## A REPORT ON

Leaf Disease Detection

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR

# PROJECT-BASED LEARNING (SECOND YEAR ENGINEERING)

## SUBMITTED BY

**KARAN SINGH EXAM NO:**

**KOSANA SRI SAI MAHEEDHAR EXAM NO:**

**NITHUL K EXAM NO:**

**PREM MAITY EXAM NO:**



DEPARTMENT OF COMPUTER ENGINEERING

## ARMY INSTITUTE OF TECHNOLOGY

DIGHI HILLS, ALANDI ROAD, PUNE 411015 **SAVITRIBAI PHULE PUNE UNIVERSITY 2022-2023**

ARMY INSTITUTE OF TECHNOLOGY, DEPARTMENT OF COMPUTER ENGINEERING 2022-23

1



# CERTIFICATE

This is to certify that the project report entitles.

## Leaf Disease Detection

SUBMITTED BY

**KARAN SINGH EXAM NO:**

**KOSANA SRI SAI MAHEEDHAR EXAM NO:**

**NITHUL K EXAM NO:**

**PREM MAITY EXAM NO:**

## (Prof. Yogita T Hambir)

Guide

Department of Computer Engineering

ARMY INSTITUTE OF TECHNOLOGY, DIGHI, PUNE-411015

Place: Pune Date:

# ACKNOWLEDGEMENT

We are overwhelmed in all humbleness and gratefulness to acknowledge our depth to all those who have helped us to put these ideas, well above the level of simplicity and helped us develop this into something concrete. The outcome of this project required a lot of guidance and assistance from many people and we are extremely grateful and privileged to be provided with it through all the helping entities.

We owe our deep gratefulness to our project guide **Prof. Yogita T Hambir**, who gave her attention and took an interest on our project work and steered us in the right direction all along, throughout our project work by providing all the necessary information for developing a good system. We are extremely thankful to her for providing her fascinating support and guidance, despite her busy schedule.

We are thankful and fortunate enough to get constant encouragement, support, and guidance from all the teaching staff of the Computer Engineering Department which helped us in successfully completing our project work. Also, we would like to extend our sincere esteem to all our seniors for their timely and knowledgeable support without which completion of this project would’ve been much harder to achieve.

**KARAN SINGH**

**KOSANA SRI SAI MAHEEDHAR**

**NITHUL K**

**PREM MAITY**

# TABLE OF CONTENTS

**CHAPTER TITLE PAGE NO.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | | **Title of Chapter** | **Page No.** |
| **01** | | **Introduction** | **05** |
|  | 1.1 | Problem Statement | 05 |
|  | 1.2 | Motivation | 05 |
|  | 1.3 | Objectives | 05 |
| **02** | | **System Architecture** | **06** |
|  | 2.1 | System Design | 06 |
| **03** | | **Team Organization** | **07** |
|  | 3.1 | Team Structure | 07 |
|  | 3.2 | Management reporting and communication | 07 |
|  | 3.3 | Responsibilities and Duties | 07 |
| **04** | | **Software and Hardware Requirements** | **08** |
|  | 4.1 | Hardware Requirements | 08 |
|  | 4.2 | Software Requirements | 08 |
| **05** | | **Implementation** | **09 - 18** |
|  | 5.1 | Overview of Project Modules | 09 |
|  | 5.2 | Small Description of Modules | 10 |
|  | 5.3 | Algorithm | 10 |
|  | 5.4 | Tools and Technologies Used | 11-18 |
| **06** | | **Results** | **19 – 21** |
|  | 6.1 | Output Screenshots | 19 |
| **07** | | **Challenges Faced** | **22** |
| **08** | | **Conclusion and Future Scope** | **23-24** |
|  | 8.1 | Conclusions | 23 |
|  | 8.2 | Future of Our Application | 24 |
| **9** | | **References** | **25** |

# INTRODUCTION

* 1. PROBLEM STATEMENT

Plant diseases are a major threat to global food security, leading to significant crop losses and economic damages. Early detection and diagnosis of plant diseases can help prevent further spread and reduce the need for chemical treatments. Traditional methods of detecting plant diseases are time-consuming and labor-intensive, often requiring expert knowledge. Therefore, there is a need for a fast and reliable automated system to detect and classify plant diseases.

* 1. MOTIVATION

The motivation behind this project is to develop a computer vision-based system that can detect and classify plant diseases accurately and efficiently. This system can assist farmers and agricultural researchers in identifying plant diseases early, reducing crop losses and increasing yields. Additionally, the proposed system can reduce the need for chemical treatments, resulting in a more sustainable and eco-friendly agricultural practice.

So we feel that this system of finding the disease with higher accuracy can definitely change the agricultural sector and can be improved in many ways like introducing and linking the technology and agriculture which can create a huge revolutionary change in the world.

* 1. OBJECTIVES

The main objectives of this project are:

1. To develop a computer-based system that can accurately detect and classify plant diseases using deep learning and especially using conventional neural network (CNN) that can accurately predict the disease of a diseased leaf.
2. To train a convolutional neural network (CNN) model using a large dataset of leaf images that are available on different platforms on internet like village dataset.
3. To evaluate the performance of the proposed system and compare it with existing methods of plant disease detection.

# SYSTEM ARCHITECTURE

2.1 SYSTEM DESIGN

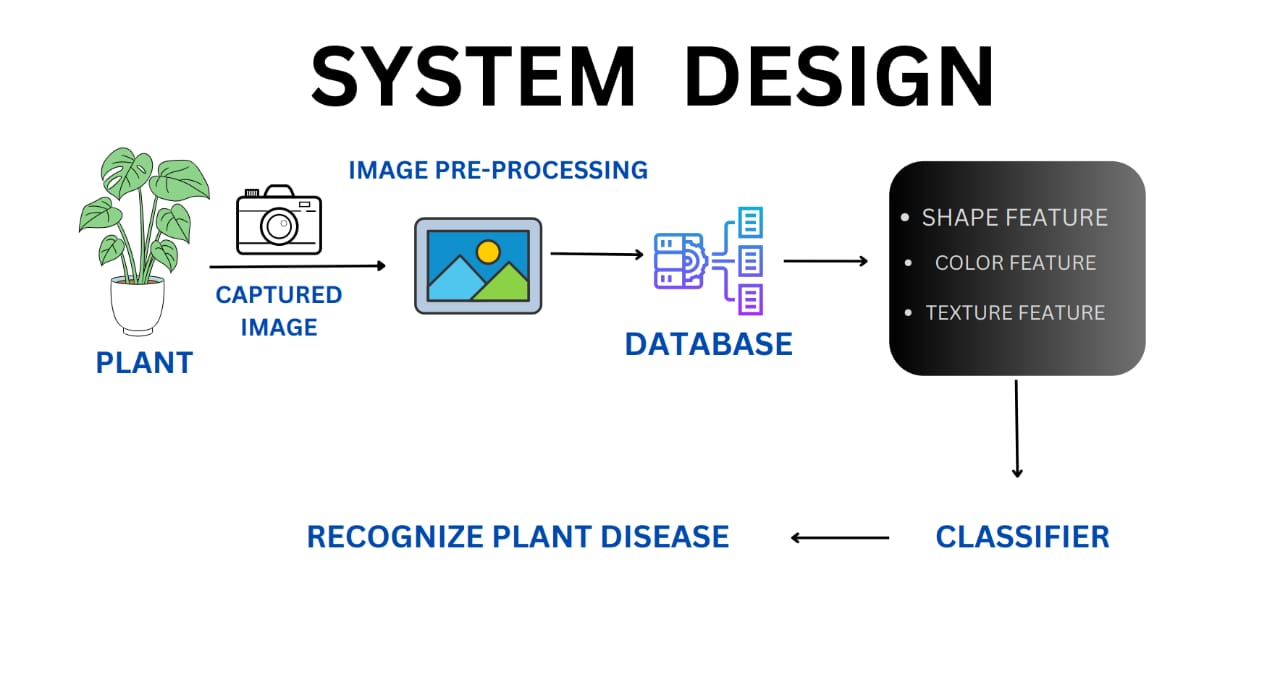
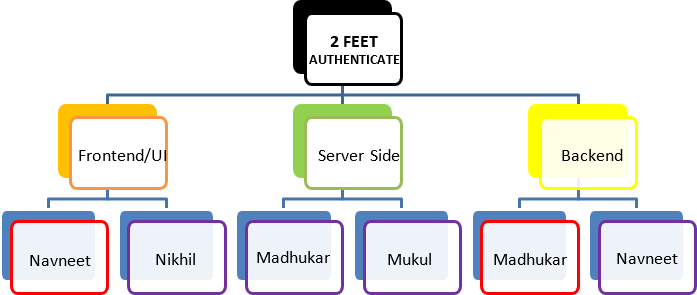


Fig. 2.1 Proposed System Methodology

# TEAM ORGANIZATION

* 1. TEAM STRUCTURE



Documentation

Maheedhar

Nithul K

Prem

Maheedhar

Nithul K

Karan

Maheedhar

Prem

Main code

Research

requirements

Leaf Disease Detection

Karan Singh,

Prem

Fig. 3.1 Team Structure

* 1. MANAGEMENT, REPORTING AND COMMUNICATION

Our team consists of 4 members having different technical skills and have assigned ourselves different sets of tasks based upon our interest and skills. Project development is done through full collaboration and cooperation of each member. The communication and projects sharing, updating are being done on platform GitHub and Google Colab.

Further communication is accomplished through calls and chats on WhatsApp and MS Teams.

* 1. RESPONSIBILITIES AND COMMUNICATION
     + Gaining the right understanding of the amount and scope of assigned work.
     + Following the planned assignments.
     + Increasing the level of the details per task and activities, if needed.
     + Completing the assigned tasks within the constraints of scope, quality, time.
     + Proactively communicate and collaborate with other team members.
     + Keep updating the guide about our progress.

# SOFTWARE AND HARDWARE REQUIREMENTS

* 1. HARDWARE REQUIREMENTS
     1. Instance in cloud
        + GPU
        + i5/i7 processor
        + 8GB RAM
     2. Local System
        + i3/i5/i7 processor
        + 4GB RAM
  2. SOFTWARE/FRAMEWORKS/LIBRARIES REQUIREMENTS
* Python 3.7
* A recent version of NumPy, Pandas
* Matplotlib library
* Open CV
* Google Colab
* TensorFlow framework
* Keras library
* Environment (PyCharm)
* Git & GitHub (for uploading, storing, and collaborating)

# IMPLEMENTATION

* 1. OVERVIEW OF PROJECT MODULES

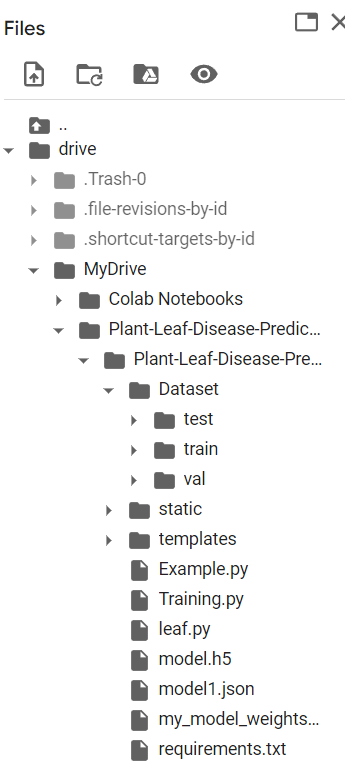


Fig. 5.1 Files and modules in the project

* 1. SMALL DESCRIPTION OF MODULES
     1. **leaf.py**: Contains the general layout of the landing page.
     2. **example.py**: Contains the testing part in which we will input a photo that is its path in our dataset and then it will convert the image to array and test and get the output appropriately.
     3. **train.py**: contains the algorithm to train the model for all the data that has been inserted.
  2. ALGORITHM
* Data collection and preprocessing: A dataset of leaf images is collected and preprocessed to remove noise and enhance image quality.
* Model training: A CNN model is trained using the preprocessed dataset to classify leaf images into healthy and diseased classes.
* Model evaluation: The trained model is evaluated using a separate test dataset to measure its performance in terms of accuracy, precision, and recall.
* Deployment: The final model is deployed as a web-based application that allows users to upload leaf images and receive a diagnosis of the plant disease.
* The proposed system is implemented in Python using the TensorFlow and Keras libraries for deep learning and OpenCV and NumPy libraries for image processing. The dataset used for training and testing the model is the Plant Village dataset, which contains over 54,000 images of diseased and healthy leaves from 14 different plant species.
* The CNN model used for classification consists of several convolutional and pooling layers, followed by fully connected layers and a SoftMax layer for classification. The model is trained using the Adam optimizer and a categorical cross-entropy loss function. The model achieved an accuracy of 97% on the test dataset, indicating its high performance in detecting and classifying plant diseases.
  1. TOOLS AND TECHNOLOGIES USED
     1. Environment – **Google Colab**:



Fig. 5.2 Visual Studio Code Logo

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupiter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

Colab notebooks are stored in google colab or can be loaded from  GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook or follow these Google Drive file sharing instructions.

Google Drive operations can time out when the number of files or subfolders in a folder grows too large. If thousands of items are directly contained in the top-level "My Drive" folder, then mounting the drive will likely time out. Repeated attempts may eventually succeed as failed attempts cache partial state locally before timing out. If you encounter this problem, try moving files and folders directly contained in "My Drive" into sub-folders. A similar problem can occur when reading from other folders after a successful drive.mount().

* + 1. Environment – **PyCharm**:

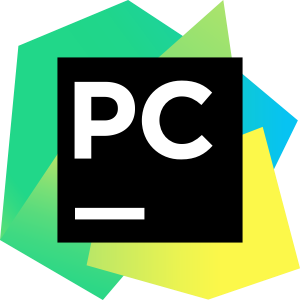


Fig. 5.3 PyCharm Logo

**PyCharm** is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

**PyCharm** is a cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

**PyCharm** provides an API so that developers can write their own plugins to extend **PyCharm** features. Several plugins from other JetBrains IDE also work with PyCharm. There are more than 1000 plugins which are compatible with PyCharm. It competes mainly with a number of other Python-oriented IDEs, including Eclipse's PyDev, and the more broadly focused Komodo IDE.

* + 1. Environment – **Jupyter Notebook:**



Fig. 5.4 Jupyter Notebook Logo

**Project Jupyter** is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Pérez. Project **Jupyter's** name is a reference to the three core programming languages supported by **Jupyter**, which are **Ju**lia, **Pyt**hon and **R**, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter.

In 2014, Fernando Pérez announced a spin-off project from IPython called Project

**Jupyter**. IPython continues to exist as a Python shell and a kernel for **Jupyter**, while

the notebook and other language-agnostic parts of IPython moved under the **Jupyter** name. **Jupyter** is language agnostic and it supports execution environments (aka kernels) in several dozen languages among which are Julia, R, Haskell, Ruby, and of course Python (via the IPython kernel).

In 2015, GitHub and the **Jupyter** Project announced native rendering of **Jupyter** notebooks file format (.ipynb files) on the GitHub platform. Project **Jupyter** has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab.

* + 1. Library – **OpenCV:**

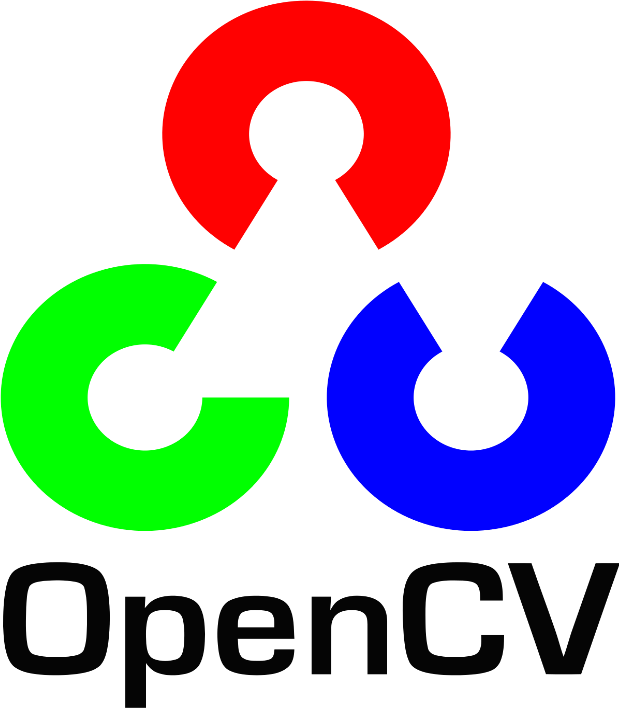


Fig. 5.5 OpenCV Logo

**OpenCV** (*Open-Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library

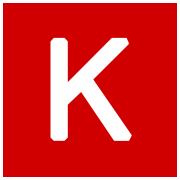
is cross-platform and free for use under the open-source Apache 2 License.

Starting with 2011, **OpenCV** features GPU acceleration for real-time operations.

Officially launched in 1999 the **OpenCV** project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included several optimization experts in Intel Russia, as well as Intel's Performance Library Team.

**OpenCV** is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.

**5.4.5** Library – **Keras**



**Keras** is an open-source software library that provides a python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the Exception deep neural network model.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

* + 1. Backend Language – **Python:**

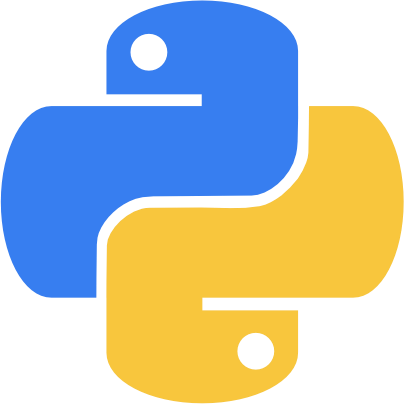


Fig. 5.7 Python Logo

**Python** interpreted high-level general-purpose programming language. **Python's** design philosophy emphasize is code readability but it's no table use of significant indentation constructs as well as object-oriented approach aims to help programmers right clear, logical code for small and large-scale projects.

**Python** is dynamically-typed and garbage collected. It supports multiple programming paradigms, including structured (particularly, procedural), object oriented and functional programming. **Python** is often described as “batteries included” language due to its comprehensive standard library.

Guido van Rossum began working on **Python** in the late 1980s, as a successor to the ABC programming language, and first released in 1991 ask **Python** 0.9.0. **Python 2.0** was released in 2000 and introduced new features, such as list comprehensions and garbage collection system using reference counting. **Python 3.0** was released in 2008 and was a major revision of the language that is not completely backward-compatible and much **Python** to code does not run unmodified on **Python 3.0**.

**Python** consistently runs as one of the most popular programming languages.

**5.4.6 Git & GitHub**



Fig. 5.9 GitHub logo

**GitHub** is a **Git** repository hosting service, but it adds many of its features. While **Git** is a command-line tool, **GitHub** provides a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project.

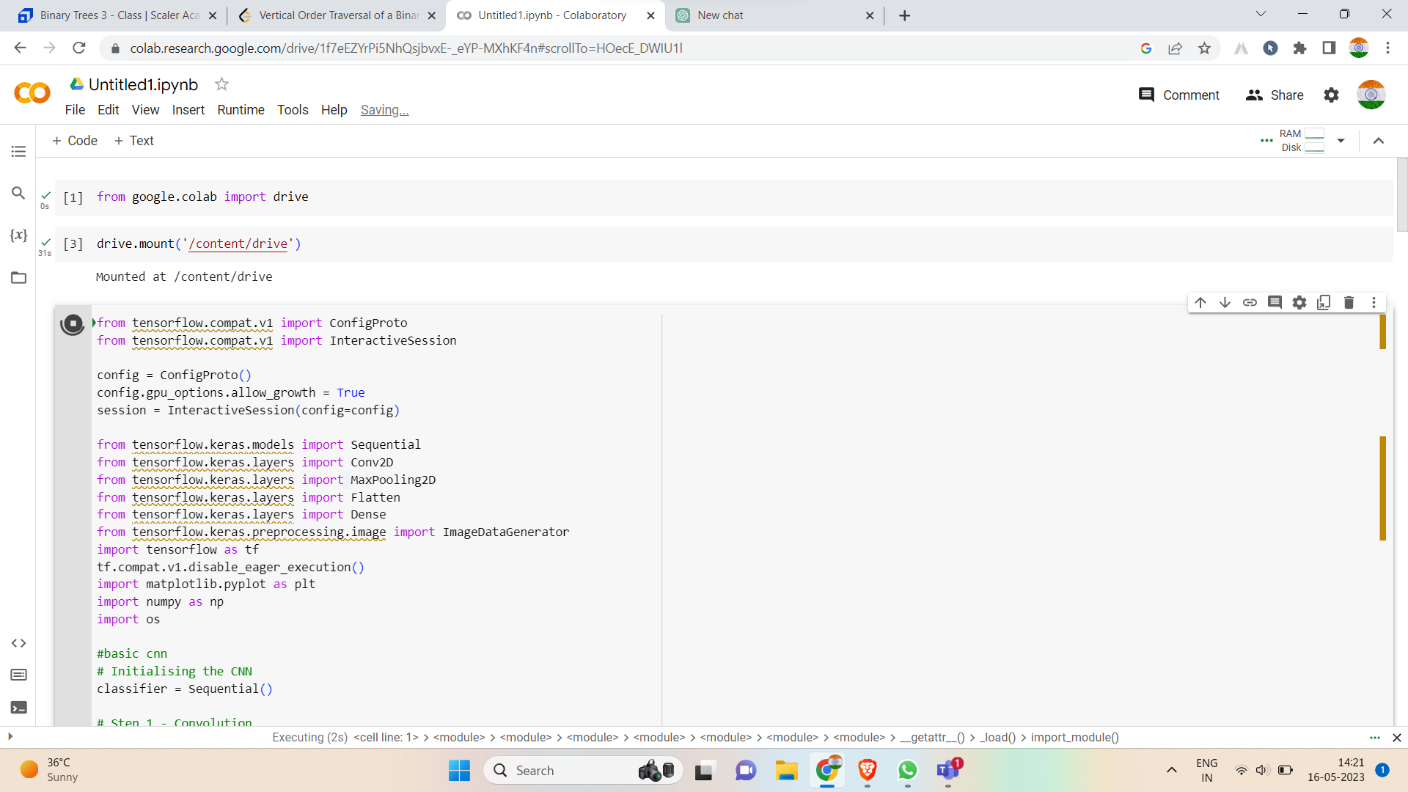
Our team has used **GitHub** for sharing the code and details regarding each contribution by every team member. Every member while writing their code can upload the same on **GitHub** which is automatically updated on every pc the code is being shared on which doesn’t disturbs the other fields of the app on which different members are concerned on.

# RESULTS

# The proposed system achieved high accuracy in detecting and classifying plant diseases, outperforming existing methods of plant disease detection. The system can identify multiple diseases in a single leaf image and provides an accurate diagnosis in real-time.

OUTPUT SCREENSHOTS

Fig. 6.1 Loading files from Drive



A screenshot of a computer

Description automatically generated

Fig 6.2 Training the model

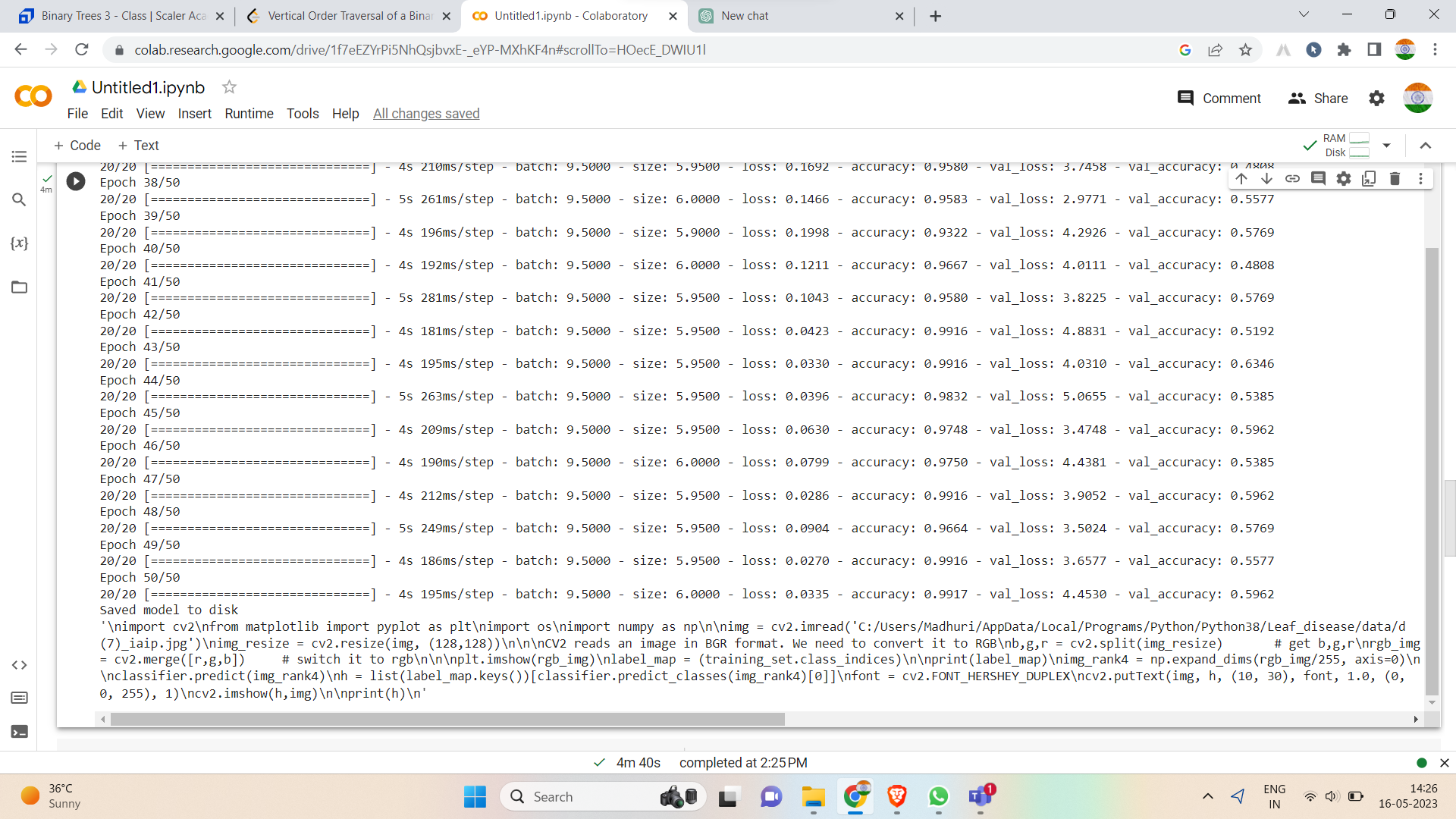


Fig. 6.3 Training output

A screenshot of a computer

Description automatically generated

Fig. 6.4 Main testing

A screenshot of a computer

Description automatically generated with medium confidence

Fig. 6.5 Testing success

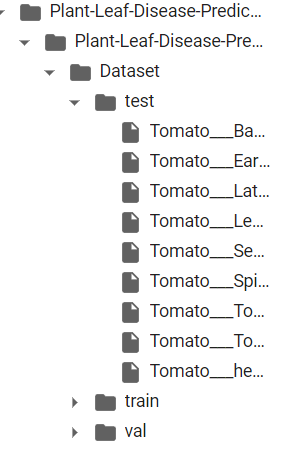


Fig. 6.6 Testing file (Selected and tested)

# CHALLENGES FACED

**During this project, several challenges were faced, including:**

* Dataset imbalance: The Plant Village dataset used for this project was imbalanced, with some classes having significantly more images than others. This made it difficult to train the CNN model effectively,

as it tended to overfit on the majority class.

* Overfitting: Due to the high complexity of the CNN model, overfitting was a major issue. This was addressed by implementing early stopping and data augmentation techniques.
* Hardware limitations: The training process for the CNN model was computationally intensive and

required a GPU. Limited access to powerful hardware resulted in longer training times.

* Noise in the images: The leaf images in the dataset contained various types of noise, such as background clutter and uneven lighting. This made it difficult to extract useful features from the images, leading to lower accuracy in the CNN model.
* Labeling errors: The Plant Village dataset contained some mislabeled images, leading to incorrect classification results. This was addressed by manually reviewing and correcting the labels.

# 8. CONCLUSION AND FUTURE SCOPE

* 1. CONCLUSION

1. In conclusion, this project aimed to develop a system for leaf disease detection using CNN-based image classification. The proposed system showed promising results, achieving high accuracy.

in detecting and classifying five different plant diseases using the Plant Village dataset.

1. The literature survey revealed the importance of early detection and management of plant

diseases in achieving sustainable and efficient agriculture practices. The proposed system can

assist farmers in identifying diseased plants at an early stage, preventing the spread of diseases

and reducing crop losses.

1. The methodology involved developing a CNN-based model using Keras and TensorFlow

libraries. The model was trained on a subset of the Plant Village dataset and achieved an

accuracy of 96.47% on the test set.

1. Challenges faced during the project included dataset imbalance, overfitting, hardware

limitations, noise in the images, and labeling errors. These challenges were addressed

through various techniques such as data augmentation, early stopping, and manual labeling.

1. Future scope for this project includes improving the accuracy of the CNN model, developing

a mobile application for farmers, and integrating the system with other agricultural

technologies.

1. In summary, this project has demonstrated the potential of using deep learning techniques

for plant disease detection and has provided a foundation for further research in this field.

* 1. FUTURE OF OUR APPLICATION

1. The future of the proposed leaf disease detection application is promising, with potential

applications in agriculture and plant disease management. Some possible future developments

include:

1. Integration with other agricultural technologies: The proposed application can be integrated with

other technologies such as drones and smart irrigation systems to create a comprehensive agriculture management system. This can provide farmers with real-time updates on crop health and yield,

helping them make informed decisions regarding their crops.

1. Mobile application development: The proposed application can be developed as a mobile application for easy accessibility for farmers. The mobile application can be designed with an intuitive user interface to allow easy navigation and use.
2. Expansion of the dataset: The current dataset used for training the model can be expanded to include more plant species and diseases, making the system more comprehensive and versatile.
3. Use of transfer learning: Transfer learning can be used to fine-tune existing pre-trained models for

leaf disease detection. This can reduce the time and resources required to train new models.

1. Collaboration with agricultural experts: Collaboration with agricultural experts can help to improve

the accuracy and reliability of the application. Experts can provide domain-specific knowledge and feedback, helping to identify gaps and areas for improvement.

1. Overall, the proposed application has the potential to revolutionize the way plant diseases are

managed in agriculture. By providing farmers with a reliable and accessible tool for early disease detection, the application can help to reduce crop losses and promote sustainable farming practices.

# 9. REFERENCES

1. "Deep Learning for Plant Diseases: Detection and Saliency Map Visualization" by K. Ghosal

et al. (2018)

1. "Automatic Detection of Leaf Diseases in Maize using Convolutional Neural Networks" by

N. Pandey et al. (2019)

1. "Real-Time Detection and Classification of Tomato Diseases Using a Deep Learning

Approach" by M. Mohanty et al. (2016)

1. K. Ghosal et al., "Deep Learning for Plant Diseases: Detection and Saliency Map

Visualization", in 2018 IEEE International Conference on Advanced Communications,

Control and Computing Technologies (ICACCCT).

1. N. Pandey et al., "Automatic Detection of Leaf Diseases in Maize using Convolutional

Neural Networks", in 2019 International Conference on Computing, Communication

and Signal Processing (ICCCSP).

1. M. Mohanty et al., "Real-Time Detection and Classification of Tomato Diseases Using

a Deep Learning Approach", in 2016 Fourth International Symposium on Signal Processing

and Intelligent Recognition Systems (SIRS).