ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 19, Issue 2, 2025

PADDY CROP DISEASE DETECTION USING CNN

Mrs. Gokavaram Prasanna[1] , B. Sneha Latha[2] , C.Maheswari[3] , S.Pujitha[4] , Y.Sandhya[5] , D.Harini[6]

[1]Assistant Professor , Dept of CSE, Gouthami Institute Of Technology and Management for Women, Andhra Pradesh, India

[2,3,4,5,6]U.G Students, Dept of CSE, Gouthami Institute Of Technology and Management for Women, Andhra Pradesh, India

ABSTRACT

Early detection of diseases in paddy crops is vital for preventing yield loss and ensuring food security. Traditional methods like visual inspection and lab testing are time-consuming and prone to errors. This project proposes an image processing-based machine learning approach using Convolutional Neural Networks (CNNs) to detect and classify rice diseases, including Rice Blast, Bacterial Leaf Blight, Sheath Blight, and healthy leaves. By analyzing a dataset of 3,671 labeled images, the model efficiently identifies diseases in rice plants captured via mobile cameras. Leveraging the TensorFlow Inception V3 model, it provides timely disease detection, helping farmers take necessary actions and improve crop productivity. To address this critical issue, we developed a model capable of identifying and arranging diseases in affected plants using images captured a mobile camera.

INTRODUCTION

Rice is a crucial global crop, especially in countries like India, Bangladesh, and China, providing sustenance and driving economic growth. As the global population is projected to reach 9.7 billion by 2050, rice remains a key food source for over half of the world's population. However, increasing demand for rice faces challenges from climate change, which impacts soil, air, and food quality, leading to significant production losses from paddy diseases, causing up to a 40% reduction in annual yields. The integration of advanced technologies like computer vision and machine learning is transforming precision agriculture, enabling early and accurate detection of diseases, which helps reduce yield losses, optimise resource use, and minimise environmental damage, making it a sustainable solution. These technologies also offer accessible agronomic advice, benefiting farmers without advanced infrastructure or expert access.

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

LITERATURE SURVEY

Traditional methods of rice disease diagnosis relied on manual inspection by experts or agricultural officers, who visually examined symptoms on leaves, stems, and grains to identify diseases and determine their severity. These approaches were labor-intensive, subjective, and prone to inaccuracies due to human error and varying interpretations. To improve efficiency, early digital image processing techniques, such as color and texture-based segmentation, were introduced to isolate affected plant areas. However, methods like histogram equalization and edge detection were still sensitive to environmental factors, like lighting changes and background noise, limiting their effectiveness in real-world conditions.

As rule-based systems had their limitations, machine learning techniques began to gain traction for rice disease detection. Algorithms like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Decision Trees were applied, but these models relied on manually extracted features, reducing their accuracy in complex settings. The introduction of Convolutional Neural Networks (CNNs) marked a breakthrough, enabling automatic learning of features directly from raw images without the need for manual extraction. Early studies, like Mohanty et al.'s (2016) work using CNNs on the PlantVillage dataset, highlighted the potential of deep learning. This led to more refined models, such as Lu et al.'s (2017) deep CNN framework for rice leaf disease classification, which achieved 95.6% accuracy, and Zhang et al.'s (2018) hybrid CNN model, combining image segmentation and classification for better detection of subtle disease symptoms.

EXISTING METHODS

To standardize rice disease diagnosis and overcome the limitations of manual inspection, image processing techniques have been introduced in agriculture. These methods enable digital analysis of rice leaf images, extracting features such as color, texture, and shape to differentiate between healthy and diseased samples. Traditional image processing pipelines commonly use techniques like histogram equalization for contrast enhancement, edge detection (e.g., Sobel or Canny operators), and segmentation methods like thresholding or region growing to isolate affected areas. Additionally, texture analysis using the Gray Level Co-occurrence Matrix (GLCM) and color feature extraction in various color spaces (e.g., RGB, HSV, YCbCr) are employed, with these features then used in classical machine learning algorithms for classification.

Classical machine learning techniques, such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, and Random Forests, have been extensively applied to rice leaf disease detection. These models rely on handcrafted features, including GLCM-based texture features. While SVMs are effective for detecting diseases like Brown Spot and Leaf Blast, and k-NN works well on smaller datasets, they often struggle with generalizing to new data. Random Forests offer better accuracy, particularly in noisy datasets, but require significant feature engineering and fine-tuning. Hybrid methods, combining image processing with machine learning (e.g., K-

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

means clustering for segmentation followed by Naive Bayes or Logistic Regression classifiers), have emerged to enhance performance. However, these methods still face challenges with high intra-class variance and poor lighting conditions, limiting their suitability for real-time or large-scale applications.

PROPOSED METHOD

The proposed method leverages a deep learning-based system using Convolutional Neural Networks (CNNs) to detect rice leaf diseases, offering a scalable, accurate, and user-friendly solution for identifying diseases like Brown Spot, Hispa, Leaf Blast, and Bacterial Leaf Blight. The system uses the powerful Inception V3 CNN architecture, which excels in learning complex patterns and distinguishing disease-specific features. By incorporating transfer learning, the model adapts effectively to rice disease datasets, even with limited labeled images. This approach also includes a comprehensive data preprocessing pipeline and a robust deployment strategy, making it suitable for real-world use by farmers.

To build the system, a diverse and high-quality dataset is crucial. Rice leaf images, both healthy and diseased, are captured using mobile cameras and drones under varying environmental conditions to ensure robustness. Datasets like the Rice Leaf Disease Dataset and the Plant Village Dataset are also used, with each image labeled into categories such as healthy or diseased. Given that raw field data often includes lighting issues, background noise, and resolution inconsistencies, a thorough preprocessing phase is employed. This includes resizing images to 299x299 pixels, normalizing pixel values, and applying techniques like Gaussian blur, median filtering, and segmentation methods such as K-means clustering and Otsu's thresholding to isolate diseased areas for accurate model training.

SOFTWARE REQUIREMENTS:

• **Operating System :** Windows 8 and above
• **Coding Language** : Python 3.12.0
• **Framework** : Django
• **Platform** : Visual Studio Code (Preferable)

HARDWARE REQUIREMENTS:

• **System** : MINIMUM i3 and above
• **Hard Disk** : 40 GB. (min)
• **RAM**: 4 GB. (min)

SOFTWARE ENVIRONMENT

What is Python:

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

Advantages of Python:

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation- generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable add **scripting capabilities** to our code in the other language. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Bryton is that it isn't that secure.

Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors.**

UnderdevelopedDatabase Access Layers

Compared to more widely used technologies like **JDBC (Java database Connectivity)**and **ODBC(Open Database Connectivity** it is less often applied in huge enterprises.

Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python:

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde end Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime.

What is Flask?

Flask is a lightweight and flexible Python web framework known for its simplicity and ease of use. It enables developers to create robust web applications with minimal setup and a clean design. Flask adopts a modular approach and provides the essential tools needed to build web apps, leaving the choice of additional libraries to the developers. This review explores Flask's core principles, functionalities, and advantages, making it an ideal starting point for developers who prefer a minimalist framework.

A Glimpse into Flask's Philosophy

Flask was developed by Armin Ronacher as part of the Pocoo project in 2010. Its minimalist approach is guided by the philosophy of being "micro but extensible." Flask avoids imposing unnecessary restrictions, giving developers full control over application design. This flexibility has made Flask a popular choice for projects ranging from small prototypes to large-scale production applications.

Flask operates on the "microframework" principle, meaning it provides only the core features necessary for web development, such as routing, request handling, and template rendering. For additional functionalities like database integration or user authentication, developers can incorporate third-party extensions, keeping the application lightweight and customizable.

One of Flask's most notable features is its "unopinionated" nature, allowing developers to structure applications as they see fit. Unlike frameworks with strict architectural patterns, Flask offers the freedom to implement custom architectures based on project requirements.

Key Features of Flask

Flask's functionality can be summarized as follows:

1. **Routing and URL Mapping**: Flask simplifies URL routing with the @app.route decorator. Developers can map URLs to specific view functions effortlessly, creating intuitive and organized endpoints.

2. **Request and Response Handling**: Flask provides tools for handling HTTP methods (GET, POST, PUT, etc.) and working with request data, cookies, and headers. The request object allows seamless interaction with incoming data.

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

2. **Jinja2 Templating**: Flask integrates Jinja2, a powerful templating engine, for dynamic content rendering. It supports template inheritance, filters, and expressions, making it easy to create reusable and interactive UI components.

3. **Blueprints for Modular Applications**: Flask introduces "Blueprints," a way to organize applications into smaller, reusable modules. Blueprints promote maintainability and scalability in larger projects.

4. **Extensions**: Flask's lightweight core can be augmented with extensions like:

Flask-SQL Alchemy for database interaction

Flask-WTF for form validation

Flask-Login for user authentication

Flask-Migrate for database migrations

5. **Built-in Development Server and Debugger**: Flask includes a built-in development server with debugging tools, enabling developers to test and debug applications efficiently during development.

6. **RESTful API Support**: Flask's flexibility makes it an excellent choice for building RESTful APIs. Libraries like Flask-RESTful and Flask-Smorest provide additional utilities for API development.

Need for Flask :

Simplicity and Flexibility:

Minimalist Design: Flask's lightweight nature allows developers to create web apps quickly and without unnecessary overhead. **Customizable**: Developers have the freedom to choose their preferred tools and libraries, tailoring the framework to their project's needs.

Scalability:

Flask's modular structure and extensibility make it scalable for small to medium-sized applications. Larger applications can use Blueprints to maintain a clean codebase.

Ideal for Prototyping:

Flask's simplicity and quick setup make it perfect for building prototypes and proof-of-concept applications.

Challenges in Flask

1. **Manual Configuration**: Flask requires developers to manually set up features like user authentication, database connections, and migrations, which can be time-consuming for larger projects.

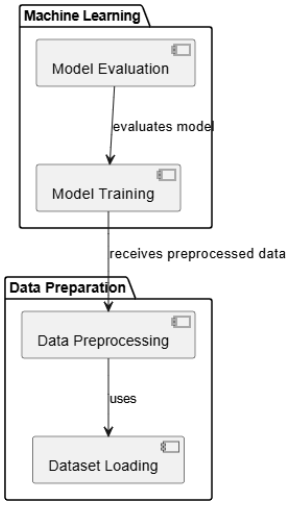
2. **Lack of Built-in Features**: Unlike Django, Flask does not include out-of-the-box solutions for common functionalities, requiring reliance on third-party extensions.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

SYSTEM TEST :

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

ARCHITECTURE



The working method for paddy crop disease detection using Convolutional Neural Networks (CNN) involves several key stages. First, a dataset of paddy leaf images—comprising healthy and diseased samples—is collected and preprocessed through techniques like resizing, normalization, and augmentation to improve model generalization. Next, the CNN model is designed or adopted using architectures such as VGG, ResNet, or custom lightweight models, and trained on the labeled dataset to learn distinguishing features of different diseases directly from image patterns. During training, the model automatically extracts hierarchical features through convolutional, pooling, and fully connected layers. After sufficient training and validation, the model is tested on unseen images to evaluate its accuracy, precision, recall, and F1-score. Finally, the trained CNN can be deployed to a user interface or mobile application to classify new paddy leaf images in real-time, enabling early disease detection and timely intervention for improved crop management.

RESULTS

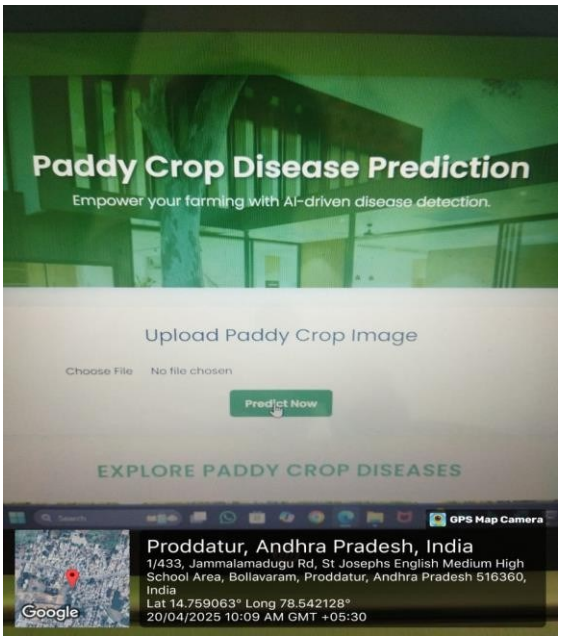


Fig: Home page of Paddy crop disease detection using CNN

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

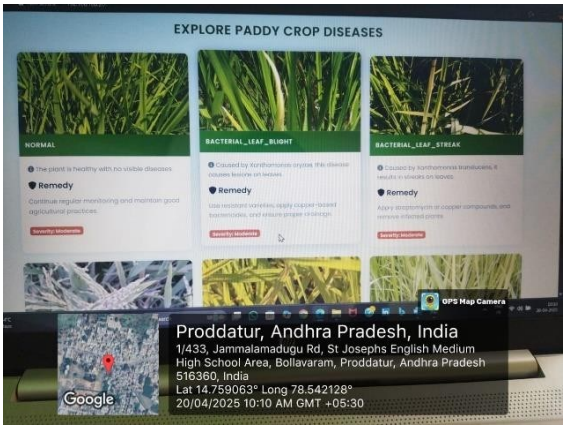


Fig : Explore paddy crop diseases

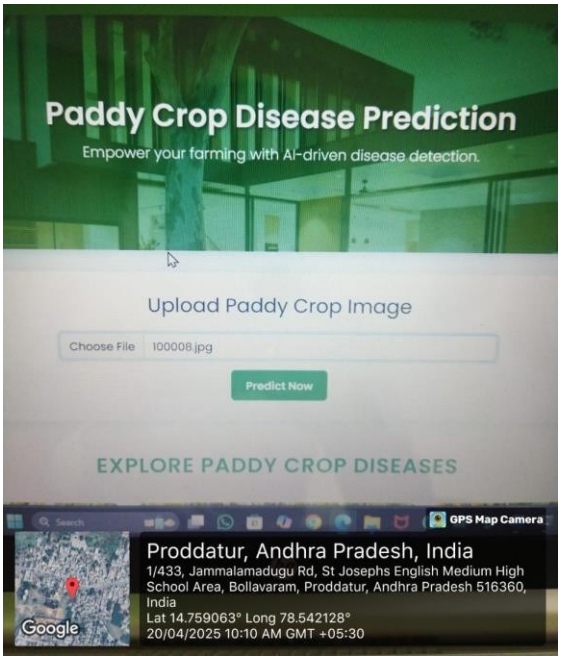


Fig : Uploading paddy crop image

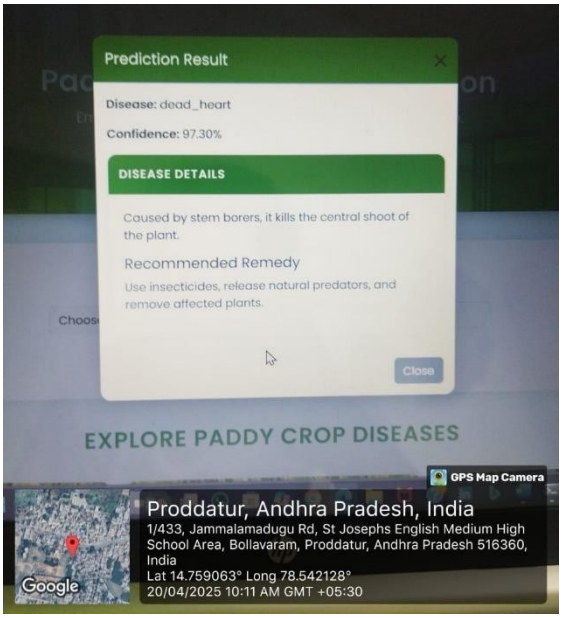


Fig : Prediction Result of disease

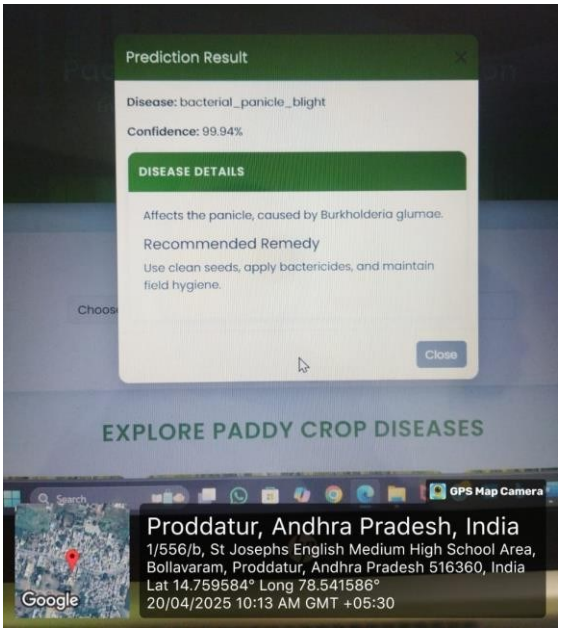


Fig : Prediction result of another crop disease

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

CONCLUSION

The growing demand for food due to population growth and climate change highlights the need for innovative solutions to tackle agricultural challenges, particularly crop diseases. Rice, a staple for over half the global population, is especially vulnerable to disease outbreaks, which can result in significant yield losses and affect food security, farmer livelihoods, and economies. The "Rice Leaf Disease Detection Using CNN" project aims to leverage deep learning, specifically Convolutional Neural Networks (CNNs) with the Inception V3 architecture and transfer learning, to create an automated, scalable solution for identifying common rice leaf diseases like Brown Spot, Hispa, Leaf Blast, and Bacterial Leaf Blight. This approach improves on traditional methods by automatically extracting features from raw image data, enhancing classification accuracy even in varied conditions. By using advanced preprocessing, data augmentation, and model evaluation techniques, the system offers a reliable tool for early disease detection and timely intervention, making it particularly valuable in regions with limited high-quality annotated datasets.

REFERENCES

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012) introduced deep convolutional neural networks for ImageNet classification, achieving significant breakthroughs in image recognition.
2. Simonyan, K., & Zisserman, A. (2015) proposed Very Deep Convolutional Networks for large-scale image recognition, enhancing the depth and accuracy of CNN models.
3. He, K., Zhang, X., Ren, S., & Sun, J. (2016) developed deep residual learning for image recognition, introducing residual networks (ResNet) to improve training deep networks.
4. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015) demonstrated human-level control through deep reinforcement learning, advancing AI decision-making capabilities.
5. Zhang, Y., & Chen, K. (2021) provided a comprehensive survey on convolutional neural networks for agricultural crop disease detection, highlighting advancements in crop disease identification.
6. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016) applied deep learning for image-based plant disease detection, setting the foundation for automated agricultural diagnostics.