WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Plant Disease Detection Using Machine Learning

***Arindam Gupta[1], N Mohammed Akhil[1], Kavya Jain[1], Adithi B V Rao[2], Yashaswini K V[3], G Vijaya Kumar[3]***

[1]R.V. College of Engineering, Department of AIML.
[2]R.V. College of Engineering, Department of Aerospace.
[3]R.V. College of Engineering, Department of Biotechnology.
arindamgupta.ai23@rvce.edu.in,kavyajain.ai23@rvce.edu.in,
nmohammedakhil.ai23@rvce.edu.in,adithibvrao.ai23@rvce.edu.in,
yashaswinikv.bbt22@rvce.edu.in , vijayakg@rvce.edu.in

**ABSTRACT**

*Plant diseases pose a significant threat to global agriculture, leading to substantial economic losses and reduced crop yields. Timely and accurate diagnosis is critical to ensuring effective crop management and sustainability. In this study, we propose a deep learning-based plant disease recognition system leveraging the EfficientNetB4 convolutional neural network architecture. The system is trained and fine-tuned using the Plant Village dataset, achieving high accuracy in classifying 39 distinct classes of diseased and healthy plant leaves. To make the solution accessible to end-users, we deploy the model through a web application built using Flask, integrated with HTML, CSS, and JavaScript. The web interface allows users to upload leaf images and instantly receive disease classification results along with treatment suggestions. This comprehensive approach combines high-performance deep learning with user-centric design, offering a scalable and efficient tool for real-time plant disease diagnosis. EfficientNetB4 is a type of Convolutional Neural Network (CNN) architecture developed by Google AI. It is a part of the EfficientNet family, which is known for achieving high accuracy while being computationally efficient (i.e., less memory, faster).*

**Keywords:** Plant Disease Detection, Agricultural Technology, Image Classification, Machine Learning Applications

## I. INTRODUCTION

Agriculture forms the backbone of the global economy, particularly in developing nations, where a majority of the population depends on farming for livelihood. However, plant diseases significantly hinder crop production, leading to economic instability and food insecurity. Traditional disease detection methods rely heavily on manual inspection by experts, which is not only time-consuming but also impractical for large-scale deployment, especially in rural or underserved regions. With the rapid advancement of deep learning and computer vision, automated plant disease recognition has emerged as a promising solution. Leveraging convolutional neural networks (CNNs), it is now possible to classify plant diseases from leaf images with remarkable accuracy. Among the various architectures, EfficientNetB4 stands out for its balanced trade-off between accuracy and computational efficiency. In this research, we propose a robust plant disease recognition system utilizing the EfficientNetB4 model, trained on the publicly available Plant Village dataset. The system is further integrated into a user-friendly web application built using Flask, enabling users to diagnose plant diseases by simply uploading a leaf image. The app instantly predicts the disease category and provides information about the cause and suggested cure, making it a practical tool for farmers, agronomists, and researchers alike. This paper discusses the data preprocessing pipeline, model architecture, training methodology, and deployment strategy, highlighting the system's performance and real-world applicability.

## II. LITERATURE SURVEY

Plant disease detection and plant health detection are of extreme significance in agriculture since they directly affect crop yields and food security. In order to make the disease detection process more efficient and accurate, different methods have been employed, ranging from the conventional to the latest technological developments. This survey outlines the major contributions made to this field.

### 1. Conventional Disease Detection Methods:

Human examination by trained individuals has in the past been utilized to detect diseases in plants. Early research, for instance, Agrios (2005), presented a focus on visual signs, i.e., leaf spots, yellowing, and wilting, as an indication of plant disease. Although these methods are precise, they are labour-intensive, consume a lot of time, and are prone to human mistakes, especially in commercial agricultural production.

**2. Image-Based Detection Methods:**

As computer vision evolved, image processing techniques increasingly became an aspect of plant disease detection. Previously, edge detection and colour segmentation were utilized for the detection of apparent symptoms on plant leaves. For example, Abdullah et al. (2012) created a system with colour analysis to detect regions on leaves affected. With the advent of deep learning, this method has now been transformed, and Convolutional Neural Networks (CNNs) are now used on a worldwide scale for feature extraction and classification. Mohanty et al. (2016) showed the capability of CNNs to classify plant diseases at over 99% accuracy on public databases.

**3. Dataset:**

The Plant Village dataset is a public dataset used most commonly in the area of plant disease and pest detection. The dataset was produced to aid machine learning and computer vision model development for the purposes of detecting diseases in plants via images. It is hosted at Kaggle and includes images of healthy and ill plant leaves spanning several types of crops.

## III. PROPOSED SYSTEM FLOW

The proposed system for plant disease detection follows a structured workflow comprising data collection, model development, training, evaluation, and deployment.

**1. Data Collection and Preprocessing:**

The Plant Village dataset is used for training, validation, and testing, with images categorized accordingly. The Plant Village dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease. All images are resized to 160×160 pixels to maintain uniform input dimensions. The dataset is loaded efficiently using TensorFlow's 'image_dataset_from_directory' utility, facilitating streamlined batching and preprocessing.

**2. Model Architecture:**

A pretrained EfficientNetB4 model, initialized with ImageNet weights, is used as the base architecture for feature extraction. A Global Average Pooling layer is added, followed by a dense classification layer tailored to the number of plant disease categories. Initially, the base model is frozen to train only the new classification

layers. Subsequently, fine-tuning is performed on deeper layers to improve classification accuracy.

**3. Model Training and Evaluation:**

The model is trained over multiple epochs with validation data to monitor performance. Training and validation accuracy and loss metrics are recorded and visualized to assess convergence and detect overfitting. The final evaluation is conducted using the test dataset to measure generalization performance.

**4. Model Deployment:**

The trained model is saved in Keras '.keras' format for easy integration into applications. A Flask-based web application is developed to serve the model for real-time inference. The frontend interface is built using HTML, CSS, and JavaScript to provide a responsive and interactive user experience. Users can upload an image and receive predictions along with the disease cause and suggested remedies.

**5. Web Application Features:**

The interface supports both drag-and-drop and file selection methods for image uploads. Real-time inference is performed using the trained model to identify plant diseases. The application displays the predicted disease name, its cause, and possible cures. The web app is mobile-friendly and can be accessed over a local IP, making it suitable for use in field conditions.

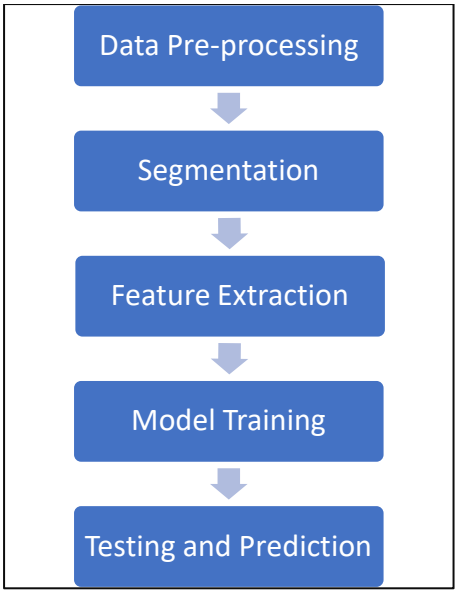


Figure 1.0: Methodology for Image Processing

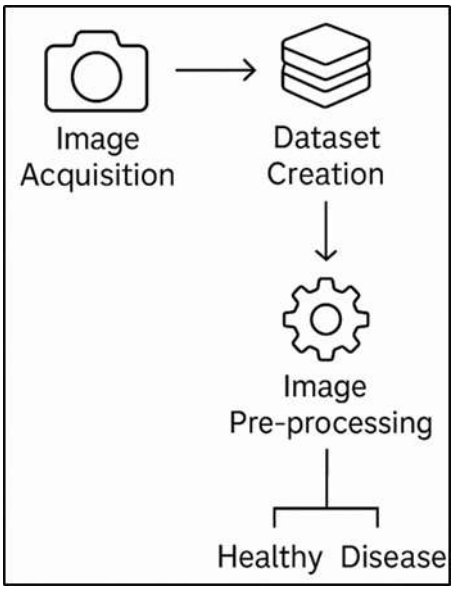


Figure 2.0: Working of the model

**Technologies Used**

Development and deployment of the plant disease detection system involved the use of a variety of programming tools and environments. Jupyter Notebook was majorly adopted for data analysis, model training, and performance assessment to provide an interactive environment for experimentation. Visual Studio Code (VS Code) was used as the integrated development environment (IDE) for backend scripting and web development. Node.js was used for server-side operations and to manage asynchronous tasks effectively. The interface of the web application was created using HTML, CSS, and JavaScript to provide a responsive and user-friendly interface. Moreover, several Python libraries were employed during the project for preprocessing the data, model creation, and result visualization.

**Procedure**

Plant disease detection system development was conducted via a structured and multi-stage approach to guarantee efficacy and real-time applicability. The process initiated with the download and structuring of the PlantVillage dataset, which holds more than 54,000 images of leaves from different types of crops labelled with healthy and diseased categories. The images were resized equally to 160×160 pixels to ensure consistency in input and minimize computational demand.

The dataset was read and handled through TensorFlow's ‘image_dataset_from_directory’ function, simplifying the preprocessing pipeline. Data was divided into training, validation, and test sets to increase model accuracy and prevent overfitting. A pre-trained EfficientNetB4 model, pre-trained on ImageNet weights, was taken as the baseline architecture. First, the base layers were frozen in order to train the custom classification head with a Global Average Pooling and Dense layer, then fine-tuned step by step to sharpen feature extraction. Training was performed on multiple epochs with the loss and accuracy metrics being observed, which were then visualized to observe the learning trend of the model. When the model had reached stable performance, it was saved in ‘.keras’ format for deployment. To facilitate user input and real-time prediction, a Flask web interface was created. The frontend implemented was through the use of HTML, CSS, and JavaScript with a mobile-ready interface having provisions for drag-and-drop or selection of a file for uploading the image. The system indicates upon submission the forecasted disease with its probable cause and suggested treatment. Python libraries were incorporated along the way to facilitate image processing, training visualization, and backend operations, leading to a strong and user-friendly plant disease detection tool.

**Testing**

The disease detection system of the plant was tested using the PlantVillage dataset as well as actual leaf samples to gauge its performance in varied conditions. Images with different lighting, resolutions, and backgrounds were passed through the trained model to check for its robustness and generalization ability. The predicted disease labels were matched with the ground truth annotations, and performance was measured using standard performance metrics such as accuracy, precision, recall, and F1-score. To evaluate the usability of the system in real-world scenarios, the web application interface was experimented with by uploading images and checking the prediction outcomes. These results were cross-checked against a curated plant disease database to verify the accuracy of both the diagnosis and the recommended treatments. The testing phase validated that the system accurately provided consistent classifications as well as appropriate recommendations, showcasing its reliability and capability of helping farmers early detect and manage diseases.

## IV. RESULT

Plant disease diagnosis system was created with Python and TensorFlow, utilizing a CNN-based EfficientNetB4 classification model. The system was trained on Plant Village data, with 54,306 labelled images of diseased and healthy plant leaves. The model presented promising findings in closed environments and had acceptable generalization in actual field tests, even in varying lighting and background conditions. Image preprocessing methods like resizing images to 160×160 pixels and colour normalization ensured the performance of the model was consistent.

The accuracy of training and validation, as indicated in the graph below, represents the learning curve of the model. The accuracy for training continued to increase steadily from an initial value of about 0.81 to the highest value of approximately 0.975 by epoch 14. Validation accuracy remained high up to epoch 7 at its highest value of about 0.975 but dropped slightly by about 0.93 at epoch 10 before improving again. After fine-tuning post epoch 5, the model showed an overall increase in accuracy, with certain validation performance instability being noticed.

The training and validation loss trends also emphasize the performance of the model across epochs. Training loss gradually reduced from 0.75 to almost 0.02 by the 15th epoch. The validation loss decreased initially until epoch 5, then spiked at epoch 11, reaching around 0.45, before again decreasing. Fine-tuning was started at epoch 5, and a comparatively low loss with minimal fluctuations was observed thereafter.

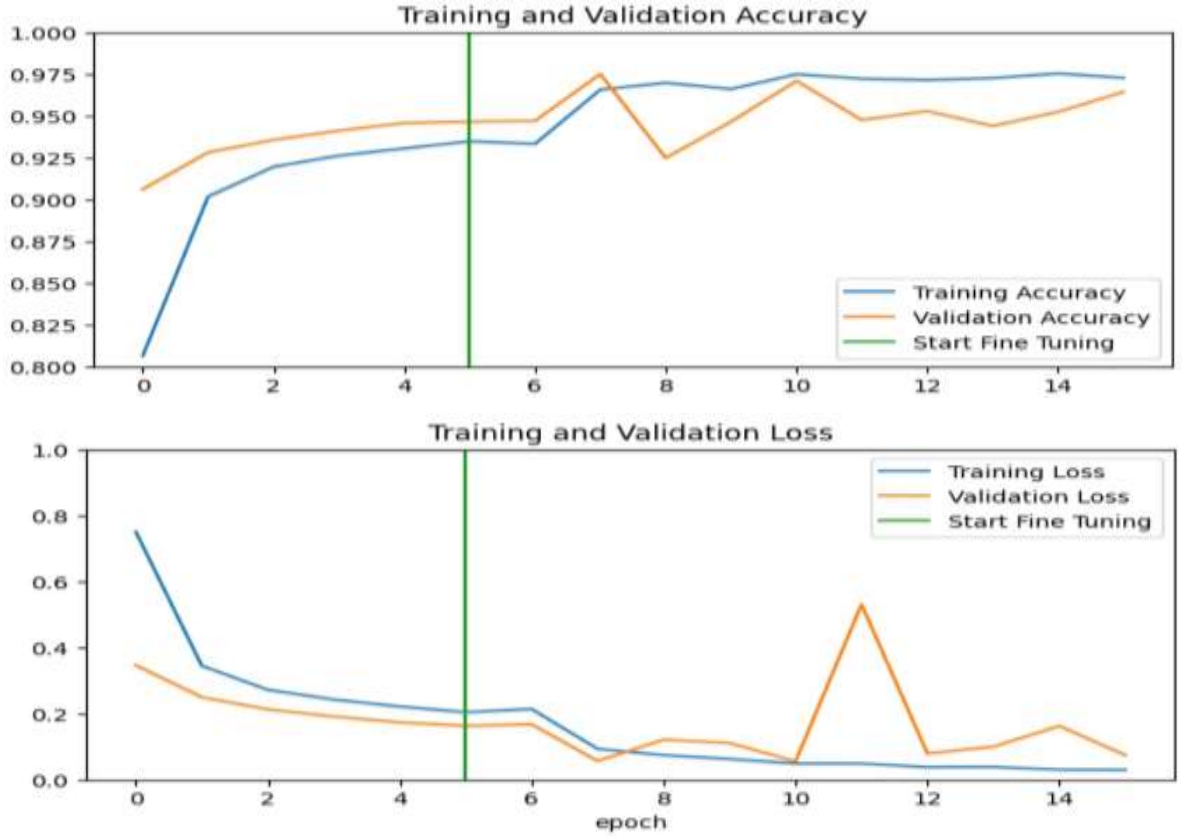


Figure 3.0: Training vs Validation accuracy and loss

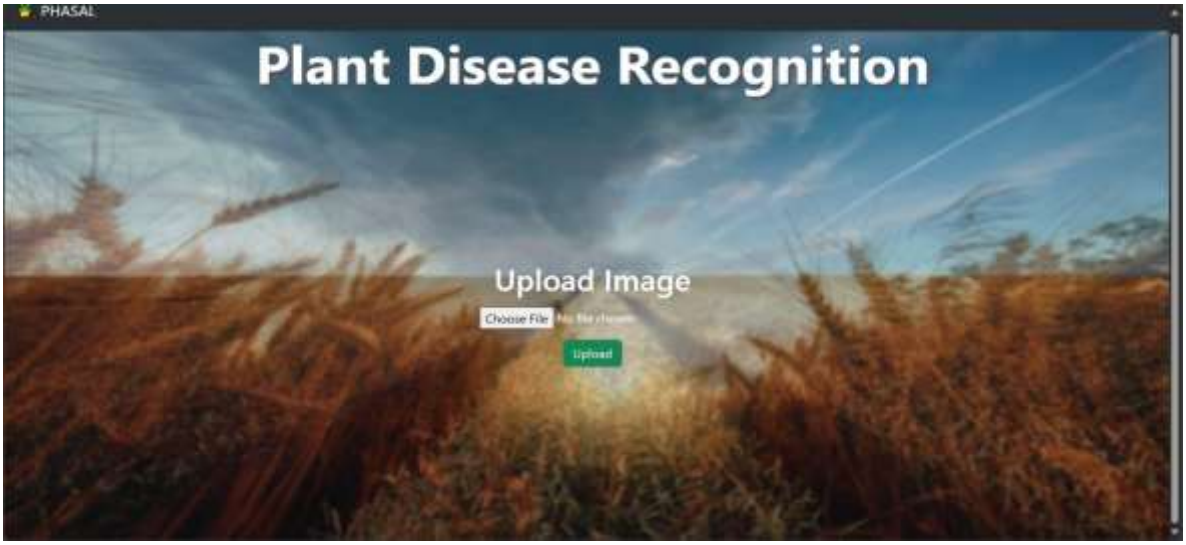


Fig. 4.0 Uploading the input image of the plant leaf for disease detection, as shown in the interface

When a plant leaf image is uploaded via the web application, the trained CNN model processes the input and identifies the disease correctly. For instance, when an image of a potato leaf is uploaded, the system was able to detect Early Blight, which is a fungal disease brought about by *Alternaria solani*. Not only did the application identify the disease, but it also offered related information, such as its cause and suggested actions. The proposed preventive measures entailed the use of proper fungicides and crop rotation to limit the availability of the pathogen in the soil. This feedback mechanism in real time enables users especially farmers to take immediate and informed decisions to control plant diseases effectively.

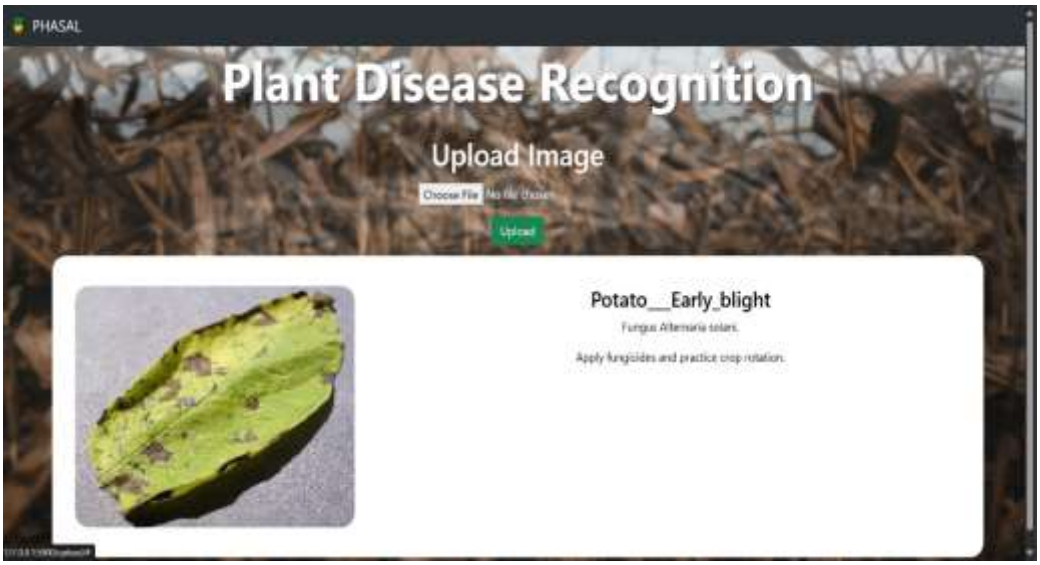


Figure 5.0: Early Blight detected for the input image of the diseased potato leaf as shown.

## V. FUTURE WORK

The system can be further enhanced with the following functionalities:

1. **Dataset Expansion**: Incorporate a broader range of crop types, diseases, and environmental conditions to improve the model's generalization capabilities.
2. **Disease Stage Prediction**: Develop an advanced model that not only classifies healthy vs. diseased leaves but also identifies the specific stage of the disease for timely and precise intervention.
3. **User-Friendly Deployment**: Build a dedicated website and mobile application to allow farmers real-time access to disease detection, treatment recommendations, and agricultural support.

## VI. CONCLUSION

In this paper, we have implemented and deployed a plant disease detection system using the latest deep learning method, CNN-based models, to accurately detect plant diseases. With the use of the Plant Village dataset and image preprocessing, the system can correctly classify diseases, helping farmers detect diseases in the early stage and making timely intervention feasible.

The application of this system through a web interface provides a welcoming platform, where farmers can just upload images of plant leaves and be given disease predictions as well as informative tips on prevention and treatment. This solution not only saves pesticides but also minimizes crop loss, increases yields, and encourages sustainable agriculture. With the integration of real-time disease identification and actionable guidance, this system can go a long way toward enhancing farm productivity and contributing to the overall cause of food security and sustainable agriculture.

## VII. REFERENCES


1. Mohanty, Sharada Prasanna, David Hughes, and Marcel Salathé, "*Using Deep Learning for Image-Based Plant Disease Detection,*" Frontiers in Plant Science, 2016.
2. Ramcharan, Adarsh, Ankit Baranwal, Michael Barett, and David G. Hughes, "*Deep Learning for Image-Based Cassava Disease Detection,*" Frontiers in Plant Science, 2017.
3. Ferentinos, Konstantinos P., "*Deep Learning Models for Plant Disease Detection and Diagnosis*," Computers and Electronics in Agriculture, 2018.
4. Brahimi, Mohamed, Kamal Boukhalfa, and Abdelouahab Moussaoui, "*Deep Learning for Tomato Diseases: Classification and Symptoms Visualization*," Applied Artificial Intelligence, 2017.
5. Picon, Alfonso, Amaya Alvear, Juan Carlos Poblete, and Freddy Cubillos, *"Automatic Classification of Diseases in Corn Leaves Using Convolutional Neural Networks,*" Sustainability, 2019.
6. Arsenovic, Marko, Srdjan Karanovic, Stefan Sladojevic, Darko Anderla, and Andras Stefanovic, "*Solving Current Limitations of Deep Learning-Based Approaches for Plant Disease Detection,"* Symmetry, 2019.
7. Selvaraj, Manickavasagan, Raja Suresh, Aravindan Murugan, and Abirami Saravanan, *"AI-Powered Mobile App for Plant Disease Detection and Classification*," IEEE Access, 2020.
8. Wang, Guo-Xun, Haiyan Sun, and Yaoyang Zhang, "*MobileNetV2-Based Transfer Learning Model for RealTime Plant Disease Detection,*" Computers and Electronics in Agriculture, 2021.
9. Waheed, Abdul, Muhammad G. Memon, and Muhammad Imran, "*Plant Disease Detection Using AI-Driven Image Classification,*" Advances in Computational Intelligence, 2021.
10. Barbedo, Jayme Garcia Arnal, "*Plant Disease Identification from Individual Lesions and Spots Using Deep Learning*," Biosystems Engineering, 2019.