**Egyptian E-Learning University**

Faculty of Computers & Information Technology

# Rice Detection System

**By**

|  |  |
| --- | --- |
| Name: Basel Emad Osman Hassan | ID: 2100555 |
| Name: Abdalla Tarek Abdelwahed Osman | ID: 2100243 |
| Name: Reham Ahmed Thabet Essa | ID: 2101496 |
| Name: Magy Emil Fawzy Awad | ID: 2100901 |
| Name: Tesbina Monir Shehata Soliman | ID: 2101667 |
| Name: Alaa younis elsayed ahmed | ID: 2100444 |
| Name: Nourhan Elsayed Abdelkareem | ID: 2101383 |

Supervised by

**Dr. Amira Idrees**

Assistant

**Eng. Yousif Ayman**

[Sohag]-2025

Contents

[Abstract 4](#_Toc200548392)

[Acknowledgments 5](#_Toc200548393)

[Introduction 6](#_Toc200548394)

[Literature Review / Related Work 10](#_Toc200548395)

[Proposed system 18](#_Toc200548396)

[Implementation 24](#_Toc200548397)

[Testing & Evaluation 55](#_Toc200548398)

[Results & Discussion 58](#_Toc200548399)

[Conclusion & Future Work 62](#_Toc200548400)

[References 68](#_Toc200548401)

[**Figure 1:libraries** 30](#_Toc200548450)

[**Figure 2:Hyperparameters** 31](#_Toc200548451)

[**Figure 3:load dataset** 31](#_Toc200548452)

[**Figure 4:split dataset** 32](#_Toc200548453)

[**Figure 5:labels** 32](#_Toc200548454)

[**Figure 6:visualizes images** 33](#_Toc200548455)

[**Figure 7:Normalize dataset** 33](#_Toc200548456)

[**Figure 8:Augmentation block** 34](#_Toc200548457)

[**Figure 9:CCN Architecture** 35](#_Toc200548458)

[**Figure 10:compile the model** 35](#_Toc200548459)

[**Figure 11:callbacks** 36](#_Toc200548460)

[**Figure 12:Train the model** 36](#_Toc200548461)

[**Figure 13:Evaluate the model** 37](#_Toc200548462)

[**Figure 14:summary** 37](#_Toc200548463)

[**Figure 15:Visual Diagram** 38](#_Toc200548464)

[**Figure 16:plot Training history** 38](#_Toc200548465)

[**Figure 17:visualize test dataset** 39](#_Toc200548466)

[**Figure 18:Evaluates model prediction** 40](#_Toc200548467)

[**Figure 19 :chatBot** 41](#_Toc200548468)

[**Figure 20:Login Page** 42](#_Toc200548469)

[**Figure 21:Home Page** 44](#_Toc200548470)

[**Figure 22:Upload Image Page** 46](#_Toc200548471)

[**Figure 23:Rice Plant Diseases Page** 47](#_Toc200548472)

[**Figure 24:Chat Bot Page** 48](#_Toc200548473)

[**Figure 25:Community Page** 49](#_Toc200548474)

[**Figure 26:Weather Forecast Page** 50](#_Toc200548475)

[**Figure 27:reports** 51](#_Toc200548476)

Abstract

This project proposes an AI-powered mobile application for early detection of rice plant diseases to enhance agricultural productivity and food security. Rice, a staple crop for over half the global population, is vulnerable to diseases like bacterial blight, brown spot, and rice blast, which can cause yield losses of 30-40%. Traditional diagnostic methods are manual, time-consuming, and require expertise, often delaying critical interventions. To address this, we developed a deep learning-based solution using Convolutional Neural Networks (CNNs) trained on a dataset of 3,829 annotated rice leaf images from Kaggle, covering multiple disease classes.

The methodology involved data preprocessing, model training with architectures like ResNet and MobileNet, and optimization for mobile deployment. The final model achieved 96.61% accuracy , surpassing the initial target of 90-95%.

Key features of the application include real-time image analysis, Farmer Communication Platform , a diagnostic chatbot, and weather-based disease alerts.

The results demonstrate that this tool can significantly reduce crop losses by enabling timely treatment, while minimizing unnecessary pesticide use. Its scalability potential extends to other crops, aligning with global sustainable agriculture goals. This project bridges the gap between AI research and practical farming needs, offering an accessible, cost-effective solution for small-scale farmers.

Acknowledgments

First, we would like to thank Allah for helping us to complete this project successfully. We are heartily thankful to Dr. Amira Idrees and TA. Yousef Ayman who not only served as our supervisors, but also guided, encouraged and challenged us throughout the project, and guided us with great dedication, never accepting less than our best efforts, and allowing for the completion and success of this project.

We also owe our deepest gratitude to our families for their great support, which without, our work would not have been successful. We also want to thank our TA. Yousef Ayman because it was very helpful to take care of all the details of the project. We also want to publish Dr. Amira Idrees because she helped us with the design of the project. We want to thank our TA. Yousef Ayman for his guidance and inspiring remarks, his support also for his presenting the study and receiving feedback from the research community. We also thank him for sharing his project with us so that we can benefit from it and take it as a reference for us. In the end, we also thank God for granting us success and honoring us.

Chapter 1

Introduction

* 1. **Introduction**

Rice is a staple food for over half of the world’s population, particularly in Asia and Africa. However, rice crops are highly susceptible to diseases that can drastically reduce yield and threaten food security. Traditional methods of disease detection are manual, time-consuming, and often require expert knowledge, leading to delays in diagnosis and treatment. This project proposes an AI-powered mobile application to revolutionize rice plant disease detection by providing accurate, real-time analysis and actionable insights for farmers.

* 1. **Background and Motivation for the Project**

The reliance on rice as a primary food source makes its production critical for global food security. Diseases such as bacterial blight, brown spot, and rice blast can cause yield losses of 30-40% if not detected and managed early. Current diagnostic methods are inefficient and inaccessible to many small-scale farmers. The motivation behind this project is to leverage artificial intelligence and machine learning to democratize access to advanced agricultural tools, enabling timely interventions and sustainable farming practices.

* 1. **Importance of the Problem Being Addressed**

The economic and social impact of rice crop failures is profound, especially in developing regions where agriculture is a primary livelihood. Early disease detection can mitigate losses, reduce unnecessary pesticide use, and improve crop yields. By addressing this problem, the project aims to enhance food security, support farmer livelihoods, and contribute to the broader goal of sustainable agriculture.

* 1. **Problem Statement**

- Clear Definition of the Problem: Manual disease detection methods are slow, error-prone, and dependent on expert knowledge, leading to delayed responses and significant crop losses.

- Justification: The lack of accessible, automated solutions exacerbates food insecurity and economic instability for rice-dependent communities. An AI-driven tool can bridge this gap by providing instant, accurate diagnoses.

* 1. **Objectives**

- Main Objective: Develop a mobile AI application to accurately predict rice plant diseases using image recognition and machine learning.

- Specific Objectives:

1. Train a deep learning model (e.g., CNN) on a diverse dataset of rice disease images.

2. Achieve a target accuracy of 90-95% in disease classification.

3. Integrate the model into a user-friendly mobile app.

4. Provide actionable recommendations for disease management and prevention.

* 1. **Brief Overview of the Proposed Solution**

The solution involves a convolutional neural network (CNN) trained on a dataset of 3,829 rice leaf images, covering diseases like bacterial blight and brown spot. Farmers can upload crop images via a mobile app to receive instant diagnoses and treatment advice. Additional features include a chatbot, weather-based alerts, and a farmer forum to foster community support. The project’s scalable framework has potential applications for other crops, aligning with global food security goals.

Chapter 2

Literature Review / Related Work

**4.1: Convolutional neural network in rice disease recognition**

➢ Year: Published 31 October 2023

There are many rice diseases, which have very serious negative effects on rice growth and final yield. It is very important to identify the categories of rice diseases and control them. In the past, the identification of rice disease types was completely dependent on manual work, which required a high level of human experience. But the method often could not achieve the desired effect, and was difficult to popularize on a large scale. Convolutional neural networks are good at extracting localized features from input data, converting low-level shape and texture features into high-level semantic features.

Algorithms Used : MobileNet, CNN.

* **MobileNet** is a family of [convolutional neural network](https://en.wikipedia.org/wiki/Convolutional_neural_network) (CNN) architectures designed for [image classification](https://en.wikipedia.org/wiki/Image_classification), [object detection](https://en.wikipedia.org/wiki/Object_detection), and other computer vision tasks.
* A **convolutional neural network** (**CNN**) is a type of [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) that learns [features](https://en.wikipedia.org/wiki/Feature_engineering) via [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or kernel) optimization. This type of [deep learning](https://en.wikipedia.org/wiki/Deep_learning) network has been applied to process and make [predictions](https://en.wikipedia.org/wiki/Prediction) from many different types of data including text, images and audio.

➢ Highest Accuracy: 98.7%

➢ Dataset Used: Public data on the Internet and photos taken by the authors in the field, and there were 2400 pictures in the dataset.

**4.2: Hyperspectral Imaging Combined With Deep Transfer Learning for Rice Disease Detection**

➢Year: Published 28 September 2021.

Various rice diseases threaten the growth of rice. It is of great importance to achieve the rapid and accurate detection of rice diseases for precise disease prevention and control. Hyperspectral imaging (HSI) was performed to detect rice leaf diseases in four different varieties of rice. Considering that it costs much time and energy to develop a classifier for each variety of rice, deep transfer learning was firstly introduced to rice disease detection across different rice varieties. Three deep transfer learning methods were adapted for 12 transfer tasks, namely, fine-tuning, deep CORrelation ALignment (CORAL), and deep domain confusion (DDC). A self-designed convolutional neural network (CNN) was set as the basic network of the deep transfer learning methods.

➢ Algorithms Used: CNN.

* A **convolutional neural network** (**CNN**) is a type of [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) that learns [features](https://en.wikipedia.org/wiki/Feature_engineering) via [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or kernel) optimization. This type of [deep learning](https://en.wikipedia.org/wiki/Deep_learning) network has been applied to process and make [predictions](https://en.wikipedia.org/wiki/Prediction) from many different types of data including text, images and audio.

➢ Highest Accuracy: 80%

➢ Dataset Used: Images of rice leaves with diseases.

4.3: **Disease Detection in Rice leaves using CNN**

➢Year: Published 17 June 2023.

Among the most essential crops for human consumption is rice, and leaf diseases can significantly affect yield and quality. Almost half of the world’s population eats this cereal grain. The identification of rice leaf diseases is crucial for the economy and food security. Rapid corrective measures could be anticipated by understanding the disorders through their peculiar characteristics. The majority of the technical issues with image recognition and classification have been solved using deep learning-based CNN techniques. This study aims to identify the optimal Convolutional Neural Network (CNN) architecture for detecting diseases on rice leaves, considering accuracy, recall, and precision as evaluation metrics. By accurately identifying diseases, the proposed procedure provides valuable assistance to farmers in ensuring healthy crop production.

➢ Algorithms Used: CNN.

* A **convolutional neural network** (**CNN**) is a type of [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) that learns [features](https://en.wikipedia.org/wiki/Feature_engineering) via [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or kernel) optimization. This type of [deep learning](https://en.wikipedia.org/wiki/Deep_learning) network has been applied to process and make [predictions](https://en.wikipedia.org/wiki/Prediction) from many different types of data including text, images and audio.

➢ Highest Accuracy: 75%

➢ Dataset Used: Diseases on rice leaves.

**4.4: A Deep Learning and SVM Hybrid Model for Rice**

➢Year: **30 July 2024.**

Agriculture plays a vital role in Bangladesh’s economy. It is essential to ensure the proper growth and health of crops for the development of the agricultural sector. In the context of Bangladesh, crop diseases pose a significant threat to agricultural output and, consequently, food security. This necessitates the timely and precise identification of such diseases to ensure the sustainability of food production. This study focuses on building a hybrid deep learning model for the identification of three specific diseases affecting three major crops: late blight in potatoes, brown spot in rice, and common rust in corn. The proposed model leverages EfficientNetB0′s feature extraction capabilities, known for achieving rapid high learning rates, coupled with the classification proficiency of SVMs, a well-established machine learning algorithm.

➢ Algorithms Used : SVM, CNN,**c**[**lassification**](https://www.mdpi.com/search?q=classification), [**feature extraction**](https://www.mdpi.com/search?q=feature+extraction)**.**

* Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. While it can handle regression problems, SVM is particularly well-suited for classification tasks.
* A **convolutional neural network** (**CNN**) is a type of [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) that learns [features](https://en.wikipedia.org/wiki/Feature_engineering) via [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or kernel) optimization.
* A Classification Algorithm determines the category to which a set of data belongs (typically faulty, fault type or healthy categories).
* Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set.

➢ Highest Accuracy: 97.29%

➢ Dataset Used: Public rice leaf image datasets.

**4.5: An Overview of Chatbot Technology**

➢Year: 6 May 2020

The use of chatbots evolved rapidly in numerous fields in recent years, including Marketing, Supporting Systems, Education, Health Care, Cultural Heritage, and Entertainment. In this paper, we first present a historical overview of the evolution of the international community’s interest in chatbots. Next, we discuss the motivations that drive the use of chatbots, and we clarify chatbots’ usefulness in a variety of areas. Moreover, we highlight the impact of social stereotypes on chatbots design. After clarifying necessary technological concepts, we move on to a chatbot classification based on various criteria, such as the area of knowledge they refer to, the need they serve and others. Furthermore, we present the general architecture of modern chatbots while also mentioning the main platforms for their creation. Our engagement with the subject so far, reassures us of the prospects of chatbots and encourages us to study them in greater extent and depth.

➢ Algorithms Used: AIML , ChatScript , RiveScript .

* AIML (Artificial Intelligence Markup Language(is an XML-based language designed for creating conversational patterns. It extends pattern matching by allowing more structured and flexible responses, often used in conjunction with Latent Semantic Analysis (LSA) to handle more complex queries.
* ChatScript An advanced scripting language that offers dynamic conversation capabilities. It supports long-term memory and emotional context, allowing chatbots to remember user information and adjust responses based on user sentiment.
* RiveScript A lightweight scripting language for creating chatbots, known for its simplicity and ease of integration with various programming languages.

**4.6: Study on the Classification Method of Rice Leaf Blast Levels**

➢Year :05 May 2022

Leaf blast is a disease of rice leaves caused by the Pyricularia oryzae. It is considered a significant disease is affecting rice yield and quality and causing economic losses to food worldwide. Early detection of rice leaf blast is essential for early intervention and limiting the spread of the disease. To quickly and non-destructively classify rice leaf blast levels for accurate leaf blast detection and timely control. This study used hyperspectral imaging technology to obtain

hyperspectral image data of rice leaves. The descending dimension methods got rice leaf disease characteristics of different disease classes, and the disease characteristics obtained by screening were used as model inputs to construct a model for early detection of leaf blast disease.

➢ Algorithms Used: CNN, VGG16, ResNet50, MobileNet, EfficientNet.

* A **convolutional neural network** (**CNN**) is a type of [feedforward neural network](https://en.wikipedia.org/wiki/Feedforward_neural_network) that learns [features](https://en.wikipedia.org/wiki/Feature_engineering) via [filter](https://en.wikipedia.org/wiki/Filter_(signal_processing)) (or kernel) optimization.
* **MobileNet** is a family of [convolutional neural network](https://en.wikipedia.org/wiki/Convolutional_neural_network) (CNN) architectures designed for [image classification](https://en.wikipedia.org/wiki/Image_classification), [object detection](https://en.wikipedia.org/wiki/Object_detection), and other computer vision tasks.
* **EfficientNet** is a family of convolutional neural networks (CNNs) developed by Google AI in 2019. It is known for achieving state-of-the-art accuracy on image classification tasks while being significantly more efficient than previous models.
* VGG16 is a convolutional neural network (CNN) architecture proposed by the Visual Geometry Group at the University of Oxford.

➢ Highest Accuracy:88.46%.

➢ Dataset Used: Rice leaves

Chapter 3

Proposed system

The proposed system utilizes a deep learning-based approach to classify rice leaf diseases. The main steps involved are as follows:

1. Dataset Preparation

Images of rice leaves are loaded from a structured dataset directory. The dataset is divided into three subsets:

- Training set: Used to train the model.

- Validation set: Used to monitor performance during training and prevent overfitting.

- Testing set: Used to evaluate the final model’s generalization capability.

2. Preprocessing and Augmentation

To enhance model performance, the following preprocessing and augmentation techniques are applied:

- Resizing: Images are resized to 128×128 pixels to ensure uniformity.

- Normalization: Pixel values are scaled to a range of [0, 1] to improve training stability.

- Data Augmentation: Techniques such as the following are used to increase dataset variability and prevent overfitting:

- Random flipping (horizontal and vertical).

- Rotation to simulate different orientations of leaves.

- Zooming to capture different scales of disease symptoms.

- Contrast adjustment to improve robustness against lighting variations.

3. Model Architecture

A custom Convolutional Neural Network (CNN) is designed with the following layers:

- Convolutional Layers: Multiple layers with ReLU activation to extract hierarchical features.

- Max-Pooling Layers: Used for downsampling and reducing computational complexity.

- Flattening Layer: Converts 2D feature maps into a 1D vector for classification.

- Fully Connected (Dense) Layers: Includes dropout to prevent overfitting.

- Output Layer: Uses softmax activation for multi-class classification.

4. Training

The model is trained with the following configurations:

- Optimizer: Adam optimizer for adaptive learning rate adjustment.

- Loss Function: Sparse categorical crossentropy to handle multi-class classification.

- Callbacks:

- EarlyStopping: Monitors validation loss and stops training if no improvement is detected.

- ReduceLROnPlateau: Reduces learning rate when validation loss stagnates to fine-tune model weights.

5. Evaluation

The trained model is evaluated using:

- Accuracy: Measures the proportion of correct predictions over the total test samples.

- Confusion Matrix: Generated using Scikit-learn to visualize classification performance across different disease categories.

Algorithms or Frameworks Used:

Algorithms or Frameworks Used:

1. CNN (Convolutional Neural Network):

- A specialized deep learning architecture designed for processing structured grid data like images.

- Uses convolutional layers with learnable filters that automatically extract hierarchical features (edges → textures → patterns → object parts).

- Incorporates ReLU activation for non-linearity and max-pooling layers for spatial downsampling.

- Final architecture includes flattening and dense layers for classification with softmax activation.

2. Data Augmentation Techniques:

- Random Flipping:Horizontally and vertically flips images to increase orientation variability (applied with 50% probability).

- Rotation: Rotates images randomly within ±30 degrees to simulate different leaf angles.

- Zooming: Applies random zoom (10–20% scale variation) to capture disease features at different magnifications.

- Contrast Adjustment: Modifies image contrast (±30% range) to improve robustness to lighting conditions.

3. Loss Function (Sparse Categorical Crossentropy):

- Measures the discrepancy between predicted probabilities and true class labels.

- Suitable for multi-class classification with mutually exclusive classes (e.g., disease types).

- Computes loss directly from integer labels without one-hot encoding, optimizing memory usage.

4. Optimizer (Adam):

- Adaptive Moment Estimation optimizer combines RMSProp and momentum concepts.

- Automatically adjusts learning rates per parameter using moving averages of gradients (β₁=0.9, β₂=0.999).

- Default learning rate of 0.001 with decay mechanisms via ReduceLROnPlateau callback.

Frameworks & Libraries:

1. TensorFlow:

- End-to-end deep learning framework used for model construction (Keras API), training, and inference.

- Provides GPU acceleration via CUDA/cuDNN for efficient convolutional operations.

- Includes utilities for dataset loading (tf.data), image augmentation (ImageDataGenerator), and callbacks.

2. NumPy:

- Fundamental package for numerical computing in Python.

- Handles array operations for image data manipulation (reshaping, normalization).

- Interfaces with TensorFlow tensors for preprocessing pipelines.

3. Pandas:

- Used for structured dataset organization (e.g., CSV metadata linking images to labels).

- Facilitates train/validation/test splits via stratified sampling methods.

4. Matplotlib:

- Generates visualizations of training curves (accuracy/loss vs. epochs).

- Plots confusion matrices and sample augmented images for qualitative analysis.

5. Scikit-learn:

- Provides metrics (accuracy\_score, classification\_report) for model evaluation.

- Functions for confusion matrix calculation (confusion\_matrix) and normalization.

- Optional integration for SVM-based hybrid models (though not used in final pipeline).

Chapter 4

Implementation

* 1. **Technologies, Tools, and Programming Languages Used**

**4.1.1 Programming Languages:**

**Python**

Python is a high-level, interpreted programming language widely used in data science and machine learning. In this project, Python was used to develop and train the machine learning model, handle data preprocessing, visualize results, and manage the backend logic. Its simplicity and extensive ecosystem of libraries make it ideal for rapid prototyping and scientific computing.

**Dart**

Dart is a client-optimized programming language developed by Google. It is mainly used for building mobile applications with the Flutter framework. In this project, Dart was used to develop the frontend of the mobile application, allowing the creation of a responsive and cross-platform user interface for Android.

**4.1.2 Tools and Development Environments:**

**Jupyter Notebook**

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It was used for developing and testing the machine

learning model in Python. Its interactive environment facilitated the step-by-step execution and visualization of results.

**Android Studio**

Android Studio is the official integrated development environment (IDE) for Android application development. It was used for writing and testing the Dart code (via Flutter), building the mobile app, debugging it, and running it on an emulator or a real device.

**Figma**

Figma is a cloud-based design tool used for creating UI/UX prototypes and high-fidelity designs. In this project, Figma was used to design the user interface of the mobile app before the implementation phase. It helped visualize the app's layout, colors, icons, and user flow.

**Postman**

Postman is a popular API development and testing platform that simplifies working with web services. It provides an intuitive interface for sending requests to APIs, inspecting responses, and automating workflows without requiring deep technical knowledge. Developers use Postman to test API endpoints, debug integrations, document services, and collaborate on API projects. The tool supports various request types (GET, POST, PUT, DELETE), stores requests in organized collections, enables environment variables for different configurations, and offers automated testing capabilities. With features like mock servers and API monitoring, Postman

streamlines the entire API lifecycle from design to production. Its user-friendly approach makes API interaction accessible to both technical and non-technical users while providing advanced functionality for professional developers.

**4.1.3 Python Libraries and Frameworks:**

**NumPy**

NumPy is a powerful numerical computing library that supports large, multi-dimensional arrays and matrices. It was used to perform array-based operations, mathematical computations, and efficient data handling throughout the project.

**Pandas**

Pandas is a data manipulation and analysis library that provides flexible data structures such as DataFrames. It was used for reading, organizing, and preprocessing tabular data, especially when handling labels or metadata associated with the images.

**TensorFlow**

TensorFlow is an open-source deep learning framework developed by Google. It was used to build and train the convolutional neural network

(CNN) model for image classification. TensorFlow provides robust tools for model definition, training, evaluation, and deployment.

**Keras Modules**

from tensorflow.keras import models, layers, optimizers

Keras is a high-level API within TensorFlow. These modules were used to:

models: define and manage the neural network architecture (Sequential, Model).

layers: add layers like Conv2D, MaxPooling2D, Dense, Dropout, etc., to build the CNN.

optimizers: choose optimization algorithms like Adam or SGD for training.

**ImageDataGenerator**

This class was used to load images from directories, resize them, and apply data augmentation (e.g., rotation, zoom, shift) to artificially expand the training dataset and improve the model's ability to generalize.

**scikit-learn**

scikit-learn is a popular machine learning library in Python. In this project, the confusion\_matrix function was used to evaluate the performance of the trained model by comparing predicted labels with actual labels.

**Matplotlib**

Matplotlib is a plotting library used for creating static, animated, and interactive visualizations. It was used to display training progress (e.g., loss

and accuracy graphs), show sample images, and visualize the confusion matrix.

**OS**

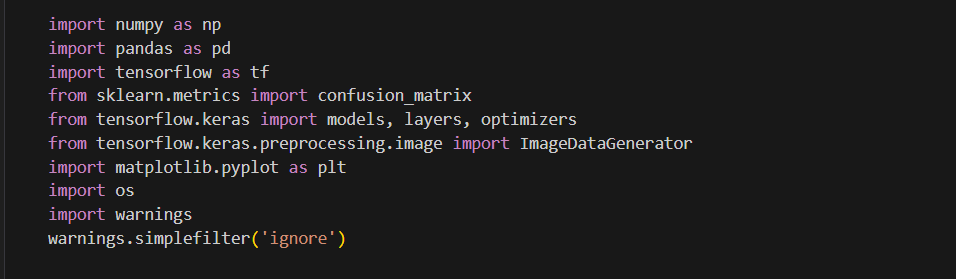
The built-in os module was used for interacting with the operating system, such as accessing directories, reading file paths, or checking if certain folders exist.

**Warnings**

This module was used to suppress unwanted warnings during execution to make the output cleaner and easier to read.

**4.2 Key components/modules of the system**

**4.2.1 AI-Model:**

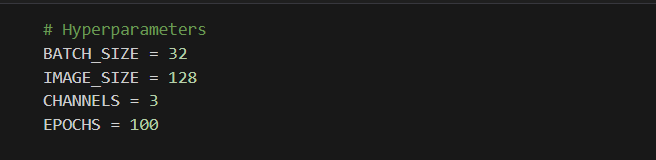


**Figure 1:libraries**

This block imports all the essential libraries and modules needed to build, train, and evaluate a deep learning model for image classification.

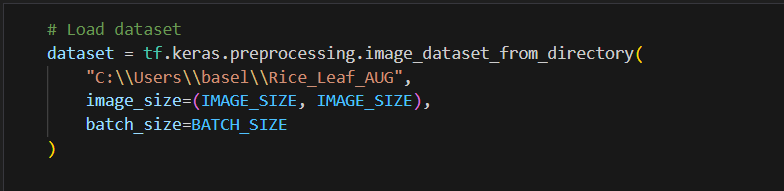
It includes tools for data manipulation (NumPy, Pandas), deep learning framework (TensorFlow and Keras), image preprocessing and augmentation (ImageDataGenerator), evaluation metrics (confusion matrix), visualization (Matplotlib), file handling (os), and suppressing unnecessary warnings to keep the output clean.

Overall, it sets up the environment with everything required for the whole machine learning workflow.



**Figure 2:Hyperparameters**

This block sets key settings for training a deep learning model, like image size, batch size, number of training rounds (epochs), and that the images are in RGB. These settings affect training speed, performance, and memory usage.

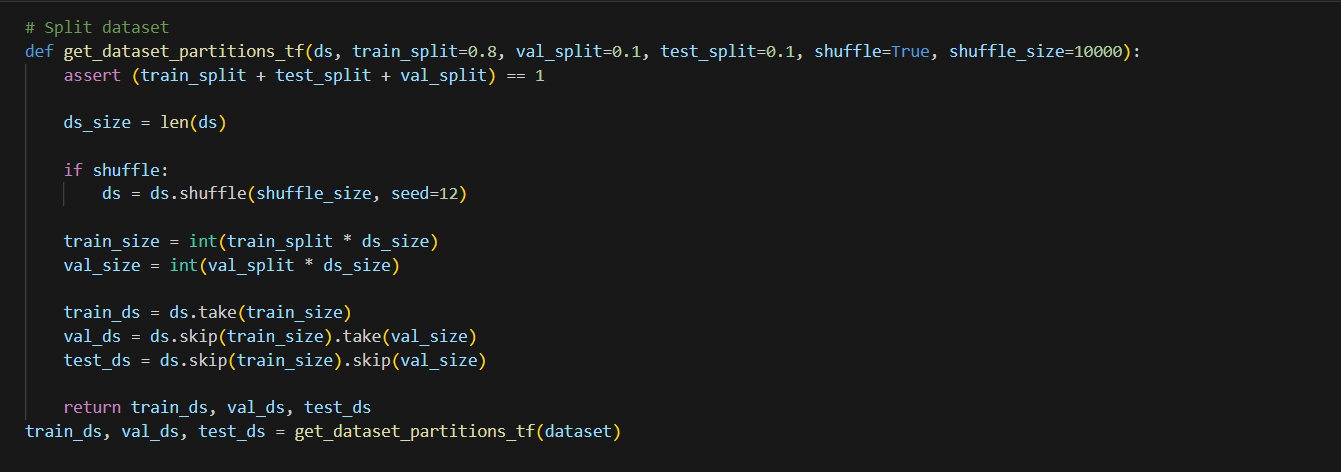


**Figure 3:load dataset**

This block is used to load an image dataset from a specific directory and automatically prepare it for training a deep learning model.

It helps by reading the images, resizing them all to the same dimensions, and batching them into manageable groups for efficient processing during training.

In short, this step is essential for organizing and preprocessing image data in a way that’s easy to feed into a neural network.

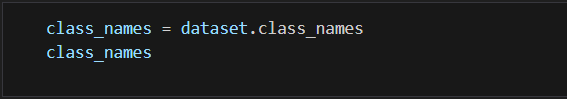


**Figure 4:split dataset**

This block defines and uses a function to split a dataset into training, validation, and testing parts based on specified proportions.

It optionally shuffles the dataset to ensure randomness, then divides it into three subsets to be used for model training, tuning, and final evaluation respectively.

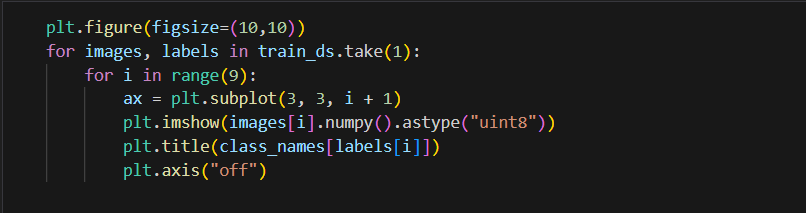
This splitting is crucial for assessing how well the model generalizes to unseen data and for avoiding overfitting during training.



**Figure 5:labels**

This block retrieves the list of class names (labels) from the loaded dataset.

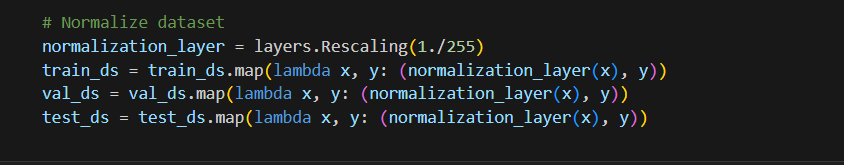
It helps to identify and understand the categories or classes that the model will be trained to recognize.



**Figure 6:visualizes images**

This block visualizes a sample of 9 images from the training dataset along with their corresponding class labels.

It helps to quickly inspect the data, verify that images are loaded correctly, and understand what the model will be learning from.

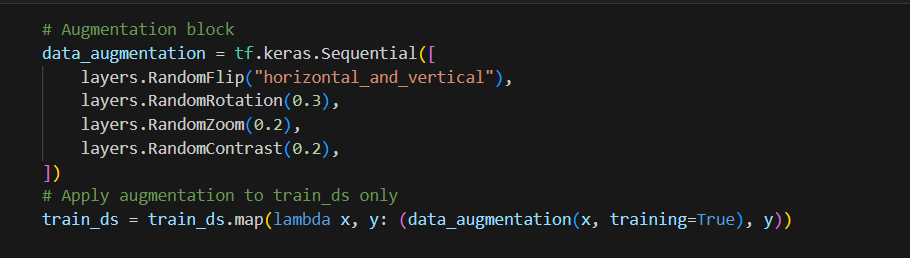


**Figure 7:Normalize dataset**

This block normalizes the image data by scaling pixel values from the range 0-255 to 0-1.

Normalizing helps the model train more efficiently and improves convergence by standardizing the input data across

the training, validation, and test datasets.

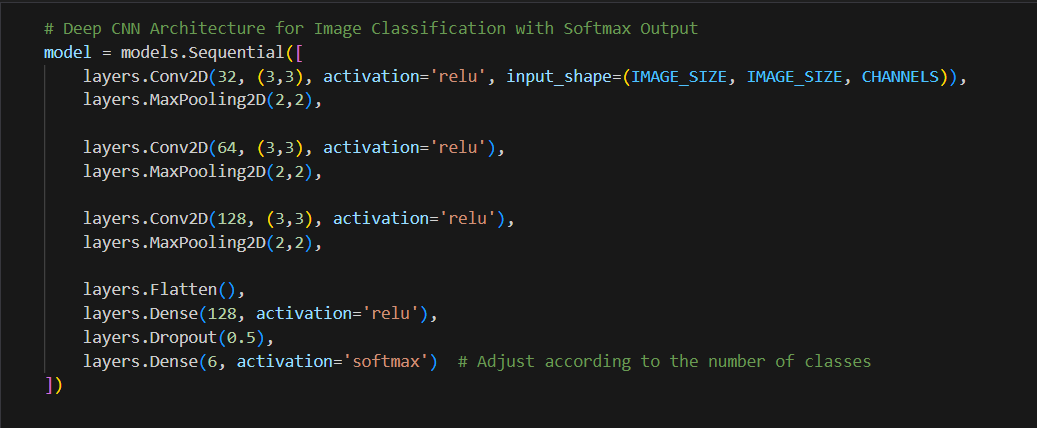


**Figure 8:Augmentation block**

This block defines a data augmentation pipeline that randomly applies transformations like flipping, rotation, zoom, and contrast changes to training images.

By applying these variations only to the training set, it helps the model generalize better by exposing it to a wider

range of image appearances, reducing overfitting and improving robustness.

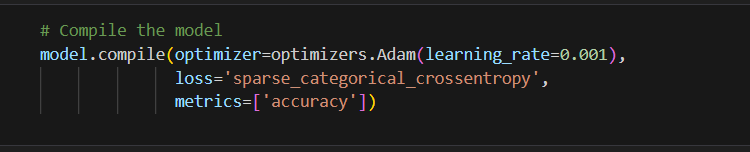


**Figure 9:CCN Architecture**

This block builds a deep convolutional neural network (CNN) designed for image classification.

It consists of multiple convolutional and pooling layers to automatically extract important features from images, followed by fully connected layers that learn to classify those features into one of several classes using a softmax output.

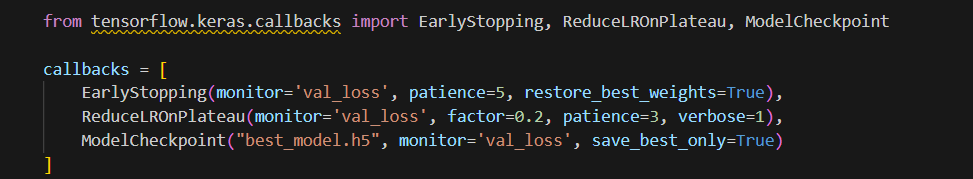
The dropout layer helps prevent overfitting by randomly disabling some neurons during training, improving the model’s ability to generalize to new data.



**Figure 10:compile the model**

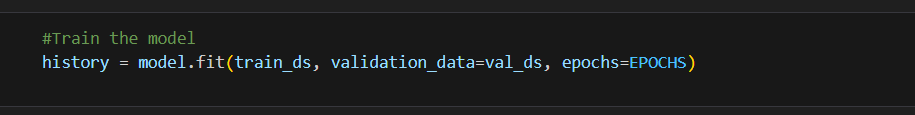
This block prepares the model for training by specifying how it should learn and evaluate performance.

It sets the Adam optimizer with a learning rate of 0.001 to adjust weights efficiently, uses sparse categorical cross-entropy as the loss function suitable for multi-class classification with integer labels, and tracks accuracy as the key metric during training and evaluation.



**Figure 11:callbacks**

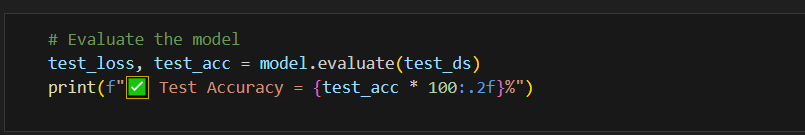
This block adds training callbacks to improve the process:  
**EarlyStopping** stops training if validation loss doesn’t improve for 5 epochs,  
**ReduceLROnPlateau** lowers the learning rate if loss plateaus,  
and **ModelCheckpoint** saves the best model during training.



**Figure 12:Train the model**

This block trains the deep learning model using the prepared training dataset while validating its performance on the validation set after each epoch.

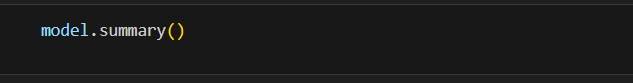
It runs for the specified number of epochs and records the training history, including metrics like loss and accuracy, which can later be used for analysis and visualization.



**Figure 13:Evaluate the model**

This block evaluates the trained model’s performance on the unseen test dataset to measure how well it generalizes.

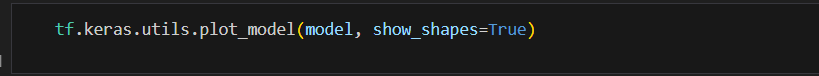
It calculates the loss and accuracy on the test data, then prints out the test accuracy as a percentage to give a clear indication of the model’s effectiveness on new, real-world data.



**Figure 14:summary**

This block prints a detailed summary of the model’s architecture, showing each layer’s type, output shape, and the number of trainable parameters.

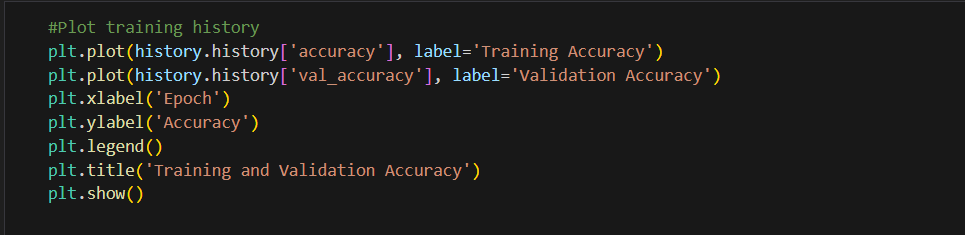
It helps to understand the model’s complexity and structure before or after training.



**Figure 15:Visual Diagram**

This block generates a visual diagram of the model’s architecture, displaying the layers and their output shapes.

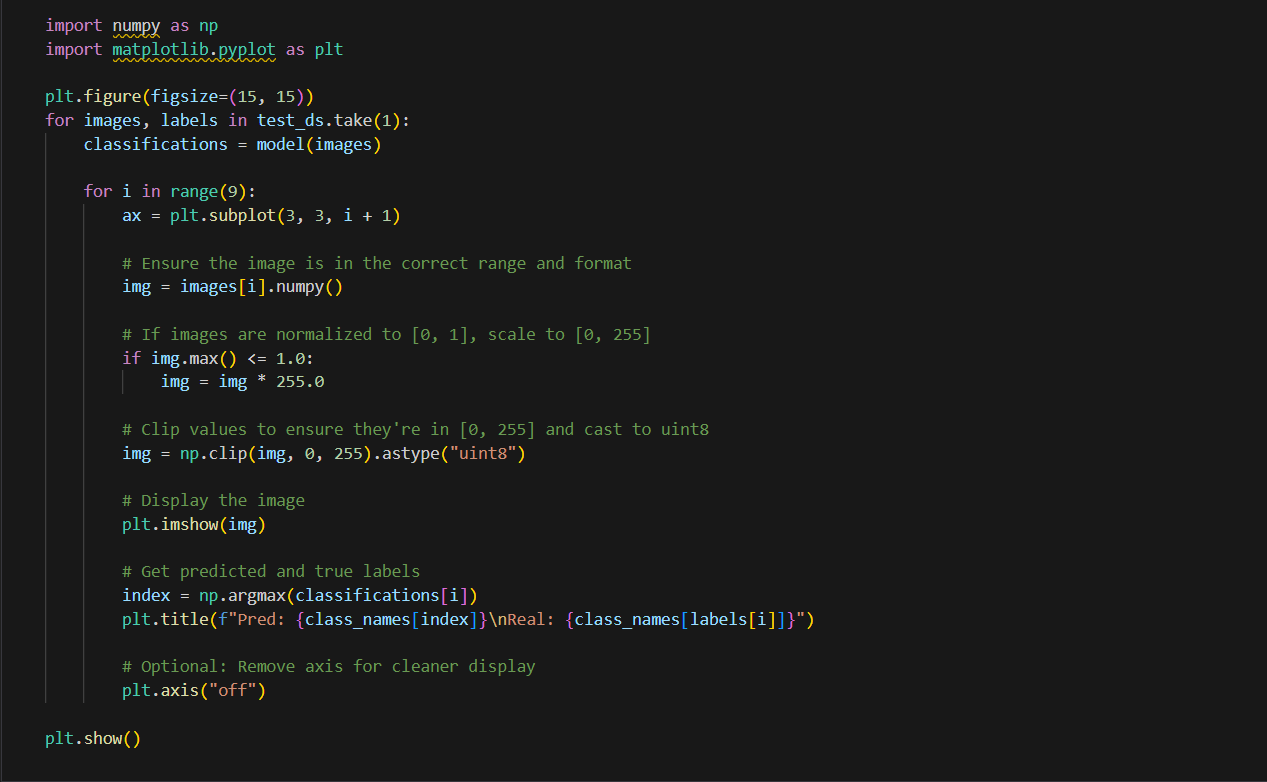
It helps to quickly understand the model structure and data flow between layers in a clear, graphical way.



**Figure 16:plot Training history**

This block plots the model’s training and validation accuracy over all epochs.

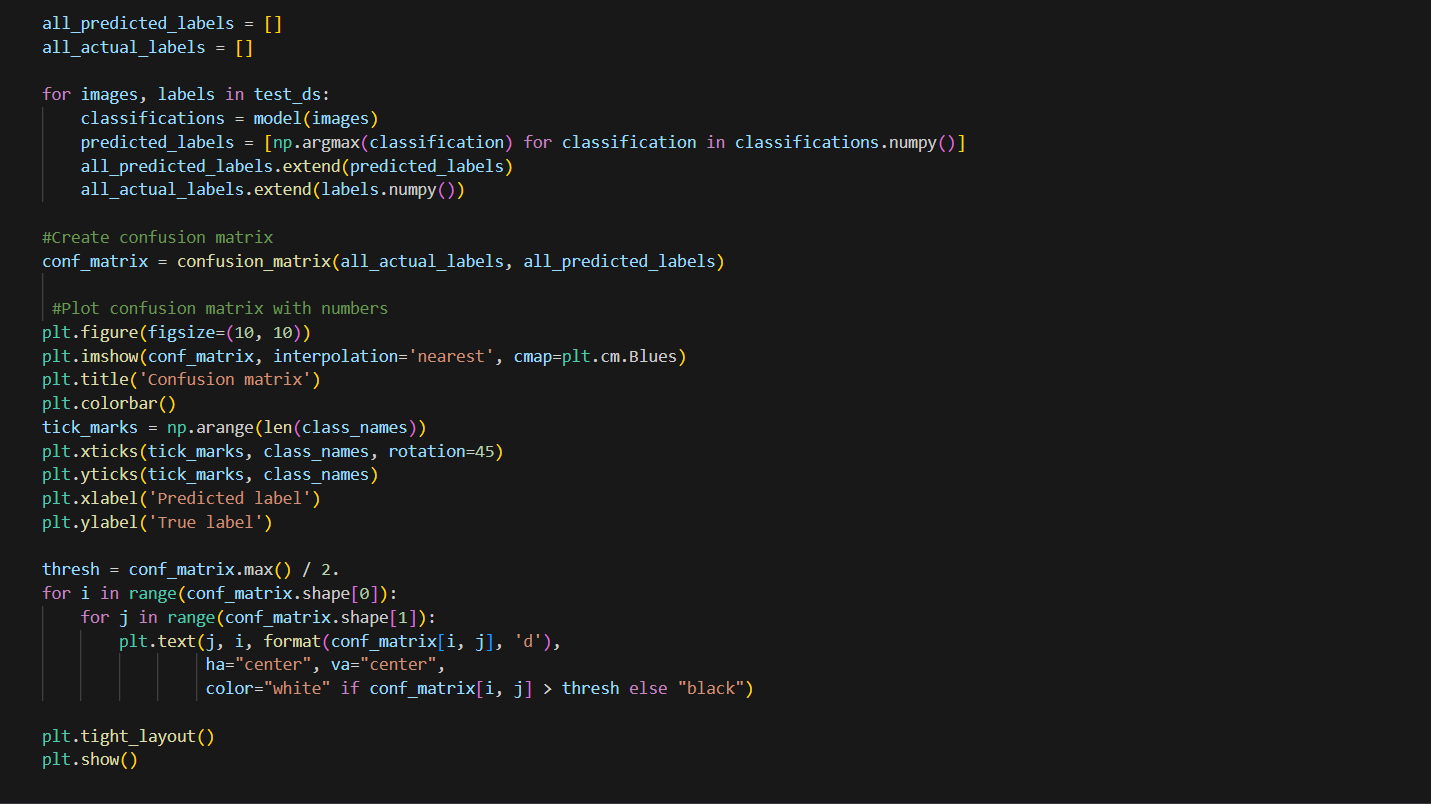
It helps visualize how well the model learned over time, showing trends like improvement, overfitting, or underfitting during training.



**Figure 17:visualize test dataset**

This block visualizes a sample of 9 images from the test dataset alongside their predicted and true class labels.

It helps to qualitatively assess the model’s prediction accuracy by showing how well it classifies individual images and identifying any mistakes visually.



**Figure 18:Evaluates model prediction**

This block evaluates the model’s predictions on the entire test dataset and generates a confusion matrix to summarize its classification performance across all classes.

It then visualizes the confusion matrix as a color-coded heatmap with numerical values, making it easy to identify where the model performs well and where it confuses different classes.

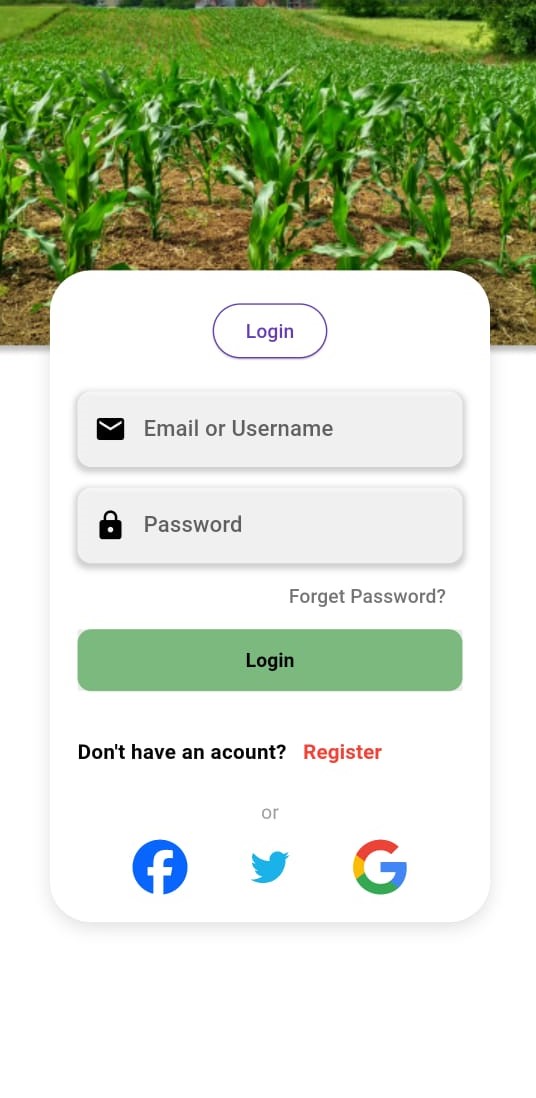
This detailed insight helps diagnose strengths and weaknesses in the model’s predictions.



**Figure 19 :chatBot**

This code implements a specialized chatbot for diagnosing and treating rice plant diseases. Developed in Python using the Flask web framework, it creates a simple yet effective web application. The bot operates by receiving user queries about rice diseases online, then forwarding these questions to OpenRouter's specialized AI platform. After obtaining precise answers from the AI system, it displays these responses to users.

**4.2.2 UI/UX Design:**



**Figure 20:Login Page**

- Function:

- This is the first page users see when opening the application.

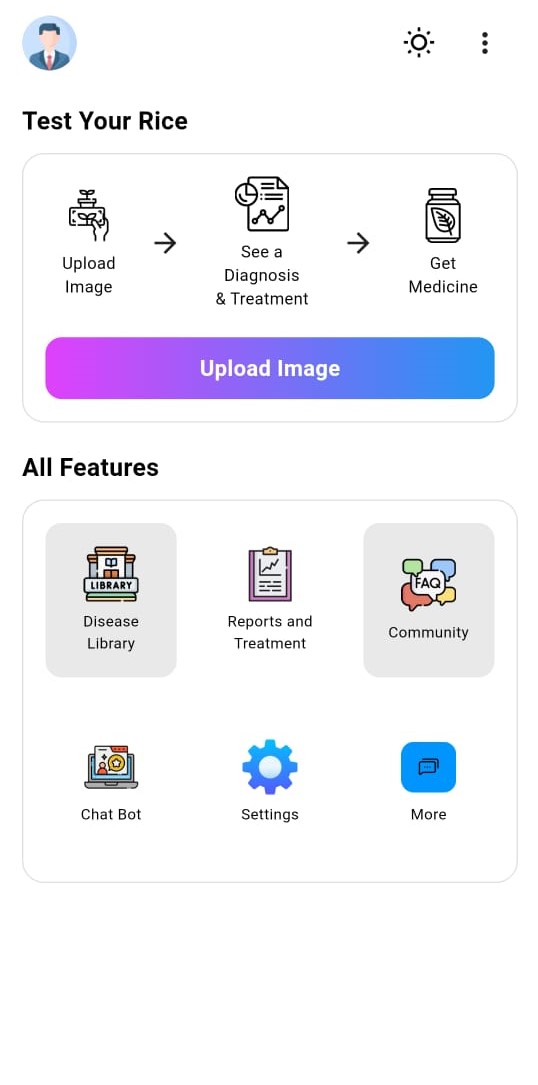
- Allows users to log in using their email or username and password.

- Features:

- "Forgot Password?" button to recover a lost password.

- "Register" button to create a new account if the user doesn’t have one.

- Benefit: Secures user access to their accounts and protects their data.



**Figure 21:Home Page**

- Function:

- The main page after logging in, offering the app’s core services.

- Features:

- Upload Image: To upload a photo of a rice plant for disease diagnosis.

- See a Diagnosis & Treatment: To view diagnostic results and treatment methods.

- Get Medicine: To receive recommendations for suitable medications or pesticides.

- All Features Section includes:

- Library: For a database of diseases and treatments.

- Reports and Treatment: For past reports and suggested treatments.

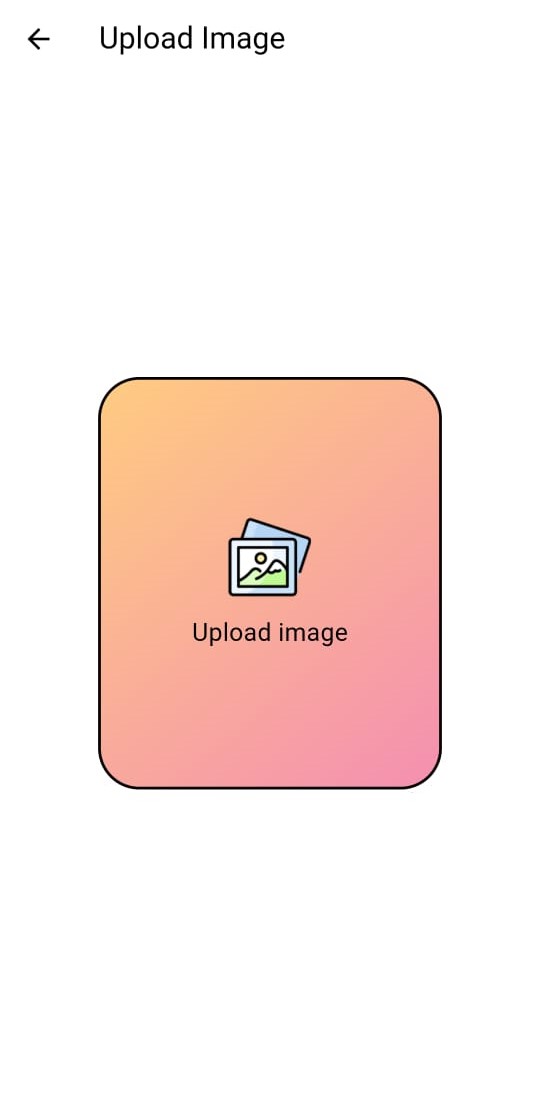
- Community: To connect with other farmers.

- Chat Bot: For instant AI-powered assistance.

- Settings: For app configurations.

- Benefit:

- Serves as the starting point for all user tasks, such as diagnosis, treatment, and research.



**Figure 22:Upload Image Page**

- Function:

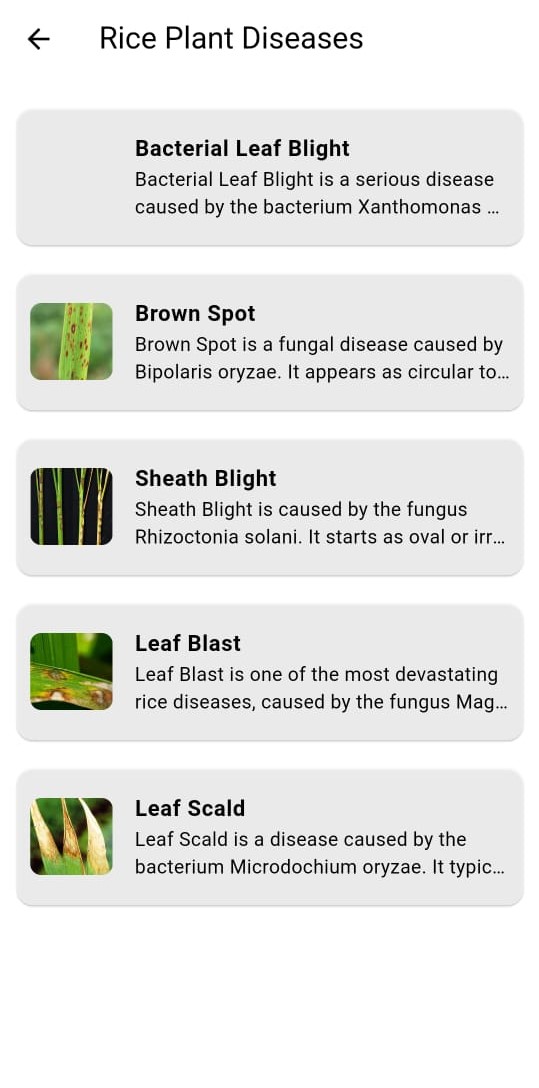
- Allows users to upload an image of a rice plant for analysis.

- Features:

- "Upload Image" button to select a photo from the device.

- Benefit:

- Initiates the automatic disease diagnosis process based on the uploaded image.



**Figure 23:Rice Plant Diseases Page**

- Function: Provides detailed information about common rice diseases.

- Features: List of diseases such as:

Bacterial Leaf Blight

Brown Spot

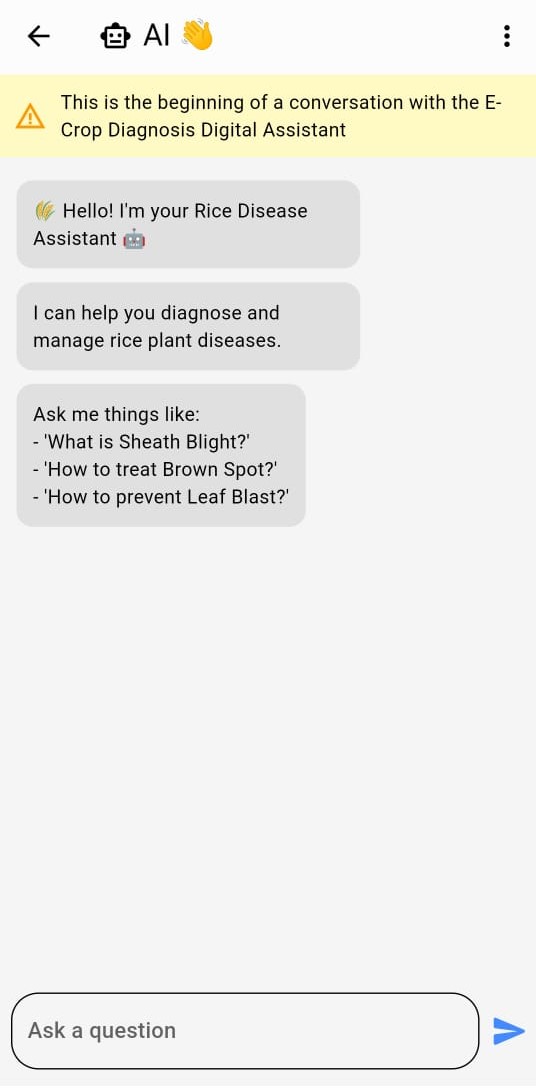
Sheath Blight

Leaf Blast

Leaf Scald

- Explanation of each disease, including causes (e.g., bacteria or fungi) and symptoms.

- Benefit: Helps users understand and identify diseases before formal diagnosis.



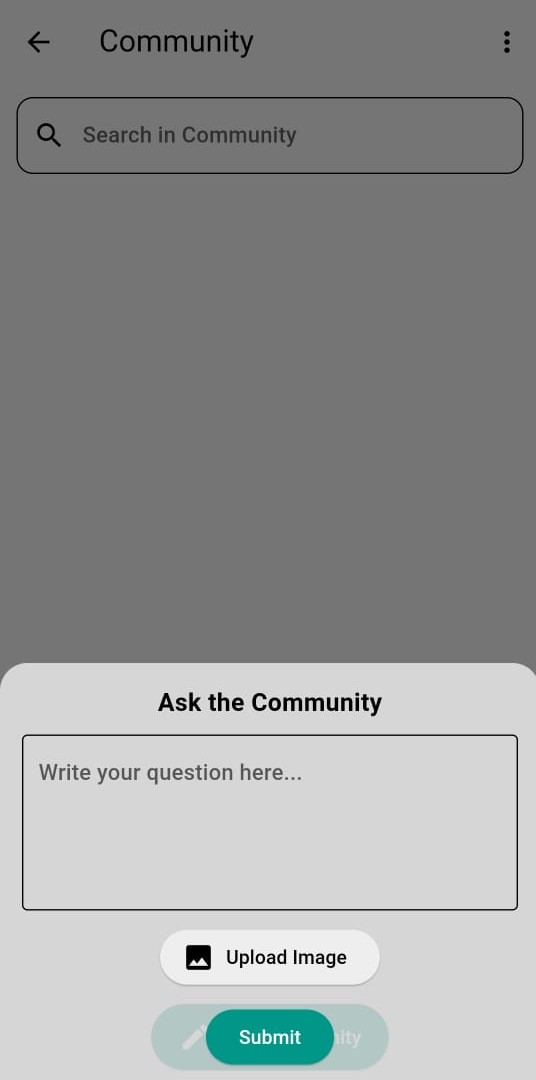
**Figure 24:Chat Bot Page**

- Function: An AI assistant that answers user questions about rice diseases and treatments.

- Features: Users can ask questions like

- User-friendly chat interface.

- Benefit: Provides instant and accurate answers without manual searching.



**Figure 25:Community Page**

- Function: Enables users to interact with other farmers and share experiences.

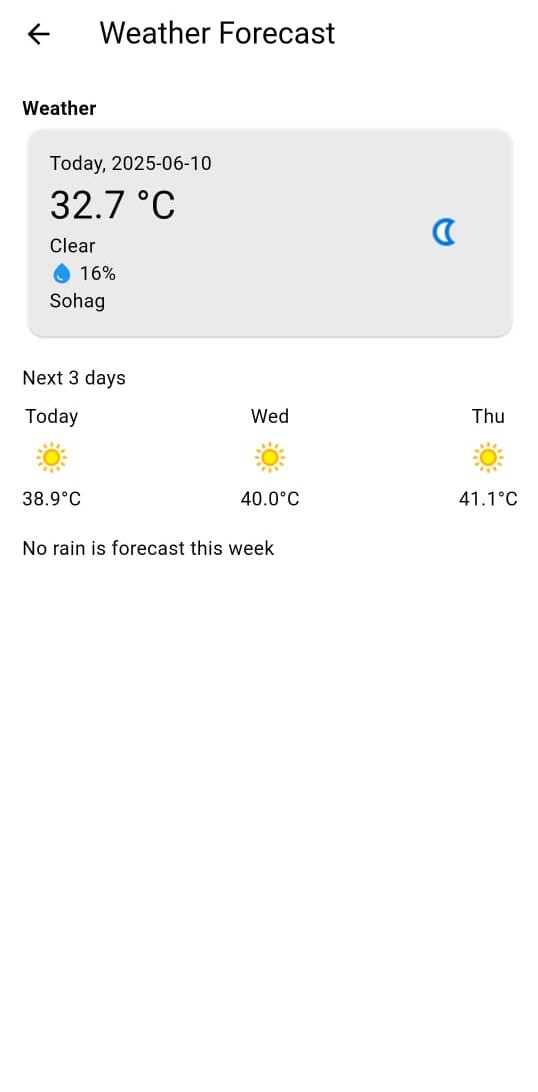
- Features:

- Search in Community: To find specific topics.

- Ask the Community: To post new questions.

- Upload Image: To share photos of agricultural issues.

- Benefit: Builds a support network among farmers and experts.



**Figure 26:Weather Forecast Page**

- Function: Displays weather forecasts for the selected location.

- Features:

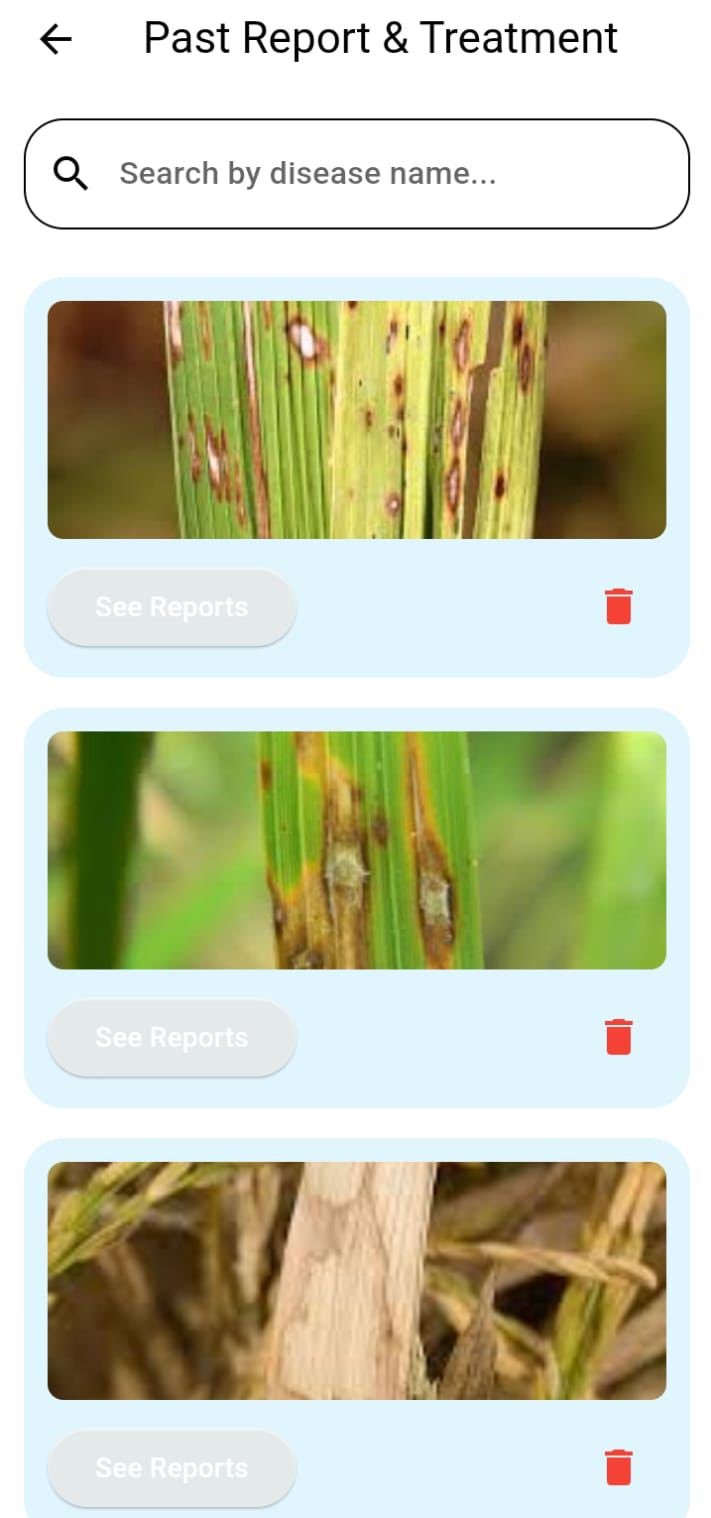
- Current temperature (e.g., 32.7°C).

- Weather conditions (e.g., Clear).

- Forecast for the next three days.

- Rainfall information.

- Benefit: Helps farmers plan agricultural activities based on weather conditions.



**Figure 27:reports**

This page allows you to search for past medical reports and treatments by entering the disease name in the search box. When you click "See Reports", it displays the search results showing details like previous diagnoses and prescribed medications.

**4.3 Challenges Faced and How They Were Resolved**

During the implementation of the Rice Disease Detection System (RDDS), several technical and practical challenges were encountered. These challenges required careful analysis and appropriate solutions to ensure a functional and efficient system:

1. Limited and Imbalanced Dataset:

One of the primary challenges was the limited availability of labeled rice disease images, particularly for some disease categories. The dataset was also imbalanced, with certain classes dominating. This was mitigated by applying data augmentation techniques using TensorFlow's ImageDataGenerator, such as rotation, horizontal and vertical flipping, zooming, and shifting. This helped to increase the dataset size and reduce class imbalance, thereby improving model generalization.

2. Overfitting During Training:

In the initial training stages, the model showed signs of overfitting achieving high accuracy on training data but poor results on validation data. To address this, dropout layers were incorporated into the CNN architecture to prevent neuron co-adaptation. Additionally, early stopping was used to halt training once the validation performance plateaued, preventing unnecessary training and overfitting.

3. Long Training Time and Resource Constraints:

Training convolutional models on image data required substantial computational resources. Initially, training was slow on local hardware. To overcome this, cloud-based platforms like Jupyter Notebook with GPU support were used, significantly reducing training time and allowing for faster experimentation and tuning.

4. Low-Quality and Noisy Images:

Some of the images collected from real-world sources were of low resolution, poorly lit, or blurry, which negatively impacted prediction accuracy. To improve model robustness, preprocessing techniques such as resizing, contrast adjustment, and noise reduction were applied. These steps helped in standardizing the inputs and minimizing the effect of noise.

5. Deployment and Integration Challenges:

After training, integrating the model into a usable interface presented another challenge. It was important to ensure that the model could run efficiently on low-resource devices or web-based platforms. This was resolved by converting the model to a lighter format (e.g., using TensorFlow Lite) and optimizing the code for deployment, making the system accessible and responsive for end users such as farmers or agricultural professionals.

6. Model Evaluation and Error Analysis:

Basic accuracy metrics did not fully reflect the model’s performance across all classes. Therefore, a confusion matrix was generated using

sklearn.metrics.confusion\_matrix to identify specific misclassifications. This analysis guided further improvements in data preprocessing and model refinement, especially for underperforming classes.

These challenges, while initially hindering progress, contributed significantly to the learning process and to the overall robustness and usability of the RDDS model.

Chapter 5

Testing & Evaluation

**Testing Strategies**

The testing strategy involves:

- Dataset Splitting: The dataset is divided into training (70%), validation (15%), and testing (15%) sets.

- Validation Set: Used during training to monitor performance and prevent overfitting using the EarlyStopping callback.

- Testing Set: Used after training to evaluate the final model’s accuracy and generalization capability.

- User Testing: Although no formal unit or integration testing frameworks were implemented, the evaluation on unseen test data serves as a practical form of validation.

**Performance Metrics**

The model’s performance is evaluated using the following metrics:

1. Accuracy:

- Calculated as the ratio of correct predictions to total test samples.

- Reflects the overall effectiveness of the model.

2. Loss Function:

- Sparse categorical crossentropy is used to guide the learning process.

- Lower loss indicates better model performance.

3. Confusion Matrix:

- Generated using Scikit-learn to visualize true vs. predicted classifications.

- Helps identify misclassified categories and improve model adjustments.

This structured approach ensures a robust deep learning model capable of accurately classifying rice leaf diseases while maintaining generalization across different data samples.

Chapter 6

Results & Discussion

* 1. **Introduction**

This section presents the results of implementing the Rice Disease Detection System (RDDS) using Convolutional Neural Networks (CNN). It discusses the model's efficiency, analyzes the significance of the obtained results, and highlights the key challenges and limitations encountered during the system's development.

* 1. **Summary of findings.**

RDDS was trained on a diverse dataset containing images of rice leaves infected with various diseases, as well as healthy leaves. The model achieved an overall accuracy of 96.61%, indicating high effectiveness in classifying and detecting rice diseases through image analysis.

During evaluation, the model demonstrated a strong ability to distinguish between healthy and infected leaves, with good generalization when handling previously unseen data. Other performance metrics showed a low error rate and consistent results across test samples, reflecting effective model training.

* 1. **Interpretation of results.**

Interpretation of Results (Did the Project Achieve Its Objectives?)

The primary goal of the project was to build an intelligent system capable of accurately identifying rice diseases through image analysis. Given the achieved accuracy of 96.61% and the model's ability to provide instant diagnoses through a user-friendly interface, it can be concluded that the project successfully met its objectives.

The integration of computer vision and deep learning techniques contributed to improving result quality and supporting early disease detection, thereby enhancing more effective agricultural decision-making. These results align with previous studies and, in some cases, outperform models mentioned in the literature review.

* 1. **Limitations of the proposed solution.**

Despite promising results, the proposed system has several limitations, the most notable of which are:

Data Diversity: Although the dataset was useful, it may not cover all forms of rice leaves, backgrounds, lighting conditions, and disease progression stages in real-world farming environments, which could affect the model's accuracy in the field.

Environmental Sensitivity: The model's performance may decline when images are captured under poor lighting conditions or using low-quality cameras, which are common in rural areas.

Model Scope: The system is currently limited to detecting a specific set of rice diseases and does not include rare diseases or other issues such as nutrient deficiencies or pest damage.

User Dependency: The accuracy of diagnosis heavily depends on the quality of the image uploaded by the user. Blurry or unclear images negatively impact the model's performance.

To improve the system in the future, it is recommended to focus on increasing data diversity, applying data augmentation techniques, fine-tuning the model more precisely, and enhancing preprocessing stages. These steps will improve the model's reliability and usability in diverse conditions.

Chapter 7

Conclusion & Future Work

* 1. **Conclusion.**

This project presents an innovative approach to one of the most pressing issues in modern agriculture early detection and management of rice plant diseases. As rice remains a fundamental food source for over half of the global population, particularly in developing regions, ensuring healthy and productive rice crops is essential for maintaining food security and economic stability. The widespread impact of rice diseases has emphasized the urgent need for smart, efficient, and accessible solutions that support farmers in protecting their crops and improving their yield.

The Rice Disease Detection System developed in this project represents a powerful integration of artificial intelligence, computer vision, and deep learning. By analyzing leaf images using advanced convolutional neural networks (CNNs), the system is capable of accurately identifying and classifying different types of rice plant diseases in real time. This automated approach significantly reduces the reliance on traditional manual inspection methods, which are often time consuming, inconsistent, and dependent on expert availability.

One of the key strengths of this system is its user centric design, ensuring that the final product is not only technically robust but also accessible and easy to use for non-technical users, particularly farmers in rural areas. The interface and user experience have been designed with simplicity and

practicality in mind, aiming to maximize usability and adoption across different user groups. By placing farmers at the center of the solution, the system bridges the gap between advanced technology and real-world agricultural challenges.

Moreover, the scalability of the solution ensures that it can be adapted and expanded to cover more crops and diseases in the future, making it a versatile tool in the fight for sustainable agriculture. The project also opens up new possibilities for further research and development in the field of precision farming, offering a pathway toward smarter, data-driven agricultural practices.

In conclusion, this project highlights the transformative potential of combining AI with agriculture. It not only addresses a significant real-world problem but also demonstrates how interdisciplinary innovation can lead to impactful, sustainable solutions. The Rice Disease Detection System stands as a testament to the role of technology in shaping the future of farming smarter, faster, and more resilient.

* 1. **future work.**

**1. Plant Health Tracking Over Time**

One of the proposed future enhancements is the implementation of a plant health tracking system that allows users to monitor the development or recovery of a rice plant over time. In this feature, the user can periodically

capture and upload images of the same plant.

The system will then analyze and store these images along a visual timeline.

The goal is to provide a clear progress overview, indicating whether the plant’s condition is improving, deteriorating, or remaining stable. This will be achieved by applying computer vision techniques to compare changes in the plant's appearance, such as:

* Leaf color and texture variations.
* Presence or absence of disease symptoms.
* Health score trends.

The timeline or graph-based visualization will give farmers better insights into treatment effectiveness, and help agricultural experts study disease progression. This feature could also be extended to generate automatic suggestions for the next steps based on health trends.

**2. Offline Mode Support**

To enhance accessibility and usability in remote agricultural areas where internet connectivity is limited or unstable, it is proposed to introduce an offline mode in future versions of the application.

This feature would involve embedding a lightweight version of the trained deep learning model directly into the mobile application using platforms like TensorFlow Lite (TFLite) or ONNX. This allows the app to function independently of cloud servers, enabling disease prediction to be done locally on the device.

In addition to local predictions, key data such as:

* Disease information.
* Common solutions.
* Previous predictions.

can be cached and made available offline. Once internet becomes available again, the app can synchronize with the cloud to update records or receive new data. This enhancement would significantly improve reliability and flexibility, making the system practical and scalable in rural settings.

**3. Voice Assistant for Illiterate Farmers**

To ensure inclusivity and make the system accessible to farmers who may have low literacy levels or are not familiar with written interfaces, a voice assistant feature is suggested as a future improvement.

In this feature, the farmer can interact with the application using voice commands in their local language. The app will use speech recognition

technologies (e.g., Google Speech-to-Text API ) to interpret spoken queries such as:

* “What disease is this?”
* “How do I treat my plant?”
* “Scan this leaf.”

The system would then process the request and respond using audio output via text-to-speech technology, making the interaction entirely voice-based and user-friendly. This approach reduces the barriers of education and technology, promoting wider adoption of the app in rural farming communities.

In addition, this feature could be paired with language localization, enabling support for various dialects or regional languages, increasing the impact of the system across different user groups.

References

* **Research Papers:**
* Convolutional neural network in rice disease recognition: accuracy, speed and lightweight
* [https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1269371/full#h1](https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1269371/full)
* Study on the Classification Method of Rice Leaf Blast Levels
* [https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2022.879668/full#h1](https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2022.879668/full)
* Hyperspectral Imaging Combined With Deep Transfer Learning for Rice Disease Detection
* [https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2021.693521/full#h1](https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2021.693521/full)
* A Deep Learning and SVM Hybrid Model for Rice
* [https://www.mdpi.com/2313-433X/10/8/183#html-abstract](https://www.mdpi.com/2313-433X/10/8/183)
* Disease Detection in Rice leaves using Convolutional Neural Network
* <https://ieeexplore.ieee.org/document/10200216>
* **Online course:**
* UI/UX Design using Figma
* <https://www.youtube.com/@ehabfayez>
* **Dataset:**
* Rice Leaf Diseases Detection dataset
* <https://www.kaggle.com/datasets/anshulm257/rice-disease-dataset>