A PROJECT REPORT ON

AN APPROACH TO RICE LEAF DISEASE DETECTION USING CNN

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

CSE- ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted By

J SASI KANTH	20MH1A4217
M CHAKRADHAR	21MH5A4204
K KIRAN KUMAR	20MH1A4222
CH NAGA SAI VAMSI	21MH5A4202

Under the Esteemed supervision of

Mrs. K. RojaRani, M.Tech

Assistant Professor

Department of CSE(AIML & IOT)

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, permanently affiliated to JNTUK & Accredited by NAAC Recognized by UGC under the sections 2(f) and 12(B) of the UGC act 1956 AdityaNagar, ADB Road –Surampalem 533437, E.G. Dist., A.P.

2023-2024

ADITYA COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUK, Kakinada,accredited by NBA & NAAC)Recognized by UGC under sections 2(f) & 12(b) of UGC Act, 1956

DEPARTMENT OF CSE(AIML & IOT)

CERTIFICATE

This is to certify that the project work entitled, **" AN APPROACH TO RICE LEAF DISEASE DETECTION USING CNN"**, is a work carried out by **J.SASI KANTH (20MH1A4217), M.CHAKRADHAR (21MH5A4204), K.KIRAN KUMAR (20MH1A4222), CH.NAGA SAI VAMSI (21MH5A4202)** in partial fulfillment of the requirement for the **award of the degree of Bachelor of Technology** in CSE (AI&ML) of **ADITYA COLLEGE OF ENGINEERING** affiliated to **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA** is a bonafide work carried out by him/her during the academic year 2023-2024.

Project Guide	**Head of the Department**
Mrs. K. RojaRani, M.Tech	**Dr. B. Kiran Kumar, PhD.**
Assistant Professor	**Professor & HOD**
Department of CSE(AIML & IOT)	**Department of CSE(AIML & IOT)**

EXTERNAL EXAMINER

DECLARATION BY THE CANDIDATE

We , **J.SASI KANTH, M.CHAKRADHAR, K.KIRAN KUMAR, CH.NAGA SAI VAMSI** bearing hall ticket numbers **20MH1A4217, 21MH5A4204,20MH1A4222, and 21MH5A4202** hereby declare that the project report titled **"AN APPROACH TO RICE LEAF DISEASE DETECTION USING CNN"** under the guidance of **Mrs. K. RojaRani,M.Tech , Assistant Professor**, is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **CSE - (AI & ML).**

This is a record of Bonafide work carried out by us, the results Embodied in this project report have not been reproduced or copied from any source and have not been submitted to any other University or Institute for the award of any other Degree.

Project Associates

J SASI KANTH	**20MH1A4217**
M CHAKRADHAR	**21MH5A4204**
K KIRAN KUMAR	**20MH1A4222**
CH NAGA SAI VAMSI	**21MH5A4202**

ACKNOWLEDGEMENT

First and foremost, we sincerely salute our esteemed institution **ADITYA COLLEGE OF ENGINEERING** for giving this golden opportunity for fulfilling our warm dreams of becoming Engineers.

We are highly obliged to our **Professor & Head of the Department, Dr.B. KIRAN KUMAR** sir for his constant inspiration, extensive help, and valuable support in every step.

We would like to express our sincere gratitude to **Mr. K. Satyanarayana, M.Tech,(PhD), Assistant Professor.,** our dedicated and insightful project coordinator, for his invaluable guidance, support, and encouragement through this project.

We would like to express our sincere gratitude to our main project internal guide, **Mrs. K Roja Rani, M.Tech, Assistant Professor.,** for his guidance, encouragement and continuing support throughout the course of this work.

We wish to thank **Dr.PULLELA.S.V.V.S.R.KUMAR, professor in CSE and Dean (Academics)** for his support and suggestions during our Internship work.

We owe a great deal to **Dr.A. RAMESH, principal** for his extending helping hand at every juncture of need. Finally, we are pleased to acknowledge our indebtedness to all those who devoted themselves directly or indirectly to make this project work a total success.

Project Associates

J SASI KANTH	**20MH1A4217**
M CHAKRADHAR	**21MH5A4204**
K KIRAN KUMAR	**20MH1A4222**
CH NAGA SAI VAMSI	**21MH5A4202**

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, Permanently Affiliated to JNTUK & Accredited by NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

Institute Vision, Mission

INSTITUTE VISION:

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Value based instruction and

to emerge as a premiere institute.

INSTITUTE MISSION:

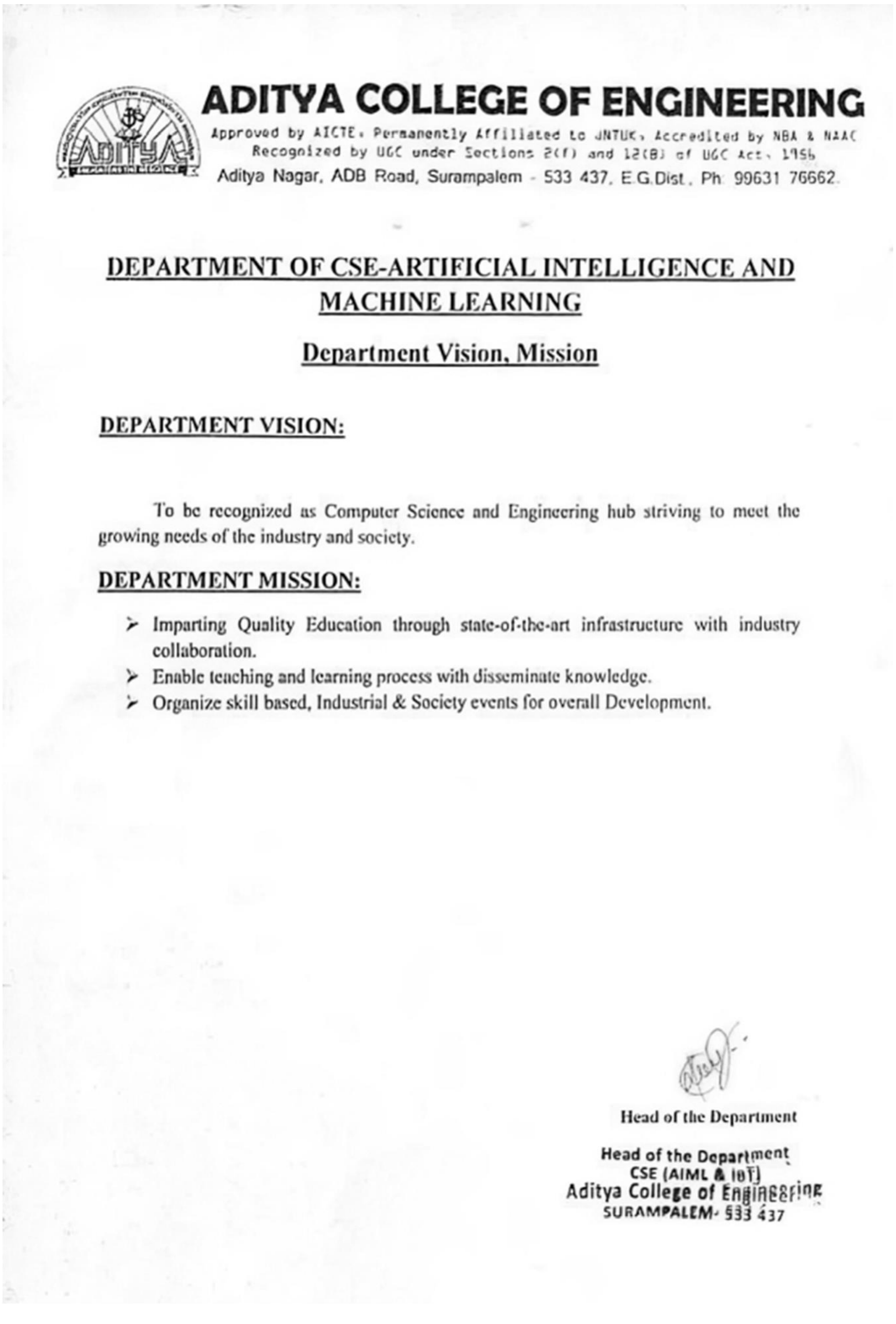
Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative Research and development
- Industry Institute Interaction
- Empowered Manpower

[illegible]

PRINCIPAL

PRINCIPAL
Aditya College of Engineering
SURAMPALEM - 533 437



ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

DEPARTMENT OF CSE-ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Department Vision, Mission

DEPARTMENT VISION:

To be recognized as Computer Science and Engineering hub striving to meet the growing needs of the industry and society.

DEPARTMENT MISSION:

- Imparting Quality Education through state-of-the-art infrastructure with industry collaboration.
- Enable teaching and learning process with disseminate knowledge.
- Organize skill based, Industrial & Society events for overall Development.

[illegible]

Head of the Department

Head of the Department
CSE (AIML & IoT)
Aditya College of Engineering
SURAMPALEM- 533 437

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

DEPARTMENT OF CSE-ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Program Specific Outcomes (PSOs)

The Graduates of B.Tech (AIML) Program shall be able to:

1. PSO 1: AI and ML System Development

Design, develop, and implement AI and ML systems by applying fundamental principles, algorithms, and techniques.

2. PSO 2: Data-driven Decision-Making

Collect, preprocess, and analyze large and diverse datasets to extract meaningful insights using AI and ML techniques to support data-driven decision-making processes in various domains.

[illegible]

Head of the Department

Head of the Department
CSE (AIML & IoT)
Aditya College of Engineering
SURAMPALEM- 533 437

ADITYA COLLEGE OF ENGINEERING

Approved by AICTE, Permanently Affiliated to JNTUK, Accredited by NBA & NAAC
Recognized by UGC under Sections 2(f) and 12(B) of UGC Act, 1956
Aditya Nagar, ADB Road, Surampalem - 533 437, E.G.Dist., Ph: 99631 76662.

DEPARTMENT OF CSE- ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Program Educational Objectives(PEO's)

The Graduates of B.Tech (AIML) Program shall be able to:

1. PEO 1: Proficiency in AI & ML Applications

Expertise in designing, developing, and deploying AI and ML solutions for healthcare, finance, and autonomous system problems.

2. PEO 2: Lifelong Learning and Adaptability

Adapt to evolving technologies and industry trends in AI and ML through continuous learning and self-improvement.

3. PEO 3: Effective Communication and Collaboration

Possess strong communication and teamwork skills, enabling them to effectively collaborate with interdisciplinary teams and solve complex problems using AI and ML concepts.

[illegible]

Head of the Department

Head of the Department
CSE (AIML & IoT)
Aditya College of Engineering
SURAMPALEM-533 437

ABSTRACT

The rice leaf suffers from several bacterial, viral, or fungal diseases and these diseases reduce rice production significantly. To sustain rice demand for a vast population globally, the recognition of rice leaf diseases is crucially important. However, recognition of rice leaf disease is limited to the image backgrounds and image capture conditions. The convolutional neural network (CNN) based model is a hot research topic in the field of rice leaf disease recognition. But the existing CNN-based models drop in recognition rates severely on independent dataset and are limited to the learning of large scale network parameters. In this paper, we propose a novel CNN-based model to recognize rice leaf diseases by reducing the network parameters. Using a novel dataset of 4199 rice leaf disease images, a number of CNN-based models are trained to identify five common rice leaf diseases. The proposed model achieves the highest training accuracy of 99.78% and validation accuracy of 97.35%. The effectiveness of the proposed model is evaluated on a set of independent rice leaf disease images with the best accuracy of 97.82% with an area under curve (AUC) of 0.99. Besides that, binary classification experiments have been carried out and our proposed model achieves recognition rates of 97%, 96%, 96%, 93%, and 95% for Blast, Brownspot, Bacterial Leaf Blight, Leaf smut and Tungro, respectively. These results demon strate the effectiveness and superiority of our approach in comparison to the state-of-the-art CNN-based rice leaf disease recognition models.

Keywords: Rice leaf diseases · Image recognition · Convolutional neural networks

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

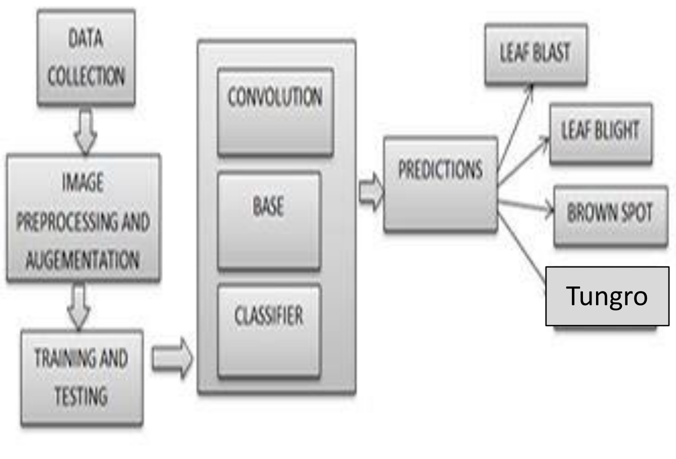
ABBREVIATION	FULL FORM
CNN	Convolutional Neural Network
RGB	Red, Green, Blue
IRRI	International Rice Research Institute
BLB	Bacterial Leaf Blight
XOO	Xanthomonas Oryzae pathovar Oryzae
AI	Artificial Intelligence
SVM	Support Vector Machine
ANN	Artificial Neural Network
GPU	Graphics Processing Unit
SSD	Solid-State Drive
RAM	Random Access Memory
API	Application Programming Interface
CPU	Central Processing Unit
OS	Operating System
IDE	Integrated Development Environment
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
SQL	Structured Query Language
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
GAN	Generative Adversarial Network
RESNET	Residual Network
VAE	Variational Autoencoder

AN APPROACH TO RICE LEAF DISEASE DETECTION USING CNN

Chapter 1

INTRODUCTION

India and other countries around the world rely on rice as their main source of food. During different stages of cultivation, rice leaves are afflicted by various diseases. Hence, early noticing and cures of such diseases are valuable to ensure high quantity and quality but this is very difficult due to the considerable costs of land under individual farmers and the enormous collection of diseases as well as the occurrence of more than one disease in the same weed. Automated Systems are required to support the difficulty of the ranchers and give further developed precision of plant diseases detection, various machine learning algorithms are using research work including Support Vector Machine (SVM) and Artificial Neural Network. The accuracy of such systems is highly dependent on feature selection techniques. The convolutional neural networks have provided a great advance in image-based recognition by eliminating the need for image processing as well as providing inbuilt feature election. We designed an automated system in which our farmers can upload images of diseased leaves onto our server, where the neural network will analyze the images to identify the disease and classify it so that the remedy and diagnosis can be returned to the farmer. Increasingly, cell phones and the internet are becoming accessible to all, so we have thought of developing an automated system that would let the farmers upload images of diseased leaves onto a server and have them uploaded to our server. "The accuracy of the achieved network is 91.23%, using stochastic gradient descent with a small batch size of thirty (30) and initial learning rate of 0.0001". Six hundred (600) images of rice plants representing the classes were used in the training by Atole & Perkin 2018. An approach is based on deep learning to detect rice plant diseases and pests by using images captured on a real-life scenario with a heterogeneous background was well reported by y. Lu, S.Yi, and y.Zhang in 2017.

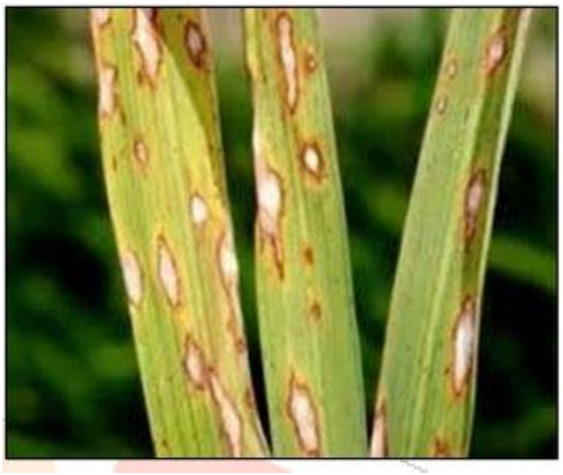


1.i)Introduction

Rice plant diseases depend on the symptoms and signs produced by the pathogens for identifying and classifying the disease leaf. The identification of diseases based on symptoms often becomes difficult. Therefore the identification based on digital images has been increasingly growing. Multispectral and hyperspectral images provide more information, but they are valuable for farmers. While the conventional cameras in cell phones are easily available to all farmers with affordable cost to capture images. This has prompted the researchers in developing systems based on RGB rich experienced experts to identify the diseases. The researchers have been encouraged by the limitations to explore automatic methods that detect and classify plant diseases simply and reliably with high accuracy. In addition, it will help the farmers to select the right pesticides. According to statistics, rice plant diseases destroy 10-15% of rice production in Asia Countries. Transfer learning is a machine learning method where a developmental model for a task is reused as the beginning point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the beginning point on computer vision and natural language processing tasks given the huge compute and time resources required to develop neural network models on these problems and from the huge jumps in the skill that they provide on related fix or worry. In this, you will discover how you can use transfer learning to speed up training and improve the performance of your deep learning model. We have utilized the CNN model and utilizing Deep learning we have calibrated the completely associated layers so we can oblige our dataset and toward the end, we have done a few blunder examinations and attempted to clarify the explanations behind the error. “The side effects and information about the illnesses have been gathered from the International Rice Research Institute (IRRI) Rice Knowledge Bank website". The dataset comprises 1649 pictures of unhealthy leaves of rice comprising of three most normal sicknesses be specific Rice Leaf Blast, Rice Leaf Bligh, Brown Spot, and 507 pictures of health leaves. The number of pictures that could be gathered from the fields is exceptionally less for preparing CNN so we have utilized several augmentation techniques like zoom, horizontal and vertical shift, and rotation.

Leaf Blast:

"Rice leaf blast disease, caused by Magnaporthe oryzae, occurs in about 80 countries on all continents where rice is grown, in both upland cultivation and paddy fields". The area of damage caused depends on environmental factors, but worldwide it is one of the most destructive rice diseases, resulting in losses of 10–30% in the rice of global yield. Initially, it appears as seedlings and small necrotic regions in rice have chlorotic margins and becomes larger and merge. In older rice plants, the symptoms of diseases can happen in leaves, collar – the intersection of the leaf edge and leaf sheath, hubs, neck, and panicle. Neck rot and panicle blast are particularly damaged causing up to 80% yield losses in severe epidemics. Triangular, purple-colored marks form on the neck node which elongates on both sides, seriously damaging grain development. When young neck nodes are occupied the panicles become white later infection in plant growth results in incomplete grain filling.



1.ii)Leaf Blast

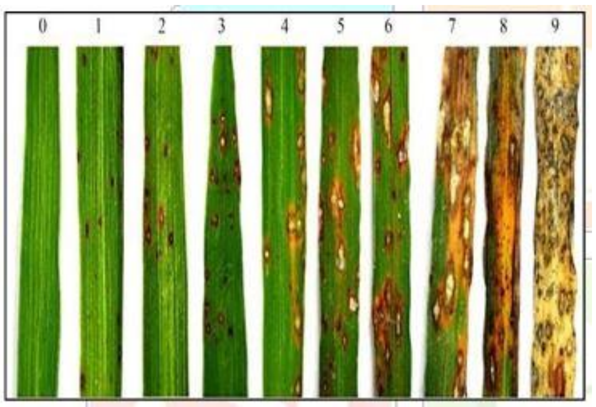
Bacterial Leaf blight:

Rice Bacterial Leaf blight is also called the bacterial disease of rice, a deadly bacterial disease that is among destructive afflictions of cultivated rice (Oryza sativa and O. glaberrima). In serious pandemics, crop misfortune might be all around as high as 75%, and a huge number of hectares of rice are contaminated yearly. The disease was first seen (1884–85) in Kyushu, Japan, and the causal specialist, the bacterium Xanthomonas oryzae pathovar oryzae (also referred to as Xoo, 1911), around then having been named Bacillus oryzae. Xanthomonas oryzae pv. Oryzae (Xoo) is widely prevalent and causes Bacterial Leaf Blight (BLB) in Basmati rice grown in different areas of Pakistan. To beat the deficiency of grain yield in rice we want to utilize ecologically safe ways to deal with defeat from this disease. The present review is expected to create a mix, in light of local adversarial microscopic organisms for biocontrol of BLB and to build the yield of Super Basmati rice assortment. Bacterial curse turns out to be first apparent as water splashed marks that spread from the leaf tips and margins, becoming bigger and ultimately delivering a smooth seepage that dries into yellow drops.

1.iii)Bacterial Leaf Blight

Brown spot:

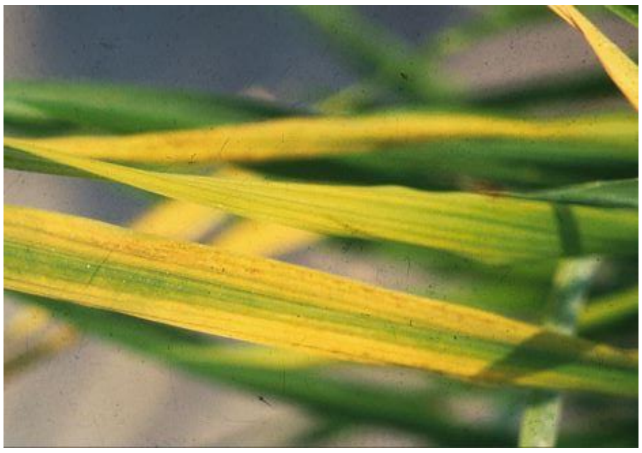
In rice diseases, the brown spot is one of the most damaging and common diseases which was historically largely ignored by farmers. Its maximum observable harm is the numerous huge spots at the leaves that may kill the entire leaf. Unfilled grains or discolored seeds are formed when the seed occurs with infection.



1.iv)Brown Spot

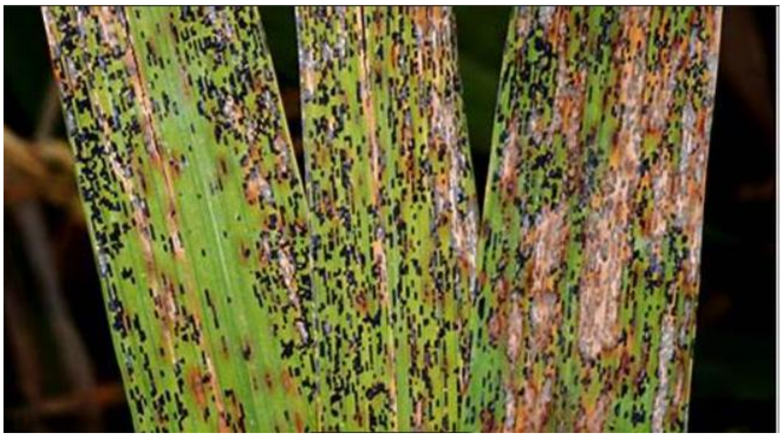
Tungro:

Tungro disease is primarily transmitted by the green leafhopper (Nephotettix virescens), which introduces the viruses into the rice plants while feeding. Control measures for tungro disease include the use of resistant rice varieties, controlling the population of the green leafhopper through insecticides or biological control methods, and the practice of clean cultivation to remove infected plants and reduce the spread of the disease


1.v)Tungro

Leaf smut:

Leaf smut, caused by the fungus Pyricularia grisea (formerly known as Magnaporthe grisea), is a common disease affecting rice plants. However, leaf smut primarily affects the panicles (the flowering part of the rice plant) rather than the leaves.


1.vi)Leaf Smut

1.1 OVERVIEW:

Our project aims to develop an automated system for detecting rice leaf diseases using deep learning techniques, specifically Convolutional Neural Networks (CNNs). Rice is a vital staple crop globally, and diseases affecting its leaves can cause significant yield losses and threaten food security. Leveraging CNNs, which have demonstrated remarkable performance in image recognition tasks, we seek to create a robust and efficient model capable of accurately identifying various rice leaf diseases.

The project will begin with the collection of a diverse dataset comprising high-quality images of rice leaves, including both healthy specimens and those infected with common diseases such as brown spot, blast, and tungro. These images will be meticulously labeled to facilitate supervised learning. Preprocessing steps, including resizing, normalization, and data augmentation, will be applied to enhance the dataset's quality and diversity.

For model development, we will design a CNN architecture tailored to the specifics of rice leaf disease detection. This architecture will consist of multiple layers with convolutional, pooling, and fully connected components optimized for feature extraction and classification. Training of the CNN model will be conducted using the prepared dataset, with performance evaluation carried out using standard metrics such as accuracy, precision, recall, and F1-score.

Through this project, we aim to contribute to the advancement of agricultural technology by providing a reliable and efficient tool for early detection and management of rice leaf diseases. The automated system developed will not only assist farmers in timely intervention and disease control but also contribute to sustainable agricultural practices and global food security.

1.2 PURPOSE OF THE SYSTEM:

The purpose of our system is to provide farmers with an automated tool for early detection and management of rice leaf diseases. By leveraging Convolutional Neural Networks (CNNs), the system can accurately identify various diseases, including brown spot, blast, and tungro, from images of rice leaves. Timely detection enables farmers to implement targeted interventions, such as pesticide application or crop rotation, mitigating yield losses and ensuring sustainable agricultural practices. Ultimately, this system contributes to food security by empowering farmers with the technology to efficiently monitor and combat diseases, thereby safeguarding rice crop yields and global food production.

1.3 SCOPE OF THE SYSTEM:

The system we're developing aims to revolutionize rice crop management by providing an automated solution for detecting leaf diseases. Leveraging Convolutional Neural Networks (CNNs), it will accurately identify common rice leaf diseases such as brown spot, blast, and tungro from uploaded leaf images. This system seeks to streamline the process of disease diagnosis, reducing reliance on manual inspection and enabling rapid, consistent analysis even across large datasets. Accessible to farmers through a user-friendly interface, it promises scalability across diverse agricultural settings and continual improvement through updates and feedback. Ultimately, it stands to significantly enhance crop health management and contribute to global food security efforts.

1.4 EXISTING SYSTEM:

In the existing system, rice leaf disease detection often relies on manual inspection by agricultural experts, which can be time-consuming and labor-intensive. Traditional methods involve visual examination of leaves for symptoms such as discoloration, lesions, or other abnormalities indicative of disease. Additionally, farmers may rely on field observations and knowledge passed down through generations to identify and diagnose leaf diseases. While these methods have been effective to some extent, they are subjective, prone to human error, and may not provide timely or accurate diagnosis, especially in cases of subtle or early-stage infections. As a result, there is a pressing need for an automated and reliable system that can enhance the efficiency and accuracy of rice leaf disease detection, ultimately improving crop management practices and ensuring food security.

Chapter 2

LITERATURE REVIEW

Image analysis techniques for measuring rust sickness found on soybean leaves. The methodology used is division of contaminated parts from multi-phantom pictures of soybean plant leaves which are finished utilizing quick manual edge setting strategy in light.

A Convenient application for paddy illness with recognizable proof structure demonstrating Fluffy entropy and Probabilistic neural framework classifier that continues running on Android which is a flexible working operating system. It incorporates forms of ailments specifically darker spots, leaf impact, tungro and micro organism leaf curse. A conventional surveys on different methods of image processing applications in the agriculture field. The main focus of the area of their survey was in two streams, one is weed detection and the other one is fruit grading systems. Weed are the dangerous crops grown along with the main crops so they found them out by applying image processing techniques and collecting the images of the particular crops.

Entire summary on disease classification in plant crops using image recognition techniques. These strategies are expected to be valuable for scientists giving a far detailed reaching diagram of vegetable pathology and also programmed discovery of plant ailments utilizing design acknowledgment systems. The main essential paddy infections. His research was more precise in finding out the leaf impact formed because of spots, bacterial infections, leaf strength and many other factors. Shape patterns are used to dismember the piece of the injuries. Discussed various data mining techniques for paddy crop disease prediction and classification.

Image Edge detection and Segmentation techniques. Initially the taken pictures are processed for enrichment at the initial stage of the process. R, G, B colour, colour, shape, boundaries, texture etc are the features which are extracted out from the target regions (disease spots) and later on other steps of the process continue to follow. This research includes the pest recommendations and also remedies for the diseases identified. This analysis work had a lot of great qualities but if failed to show the remedies and also could not reach the needed accuracy level. A process that has only eyesight and needs precise observation and additional scientific methods and technologies. Here the images of unhealthy leaves of that particular plant are captured and the features (Hue, Saturation, and Value) are extracted once the segmentation phase is completed. CNN is trained to distinguish the healthy and unhealthy plant samples and gives 80% accuracy. A model that explains about the images which are captured and stored in the database by mobile phones or any digital application. After this stage these images are analysed and judged by the senior experienced persons and a final conclusion is passed out.

Computer vision techniques help in recognising the diseased places in the images and then undergo classification process. Color difference process which is a primary approach is used for the segmentation of the not well plant areas.

The system allows the conclusion of the senior persons and also sends the feedback to the farmers through any means of digital communication. The main goal of this proposal is to develop an image recognition system that can identify the crop diseases and help out the agriculture field and also the future generations from food crisis and economic downfall. Image processing is a picture identification system which helps in the digitization of the color image. Arithmetic science is the primary logic behind this segmentation process. Then after a little more research it is found that this process depends on mankind (drawback of many other proposals) and also the time taken to complete this process is much more than many other existing and upcoming methods.

In summary, various image processing techniques have been proposed for plant disease detection. Yet none of them proved to be efficient for identification of all kinds of paddy crop diseases.

Chapter 3

SYSTEM ANALYSIS

3.1 FUNCTIONAL REQUIRMENTS

Image Classification: The system must accurately classify rice leaf images into disease categories, including Leaf Blast, Bacterial Leaf Blight, Brown Spot, Leaf Smut, and the fifth disease. This involves training the model to recognize specific visual patterns associated with each disease type.

Multi-Class Classification: It should support multi-class classification, distinguishing between different diseases present in the same leaf image. This capability allows the system to identify multiple diseases affecting a single rice plant, providing comprehensive diagnostic information to farmers.

Real-Time Inference: The system must provide real-time inference, ensuring prompt diagnosis and intervention for disease management. This requires efficient processing of input images and rapid classification to enable timely decision-making by farmers.

Scalability: It should be scalable to accommodate a growing dataset and increasing computational demands as more data is collected and model complexity evolves. This involves designing a flexible architecture that can handle large volumes of data and adapt to changing requirements over time. **User Interface:** The system should have a user-friendly interface for uploading images, displaying classification results, and providing feedback or recommendations to users. The interface should be intuitive and accessible to farmers with varying levels of technical expertise.

3.2 NON-FUNCTIONAL REQUIRMENTS:

Accuracy: The system must achieve a high level of accuracy in disease classification to minimize misclassifications and ensure reliable diagnosis. This requires rigorous testing and validation of the model's performance using representative datasets and real-world scenarios.

Performance: It should exhibit fast inference times and minimal latency to provide a responsive user experience. This involves optimizing model architecture, algorithms, and computational resources to maximize processing speed without compromising accuracy.

Robustness: The system should be robust to variations in lighting conditions, leaf orientation, and background clutter to ensure accurate classification in diverse environments. This requires training the model on a diverse range of images and incorporating techniques for noise reduction and image normalization.

Scalability: The system architecture should be scalable to accommodate future enhancements, such as additional disease categories or improvements in model performance. This involves designing modular components and using scalable infrastructure to support future growth and expansion.

Security: It should incorporate measures to ensure data privacy and security, protecting sensitive information uploaded by users. This includes encryption of data during transmission and storage, access control mechanisms, and compliance with relevant data protection regulations.

Resource Efficiency: The system should be resource-efficient, minimizing computational resources required for training and inference to optimize cost-effectiveness and environmental sustainability. This involves optimizing algorithms, minimizing memory usage, and leveraging parallel processing capabilities.

Usability: The user interface should be intuitive and user-friendly, allowing farmers and agricultural stakeholders to easily upload images and interpret the model's output. This requires conducting user testing and feedback sessions to refine the interface and ensure it meets the needs of the target audience.

3.3 SYSTEM REQUIRMENTS:

3.3.1 HARDWARE REQUIRMENTS

Computing Resources: High-performance computing resources with sufficient processing power and memory capacity to train and deploy deep learning models effectively. Recommended: Multi-core processors (e.g., Intel Core i7 or higher), ample RAM (16GB or more), and dedicated GPUs (NVIDIA GeForce GTX 1080 or higher) for accelerated model training.

Storage: Adequate storage space to store the dataset, model checkpoints, and associated files. Recommended: Solid-state drives (SSDs) for fast data access and efficient model training.

Networking: Stable internet connectivity for accessing cloud-based services, downloading datasets, and deploying model updates. Recommended: High-speed broadband connection with reliable uptime for seamless operation

Camera (Optional): High-resolution digital camera or smartphone camera for capturing images of rice leaves in the field, if acquiring new data is part of the system workflow.

3.3.2 SOFTWARE REQUIRMENTS

Operating System: Compatible with major operating systems such as Windows, macOS, or Linux. Recommended: Linux-based distributions (e.g., Ubuntu) for optimal compatibility with deep learning frameworks and libraries.

Deep Learning Framework: Framework for developing and training deep learning models, such as TensorFlow, PyTorch, or Keras. Required: TensorFlow or PyTorch for building and training Convolutional Neural Networks (CNNs) for rice leaf disease detection.

Python Environment: Python programming language environment for implementing machine learning algorithms and data processing tasks. Required: Python 3.x for developing the system components and integrating third-party libraries.

Development Tools: Integrated Development Environment (IDE) or code editor for writing, debugging, and testing code. Recommended: Visual Studio Code, PyCharm, or Jupyter Notebook for Python development.

Image Processing Libraries: Libraries for image processing and manipulation, such as OpenCV or Pillow. Required: OpenCV for preprocessing rice leaf images, including resizing, normalization, and augmentation.

Additional Libraries: Libraries for data manipulation, visualization, and evaluation, such as NumPy, Pandas, Matplotlib, and Scikit-learn. Required: NumPy for numerical operations, Pandas for data handling, Matplotlib for plotting, and Scikit-learn for evaluating model performance.

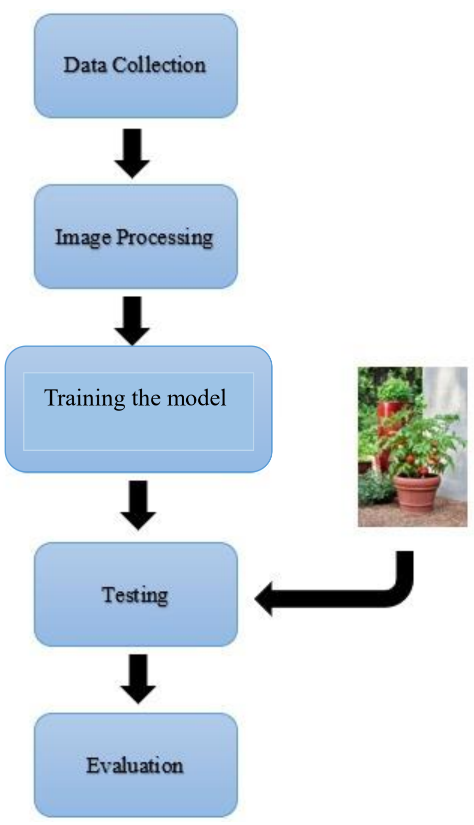
Web Development Tools (Optional): Tools and frameworks for building web-based user interfaces, if developing a web application for accessing the system. Recommended: Flask or Django for backend development, HTML, CSS, and JavaScript for frontend development.

Deployment Tools: Tools for deploying trained models into production environments, such as Docker or Kubernetes. Required: Docker for containerization and managing dependencies, Kubernetes for orchestrating containerized applications at scale.

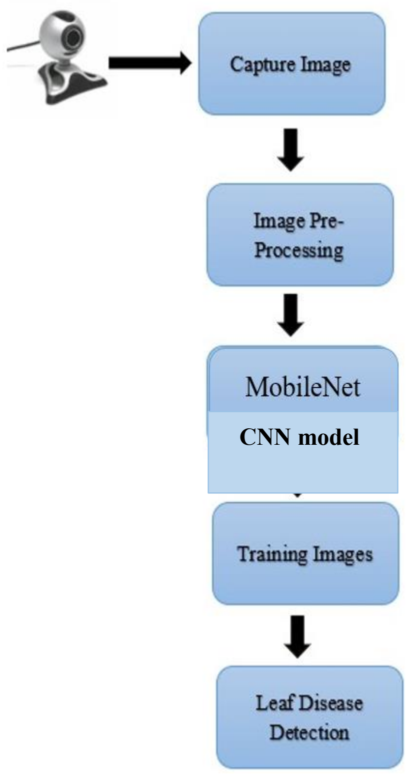
Chapter 4

PROJECT ARCHITECTURE

4.1SYSTEM ARCHITECURE



4.i) *Training the CNN model adapting to our dataset.*



4.ii) Predicting the new leaves

The system architecture of the rice leaf disease classification project follows a client-server model, with the client-side comprising the web interface for user interaction and the server-side handling the machine learning model inference. The client-side utilizes HTML templates rendered by the Flask web framework to provide a user-friendly interface for uploading images. On the server-side, TensorFlow/Keras is used to load and preprocess the images, while the trained convolutional neural network (CNN) model performs disease classification. The system architecture leverages cloud computing resources for scalability and efficiency, allowing seamless deployment and accessibility over the internet

Chapter 5

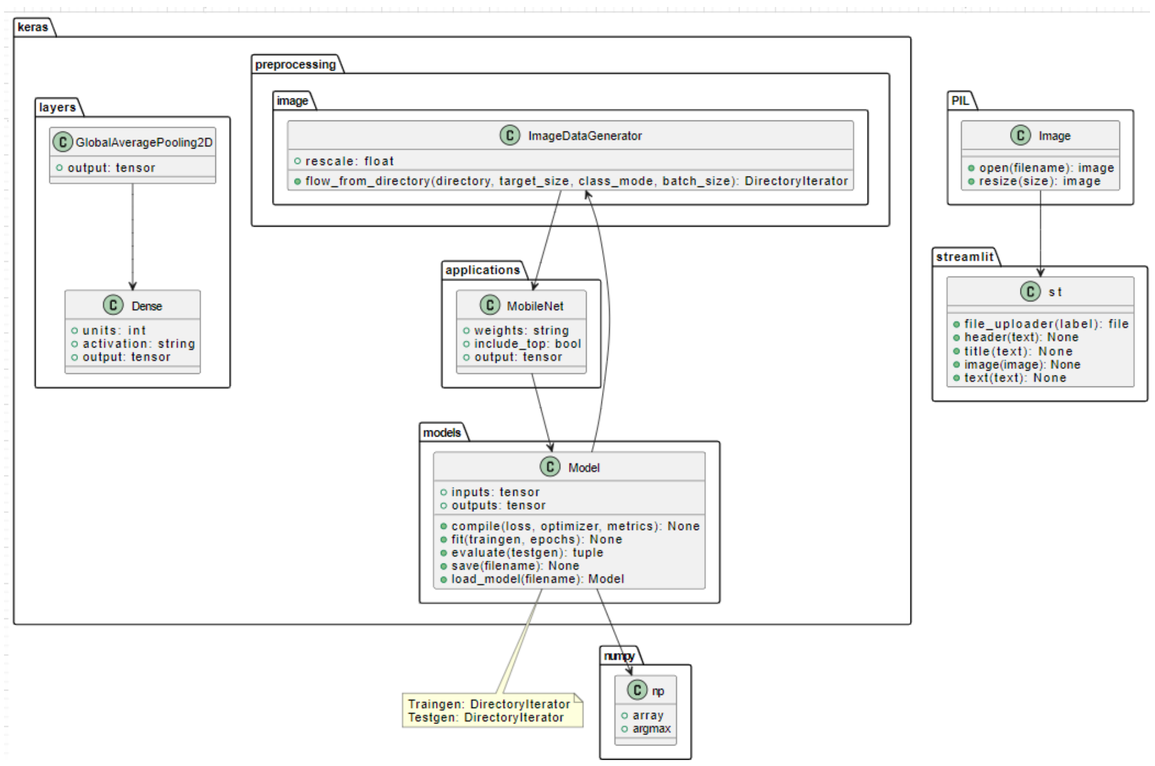
INTEGRATION OF CLASS AND OBJECT DIAGRAMS

The integration of class and object diagrams in a software project plays a crucial role in understanding the structure and behavior of the system's components. Class diagrams provide a static view of the system, depicting the classes, attributes, methods, and relationships between them. On the other hand, object diagrams provide a snapshot of the system at a particular point in time, showing instances of classes and the relationships between them.

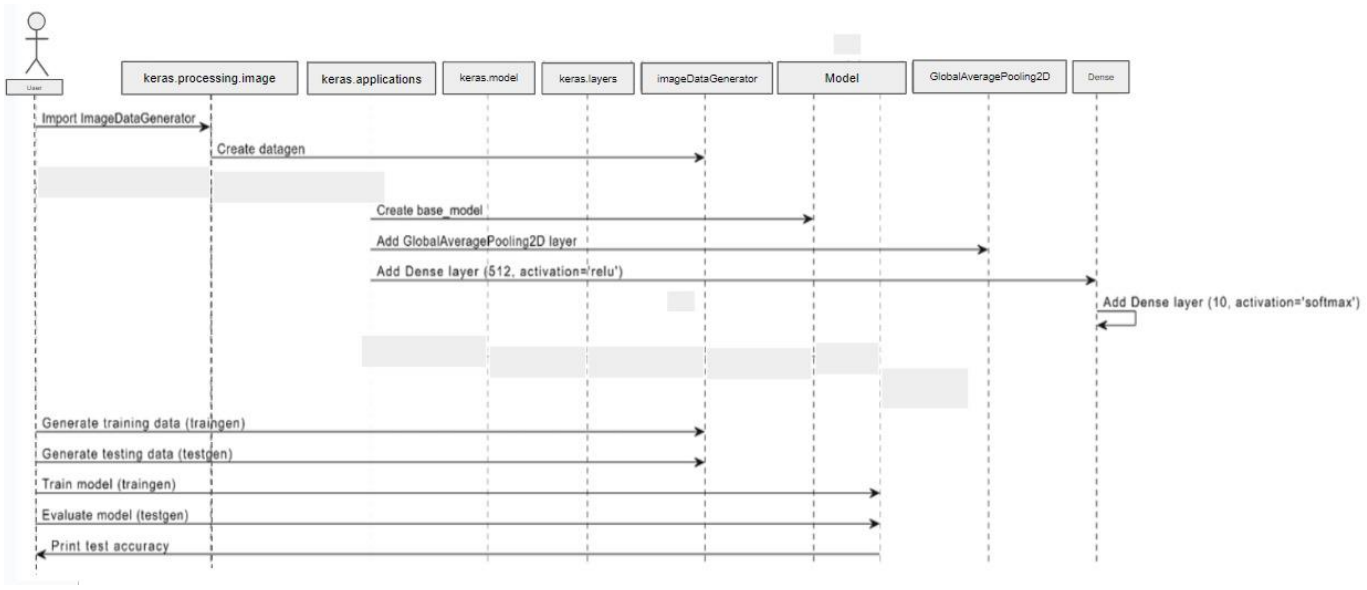
Integrating class and object diagrams involves creating a cohesive representation of the system's structure and dynamics. This integration helps developers and stakeholders visualize how the classes defined in the class diagram translate into actual objects instantiated during runtime. It also allows for a better understanding of how objects interact with each other and how data flows through the system.

By combining class and object diagrams, developers can gain insights into the system's design and implementation, identify potential design flaws or inconsistencies, and ensure that the system behaves as intended during runtime. This integration is particularly valuable in object-oriented software development, where understanding the relationships and interactions between objects is essential for building robust and maintainable systems.

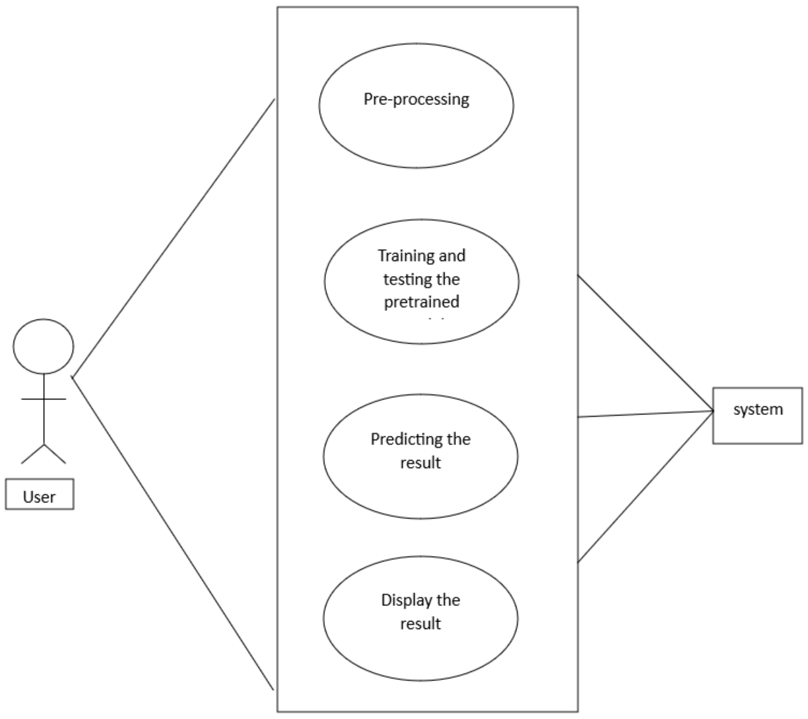
5.1 CLASS DIAGRAM



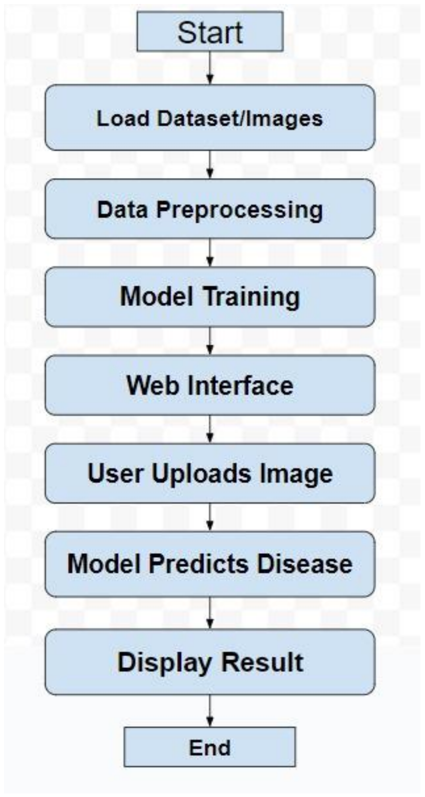
5.2 SEQUENCE DIAGRAM



5.3 USE CASE DIAGRAM



5.4 ACTIVITY DIAGRAM



In software development, various types of diagrams are utilized to depict different aspects of a system's architecture, behavior, and functionality. The class diagram offers a static view, showcasing the structure of the system through classes, attributes, methods, and their relationships. Sequence diagrams focus on dynamic interactions between objects or components, detailing the sequence of messages exchanged during a specific scenario or use case. Activity diagrams visualize the workflow or process flow within the system, highlighting activities, decisions, and parallel flows. Use case diagrams illustrate the interactions between actors and the system, identifying the functionalities offered and the roles of actors in achieving them. Together, these diagrams provide a comprehensive overview, facilitating communication among stakeholders, understanding system requirements, and guiding system design and implementation.

Chapter 6
METHODOLOGY

6.1 INTRODUCTION

This chapter delineates the methodology employed in the development of a robust deep learning model aimed at effectively classifying rice leaf diseases. Leveraging the power of TensorFlow and Keras, the methodology encompasses a systematic approach from data collection and preprocessing to model development, training, evaluation, and visualization of results.

Rice is a staple food for a significant portion of the world's population. However, various diseases can affect rice plants, leading to reduced yield and quality. Early detection and classification of these diseases are crucial for implementing timely interventions and ensuring sustainable rice production.

The objective of this project is to utilize convolutional neural networks (CNNs) to automate the process of rice leaf disease classification. By leveraging state-of-the-art deep learning techniques, the model aims to accurately identify and classify different types of diseases based on images of affected rice leaves.

The methodology is structured to ensure a comprehensive and systematic approach to model development and evaluation. It encompasses various stages, including data collection and preprocessing, model architecture design, training, evaluation, and visualization of results. Each stage is carefully designed to ensure the robustness, accuracy, and generalization of the developed model.

By following a rigorous methodology, this project seeks to contribute to the advancement of agricultural technology by providing farmers and agricultural experts with a reliable tool for early disease detection and management in rice crops.

6.2. Data Collection and Preprocessing

Data Collection:

6.2.1 Aggregation of Dataset Images:

The process of aggregating dataset images for this project was comprehensive and meticulous. It began with an extensive search across various repositories and sources to gather a diverse range of images depicting rice leaves affected by various diseases. Publicly available datasets from platforms like Kaggle, GitHub, and the UCI Machine Learning Repository were explored, providing a foundation of annotated images for rice leaf diseases.

In addition to publicly available datasets, collaborations were established with agricultural experts, research institutions, and domain specialists. These collaborations were instrumental in accessing specialized datasets containing high-quality images of rice leaf diseases. Through these partnerships, access to curated datasets with detailed annotations and expert knowledge was secured, enriching the diversity and authenticity of the dataset.

Each image was subjected to careful curation to ensure its relevance and accuracy in depicting specific rice leaf diseases. Quality control measures were implemented to filter out low-quality or irrelevant images, ensuring the integrity of the dataset. Emphasis was placed on sourcing images that accurately represented the visual manifestations of various rice leaf diseases, including bacterial leaf blight, blast, brown spot, leaf smut, and tungro, among others.

One of the key aspects of the aggregation process was the consolidation of images from multiple sources into a unified dataset. This merging process involved organizing and categorizing images based on disease type and severity. Images depicting similar diseases were grouped together to ensure comprehensive representation of each disease class within the dataset. Special attention was given to maintaining consistency in image resolution, format, and annotation across merged datasets, minimizing discrepancies and facilitating seamless integration into the training pipeline.

6.2.2 Merging of Dataset Images:

The merging of dataset images was a critical step in the data aggregation process, aimed at creating a cohesive and comprehensive dataset for model training. This process involved consolidating images from multiple sources into a unified dataset, ensuring comprehensive representation of each disease class and minimizing redundancy.

1. Dataset Compilation:

The process began with the compilation of images from various sources, including publicly available repositories, research databases, and collaborations with domain experts. Each dataset contained images depicting rice leaves affected by specific diseases, with varying levels of annotation and quality.

2. Identification of Similar Diseases:

Images depicting similar rice leaf diseases were identified across different datasets. Diseases such as bacterial leaf blight, blast, brown spot, leaf smut, and tungro were categorized based on visual similarities and disease characteristics.

3. Grouping and Categorization:

Once similar diseases were identified, images were grouped and categorized based on disease type and severity. This categorization ensured that images depicting the same disease were organized together, facilitating efficient management and analysis of the dataset.

4. Consistency Checks:

Special attention was given to maintaining consistency in image resolution, format, and annotation across merged datasets. Consistency checks were performed to verify that all images adhered to standardized specifications, minimizing discrepancies and ensuring data integrity.

5. Removal of Redundant Images:

Redundant or duplicate images were identified and removed from the merged dataset to avoid data duplication and bias. Quality control measures were implemented to filter out low-quality or irrelevant images, ensuring the overall quality of the dataset.

6. Annotation Alignment:

Annotations, if present, were aligned and standardized across merged datasets to ensure uniformity in disease labeling and classification. This alignment facilitated accurate model training and evaluation by providing consistent ground truth labels for each image.

7. Dataset Consolidation:

Finally, the merged dataset was consolidated into a cohesive and unified dataset, ready for preprocessing and model training. The consolidated dataset encompassed a diverse range of images depicting various rice leaf diseases, providing a comprehensive foundation for subsequent stages of the project.

The merging of dataset images was a meticulous and iterative process, involving careful organization, categorization, and quality control measures. By consolidating images from multiple sources into a unified dataset, the merging process ensured the diversity, authenticity, and integrity of the dataset, setting the stage for successful model development and evaluation

6.2.3 Generation of Synthetic Images:

In addition to curated dataset images, synthetic images were generated using advanced AI-powered image generation tools such as Microsoft Designer and Gencraft.ai. The generation of synthetic images was a crucial step aimed at augmenting the dataset and enhancing the diversity and richness of the training data.

1. Utilization of AI-Powered Tools:

AI-powered image generation tools leverage cutting-edge deep learning techniques, including generative adversarial networks (GANs) and variational autoencoders (VAEs), to create realistic synthetic images. These tools enable the generation of high-quality images that closely resemble real-world rice leaves affected by various diseases.

2. Training of Generative Models:

The image generation process begins with the training of generative models using a large corpus of real images as training data. During training, the generative models learn to capture the underlying patterns and characteristics of rice leaf diseases, enabling them to generate realistic synthetic images.

3. Diversity and Customization:

One of the key advantages of AI-powered image generation tools is their ability to produce diverse and customizable images. Users have the flexibility to specify parameters such as disease type, severity, leaf morphology, and environmental conditions, allowing for the generation of synthetic images tailored to specific requirements.

4. Validation and Quality Assurance:

The synthetic images generated by AI-powered tools undergo rigorous validation and quality assurance checks to ensure their realism and relevance to real-world rice leaf diseases. Validation processes may include visual inspection by domain experts, quantitative analysis of image features, and comparison with real images from the dataset.

5. Integration with the Dataset:

Once validated, the synthetic images are seamlessly integrated into the dataset, augmenting the existing collection of curated images. The synthetic images are categorized and labeled based on disease type and severity, ensuring consistency with the rest of the dataset.

6. Diversification of Training Data:

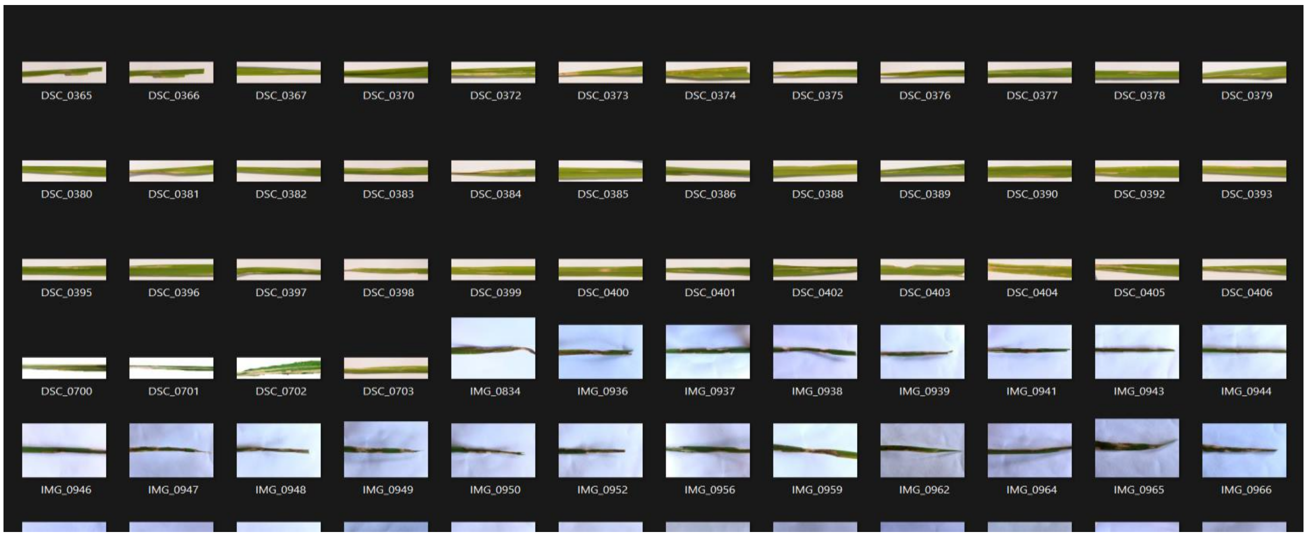
The addition of synthetic images diversifies the training data, introducing variability and complexity that enhances the model's ability to generalize to unseen data. By supplementing the dataset with synthetic images, the model becomes more robust and resilient to variations in disease presentation and environmental conditions.

7. Continuous Improvement:

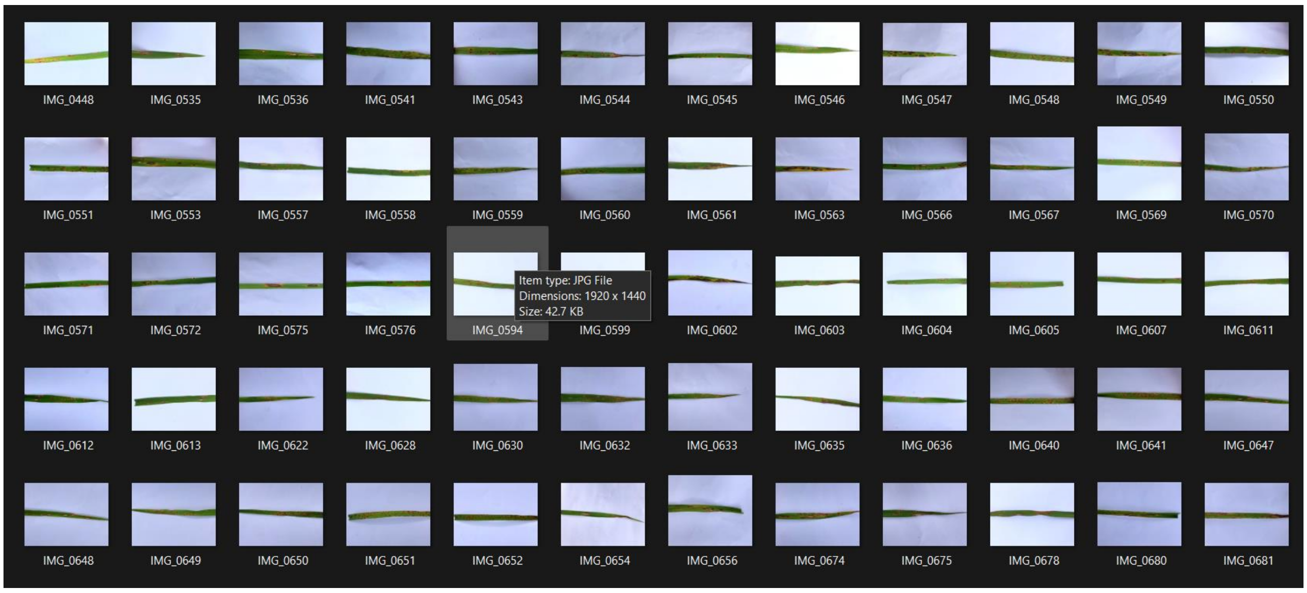
The generation of synthetic images is an iterative process that can be continuously refined and improved over time. As generative models are trained on larger and more diverse datasets, their ability to generate realistic synthetic images improves, leading to further enhancements in dataset quality and model performance.

The generation of synthetic images using AI-powered tools represents a powerful strategy for augmenting dataset diversity and enriching the training data for deep learning models. By leveraging advanced generative techniques, synthetic images contribute to the development of more robust and accurate models for rice leaf disease classification.

6.2.4 IMAGES OF DATASET



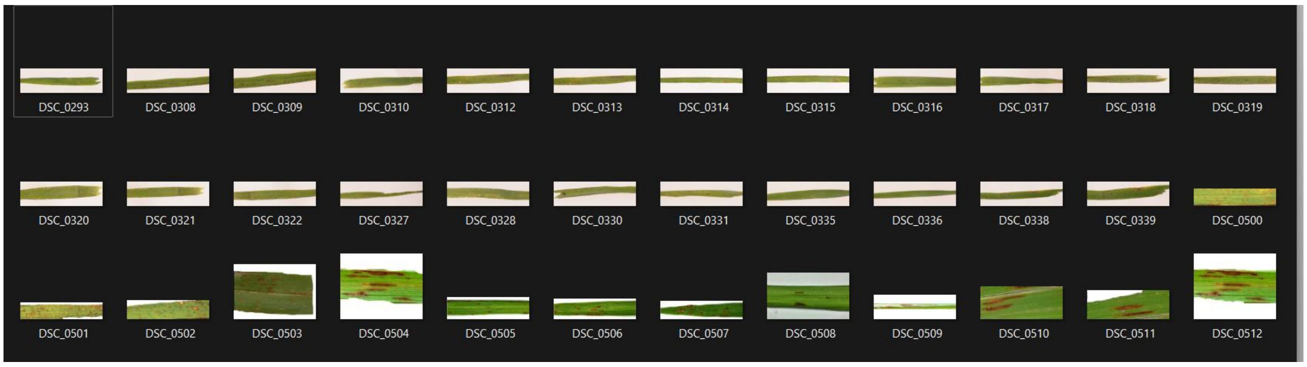
a)Bacterial Leaf Blight


IMG_0448 IMG_0535 IMG_0536 IMG_0541 IMG_0543 IMG_0544 IMG_0545 IMG_0546 IMG_0547 IMG_0548 IMG_0549 IMG_0550
IMG_0551 IMG_0553 IMG_0557 IMG_0558 IMG_0559 IMG_0560 IMG_0561 IMG_0563 IMG_0566 IMG_0567 IMG_0569 IMG_0570
IMG_0571 IMG_0572 IMG_0575 IMG_0576 IMG_0594 IMG_0599 IMG_0602 IMG_0603 IMG_0604 IMG_0605 IMG_0607 IMG_0611
Item type: JPG File
Dimensions: 1920 x 1440
Size: 42.7 KB
IMG_0612 IMG_0613 IMG_0622 IMG_0628 IMG_0630 IMG_0632 IMG_0633 IMG_0635 IMG_0636 IMG_0640 IMG_0641 IMG_0647
IMG_0648 IMG_0649 IMG_0650 IMG_0651 IMG_0652 IMG_0654 IMG_0656 IMG_0674 IMG_0675 IMG_0678 IMG_0680 IMG_0681

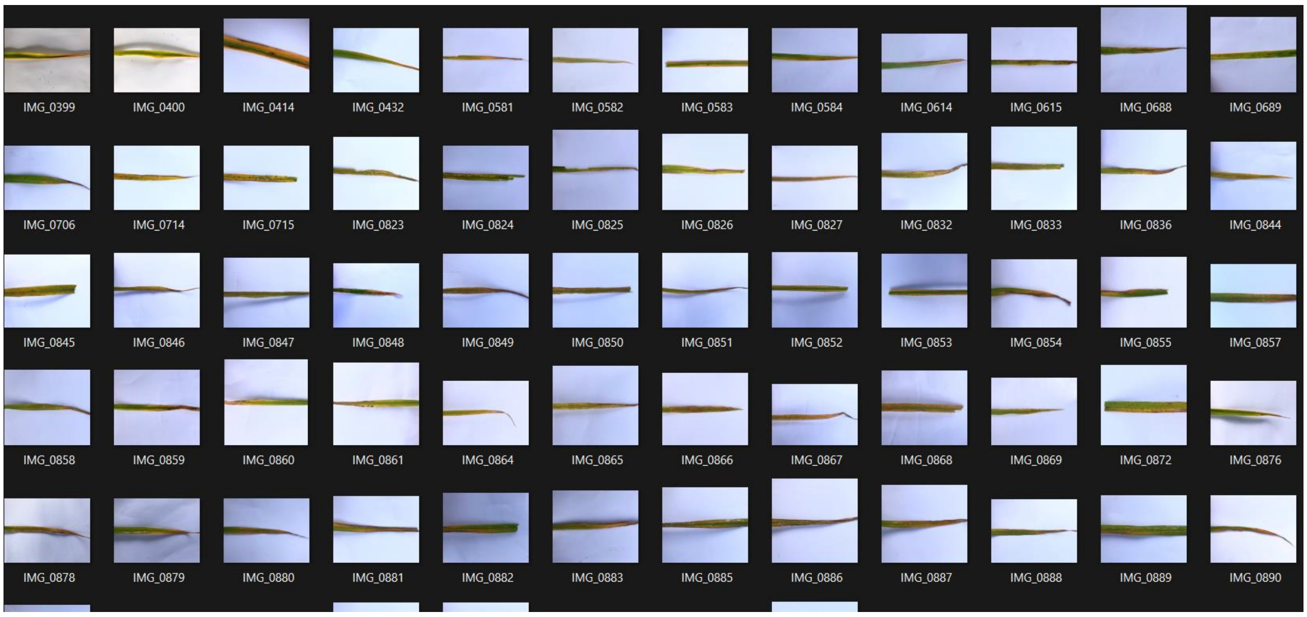
b)Blast

brownspot_orig_001 brownspot_orig_002 brownspot_orig_003 brownspot_orig_004 brownspot_orig_005 brownspot_orig_006 brownspot_orig_007 brownspot_orig_008 brownspot_orig_009 brownspot_orig_010 brownspot_orig_011 brownspot_orig_012
brownspot_orig_013 brownspot_orig_014 brownspot_orig_015 brownspot_orig_016 brownspot_orig_017 brownspot_orig_018 brownspot_orig_019 brownspot_orig_020 brownspot_orig_021 brownspot_orig_022 brownspot_orig_023 brownspot_orig_024
brownspot_orig_025 brownspot_orig_026 brownspot_orig_027 brownspot_orig_028 brownspot_orig_029 brownspot_orig_030 brownspot_orig_031 brownspot_orig_032 brownspot_orig_033 brownspot_orig_034 brownspot_orig_035 brownspot_orig_036
brownspot_orig_037 brownspot_orig_038 brownspot_orig_039 brownspot_orig_040 brownspot_orig_041 brownspot_orig_042 brownspot_orig_043 brownspot_orig_044 brownspot_orig_045 brownspot_orig_046 brownspot_orig_047 brownspot_orig_048

c)Brown Spot


DSC_0293 DSC_0308 DSC_0309 DSC_0310 DSC_0312 DSC_0313 DSC_0314 DSC_0315 DSC_0316 DSC_0317 DSC_0318 DSC_0319
DSC_0320 DSC_0321 DSC_0322 DSC_0327 DSC_0328 DSC_0330 DSC_0331 DSC_0335 DSC_0336 DSC_0338 DSC_0339 DSC_0500
DSC_0501 DSC_0502 DSC_0503 DSC_0504 DSC_0505 DSC_0506 DSC_0507 DSC_0508 DSC_0509 DSC_0510 DSC_0511 DSC_0512

d)Leaf Smut



e)Tungro

Data Preprocessing:

6.2.5 Image Resizing and Standardization:

Image resizing and standardization are essential preprocessing steps aimed at ensuring uniformity and compatibility of the dataset for model training. This process involves resizing all collected and generated images to a standardized dimension of (224, 224) pixels and standardizing pixel values to facilitate seamless integration into the training pipeline.

1. Uniform Image Dimensions:

Image resizing involves adjusting the dimensions of each image in the dataset to a standardized size, typically (224, 224) pixels. Standardizing image dimensions ensures uniformity and consistency across all images, regardless of their original size or aspect ratio.

2. Compatibility with Model Architecture:

Resizing images to a standardized dimension is crucial for compatibility with the model architecture. Many deep learning models, including convolutional neural networks (CNNs), require input images of fixed dimensions to operate efficiently. Resizing ensures that all images fed into the model have consistent dimensions, enabling seamless integration into the training process.

3. Preservation of Aspect Ratio:

During the resizing process, care is taken to preserve the aspect ratio of the original images. This ensures that the spatial relationships and proportions within the images remain intact, preventing distortion or skewing of the visual content.

4. Resampling Techniques:

Various resampling techniques may be employed during the resizing process to interpolate pixel values and adjust image resolution. Common techniques include nearest-neighbor interpolation, bilinear interpolation, and cubic interpolation. The choice of resampling technique depends on factors such as image content, computational efficiency, and desired output quality.

5. Pixel Value Standardization:

Following resizing, pixel values of the images are standardized to facilitate numerical stability during model training. Standardization typically involves scaling pixel values to a range of [0, 1] by dividing each pixel value by 255.0, which is the maximum pixel value for images represented in the RGB color space.

6. Numerical Stability:

Standardizing pixel values ensures numerical stability during model training and convergence. By scaling pixel values to a common range, computational issues such as vanishing or exploding gradients are mitigated, leading to more stable and reliable model training.

7. Compatibility with Pretrained Models:

Standardizing image dimensions and pixel values is particularly important when utilizing pretrained deep learning models or transfer learning techniques. Pretrained models are often trained on datasets with standardized inputs, making it essential to preprocess new data in a similar manner to ensure compatibility and optimal performance.

Image resizing and standardization are fundamental preprocessing steps that lay the foundation for robust and efficient model training. By ensuring uniformity, compatibility, and numerical stability, these preprocessing techniques contribute to the overall quality and effectiveness of the deep learning model for rice leaf disease classification.

6.2.6 Image Augmentation:

Image augmentation is a crucial preprocessing technique used to artificially increase the diversity and richness of the dataset by applying a variety of transformations to the existing images. These transformations introduce variations in the dataset, simulating real-world scenarios and improving the model's ability to generalize to unseen data.

1. Purpose of Image Augmentation:

The primary purpose of image augmentation is to increase the robustness of the model by exposing it to a wide range of variations in the input data. By augmenting the dataset with transformed versions of the original images, the model becomes more resilient to variations in lighting conditions, camera angles, and other environmental factors.

2. Types of Augmentation Techniques:

Image augmentation encompasses a variety of techniques, including:

Rotation: Rotating the image by a certain angle to simulate variations in camera orientation.

Width and Height Shifting: Shifting the image horizontally or vertically to simulate changes in perspective.

Shear: Applying shear transformation to the image to simulate skewing or distortion.

Zoom: Zooming in or out of the image to simulate variations in focal length.

Horizontal Flip: Flipping the image horizontally to create a mirror image.

Brightness and Contrast Adjustment: Modifying the brightness and contrast of the image to simulate changes in lighting conditions.

Noise Addition: Adding random noise to the image to simulate imperfections or artifacts.

3. Implementation using ImageDataGenerator:

Image augmentation is typically implemented using the ImageDataGenerator class from deep learning libraries such as TensorFlow or Keras. This class provides a convenient interface for applying various augmentation techniques to the dataset during model training.

Each image in the dataset is randomly transformed according to specified augmentation parameters, ensuring diversity and variability in the training data.

4. Benefits of Image Augmentation:

Image augmentation helps mitigate overfitting by exposing the model to a broader range of training examples. Introducing variations in the dataset prevents the model from memorizing specific patterns in the training data and encourages it to learn more robust and generalizable features.

Additionally, image augmentation enhances the model's ability to generalize to unseen data by simulating real-world variations that may be encountered during inference. This improves the model's performance on unseen images and enhances its practical utility in real-world applications.

5. Considerations and Best Practices:

When applying image augmentation, it's important to strike a balance between introducing variability and preserving the semantic content of the images. Excessive augmentation may distort or obscure important features, leading to degradation in model performance.

Careful selection of augmentation parameters, such as rotation angles, shift ranges, and noise levels, is essential to ensure that the augmented images remain representative of the original data distribution.

Image augmentation is a powerful technique for enhancing dataset diversity and improving model robustness. By simulating real-world variations in the training data, equips the model with the flexibility and adaptability needed to perform effectively in diverse environments.

6.2.7 Normalization:

Normalization is a crucial preprocessing step in deep learning, especially when dealing with image data. It involves scaling the pixel values of images to a standard range to ensure consistency and numerical stability during model training. In the context of image data, normalization typically involves scaling pixel values to lie within the range [0, 1] or [-1, 1].

1. Purpose of Normalization:

The primary purpose of normalization is to ensure that all input features (in this case, pixel values) contribute equally to the model's learning process. By scaling pixel values to a common range, normalization prevents features with larger magnitudes from dominating the learning process and ensures that the optimization algorithm converges more efficiently.

Additionally, normalization helps mitigate issues such as vanishing or exploding gradients, which can hinder model training by causing numerical instability. By constraining the range of input values, normalization promotes smoother and more stable gradient updates during the optimization process.

2. Scaling Pixel Values:

In the context of image data, pixel values typically range from 0 to 255 for each color channel (red, green, and blue) in the RGB color space. Normalization involves dividing each pixel value by the maximum value (255) to scale the values to the range [0, 1].

Alternatively, pixel values can be scaled to lie within the range [-1, 1] by subtracting the mean value and dividing by the standard deviation of the pixel values across the dataset. This zero-centered normalization approach helps center the data distribution around zero and facilitates convergence during model training.

3. Implementation:

Normalization is typically implemented as a preprocessing step using libraries such as TensorFlow or PyTorch. In TensorFlow, for example, the ***ImageDataGenerator*** class provides built-in support for rescaling pixel values during data augmentation.

Alternatively, normalization can be implemented manually by iterating over the dataset and applying the appropriate scaling transformation to each image.

4. Benefits of Normalization:

Improved Convergence: Normalization ensures that all input features are on a similar scale, promoting more stable and efficient convergence during model training.

Enhanced Generalization: By constraining the range of input values, normalization helps the model generalize better to unseen data by preventing overfitting and improving the model's ability to capture underlying patterns in the data.

Numerical Stability: Normalization mitigates issues such as vanishing or exploding gradients by constraining the range of input values, leading to more stable and reliable model training.

5. Considerations and Best Practices:

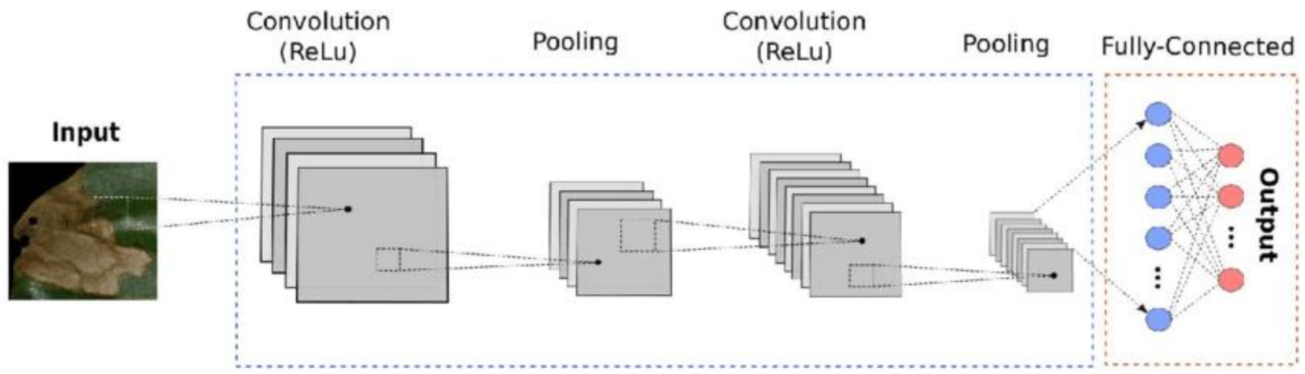
When normalizing image data, it's important to apply the same normalization parameters (e.g., mean and standard deviation) to both the training and validation/test datasets to ensure consistency.

Additionally, normalization should be performed after any other preprocessing steps, such as resizing or augmentation, to ensure that the data distribution remains consistent across all stages of model training and evaluation.

Normalization plays a crucial role in preparing image data for deep learning models, promoting stability, efficiency, and generalization. By scaling pixel values to a standard range, normalization ensures consistent and reliable model performance across diverse datasets and environments.

6.3 MODEL ARCHITECTURE

Within the model architecture designed for rice leaf disease classification, several key components interact to enable effective feature extraction, learning, and classification.



6.3.1 Convolutional Layers:

The foundational layers of the model are the convolutional layers, which serve as feature extractors by convolving learnable filters across input images. These filters are responsible for capturing spatial patterns and structures indicative of rice leaf diseases. The architecture comprises multiple convolutional layers, each progressively extracting higher-level features through hierarchical representations. The convolutional layers are followed by max-pooling layers, which reduce the spatial dimensions of the feature maps while retaining essential information.

6.3.2 Fully Connected Layers:

Following the convolutional layers, the flattened feature maps are passed through fully connected layers for classification. These dense layers facilitate high-level feature interpretation and decision-making by transforming the extracted features into class predictions. The architecture incorporates a dense layer with 512 neurons, followed by a dropout layer with a dropout rate of 0.5 to mitigate overfitting. The output layer consists of 5 neurons, each representing one of the five classes of rice leaf diseases. The softmax activation function computes the probability distribution over the classes, enabling multi-class classification.

6.3.3 Model Compilation and Optimization:

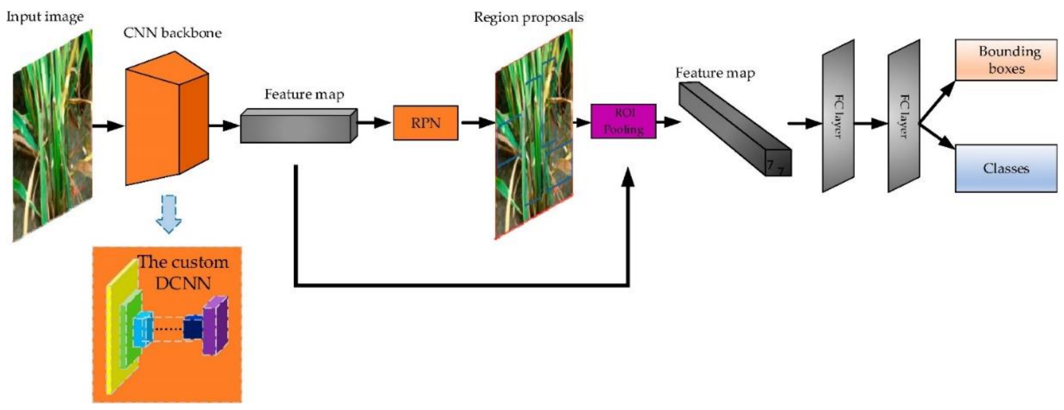
To facilitate model training, compilation, and optimization steps are employed. The Adam optimizer is utilized with a learning rate of 0.0001 to adaptively adjust the model parameters during training. This optimizer is well-suited for deep learning tasks and offers efficient convergence properties. The categorical cross-entropy loss function is employed to quantify the difference between predicted and actual class labels, providing an objective measure of model performance. By minimizing this loss function, the model learns to accurately classify rice leaf diseases.

6.3.4 Data Preprocessing and Augmentation:

Prior to model training, image data undergo preprocessing and augmentation to enhance dataset diversity and model robustness. The ImageDataGenerator class from TensorFlow/Keras is leveraged to perform data augmentation, which includes rescaling, rotation, shifting, shearing, zooming, and horizontal flipping. These augmentations introduce variations in the training data, enabling the model to generalize better to unseen images and environmental conditions.

6.3.5 Model Training and Evaluation:

The model is trained using the fit method, with training data generated by the train_generator and validation data generated by the validation_generator. Early stopping is implemented to monitor the validation loss and halt training if it does not improve for a specified number of epochs, thereby preventing overfitting. Once trained, the model's performance is evaluated using validation data, and metrics such as accuracy, confusion matrix, and classification report are computed to assess its efficacy in classifying rice leaf diseases accurately.



6.3.i)CNN in Rice leaf disease prediction

6.3.6 Parameter and Hyper-parameter Tuning:

Parameter and hyperparameter tuning are crucial processes in optimizing the performance of machine learning models, including the convolutional neural network (CNN) model used for rice leaf disease classification. Parameter tuning involves optimizing internal parameters like weights and biases, learned from training data, while hyperparameter tuning focuses on external parameters like learning rates and batch sizes, set before training. In the provided code, parameter tuning involves adjusting the CNN model architecture, while hyperparameter tuning optimizes settings such as learning rates and batch sizes using techniques like grid search or random search. These processes fine-tune the model for improved disease classification accuracy.

Chapter 7
INTEGRATION WITH WEB-APP

7.1 Integration with Web Application:

The integration of the rice leaf disease classification model with a web application represents a significant advancement in making machine learning accessible and user-friendly. This section delves deeper into the integration process, highlighting key components and functionalities of the web application.

7.2 Flask Application Setup:

The backbone of the integration is the Flask web framework, chosen for its simplicity and flexibility in building web applications with Python. The **app.py** file serves as the core of the Flask application, orchestrating the model loading, request handling, and result rendering processes.

7.3 Model Loading and Initialization:

At the heart of the integration lies the trained rice leaf disease classification model, saved as a Keras HDF5 file (**rice_leaf_disease_model_updated.h5**). The Flask application loads this model during initialization, leveraging the TensorFlow/Keras API. Additionally, a dictionary is constructed to map the numerical class indices to their corresponding disease labels, facilitating the interpretation of model predictions.

7.4 Image Preprocessing:

Before making predictions, the web application preprocesses the uploaded images to ensure compatibility with the model's input requirements. The **preprocess_image** function handles this preprocessing task, resizing the image to the required dimensions (224x224 pixels), converting it to a NumPy array, and normalizing the pixel values to a range between 0 and 1. This standardized preprocessing pipeline ensures consistency and reliability in model predictions.

7.5 Request Handling and Routing:

The Flask application defines two primary routes to handle user requests:

Home Route ('/'): This route renders the home page template (**index.html**), providing users with an intuitive interface to upload rice leaf images for classification.

Classification Route ('/classify'): Upon receiving an image upload via a POST request, this route triggers the classification process. It preprocesses the uploaded image, makes predictions using the loaded model, and renders the result page template (**result.html**) with the predicted disease class label.

7.6 User Interface Design:

The user interface (UI) of the web application is meticulously designed to offer a seamless and intuitive experience for users. The home page (**index.html**) features a user-friendly interface for uploading images, while the result page (**result.html**) presents the classification outcome in a clear and visually appealing manner.

7.6.1 index.html code:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <style>

    body{

  background-image:                    url('https://images.unsplash.com/photo-1604213410393-
89f141bb96b8?q=80&w=2070&auto=format&fit=crop&ixlib=rb-
4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D');

  font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu,
Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;

  background-size: cover;
```

```
}


.main{
  background-size: cover;


}
.heading{
    border-radius: 12px;
    height: 20vh;
    text-align: center;
    font-size: 30px;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;
    color: rgb(255, 255, 255);
    font-weight: 500;
    background: transparent;
    backdrop-filter: blur(2px);
    border: 1px solid rgba(128, 128, 128, 0);
}


.container{
    display: flex;
    justify-content: space-evenly;
    align-items: center;
```

```
}


.box-1{
   height: 500px;
   width: 600px;
   display: flex;
   flex-direction: column;
   justify-content: center;
   align-items: center;
   border-radius: 8px;
   background: transparent;
   backdrop-filter: blur(8px);
   border: 1px 0px 0px 1px solid gray;
}
.result{
   font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;
   color: white;
   font-weight: bold;
   font-size: 36px;
}
label{
   font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;
   color: white;
```

```
    font-weight: bold;

    font-size: 36px;

}

input[type="file"] {

    /* Add your styles here */

    border: 2px solid #ccc;

    padding: 10px;

    border-radius: 5px;

    font-size: 26px;

    color: white;

  }


/* result container styling */

.box-2{

    height: 500px;

    width: 600px;

    background: transparent;

    backdrop-filter: blur(8px);

    border: 1px solid rgba(128, 128, 128, 0);

    border-radius: 8px;

    display: flex;

    flex-direction: column;

    align-items: center;

}
```

```
/* result container styling */

.button {
 --color: #eee8e8;
 padding: 0.8em 1.7em;
 background-color: transparent;
 border-radius: .3em;
 position: relative;
 overflow: hidden;
 cursor: pointer;
 transition: .5s;
 font-weight: 400;
 font-size: 17px;
 border: 2px solid;
 font-family: inherit;
 text-transform: uppercase;
 color: var(--color);
 z-index: 1;
}

.button::before, .button::after {
 content: '';
 display: block;
 width: 50px;
 height: 50px;
 transform: translate(-50%, -50%);
 position: absolute;
 border-radius: 50%;
 z-index: -1;
 background-color: var(--color);
 transition: 1s ease;
}

.button::before {
 top: -1em;
 left: -1em;
}

.button::after {
 left: calc(100% + 1em);
 top: calc(100% + 1em);
}

.button:hover::before, .button:hover::after {
 height: 410px;
 width: 410px;
}
```

```
.button:hover {
 color: rgb(15, 14, 14);
 font-weight: bold;
}

.button:active {
 filter: brightness(.8);
}

/* result.html styling */
    </style>
    <title>Rice Leaf Disease Classification</title>
</head>
<body>
    <div class="main">
        <div class="heading">
            <h1>Rice Leaf Disease Classification</h1>
        </div>
        <div class="container">
            <form class="box-1" action="/classify" method="post" enctype="multipart/form-data">
                <div>
                    <label for="image">Upload an image:</label>
                    <br >
                <input type="file" name="image" accept="image/*" required>
```

```
          </div>

          <br>

          <button class="button" type="submit">Classify</button>

        </form>

        <div class="box-2" id="result-container" style="display: none;">

          <h2 class="result">Result:</h2>

          <p id="result-message"></p>

        </div>

      </div>


    </div>

    <script>

      // Display result message when received from the server

      const resultContainer = document.getElementById('result-container');

      const resultMessage = document.getElementById('result-message');


      // Function to show the result

      function showResult(message) {

        resultContainer.style.display = 'block';

        resultMessage.textContent = message;

      }

    </script>

  </body>
```

```
</html>
```

7.6.2 result.html Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Rice Leaf Disease Classification Result</title>
   <style>
      body{
   background-image: url('https://plus.unsplash.com/premium_photo-1664117187580-c48528437e04?q=80&w=2070&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D');
   background-size: cover;
   /* height: 100%;
   display: flex;
   flex-direction: column;
   justify-content: center; */
}

.main-container{

   padding: 10px;
```

```
    border: 0px solid ;
    border-radius: 10px;
    text-align: center;
    color: white;
    word-spacing: 3px;
    letter-spacing: 3px;
    background: rgba(0, 0, 0, 0.411);
    margin: 20px;
    backdrop-filter: blur(50px);
    border: 2px solid rgb(255, 251, 251);
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;
}
strong{
    font-size: large;
    color: black;
    font-weight: bold;
    letter-spacing: 0px;
}
.box-container{
    font-family:'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
    padding: 10px;
    border: 0px solid ;
    border-radius: 10px;
    background-image: linear-gradient(to right, rgb(212, 160, 232), rgb(226, 253, 145));
    text-align: center;

    color: white;

    background: rgba(0, 0, 0, 0.386);

    margin: 20px;

    backdrop-filter: blur(50px);

    border: 2px solid rgba(255, 255, 255, 0.989);

    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;

}

li{

    list-style: none;

}

.back-btn{

    padding: 15px;

    margin: 10px;

    border-radius: 4px;

    height: 100px;

    text-decoration: none;

    width: 300px;

    background-color: black;

    color: antiquewhite;

    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva,
Verdana, sans-serif;

}

    </style>
```

```
</head>
<body>
  <div class="main-container">
    <h1>Rice Leaf Disease Classification Result</h1>
    <p>The predicted class is: <strong>{{ class_label }}</strong></p>
  </div>
  <!-- Bio Fertilizer Making Process based on predicted disease class -->
  {% if class_label == 'Bacterial leaf blight' %}
    <div class="box-container">
      <h1>Bio Fertilizer Making Process for Bacterial Leaf Blight</h1>
    <ul>
      <li>Step 1: Collect organic waste such as kitchen scraps, plant trimmings, and manure.</li>
      <li>Step 2: Mix the organic waste with equal parts water and allow it to ferment for 2-3 weeks
in a covered container.</li>
      <li>Step 3: Strain the fermented mixture and dilute it with water in a 1:3 ratio before using it
as fertilizer for your rice plants.</li>
    </ul>
    </div>
  {% elif class_label == 'Blast' %}
  <div  class="box-container">
    <h2>Bio Fertilizer Making Process for Blast</h2>
    <ol>
      <li>Step 1: Collect dried leaves and stems of Neem (Azadirachta indica) and crush them into
small pieces.</li>
```

```
            <li>Step 2: Mix the crushed Neem leaves and stems with water and allow it to soak for 24 hours.</li>
            <li>Step 3: Filter the mixture and dilute it with water in a 1:5 ratio before using it as a foliar spray to control the blast disease.</li>
        </ol>
    </div>
    {% elif class_label == 'Brown spot' %}
    <div  class="box-container">
        <h2>Bio Fertilizer Making Process for Brown Spot</h2>
        <ol>
            <li>Step 1: Collect compost or well-rotted manure from your compost pile.</li>
            <li>Step 2: Mix the compost or manure with water in a 1:5 ratio to create a nutrient-rich liquid fertilizer.</li>
            <li>Step 3: Apply the liquid fertilizer to the soil around the base of your rice plants to promote healthy growth and resistance to brown spot disease.</li>
        </ol>
    </div>
    {% elif class_label == 'Leaf smut' %}
    <div  class="box-container">
        <h2>Bio Fertilizer Making Process for Leaf Smut</h2>
        <ol>
            <li>Step 1: Collect fresh cow dung and mix it with water in a 1:10 ratio to create a slurry.</li>
            <li>Step 2: Allow the slurry to ferment for 7-10 days in a shaded area.</li>
            <li>Step 3: Dilute the fermented slurry with water in a 1:3 ratio and apply it to the soil around the base of your rice plants to prevent leaf smut disease.</li>
        </ol>
    </div>

    {% elif class_label == 'Tungro' %}

    <div  class="box-container">

        <h2>Bio Fertilizer Making Process for Tungro</h2>

        <ol>

            <li>Step 1: Collect rice straw and soak it in water for 5-7 days to initiate decomposition.</li>

            <li>Step 2: Add nitrogen-rich materials such as poultry manure or green manure to the decomposing rice straw to speed up the decomposition process.</li>

            <li>Step 3: Once the mixture has decomposed into a dark, crumbly compost, spread it around the base of your rice plants as a natural fertilizer to prevent tungro disease.</li>

        </ol>

    </div>

    {% endif %}


    <p><a  class="back-btn" href="/">Go back</a></p>

</body>

</html>
```

7.7 Deployment Considerations

Once developed, the Flask web application can be deployed on various hosting platforms to make it accessible to users worldwide. Deployment options range from traditional web hosting services to cloud platforms such as Heroku or AWS. Containerization solutions like Docker offer additional flexibility and scalability for deployment in diverse environments.

7.8 Flask Code:

```
from flask import Flask, render_template, request
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing import image

app = Flask(__name__)

# Load the trained model
model = load_model('rice_leaf_disease_model_updated.h5')

# Dictionary to map class indices to class names
class_labels = {0: 'Bacterial leaf blight', 1: 'Blast', 2: 'Brown spot', 3: 'Leaf smut', 4: 'Tungro'}


# Function to preprocess the image
def preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array /= 255.0  # Normalize the image
    return img_array
```

```
# Route for the home page
@app.route('/')
def home():
    return render_template('index.html')


# Route to handle the classification request
@app.route('/classify', methods=['POST'])
def classify():
    try:
        # Get the image file from the request
        img_file = request.files['image']

        # Save the image to a temporary file
        temp_img_path = 'temp_img.jpg'
        img_file.save(temp_img_path)

        # Preprocess the image
        img_array = preprocess_image(temp_img_path)

        # Make prediction
        prediction = model.predict(img_array)

        # Get the predicted class index
        predicted_class_index = np.argmax(prediction)
        # Get the predicted class label
        predicted_class_label = class_labels[predicted_class_index]


        # Render the result page with the predicted class label
        return render_template('result.html', class_label=predicted_class_label)


    except Exception as e:
        return str(e)


if __name__ == '__main__':
    app.run(debug=True)
```

The **app.py** file serves as the entry point for your Flask application. Within this file:

Flask Initialization: You initialize a Flask instance using **Flask(__name__)**, setting up the foundation for your web application.

Model Loading: The trained rice leaf disease classification model is loaded using TensorFlow/Keras's **load_model** function. This step ensures that the model is ready to make predictions based on user-uploaded images.

Preprocessing Function: The **preprocess_image** function is defined to preprocess the uploaded images before passing them to the model for classification. This function resizes the images to the required dimensions, converts them to NumPy arrays, and normalizes the pixel values.

Route Definitions: Two routes are defined to handle user requests:

The home route (**'/'**) renders the home page template (**index.html**), where users can upload rice leaf images for classification.

The classification route (**'/classify'**) handles the classification process. It receives the image file via a POST request, preprocesses the image, makes predictions using the loaded model, and renders the result page template (**result.html**) with the predicted disease class label.

Error Handling: Exception handling is implemented to catch any errors that may occur during the classification process. If an error occurs, the user is notified with an error message.

Application Execution: The **if __name__ == '__main__':** block ensures that the Flask application runs when the script is executed directly. The application runs in debug mode (**debug=True**), allowing for easier debugging and development.

In summary, this Flask code seamlessly integrates machine learning capabilities into a web application, providing users with a convenient and intuitive interface to classify rice leaf images and obtain instant results. Through the combination of Flask, TensorFlow/Keras, and HTML templates, this web application empowers users to make informed decisions in agricultural settings.

Chapter 8

SYSTEM TESTING

8.1 Importance of Testing

Testing the model is a pivotal phase in the development of any machine learning system, including the convolutional neural network (CNN) model tailored for rice leaf disease classification. In this critical stage, the aim is to rigorously evaluate the model's performance across a diverse set of scenarios, ensuring its robustness, generalization capabilities, and reliability in real-world applications.

8.2 Dataset Splitting:

To embark on testing, the dataset undergoes meticulous splitting into distinct subsets, typically comprising training, validation, and testing sets. The testing set, isolated from the training and validation sets, serves as the litmus test for the model's efficacy post-training.

8.3 Evaluation Metrics:

A plethora of evaluation metrics come into play to comprehensively gauge the model's performance. These metrics encompass traditional measures such as accuracy, precision, recall, and F1-score, offering nuanced insights into the model's classification prowess across diverse classes.

8.4 Testing Procedure:

Testing unfolds through a structured procedure where the trained model ingests images from the testing set, generating predictions for each image. These predictions are juxtaposed against ground truth labels to compute various performance metrics, shedding light on the model's accuracy, sensitivity, specificity, and overall performance characteristics.

8.5 Visualization Techniques:

Moreover, the deployment of visualization techniques plays a pivotal role in elucidating the model's decision-making processes. Visualizations, such as confusion matrices, highlight the model's performance across different classes, unveiling potential misclassifications or confusions between closely related disease categories.

Additionally, leveraging advanced visualization techniques like heatmaps or saliency maps can uncover the critical regions within input images that significantly influence the model's predictions.

Such insights are invaluable in understanding the model's behavior and identifying areas for refinement.

8.6 Cross-Validation:

Furthermore, the utilization of cross-validation methodologies, such as k-fold cross-validation, adds an extra layer of robustness to the testing process. By systematically training and evaluating the model on distinct subsets of the data, cross-validation provides a more comprehensive assessment of the model's performance, mitigating the risk of overfitting and ensuring its generalization capabilities.

Chapter 9

IMPLEMENTATION & CODING

9.1 Mounting Google Drive

```
from google.colab import drive

# Mount Google Drive

drive.mount('/content/drive')
```

This code imports the `drive` module from the `google.colab` package. Then, it mounts Google Drive to the Colab environment, allowing access to files stored in Google Drive.

9.2 MODEL TRAINING

```
import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import EarlyStopping

import numpy as np

from sklearn.metrics import classification_report, confusion_matrix

import seaborn as sns

import matplotlib.pyplot as plt


# Set the path to your dataset

dataset_path = '/content/drive/MyDrive/rice_leaf_diseases_dataset'


# Set up the model

model = models.Sequential()
```

```
# Convolutional layers

model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(224, 224, 3)))

model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Conv2D(256, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Conv2D(512, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))


model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dropout(0.5))

model.add(layers.Dense(5, activation='softmax'))  # Updated to 5 classes


# Compile the model

model.compile(optimizer=Adam(lr=0.0001),

              loss='categorical_crossentropy',

              metrics=['accuracy'])
```

```
# Data preprocessing using ImageDataGenerator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=45,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(224, 224),
    batch_size=64,
    class_mode='categorical',
    subset='validation'
)
# Implement Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
# Train the model
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator,
    callbacks=[early_stopping]
)
# Save the model
model.save('rice_leaf_disease_model_updated.h5')
# Evaluate the model
validation_generator.reset()
y_pred = model.predict(validation_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = validation_generator.classes
# Compute confusion matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)
# Compute classification report
class_report = classification_report(y_true, y_pred_classes)
```

```
# Print evaluation results
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=validation_generator.class_indices.keys(),
            yticklabels=validation_generator.class_indices.keys())
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

This code utilizes TensorFlow and Keras to implement a convolutional neural network (CNN) for classifying rice leaf diseases. It defines a CNN architecture with convolutional and pooling layers. The model is compiled with the Adam optimizer and categorical cross-entropy loss. Data augmentation is performed using ImageDataGenerator. Early stopping is implemented to prevent overfitting. The model is trained and evaluated using validation data. Finally, it saves the model, evaluates it using confusion matrix and classification report, and visualizes the confusion matrix using seaborn and matplotlib.

9.3 MODEL EVALUATION

```
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
# Assuming you have validation data stored in validation_generator
# Get predictions for the validation data

y_pred = model.predict(validation_generator)

y_pred_classes = np.argmax(y_pred, axis=1)

# Get true labels

y_true = validation_generator.classes

# Compute accuracy

accuracy = np.mean(y_true == y_pred_classes)

print("Accuracy:", accuracy)

# Compute confusion matrix

conf_matrix = confusion_matrix(y_true, y_pred_classes)

print("Confusion Matrix:")

print(conf_matrix)

# Compute classification report

class_report = classification_report(y_true, y_pred_classes)

print("Classification Report:")

print(class_report)
```

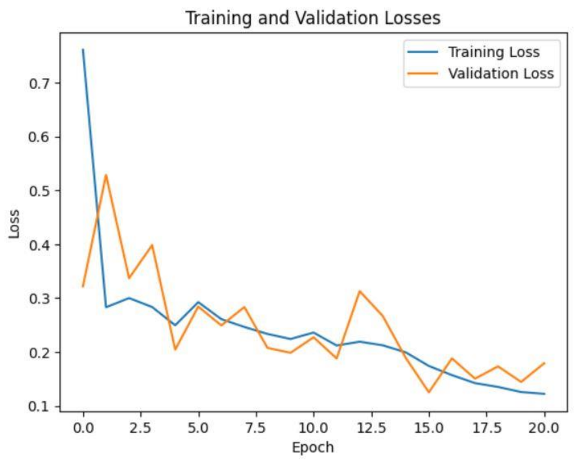
This code calculates the accuracy of the model by comparing predicted classes with true labels. Then, it computes the confusion matrix and the classification report using scikit-learn's functions. Finally, it prints the accuracy, confusion matrix, and classification report.

9.4 Training and Validation Loss Curves

```
import matplotlib.pyplot as plt

# Assuming you have stored the training history in the variable 'history'

# Get training and validation losses

train_loss = history.history['loss']

val_loss = history.history['val_loss']
```

```
# Plot the training and validation losses

plt.plot(train_loss, label='Training Loss')

plt.plot(val_loss, label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.title('Training and Validation Losses')

plt.legend()

plt.show()
```

This code plots the training and validation losses over epochs using matplotlib. It retrieves the training and validation losses from the 'history' variable and then plots them. The x-axis represents the epochs, and the y-axis represents the loss values. Finally, it adds labels, title, legend, and displays the plot.



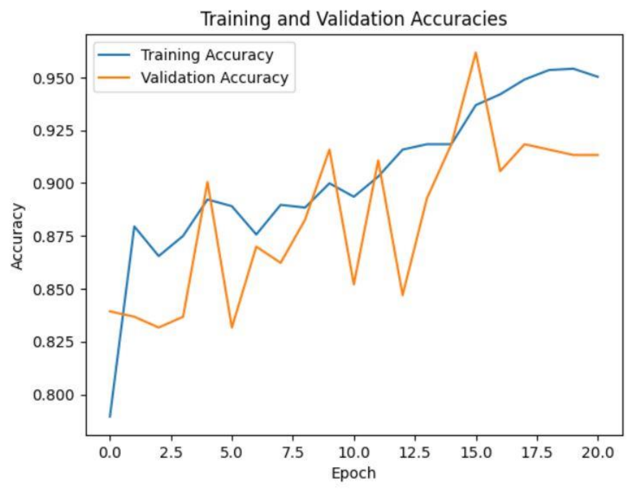
9.4.i)Training and Validation Loss Curves

9.5 Training and Validation Accuracy Curves

```
# Get training and validation accuracies

train_acc = history.history['accuracy']

val_acc = history.history['val_accuracy']


# Plot the training and validation accuracies
```

```
plt.plot(train_acc, label='Training Accuracy')

plt.plot(val_acc, label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.title('Training and Validation Accuracies')

plt.legend()

plt.show()
```

This code snippet retrieves the training and validation accuracies from the 'history' variable. It then plots these accuracies over epochs using matplotlib, with the x-axis representing epochs and the y-axis representing accuracy values. Finally, it adds labels, title, legend, and displays the plot, illustrating the training and validation accuracies.



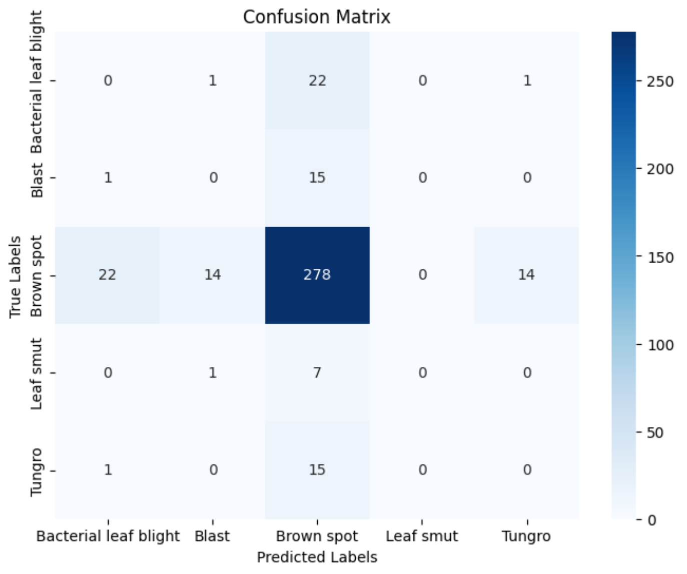
9.5.i)Training and Validation Accuracy Curves

9.6 Confusion Matrix

```
import seaborn as sns

# Dictionary to map class indices to class names

class_labels = {0: 'Bacterial leaf blight', 1: 'Blast', 2: 'Brown spot', 3: 'Leaf smut', 4: 'Tungro'}

# Plot confusion matrix

plt.figure(figsize=(8, 6))
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_labels.values(),
            yticklabels=class_labels.values())
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

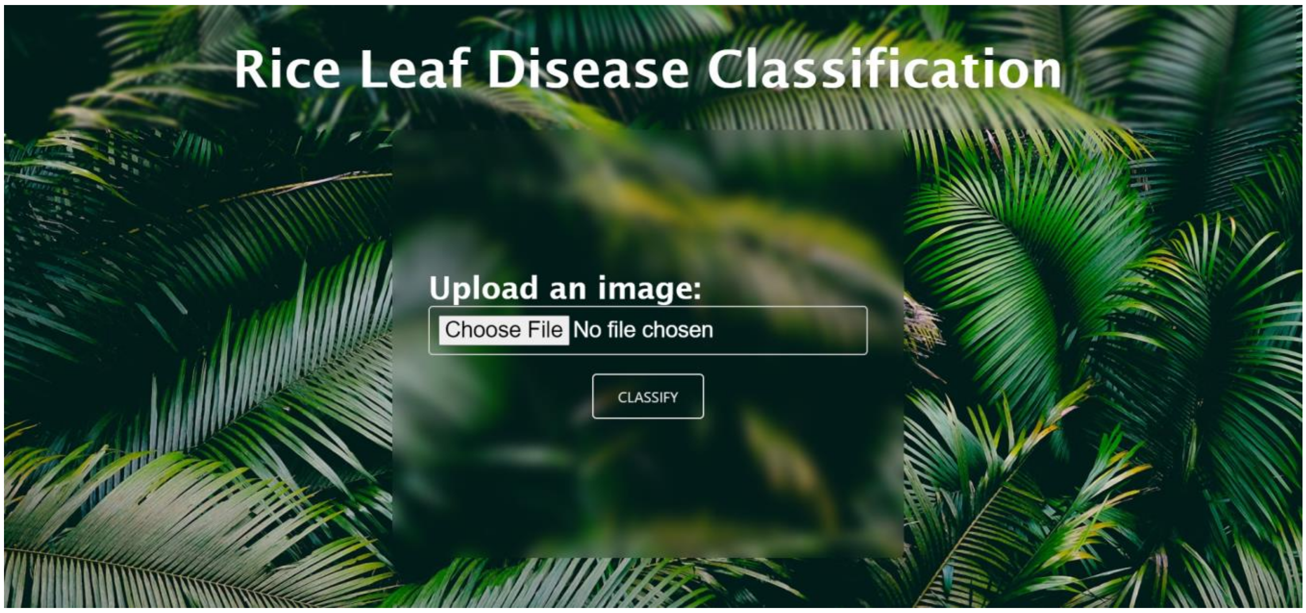
This code creates a heatmap visualization of the confusion matrix using seaborn and matplotlib. It plots the confusion matrix with annotations, using a blue color scheme ('Blues'). The x-axis represents predicted labels, while the y-axis represents true labels. Class labels are provided for both axes using the dictionary 'class_labels'. Finally, it adds labels, title, and displays the heatmap.



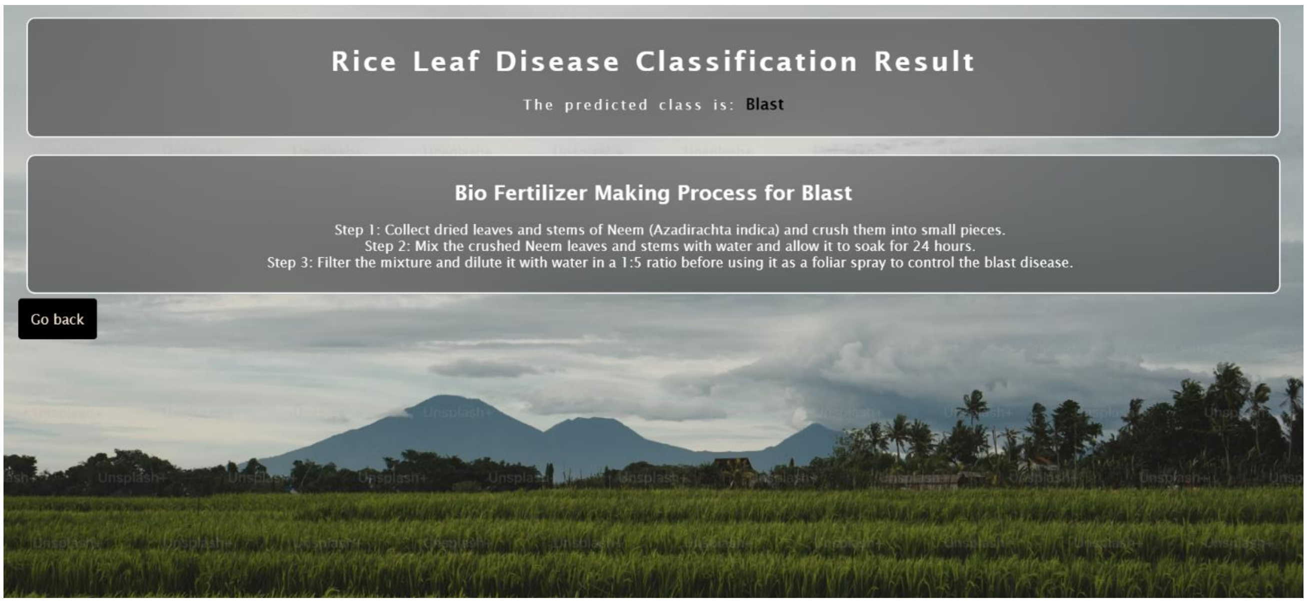
9.6.i)Confusion Matrix

Chapter 10

OUTPUT SCREENS



10.i)Home Page



Rice Leaf Disease Classification Result

The predicted class is: Blast

Bio Fertilizer Making Process for Blast

Step 1: Collect dried leaves and stems of Neem (Azadirachta indica) and crush them into small pieces.
Step 2: Mix the crushed Neem leaves and stems with water and allow it to soak for 24 hours.
Step 3: Filter the mixture and dilute it with water in a 1:5 ratio before using it as a foliar spray to control the blast disease.

10.ii)Result Page

Chapter 11

CONCLUSION & FUTURE SCOPE

1. CONCLUSION

Classification of four different rice leaf diseases was done utilising a few DL techniques. We utilized the images of the leaves with various diseases which are different in nature compared to each other, which we then analysed using a variety of well-known deep learning techniques, including VGG19, VGG16, Xception, Resnet, as well as a custom Five layered convolutional network

The accuracy of our suggested Five layered CNN model is roughly 6% higher than that of the other common deep learning models. Additionally, we discovered that by modifying the training parameters, such as the learning rate, the number of epochs, and the optimizer techniques. A handmade model can also give a high accuracy rate with fewer layers than conventional models. It will be easier for farmers to secure their crops the more effectively we can detect illnesses.

2. FUTURE SCOPE

We shall increase the reach of disease detection in the future so that it is widespread, simple, and quick. Creating extremely effective detection methods using massive datasets of various plant leaf diseases will have a critical future influence. By requiring substantial generalized datasets, this would also address the class disparity. The only thing done in this paper is recognition; moving forward, we'll further widen the recovery procedures. We would also attempt to put into practice the feature to offer remedies for the ailments affecting rice leaves. To a considerable extent and to prevent the influence on food yield, this would be helpful to the farmer.

Chapter 12

REFERENCES & BIBLIOGRAPHY

Qi, H., Liang, Y., Ding, Q., & Zou, J. (2021). Automatic Identification of PeanutLeaf Diseases Based on Stack Ensemble. Applied Sciences, 11(4), 1950.

Rahman, C. R., Arko, P. S., Ali, M. E., Khan, M. A. I., Apon, S. H., Nowrin, F., & Wasif, A. (2020). Identification and recognition of rice diseases and pests using convolutional neural networks. Biosystems Engineering, 194, 112-120.

Ramakrishnan, M. (2015, April). Groundnut leaf disease detection and classification by using backpropagation algorithms. In 2015 international conference on communications and signal processing (ICCSP) (pp. 0964 0968). IEEE.

Ramesh, S., Hebbar, R., Niveditha, M., Pooja, R., Shashank, N., & Vinod, P. V. (2018, April). Plant disease detection using machine learning. In 2018 International Conference on Design Innovations for 3Cs compute communicate control (ICDI3C) (pp. 41-45). IEEE.

R Salini, A.J Farzana, B Yamini, (2021). Pesticide Suggestion and Crop Disease Classification using Machine Learning, 63(5), 9015-9023.

https://www.youtube.com/results?search_query=ric e+leaf+disease+detection+using+cnn+ppt

https://www.researchgate.net/publication/34824311 2_Rice_Leaf_Diseases_Recoition_Using_Convolutional_Neural_Networks

https://link.springer.com/chapter/10.1007/978-3 030-65390-3_23