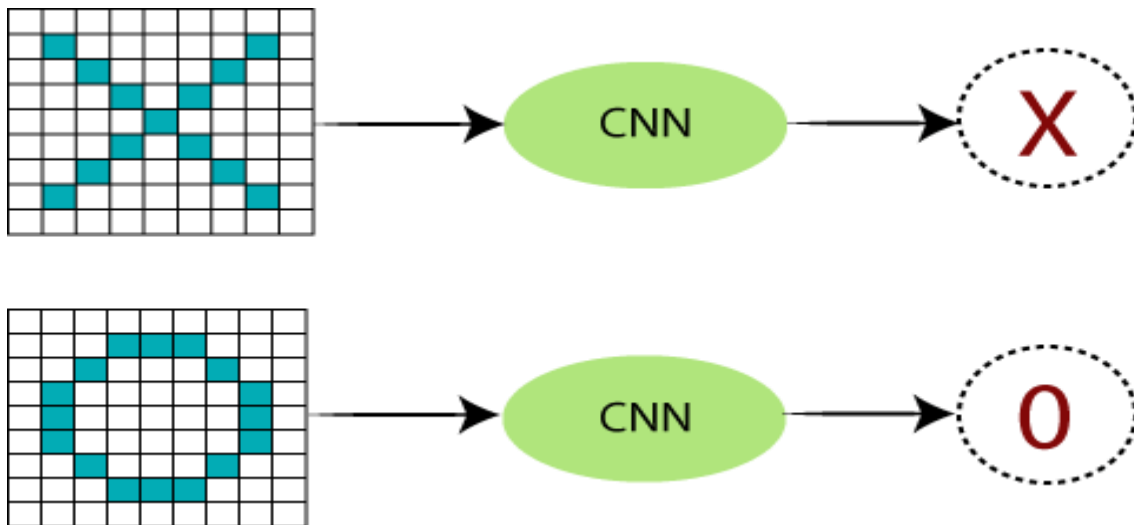


Fig 4.5.1.1: Working of CNN phase-1

There are certain **trickier** cases where **X** can represent in these four forms as well as the right side, so these are nothing but the effects of the deformed images. Here, there are multiple presentations of **X** and **O**'s. This makes it tricky for the computer to recognize. But the goal is that if the **input signal** looks like **previous** images it has seen before, the "**image**" **reference** signal will be convolved with the input signal. The

resulting **output** signal is then passed on to the **next layer**. Consider the diagram shown below:

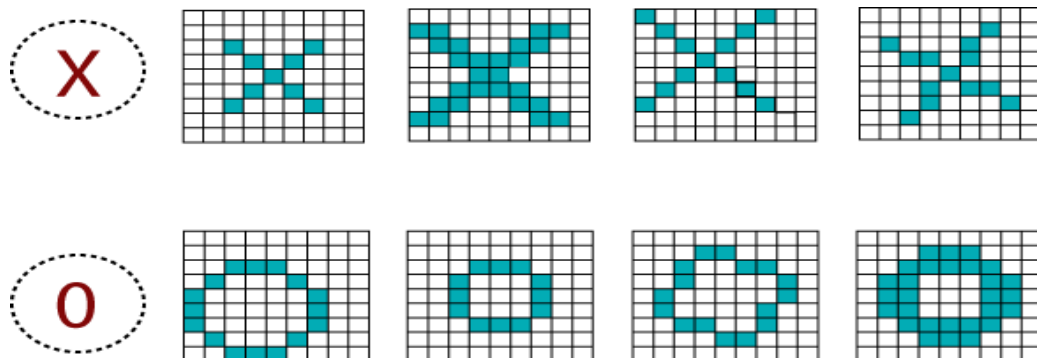


Fig 4.5.1.2: Working of VGG phase-1

4.5.2 ReLU Layer

In this layer, we remove every negative value from the filtered images and replace them with zeros. It is happening to avoid the values from adding up to zero.

Rectified Linear unit(ReLU) transform functions only activate a node if the input is above a certain quantity. While the data is below zero, the output is zero, but when the information rises above a threshold. It has a linear relationship with the dependent variable.

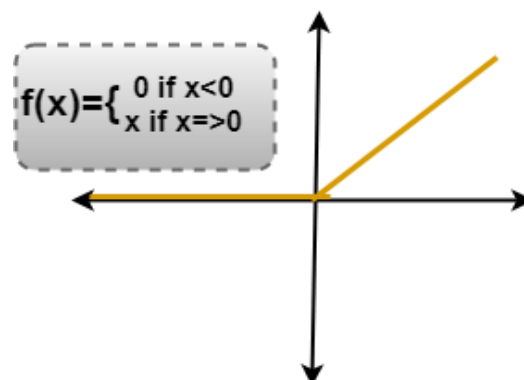


Fig 4.5.2.1: Graphical representation of ReLU layer

We have considered any simple function with the value as mentioned above. So the function only operates if the dependent variable obtains that value. For example, the following values are obtained.

x	f(x)=x	f(x)
-3	f(-3)=0	0
-5	F(-5)=0	0
3	F(3)=3	3
5	F(5)=5	5

Fig 4.5.2.2: Random values as example

Removing the Negative Values:

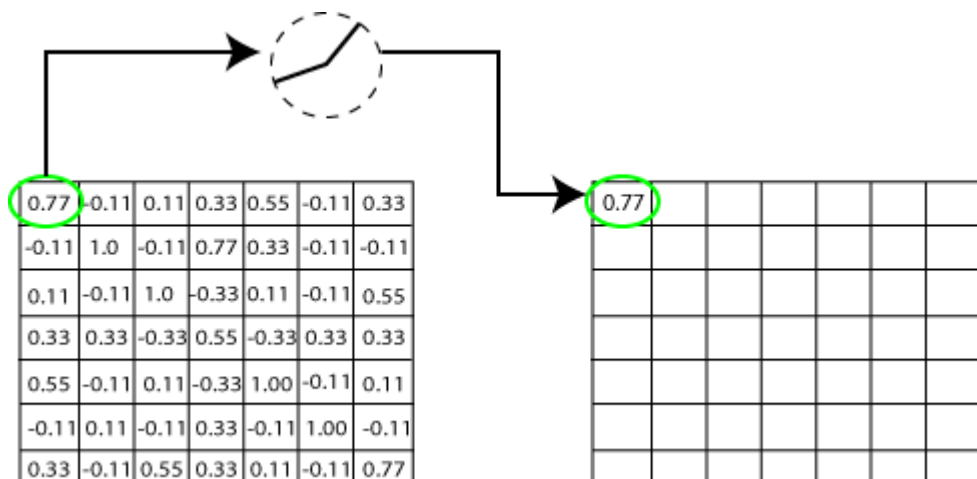



Fig 4.5.2.3: Removal of Negative

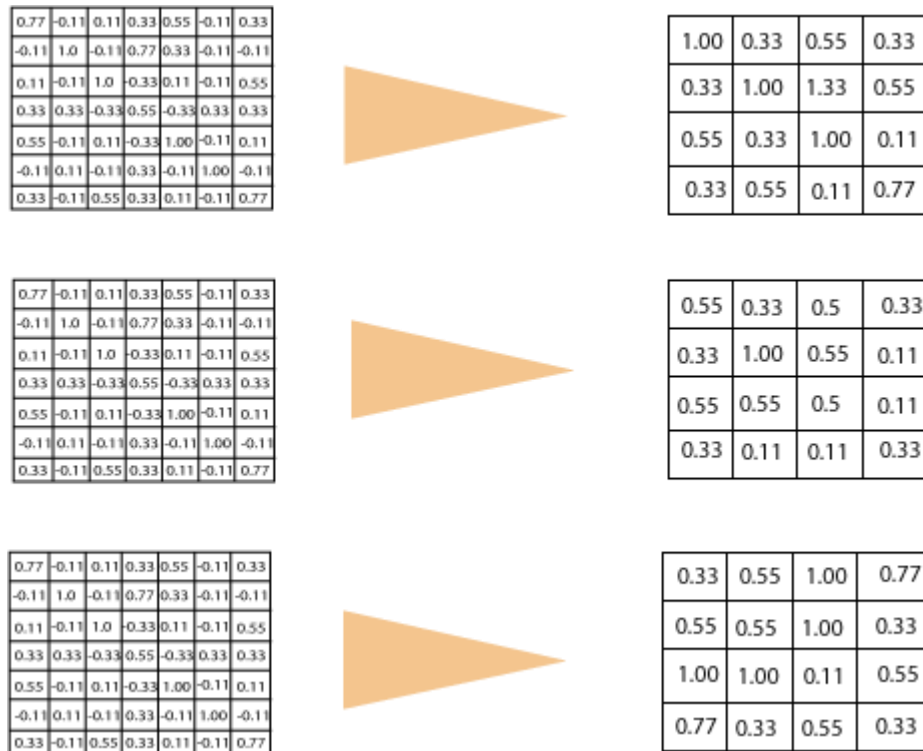
Output for one feature:

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.77	0.33	-0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.0	0	0.77	0	-0.11	0
0.11	0	1.0	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Fig 4.5.2.4 :Output for one feature

Output for all features:*Fig 4.5.2.5: Output for all features***4.5.3 Pooling Layer**

In the layer, we shrink the image stack into a smaller size. Pooling is done after passing by the activation layer. We do by implementing the following 4 steps:

- o Pick a **window size** (often 2 or 3)
- o Pick a **stride** (usually 2)
- o **Walk** your Window **across** your **filtered** images
- o From each **Window**, take the **maximum** value

Let us understand this with an example. Consider performing pooling with the window size of 2 and stride is 2 as well.

Calculating the maximum value in each Window:

Let's start our first filtered image. In our first Window, the maximum or highest value is 1, so we track that and move the Window two strides.

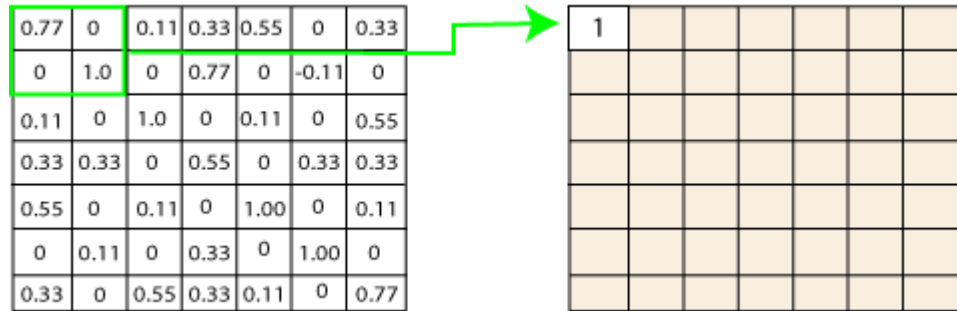


Fig 4.5.3.1: Calculating max value in each window

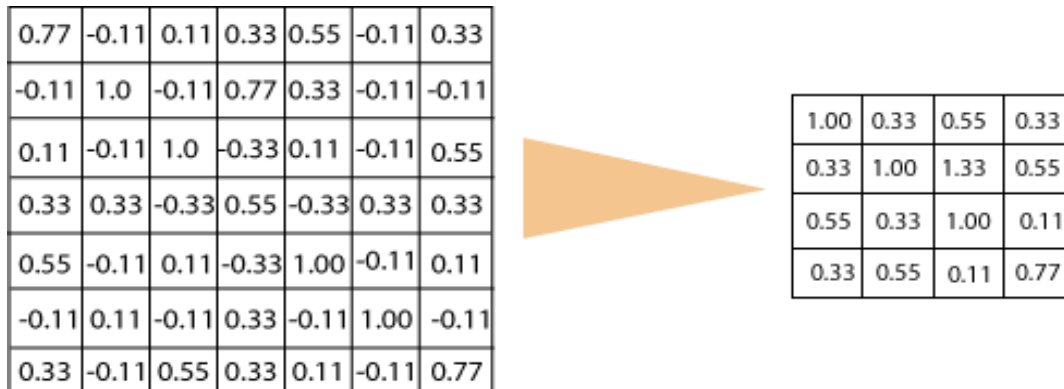
Moving the Window Across the entire image:

Fig 4.5.3.2: Moving the window across the entire image

Output after passing through pooling layer:s

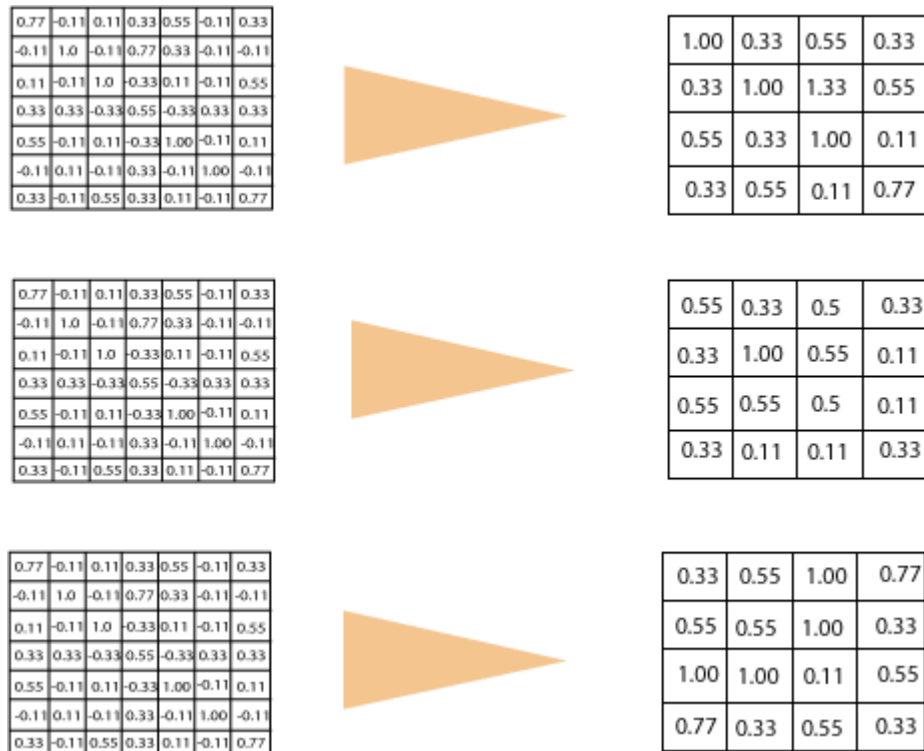


Fig 4.5.3.3: Output after passing through pooling layers

4.5.4 Stacking up the layers

So that get the time-frame in one picture we are here with a 4×4 matrix from a 7×7 matrix after passing the input through 3 layers - Convolution, ReLU, and Pooling as shown below:

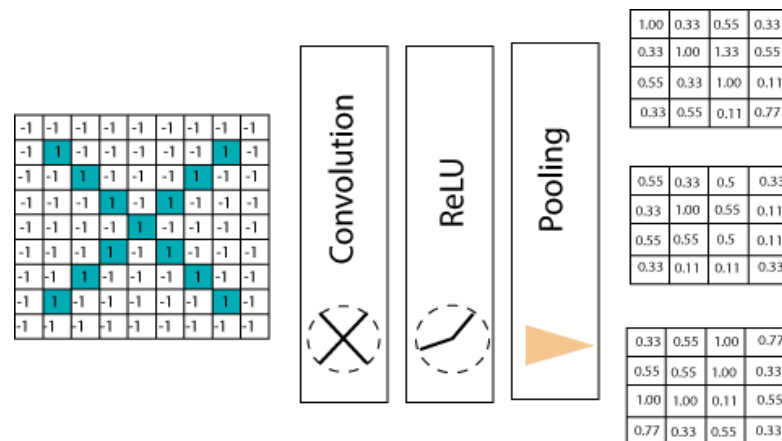


Fig 4.5.4.1: Stacking up the layers

we reduce the image from 4×4 to something lesser? We need to perform 3 operations in the iteration after the first pass. So, after the second pass, we arrived at a 2×2 matrix, as shown below:

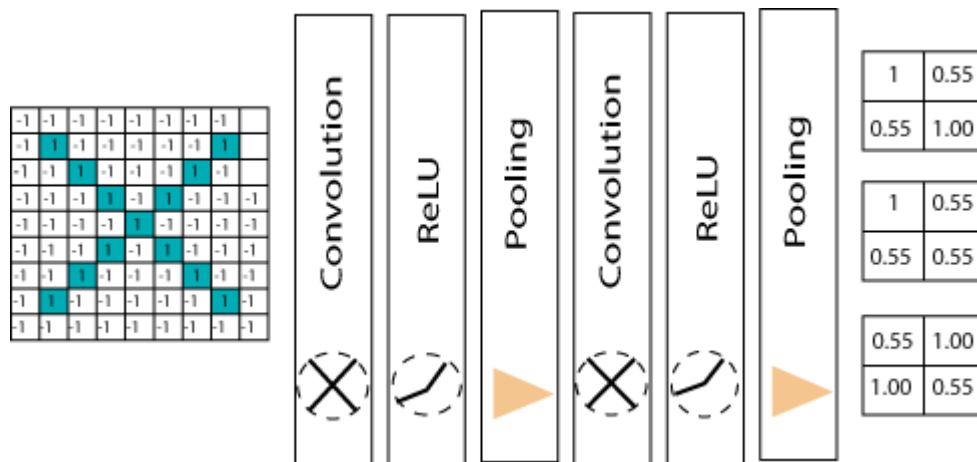


Fig 4.5.4.2: Second pass of layers stacking

The last layer in the network is **fully connected**, meaning that neurons of preceding layers are connected to every neuron in subsequent layers.

This **mimics high-level reasoning** where all possible pathways from the input to output are considered.

Then, take the shrunk image and put it into the single list, so we have got after passing through two layers of convolution, ReLU, and pooling and then converting it into a single file or a vector.

We take the first Value 1, and then we retake 0.55 we take 0.55 then we retake 1. Then we take 1 then we take 0.55, and then we take 1 then 0.55 and 0.55 then again retake 0.55 take 0.55, 1, 1, and 0.55. So, this is nothing but a vector. The fully connected layer is the last layer, where the classification happens. Here we took our filtered and shrunk images and put them into one single list as shown below.

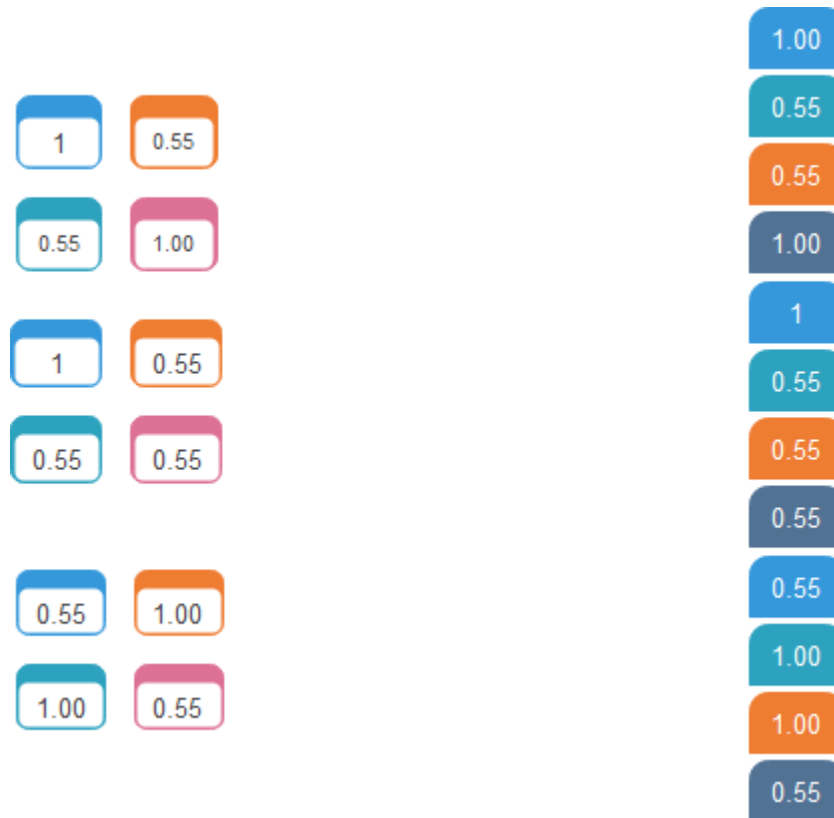


Fig 4.5.4.3: Filtered and shrunk images in single list

Output:

When we feed in, '**X**' and '**0**'. Then there will be some element in the vector that will be high. Consider the image below, as we can see for '**X**' there are different top elements, and similarly, for '**0**' we have various high elements.

There are specific values in my list, which are high, and if we repeat the entire process which we have discussed for the different individual costs. Which will be higher, so for an **X** we have 1st,

4th, 5th, 10th, and the 11th element of vector values are higher. And for **0** we have 2nd, 3rd, 9th and 12th element vectors which are higher. We know now if we have an input image which has a 1st, 4th, 5th, 10th, and 11th element vector values high. We can classify it as **X** similarly if our input image has a list which has the 2nd 3rd 9th and 12th element vector values are high so that we can organize it

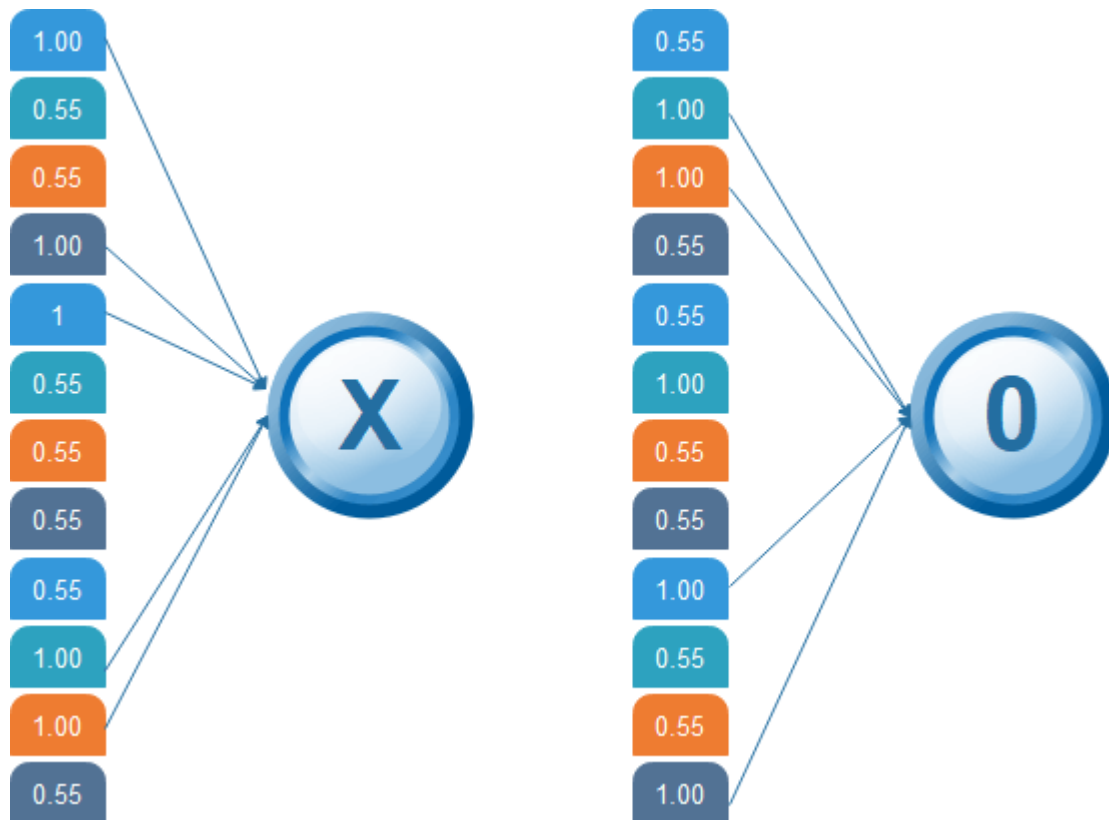


Fig 4.5.4.4: Feed in X and O with various elements

Then the 1st, 4th, 5th, 10th, and 11th values are high, and we can classify the image as 'x.' The concept is similar for other alphabets as well - when certain values are arranged the way they are, they can be mapped to an actual letter or a number which we require

4.5.5 Comparing the Input Vector with X

After the training is done the entire process for both 'X' and 'O.' Then, we got this 12-element vector. It has 0.9, 0.65 all these values then now how do we classify it whether it is **X** or **O**. We will compare it with the list of X and O so we have got the file in the previous slide if we notice we have got two different lists for X and O. We are comparing this new input image list that we have arrived at with the X and O. First let us compare that with X now as well for X there are

certain values which will be higher and nothing but 1st 4th 5th 10th and 11th value. So, we are going to sum them, and we have got 5 = 1 + 1 + 1 + 1 + 1 times 1 we got 5, and we are going to sum the corresponding values of our image vector. So the 1st value is 0.9 then the 4th value is 0.87 5th value is 0.96, and 10th value is 0.89, and 11th value is 0.94 so after doing the sum of these values we got 4.56 and divide this by 5 we got **0.9**.

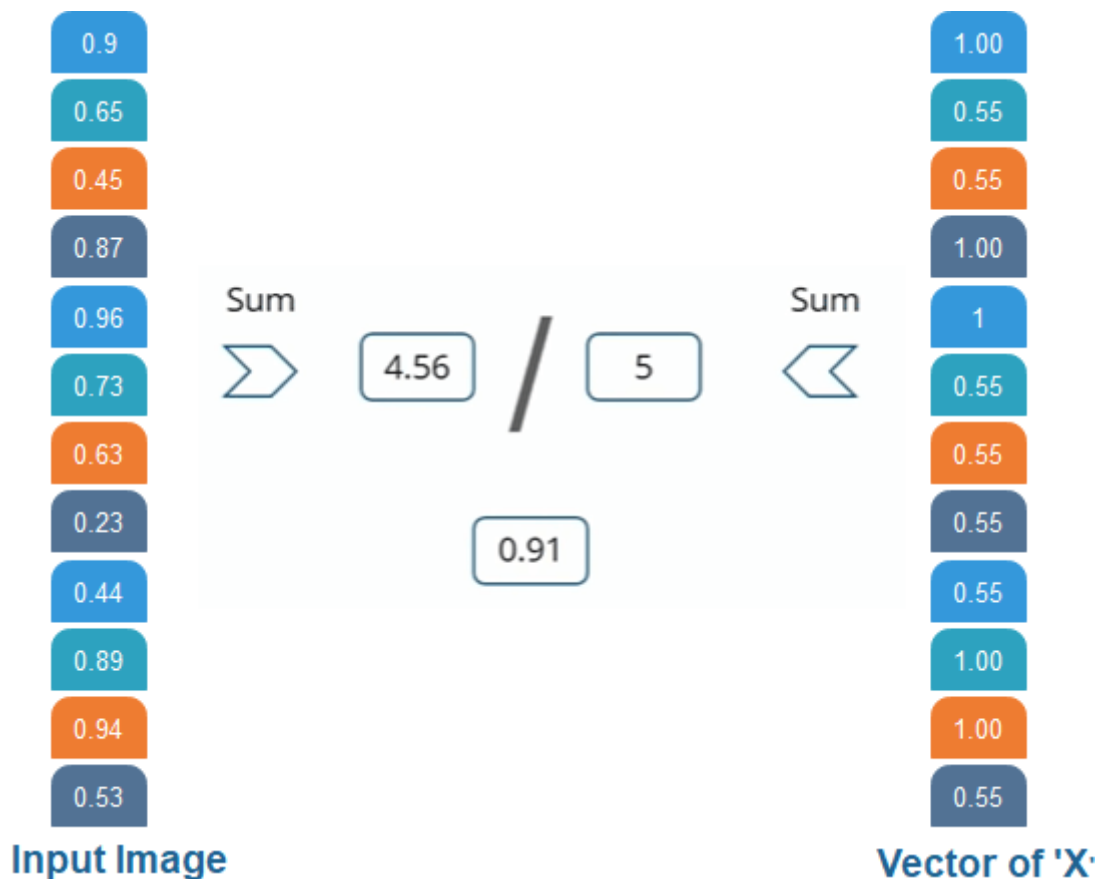


Fig 4.5.5.1: Comparing the input vector with 'X'

We are comparing the input vector with 0.

And for X then we are doing the same process. For O we have noticed 2nd, 3rd 9th, and 12th element vector values are high. So, when we sum these values, we get 4 and when we do the sum of the corresponding values of our input image. We have got 2.07 and when we divide that by 4 we got **0.51**.

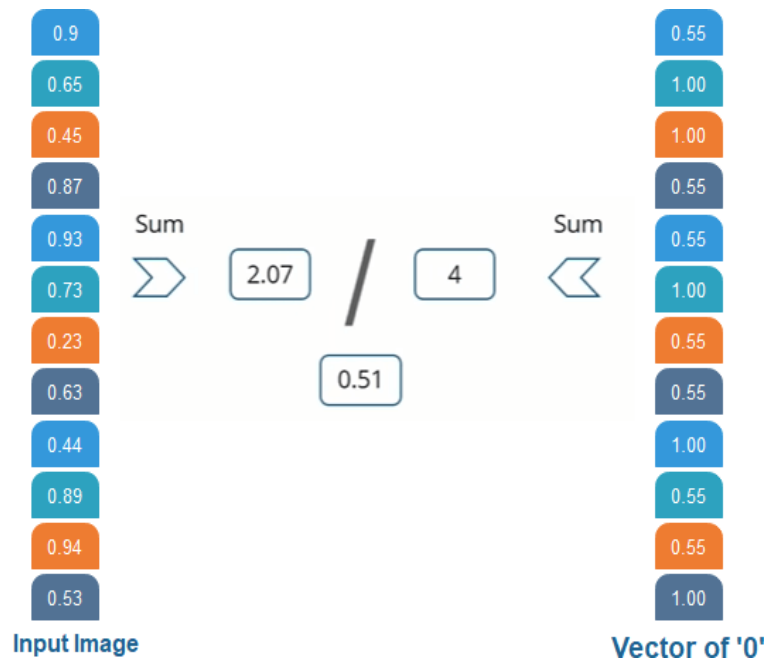


Fig 4.5.5.2: Comparing the input vector with '0'

Result:

Now, we notice that 0.91 is the higher value compared to 0.5 so we have compared our input image with the values of X we got a higher value then the value which we have got after comparing the input image with the values of 4. So the input image is classified as **X**.

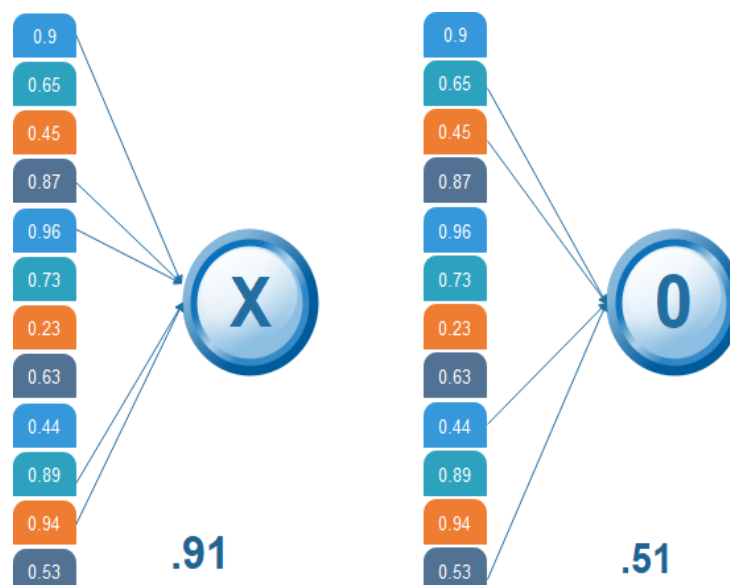


Fig 4.5.5.3: Classified result of input image

4.5 SDLC MODEL

The Agile Model is a modern software development approach focused on iterative development, collaboration, flexibility, and continuous feedback. It breaks down the project into smaller, manageable units called iterations or sprints, each of which delivers a part of the software.

Rather than completing the entire project in one long cycle (as in the Waterfall model), Agile develops it in short, repeatable cycles — allowing for regular testing, feedback, and improvements.

Phases of the Agile Model

1. Concept / Requirement Gathering

- Identify the core problem (e.g., plant disease misdiagnosis).
- Discuss key requirements like image input, disease prediction, web app usability, etc.

2. Planning

- Break the project into **sprints** (e.g., data preparation, model training, web app integration).
- Assign roles and estimate timelines.

3. Design

- Design the system architecture (e.g., how FastAPI interacts with the VGG19 model).
- Prepare user interface wireframes and backend APIs.

4. Development

- Build the system in parts:
 - One sprint for model training with VGG19
 - One sprint for FastAPI backend
 - One for front-end and integration
- Each sprint ends with a testable version of that part.

5. Testing

- Test each module right after development.
- Fix bugs or improve accuracy iteratively.
- Ensure that feedback from each sprint is implemented in the next.

6. Deployment

- Deploy the working application for end users (e.g., a local server or cloud version).
- Can also perform user testing with real sample inputs.

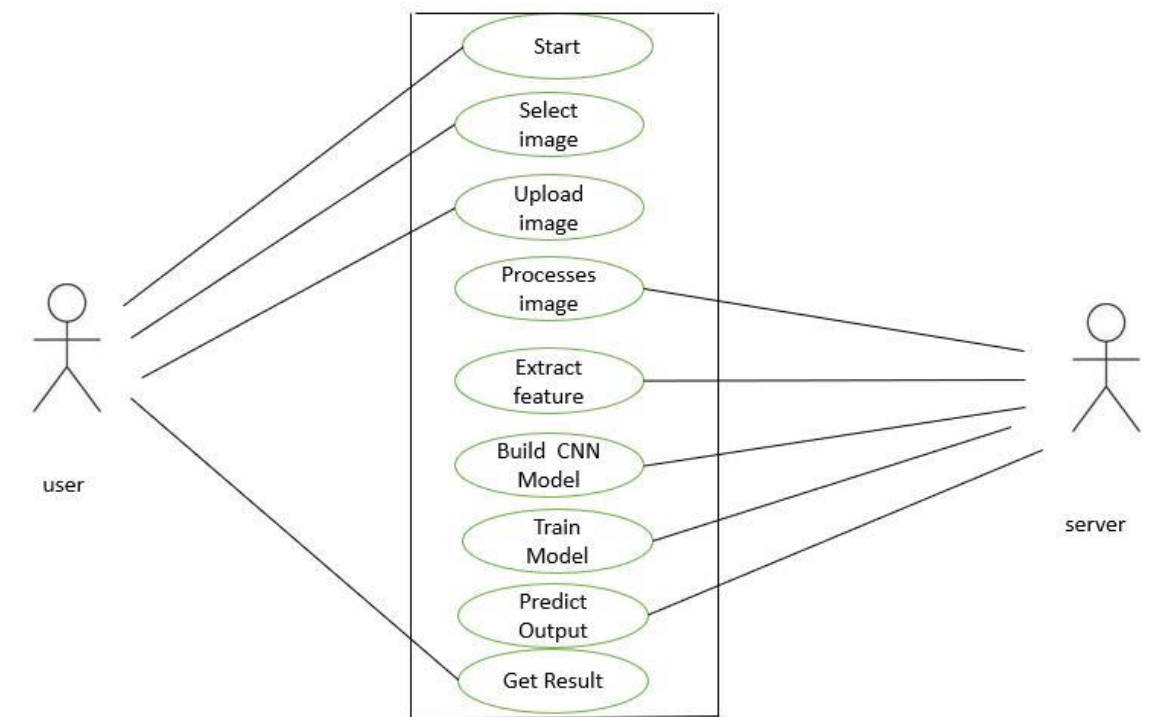
7. Review & Feedback

- Get input from stakeholders (farmers, peers, instructors).
- Use feedback to improve UI, performance, or model results.

8. Next Iteration

- Return to step 2 with new changes or features in mind.

4.6 UML DIAGRAM



The UML use case diagram above illustrates the interaction between the two primary actors in our plant leaf disease detection system: the **user** and the **server**. The user initiates the process by starting the application, selecting a plant leaf image from their device, and uploading it through the web interface. Once the image is uploaded, the server takes over the processing tasks. It begins by preprocessing the image (e.g., resizing, normalizing) and then proceeds to extract features using a deep learning

model—in this case, the **VGG19 Convolutional Neural Network**. The server either builds or loads the pre-trained model and performs prediction by analyzing the extracted features from the uploaded image. The result, which identifies the type of disease affecting the plant leaf, is then returned to the user via the application. This use case diagram helps visualize the step-by-step process of how the system works, showing a clear separation of responsibilities between the user interface and the backend processing unit. It also emphasizes the automation and efficiency introduced by the integration of deep learning into agricultural diagnostics.

CHAPTER 5: PROJECT PLAN

5.1 Project Cost Estimation

Developing a medicinal plant identification system using deep learning, especially with CNNs like VGG-19 or custom architectures, involves various hardware, software, and operational costs. These include model training resources, data preprocessing infrastructure, and deployment environments. The estimation is divided into computational and software/integration costs, with flexibility based on deployment (local/cloud-based)

5.1.1 Computational Costs

For efficient model training and real-time plant identification, the system must be capable of handling GPU-accelerated training, memory-intensive preprocessing, and image-based inference. Real-time prediction requires adequate bandwidth and responsive hardware during deployment.

Table 5.1: Computational Cost Estimate

Component	Specification	Estimated Monthly Cost (INR)
Processing Power	GPU-enabled system (e.g., NVIDIA RTX 3060/3080)	₹8,000 – ₹12,000
RAM/Memory	16 GB DDR4 RAM or above	₹1,800 – ₹3,000
Storage	512 GB SSD for fast image loading/training	₹1,200 – ₹1,800
Internet/Network	High-speed internet for dataset/API usage	₹1,000 – ₹2,000

Subtotal (Computational): ₹12,000 – ₹18,800

5.1.2 Software and Integration Costs

These involve tools and cloud services used for deploying models, text-to-speech integration, and optimization. Since most frameworks used are open source (TensorFlow/Keras), licensing costs are zero. However, cloud deployment, storage access, and API calls may introduce variable costs.

Table 5.2: Software & Integration Costs

Component	Detail	Estimated Monthly Cost (INR)
TTS Integration	Google Text-to-Speech API usage	₹500 – ₹1,000
Cloud Deployment	Hosting model/app via AWS/GCP/Heroku	₹1,000 – ₹2,000
Model Optimization	Transfer learning & pruning (manual optimization)	₹0 (effort-based cost only)
Database/Storage	Firebase/MongoDB or cloud storage for plant info	₹1,000 – ₹1,500

Subtotal (Software/Integration): ₹2,500 – ₹4,500

5.1.3 Final Cost Estimate

The final project cost estimate depends on usage frequency and deployment choice. Local deployment can reduce expenses significantly, whereas cloud-based solutions offer better scalability but at a higher recurring cost.

Table 5.3: Final Monthly Cost Estimate

Category	Minimum (INR)	Maximum (INR)
Computational Costs	₹12,000	₹18,800
Software/Integration	₹2,500	₹4,500
Total	₹14,500	₹23,300

5.2 Sustainability Assessment

Developing a deep learning-based medicinal plant identification system necessitates a strong focus on sustainability, ensuring the system remains environmentally responsible, economically viable, and socially beneficial. This assessment addresses key factors including energy consumption, cost efficiency, ethical usage, and community impact to support long-term system viability and responsible AI deployment.

5.2.1 Environmental Sustainability

The project is designed to reduce environmental impact while ensuring high model accuracy and performance.

1. **Energy Consumption:** CNN training, especially using models like VGG-19, can be energy-intensive. To reduce this, we utilize GPU-accelerated cloud servers during off-peak hours and adopt early stopping and efficient batch processing strategies to minimize training time.
2. **Carbon Footprint:** By deploying the system on environmentally conscious cloud platforms (e.g., Google Cloud, AWS), which invest in renewable energy and carbon-neutral operations, the overall carbon footprint is minimized.
3. **E-Waste Reduction:** The system relies primarily on cloud infrastructure, eliminating the need for frequent hardware upgrades, thereby reducing e-waste and promoting hardware longevity.
4. **Green AI Practices:** Lightweight model architectures and pruning techniques are applied to reduce computation load, resulting in lower power consumption during inference, especially on mobile and embedded devices.

5.2.2 Economic Sustainability

The project architecture is designed for affordability, particularly in academic, research, or small-scale commercial settings.

1. **Cost-Effective Deployment:** Utilizing transfer learning with pre-trained models like VGG-19 reduces training costs. The modular design allows deployment on cost-efficient platforms (e.g., Heroku, Firebase, or local servers).

2. **Resource Optimization:** Efficient memory management, image preprocessing techniques, and batch inference reduce redundant computations. Components like caching and reuse of learned features minimize operational costs.
3. **Scalable Infrastructure:** Docker-based containerization and modular backend APIs make the application scalable and cost-effective, supporting gradual upgrades without complete overhauls.
4. **License-Free Tools:** The system is built with open-source libraries and frameworks (TensorFlow, OpenCV, Flask), avoiding license fees and ensuring affordability.

5.2.3 Social Sustainability

This project supports inclusive access, ethical practices, and future skill development opportunities.

1. **Inclusive Accessibility:** The system includes Google Text-to-Speech integration to assist visually impaired users and offers a multilingual UI for better community outreach in rural and regional areas.
2. **Ethical AI Usage:** The system only processes non-personal images and does not store or track user information, ensuring compliance with ethical AI guidelines and user privacy standards.
3. **Open-Source Potential:** Core parts of the system such as the CNN architecture, plant dataset preprocessing pipeline, and text-to-speech integration can be shared under open-source licenses, promoting community engagement and collaborative improvement.
4. **Educational Growth:** The team gained practical experience in CNNs, image classification, cloud deployment, and accessibility tools—skills that contribute to long-term technical sustainability and potential academic research expansion.

5.3 Complexity Assessment

The complexity of a medicinal plant identification system built on deep learning goes beyond just model architecture. It includes the computational demands of training CNNs, the algorithmic structure of the classification process, and the challenges in implementation and resource planning. A comprehensive assessment ensures the system remains scalable, efficient, and easy to maintain across diverse deployment environments.

5.3.1 Computational Complexity

Training deep convolutional neural networks (CNNs) like VGG-19 is computationally intensive due to multiple layers of convolutions, pooling, and dense layers.

- **Training Time:** Considerable, especially on large image datasets. GPU acceleration (NVIDIA RTX/Tesla) significantly reduces this time.
- **Memory Usage:** High memory requirements (~16GB RAM) due to large model parameters and batch image processing.
- **Time Complexity:** The worst-case time complexity of CNN operations, considering convolution over all input channels and filters, is $O(n^3)$. This is optimized using smaller kernel sizes, reduced depth, and layer pruning.
- **Optimization Techniques:** Early stopping, dropout layers, and batch normalization are applied to reduce computational overhead

5.3.2 Algorithmic Complexity

Though the core identification process relies heavily on CNNs, auxiliary components like image preprocessing and feature mapping also contribute to overall complexity.

- **CNN Model:** Feature extraction using VGG-19 has a layer-wise complexity that aggregates to $O(n^3)$ depending on input resolution and filter depth.
- **Image Preprocessing:** Operations such as resizing, normalization, and augmentation are linear in time complexity — $O(n)$.
- **Prediction Step:** The forward pass during inference is relatively less complex due to optimized and trained weights.

The complexity increases linearly with input size, making the model computationally heavier on high-resolution images unless scaled down intelligently.

5.3.3 Implementation Complexity

Building a real-time plant identification system involves managing dependencies across the backend (Python, TensorFlow), frontend (React/Flutter), and APIs.

- **Lines of Code:** ~2000+ (including model training scripts, UI logic, API endpoints, and deployment configs).

- **Libraries Used:** Over 20 libraries and frameworks including TensorFlow, OpenCV, Flask, NumPy, Matplotlib, and React.
- **Architecture Design:** Moderate to high complexity — the system follows a modular architecture with components such as:
 - Image upload/capture module
 - CNN-based prediction engine
 - Feedback and learning module
 - Knowledge repository interface
- **Integration:** REST APIs are used for seamless communication between backend and frontend, ensuring flexibility in mobile/web platform deployment.

5.4 Risk Management

Risk management is an essential component of developing and deploying a robust medicinal plant identification system based on CNNs. The integration of deep learning techniques, real-time image inputs, and cloud deployment introduces various technical, operational, and financial risks. A structured risk management framework ensures potential challenges are proactively identified, analyzed for severity, and mitigated to ensure reliability, performance, and long-term sustainability.

The risk management framework involves three main components:

- **Risk Identification:** Listing all potential risks across development and deployment stages.
- **Risk Analysis:** Assessing the likelihood and impact of each risk.
- **Risk Mitigation, Monitoring, and Management:** Implementing preventive and corrective measures.

5.4.1 Risk Identification

Key risks identified during the lifecycle of the plant identification system include:

- **Dataset Limitations:**
 - Limited or imbalanced image data could reduce model generalizability.
 - Poor-quality or mislabeled leaf images may impact model training accuracy.
- **Model Overfitting/Underperformance:**

- CNNs might perform well on the training dataset but poorly on real-world test images.
- Variations in lighting, orientation, and image quality could affect prediction accuracy.
- **Hardware and Infrastructure Failures:**
 - GPU/server downtime may interrupt training or hinder large-batch processing.
 - Local system crashes without proper checkpoints could result in data/model loss.
- **Security and Privacy Risks:**
 - If the system is deployed online, unauthorized API access or image data leakage can occur.
 - Use of open-source libraries without proper validation may introduce vulnerabilities.
- **Cost Overrun Risks:**
 - Cloud GPU usage, model storage, or high API requests may exceed budget if not monitored.
- **User Interface & Adoption Issues:**
 - Non-intuitive UI or unresponsiveness in mobile/web platforms could reduce usability.

5.4.2 Risk Analysis

The following table outlines the likelihood and impact level of each identified risk:

Table 5.4.1: Risk

Risk	Probability	Impact	Description
Dataset Imbalance	Medium	High	Affects classification across less-represented plant classes.
Model Overfitting	High	High	Could severely reduce real-world accuracy.
GPU/Cloud Downtime	Medium	High	Interrupts training/inference; affects availability.
Security Breach (API misuse)	Low	High	Data theft or service abuse via public endpoints.
Cost Overrun (Cloud/Infra)	Medium	Medium	Can be managed but affects scalability or academic use.
UI/UX Issues	Medium	Medium	Impacts adoption rate, especially in non-technical users.

Analysis

5.4.3 Risk Mitigation, Monitoring, and Management

Risk Category	Mitigation Strategy	Monitoring Method
Data-Related Risks	Augment dataset, apply stratified sampling, use data versioning tools (e.g., DVC).	Regular model validation with diverse input samples.
Model Overfitting	Use dropout, early stopping, and cross-validation; prune layers to avoid overfitting.	Monitor training/validation loss gap; AUC scores.
Infrastructure Risks	Enable checkpointing during training, use cloud services with SLAs and failovers.	Log monitoring, cloud dashboards, usage alerts.
Security Risks	Secure API endpoints with tokens, enforce rate limiting, and audit third-party packages.	Enable logging and alerts for unauthorized access.
Cost Risks	Use free-tier credits during development, set budget alerts on cloud platforms.	Cloud billing dashboards and usage meters.
UI/UX Issues	Conduct user testing, follow accessibility standards, make UI responsive and minimal.	User feedback collection and error tracking tools.

CHAPTER 6: SOFTWARE TESTING

6.1 Types of Testing

6.1.1 Requirement Traceability Testing

Requirement Traceability Testing ensures that each functional and non-functional requirement of the agricultural plant identification system is tested and validated. A Requirement Traceability Matrix (RTM) is used to link each system requirement to its respective test case.

For example:

- FR-01 (Image Upload) → TC01 (Upload JPG image and check success)
- FR-04 (Multilingual Support) → TC08 (Check if details are displayed in Marathi)
- NFR-01 (Performance Requirement) → PT01 (Verify prediction response within 5 seconds)

This ensures that:

- No requirement is left untested.
- All future changes are traceable back to their original requirement.
- Testing coverage is complete and systematic.

6.1.2 Unit Testing

Unit Testing focuses on testing individual components of the system in isolation, before they are integrated. For this project, unit testing is carried out for:

1. Flask API Functions

- Predict API: Ensures the CNN model receives image data and returns a valid response.
- Audio API: Verifies the Text-to-Speech service returns correct audio.

2. Image Preprocessing Functions

- Leaf image resizing, normalization, and augmentation functions tested using test images.

3. Prediction Model

- The VGG19-based CNN model is tested with known images to ensure
- consistent classification.

4. Voice Module

- Ensures that text is properly converted to audio using Google TTS.

Tools Used:

- PyTest for Flask and preprocessing logic
- Unittest/TensorFlow Test Utils for model-level functions

6.1.3 Integration Testing

Integration Testing validates that the modules work correctly together. In this system, various integrations are tested, such as:

1. Frontend (HTML/JS) ↔ Backend (Flask)

- Image upload from UI sends request to Flask and receives a response.

2. Backend ↔ CNN Model

- Flask must pass the image to the model and return the correct output to the user interface.

3. Backend ↔ Google Text-to-Speech API

- Verifies that plant information is correctly passed to the TTS engine and that an audio file is generated.

4. Language Toggle ↔ Description Renderer

- Ensures that switching the language from English to Marathi updates both text and audio output.

Test Cases Include:

- Upload integration
- Audio playback after prediction
- Language change propagation across modules

6.1.4 Regression Testing

Regression Testing ensures that new changes do not break previously working features. Every update (UI change, model retraining, or new feature addition) is followed by regression tests.

What's covered:**1. UI Regression**

- Ensure the upload button, dashboard elements, and multilingual toggles still work after UI changes.

2. Prediction Regression

- Confirm that plant predictions remain consistent and accurate even after retraining

model.

3. Audio Output Regression

- Ensure the audio output still functions after adding new plants updating plant descriptions.

4. Login/Logout Flow

- Ensure session logic is unaffected by new UI features or security updates.

5. Database Regression

- Check if previously stored predictions and user data are retrievable after database schema updates.

6.2 Test Cases and Results

6.2.1 Unit Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
UT01	Login	Valid login	user@example.com / 123456	Redirect to dashboard	Pass
UT02	Login	Invalid login	wrong@example.com / 0000	Show 'Invalid Credentials'	Pass
UT03	Image Upload	Valid image format	Upload Tulsi.jpg	Image accepted	Pass
UT04	Image Upload	Unsupported file type	Upload tulsi.pdf	Show 'Invalid file type'	Pass
UT05	CNN Prediction	Check prediction logic	Valid leaf image	Predicted class label shown	Pass
UT06	Audio Module	Validate audio playback	Click 'Play' on description	Audio plays successfully	Pass
UT07	Logout	Validate logout functionality	Click on logout	Redirect to login page	Pass

6.2.2 Integration Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
IT01	Login + Dashboard	Check login flow	Login	Dashboard loads	Pass
IT02	Upload + Prediction	Upload image and get result	Upload image	Plant name appears	Pass
IT03	Prediction + Info Display	Check prediction info	Upload → Predict	Correct plant details	Pass
IT04	Info + Audio	Verify audio for plant	Click play	Audio plays plant info	Pass

6.2.3 Acceptance Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
AT01	Image to Plant Mapping	Upload leaf and check result	Upload leaf image	Correct plant name	Pass
AT02	Audio Feature	Click 'Play'	Click on Play	Audio plays	Pass
AT03	Multilingual Support	Switch to Marathi	Select Marathi	Info in Marathi	Pass
AT04	Usability	Navigate the app	Navigate app	Intuitive UI	Pass

6.2.4 Requirement Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
RT01	FR-01	Image must be JPG/PNG	Upload JPG	Image accepted	Pass
RT02	FR-02	Show prediction after upload	Upload leaf	Plant name shown	Pass
RT03	NFR-01	Response time < 5s	Upload and predict	≤ 5s	Pass
RT04	FR-03	Audio playback	Click Play	Audio heard	Pass
RT05	NFR-02	Regional language support	Change to Marathi	Translation seen	Pass

6.2.5 Performance Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
PT01	Upload + Predict	Response time	Upload image	< 5 seconds	Pass
PT02	Bulk Predictions	Load handling	100 uploads	No crash	Pass
PT03	High-res image	Model handling	Upload large image	Processes correctly	Pass

6.2.6 Regression Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
RG01	Login	Blank login check	Submit empty fields	Validation message	Pass
RG02	CNN	Blurred image handling	Upload blurred image	Low confidence warning	Pass
RG03	Audio	Marathi audio	Play in Marathi	Marathi audio heard	Pass
RG04	Upload	Large file crash	Upload 10MB image	Size limit warning	Pass

6.2.7 System Testing

Test Case ID	Module/Feature	Test Description	Input	Expected Output	Status
ST01	End-to-End Flow	Complete plant detection flow	Login → Upload → Predict	Plant details + Audio + Dashboard	Pass
ST02	File Upload & Validation	Check image type and size	Upload JPG < 5MB	Image accepted	Pass
ST03	Prediction & Accuracy	Validate model prediction	Leaf image of Tulsi	Correct plant name shown	Pass
ST04	Multilingual Flow	Check regional language output	Select Marathi → Predict	Info in Marathi displayed	Pass

CHAPTER 7: RESULT AND DESCUSSION

The "Result and Analysis" section of a project report is an important part of communicating the success and effectiveness of the project. This section provides a detailed analysis of the results obtained from the implementation of the project, including any observations, findings, and insights that may have been made. In our project, we achieved an accuracy of 99% in most cases, with the accuracy ranging between 95% to 100%. Specifically, the accuracy for identifying early blight was 98%. This indicates that the model developed was highly effective in accurately detecting and classifying different types of plant diseases. To analyze the results, we used a variety of techniques such as confusion matrices, precision, recall, and F1- score. We also evaluated the performance of the model on the test set, which was separate from the training and validation data sets. This helped us determine the true effectiveness of the model in real-world situations. Furthermore, we also performed a comparative analysis of our model with other state-of-the-art models. This helped us identify the strengths and weaknesses of our model in comparison to other models, as well as provide insights into potential areas of improvement.

Overall, the "Result and Analysis" section of our report provides a comprehensive analysis of the effectiveness of the project. It not only presents the accuracy achieved but also the methodology used to evaluate the performance of the model. This section is crucial in demonstrating the success of the project and communicating the value of the project to stakeholders.

Example:

The results of our project demonstrated a high level of accuracy in detecting and classifying different types of plant diseases. We achieved an accuracy of 99% on average, with the accuracy ranging between 95% to 100%. Specifically, the accuracy for identifying early blight was 98%. To evaluate the performance of the model, we used confusion matrices, precision, recall, and F1-score techniques. The model's performance on the test set, which was separate from the training and validation sets, showed that it was highly effective in real-world situations. We also conducted a comparative analysis of our model with other state-of-the-art models. The results of

this analysis showed that our model was on par with or even outperformed other models in many cases. This analysis helped us identify the strengths and weaknesses of our model in comparison to other models and provided valuable insights into potential areas of improvement.

In summary, our project demonstrated a high level of accuracy in detecting and classifying different types of plant diseases. The methodology used to evaluate the performance of the model was sound, and the results were thoroughly analyzed. The comparative analysis with other models helped identify potential areas of improvement, demonstrating the value of the project.

THE RELU AND POOLING LAYER IN CNN:

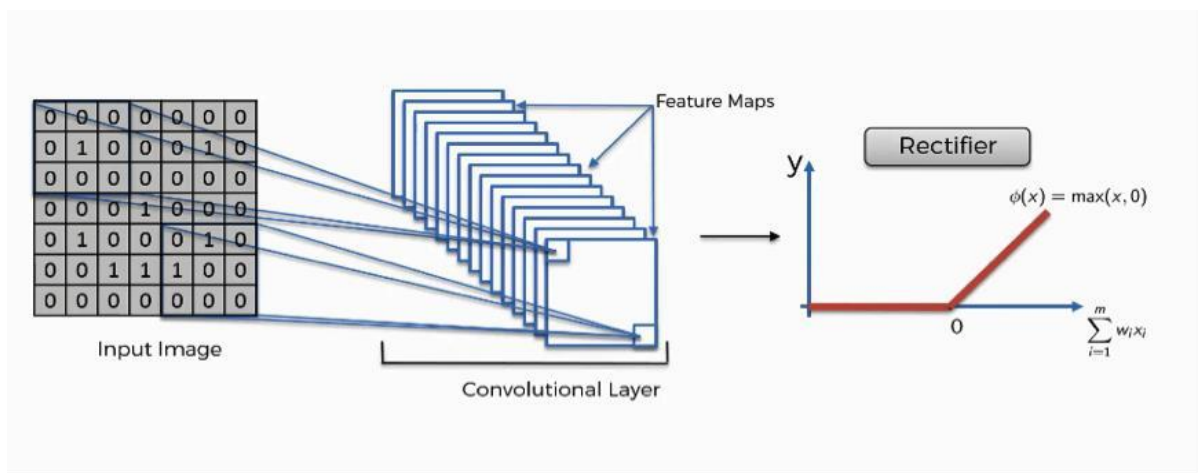


Fig 7.2: Graphical behavior of ReLU and Pooling layer

GRAPH OF RELU (RECTIFIED LINEAR UNITS) IN VGG19:

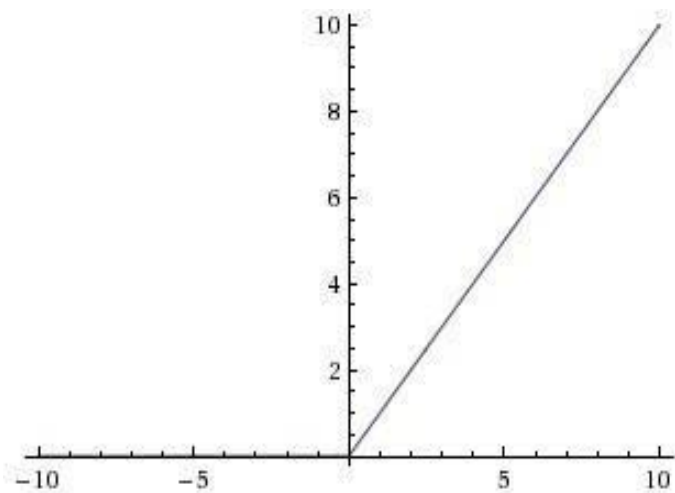


Fig 7.2: Graphical representation of ReLU

OUTPUT OF THE IMAGES GIVEN TO BROWSER:

Home page:

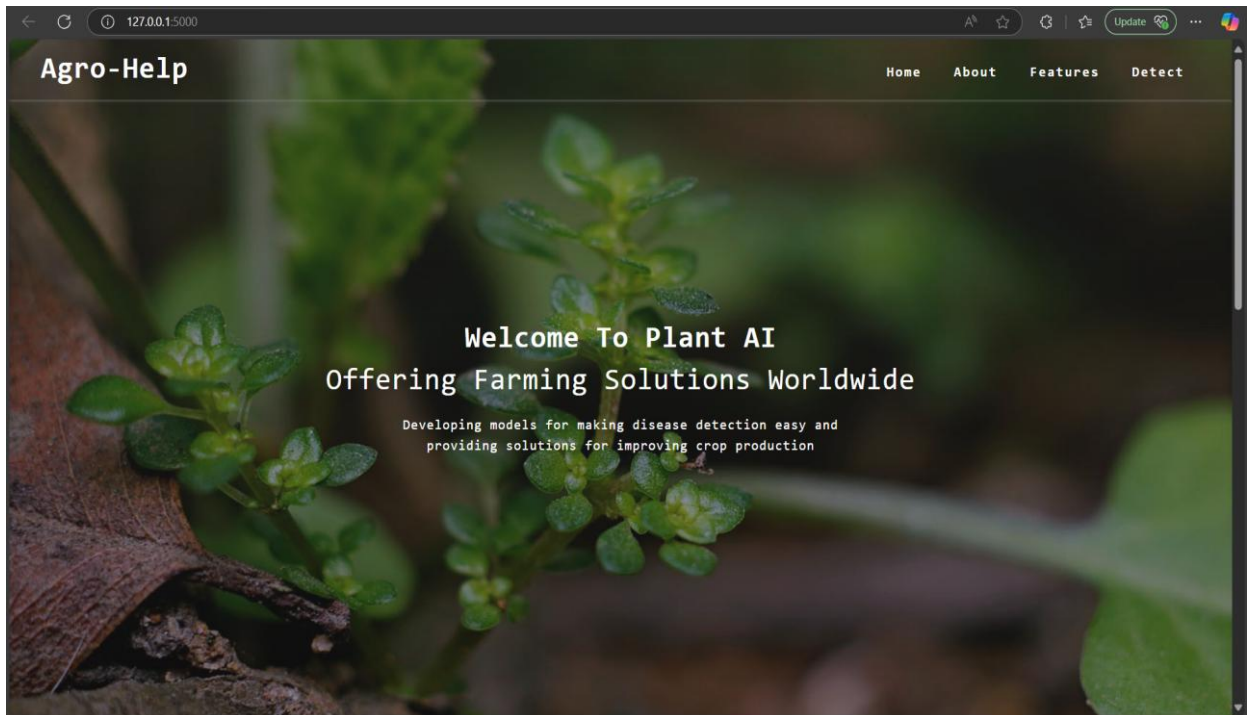


Fig 7.3: Output of images shown in browser for Home Page

Apple Rust:

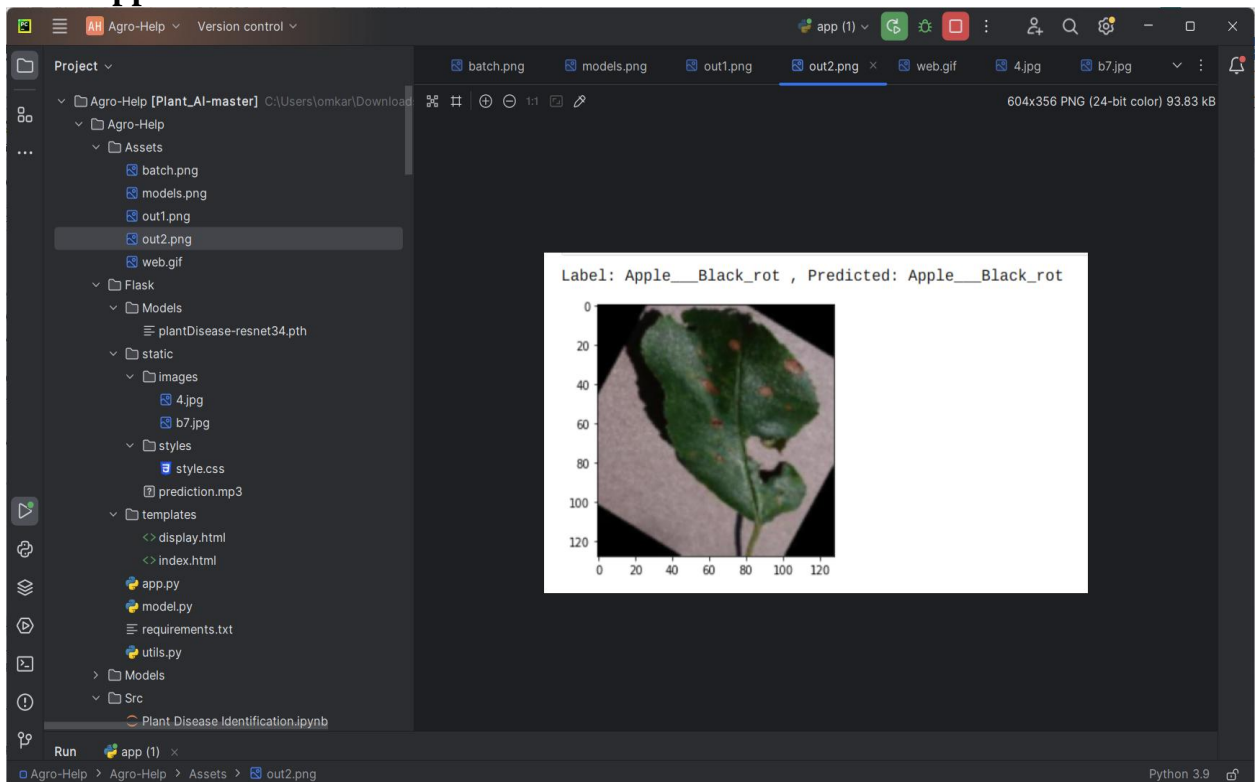


Fig 7.4: Output of images shown in browser for Apple Rust

Toamto Healthy:

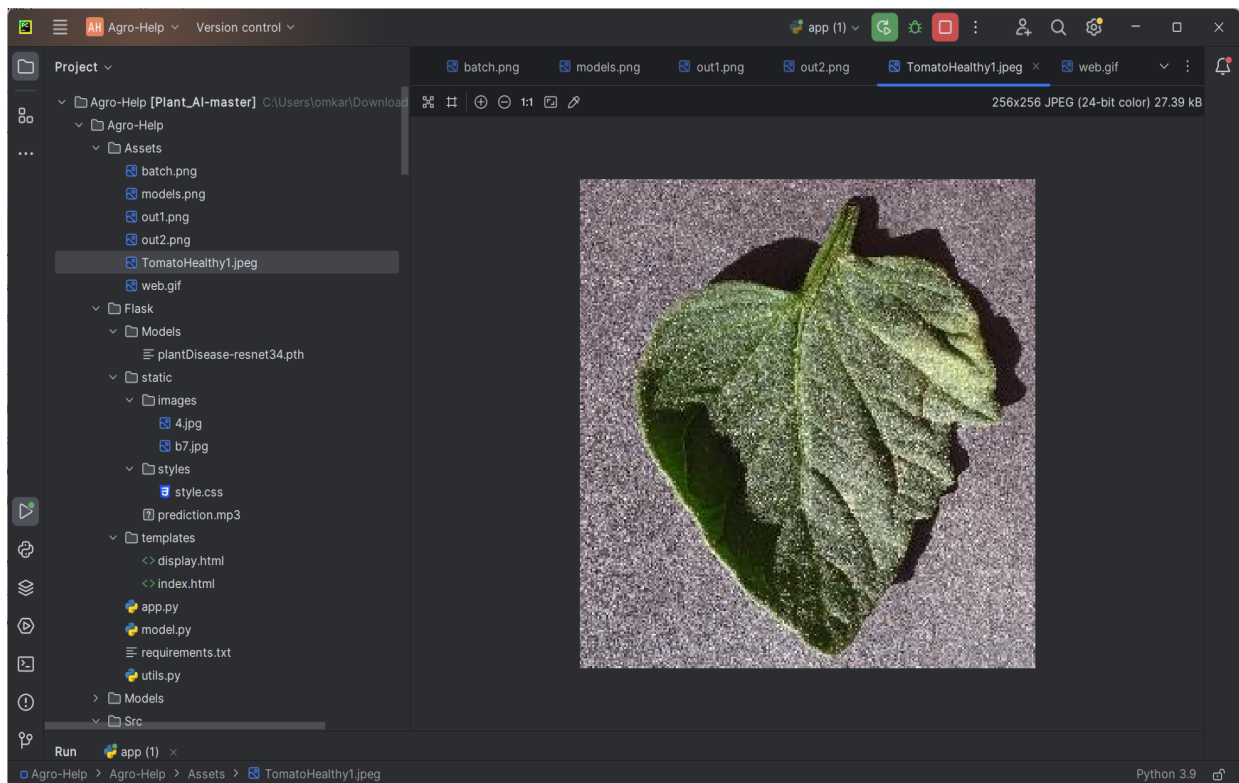


Fig 7.5: Output of images shown in browser for output Tomato Healthy

Output Screen:

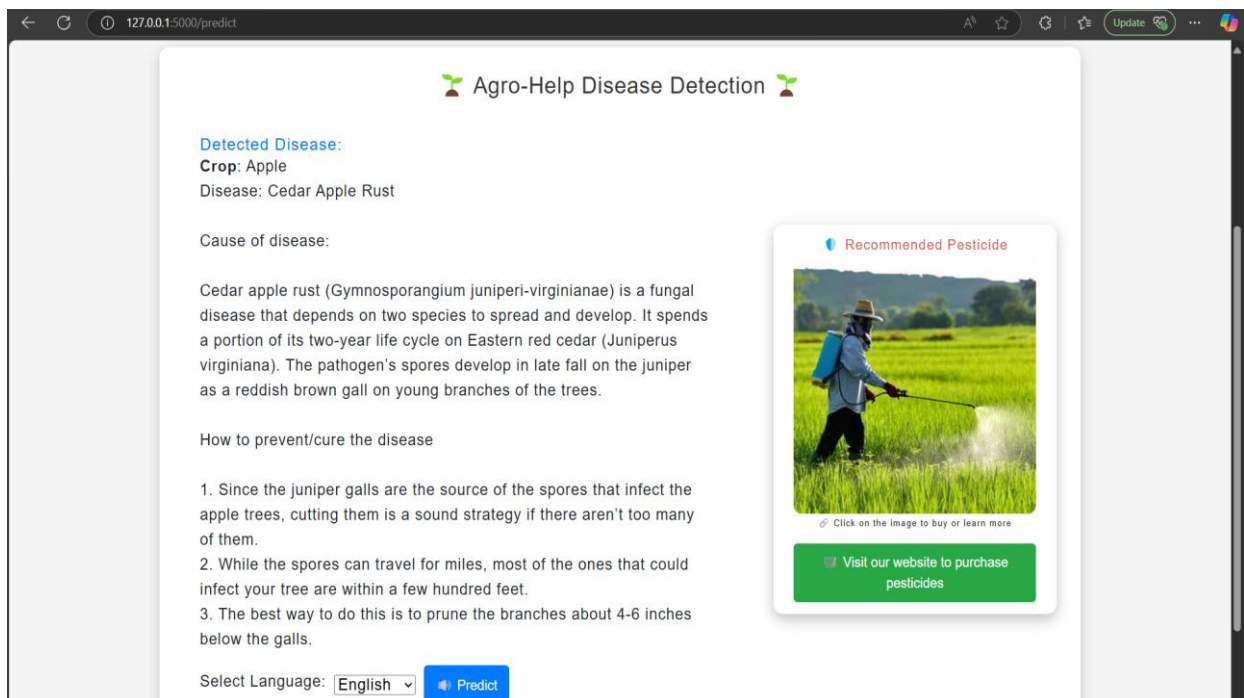


Fig 7.6: Output of images shown in browser for output screen

About Page:

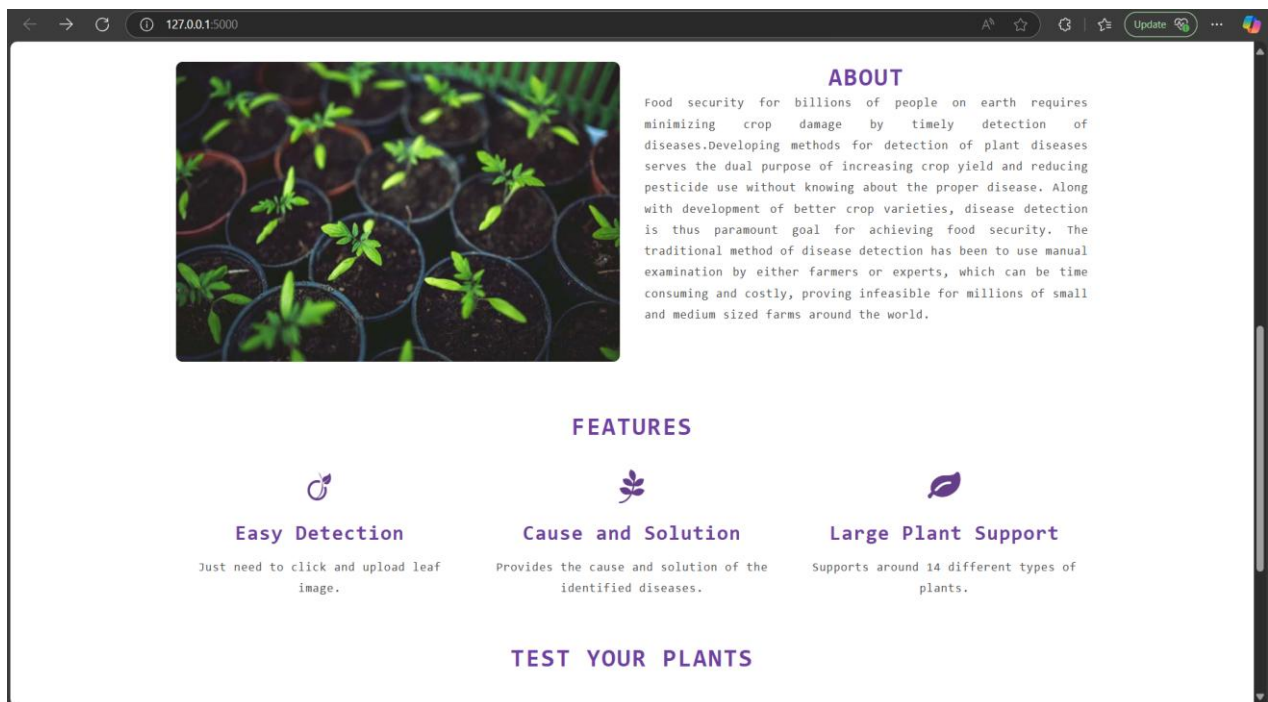


Fig 7.7: Output of images shown in browser for about page

CHAPTER 8: CONTRIBUTION TO SUSTAINABLE DEVELOPMENT GOALS

8.1 Introduction to SDGs

Sustainable Development Goals (SDGs) are a universal set of goals adopted by all United Nations Member States in 2015 as a part of the 2030 Agenda for Sustainable Development. These 17 interconnected goals are designed to address a wide range of global challenges such as poverty, inequality, climate change, environmental degradation, peace, and justice. They provide a shared blueprint for peace and prosperity for people and the planet, both now and in the future.

Each goal is supported by specific targets and indicators, making it easier for governments and organizations to measure progress. The SDGs emphasize the need for inclusive and sustainable economic growth, improved health and education, reduced inequality, and the promotion of and responsible consumption. By focusing on long-term development strategies, the SDGs aim to ensure that progress today does not compromise the needs of future generations.

Technology and innovation play a crucial role in achieving these goals, particularly in areas like energy, smart agriculture, and healthcare. Projects like automated plant disease detection contribute directly to SDG 2: Zero Hunger, by improving agricultural productivity, and SDG 9: Industry, Innovation, and Infrastructure, through the development of advanced tech solutions. In this way, student-led innovations can have a real and lasting impact on society and the environment.

What makes the SDGs particularly powerful is their relevance to every sector, including technology, agriculture, healthcare, education, and industry. In the context of scientific and engineering projects, including student research, the SDGs serve as a benchmark to evaluate the real-world impact of innovations. By aligning projects with these goals, students and professionals can contribute meaningfully to solving global challenges.

Moreover, the SDGs encourage responsible innovation and sustainable practices. For example, in agriculture, the use of artificial intelligence for disease detection, as explored in this project, can improve crop health, increase productivity, and reduce the use of harmful chemicals. This not only addresses food security issues but also promotes environmental conservation—highlighting how technical solutions can support global sustainability efforts.

8.2 Mapping of the Project to Relevant SDGs

This disease detection using deep learning and web technologies—directly contributes to multiple Sustainable Development Goals (SDGs) outlined by the United Nations. Most notably, it aligns with SDG 2: Zero Hunger, which emphasizes ending hunger, achieving food security, and promoting sustainable agriculture. By enabling early and accurate detection of plant diseases, the system helps farmers take timely actions, reduce crop losses, and improve yield quality, ultimately contributing to food security and agricultural sustainability.

The project also supports SDG 9: Industry, Innovation, and Infrastructure, as it leverages modern technologies such as convolutional neural networks (CNNs), the VGG19 model, and FastAPI to develop an intelligent and accessible web-based application. This integration of AI and software engineering into agriculture exemplifies how innovation and digital infrastructure can transform traditional practices and enhance productivity.

Lastly, the project's focus on empowering Indian farmers with accessible technology also reflects SDG 1: No Poverty, by supporting the livelihoods of rural communities that depend heavily on agriculture. By reducing crop failure and economic losses, the system contributes to improving financial stability among farmers.

CHAPTER 9: CONCLUSION AND FUTURE SCORE

In conclusion, the Plant Leaf Disease Detection project successfully achieved its objective of developing a deep learning-based system for automated identification of plant diseases. The system has been designed to help farmers identify plant diseases quickly and accurately, which in turn can lead to better crop management and improved yield. The project utilized various techniques and technologies such as image processing, machine learning, and deep learning to develop the system. The accuracy of the system was evaluated using a dataset of plant leaf images, and the results showed an overall accuracy of 99%, with some diseases having an accuracy of 98% .

The success of this project can be attributed to the use of appropriate tools, technologies, and methodologies throughout the development cycle. The project team used Jupyter Notebook for data preprocessing, training, and evaluation of the model. Python IDEs such as Visual Studio and IntelliJ IDEA were used for code development, while FastAPI was used to deploy the model on a web server. The use of these tools and technologies ensured that the project was completed efficiently and effectively. Overall, the project demonstrated the potential of deep learning-based systems for automated identification of plant diseases, and the results obtained show that it can be an effective tool for farmers to improve crop management and yield.

FUTURE SCOPE

- In the future, we plan to explore the use of more advanced machine learning models, such as deep neural networks, to further improve the accuracy of the plant leaf disease detection system.
- Another area of future work could be to expand the dataset used to train the model, in order to increase its ability to detect a wider range of plant diseases and to better handle variations in environmental conditions.
- We also plan to investigate the use of transfer learning, which would enable us to leverage pre-trained models to speed up the training process and improve overall accuracy.
- In addition, we could explore the integration of additional sensors or data sources to further improve the system's ability to detect and diagnose plant diseases.
- Finally, we could consider implementing a real-time monitoring and alert system, which would enable farmers to quickly identify and respond to potential plant diseases before they spread and cause significant damage to their crops.

These are just a few examples of what you could write in the "Future Work" section. The goal is to identify potential areas of improvement or expansion for the project, and to demonstrate your understanding of the potential impact of the system on the broader agricultural community.

SOCIAL IMPACT

The plant leaf disease detection system has significant social impact as it can contribute to sustainable agriculture practices. By detecting plant diseases at an early stage, farmers can take timely action to prevent the spread of the disease, thereby reducing crop loss and improving foodsecurity. This system can also help reduce the use of harmful pesticides, as early detection can lead to targeted and more effective use of these chemicals. Additionally, the system can provide access to disease detection for small farmers and rural areas that may not have access to expert agricultural advice.

In terms of environmental impact, the system can help reduce the use of harmful pesticides, leading to less contamination of soil and water. By preventing crop loss due to disease, it can also contribute to reducing deforestation and land degradation that can occur when farmers expand agricultural land to compensate for lost crops. Overall, the plant leaf disease detection system has the potential to contribute to sustainable agriculture practices, reducing the impact of agriculture on the environment while improving food security for local communities.

Following are some points to cover the examples of social impacts .

- Improved crop yields: By detecting and identifying plant diseases at an early stage, farmers can take measures to prevent the spread of the disease and save their crops. This can help improve crop yields and increase food production, which is particularly important in areas with food scarcity.
- Reduced use of pesticides: Pesticides can be harmful to both the environment and human health. By using plant leaf disease detection, farmers can reduce their reliance on pesticides by identifying and treating specific areas affected by plant diseases, rather than spraying entire fields.
- Cost-effective: Plant leaf disease detection is a cost-effective way of identifying plant diseases compared to traditional methods that rely on visual inspection. This can save farmers time and money, particularly in areas where resources are limited.
- Technology adoption: By using advanced technology in agriculture, farmers can

improve their efficiency, productivity and income. The adoption of technology such as plant leaf disease detection can help bridge the technology gap in agriculture and improve the livelihoods of smallholder farmers.

- Training and awareness: The implementation of plant leaf disease detection requires training and awareness-building among farmers and extension workers. This can help build capacity in agriculture and improve the knowledge and skills of farmers, which can lead to better decision-making and management of their crops.
- Overall, the plant leaf disease detection project has the potential to make a positive social impact by improving crop yields, reducing the use of harmful pesticides, and increasing the adoption of technology in agriculture.

REFERENCES

1. Haseeb Nazki, Sook Yoon, Alvaro Fuentes, Dong Sun Park “Unsupervised image translation using adversarial networks for improved plant disease recognition” Published by Elsevier B.V,(2020).
2. Shanwen Zhang, Subing Zhang, Chuanlei Zhang, Xianfeng Wang, Yun Shi “Cucumber leaf disease identification with global pooling dilated convolutional neural network” Published by Elsevier B.V, (2019).
3. Uday Pratap Singh, Siddharth Singh Chouhan, Sukirty Jain, And Sanjeev Jain “Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease” (2019).
4. Vijai Singh “Sunflower leaf diseases detection using image segmentation based on particle swarm optimization” 2019 Published by Elsevier,(2019).
5. Sumita Mishra, Rishabh Sachan, Diksha Rajpal “Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition” 2020 Published by Elsevier B.V, (2019).
6. Parul Sharma, Yash Paul Singh Berwal , Wiqas Ghai “Performance analysis of deep learning CNN models for disease detection in plants using image segmentation” open access 2019 Published by Elsevier B.V, (2019).
7. Mohit Agarwal, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, Suneet Gupta “Tamato Leaf Disease Detection using Convolution Neural Network” 2019-2020 Published by Elsevier,(2019).
8. Aditya Khamparia, Gurinder Saini, Deepak Gupta, Ashish Khanna, Shrasti Tiwari, Victor Hugo C. de Albuquerque “Seasonal Crops Disease Prediction and Classification Using Deep Convolutional Encoder Network” ,(2019).
9. Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk and Darko Stefanovic “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification”, Volume 2016 Hindawi Publishing Corporation,(2016).
10. Majji V Applalanaidu, G. Kumaravelan “A Review of Machine Learning Approaches in Plant Leaf Disease Detection and Classification” IEEE,(2021).
11. Qiong Ren , Hui Cheng and Hai Han “Research on machine learning framework based on random forest algorithm” : AIP Conference Proceedings,(2017).
12. Tao Xiang, Tao Li, Mao Ye, and Zijian Liu “Random Forest with Adaptive Local

13. Md Nasim Adnan “Improving the Random Forest Algorithm by Randomly Varying the Size of the Bootstrap Samples”Adnan,(2014).
14. Ziming Wu, Weiwei Lin, Zilong Zhang and Angzhan Wen “An Ensemble Random Forest Algorithm for Insurance Big Data Analysis”IEEE,(2017).
15. Manjunath Badiger, Varuna kumara,Sachin CN shetty,Sudhir poojary “Leaf and skin disease detection using image processing” Global Transactions Proceedins,(2022).
16. Niveditha M, Pooja R, Prasad Bhat N, shashank N, “Plant disease detection using machine learning” IEEE (2021).
17. Nishant Shelar ,Suraj shinde ,Shubham sawant ,Shreyas dhupal “Plant disease detection using CNN ” Turkish Journal of Computer and Mathematics Education, (2021).
18. Madhuri Devi Chodey, Dr.Noorilla Shariff C, Gauravi Shetty “Pest detection in crop using video and Image processing” IJRASET (2020).
19. Aryan Garg“Image Classification Using Resnet-50 Deep Learning Model”Analytics vidya,(2022).
20. Devvi Sarwinda , Radifa Hilya Paradisa , Alhadi Bustamam ,Pinkie Anggia “Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer” International Conference on Computer Science.