

# PLANT DISEASE DETECTION USING CNN

## Introduction

Agriculture is one of the most vital industries in the world, and early detection of plant diseases is crucial for maximizing crop yield and ensuring food security. Manual identification of plant diseases is time-consuming and often requires expert knowledge, making it challenging for small-scale farmers.

With advancements in artificial intelligence and computer vision, it has become possible to automate the disease detection process using deep learning techniques. This project presents a solution that utilizes a Convolutional Neural Network (CNN) to detect plant diseases from leaf images. The final model is deployed using a Streamlit interface, making it accessible and easy to use for non-technical users.

## Objective

The primary aim of this project is to build an automated plant disease classification system that can:

- Accurately classify multiple plant diseases from leaf images using a trained deep learning model.
- Provide a user-friendly web interface to enable non-technical users to utilize the model.
- Integrate model predictions into a lightweight and deployable web application.
- Ensure the application performs efficiently on common hardware and generalizes well to unseen images.

## Tools & Technologies Used

- **Python:** Programming language used for scripting and model development
- **TensorFlow / Keras:** Deep learning framework used to build and train the CNN model
- **NumPy, PIL:** Used for numerical operations and image preprocessing
- **Streamlit:** Python-based web framework used to build the UI
- **Google Colab:** Cloud-based Jupyter notebook platform with free GPU access
- **PlantVillage Dataset:** A publicly available dataset with over 43,000 labeled leaf images
- **GitHub:** Used for version control and sharing the final deliverables

## Methodology

### 1.Data Preparation

The PlantVillage dataset was split into training, validation, and test sets. Images were resized to 224×224 pixels and normalized to [0, 1] scale. Augmentation techniques like rotation, flipping, zooming, and brightness adjustments were applied using ImageDataGenerator to increase robustness.

### 2.Mode IDesign

A CNN architecture was built with two convolutional layers followed by max pooling, flattening, and two dense layers. ReLU was used as the activation function in hidden layers, and softmax in the output layer. The final model outputs 38 classes corresponding to various plant diseases.

### 3.Training & Evaluation

The model was trained for multiple epochs using the Adam optimizer and categorical cross-entropy loss function. It achieved over 97% accuracy on training and around 95–97% on test data. Evaluation was performed using accuracy, loss metrics, and confusion matrix.







### 4.Saving and Loading

The trained model was saved in .keras format. A class mapping dictionary was saved as a .json file to assist with prediction labeling during deployment.

## 5. Deployment via Streamlit

A responsive Streamlit frontend was created where users can upload an image, press a button, and see the predicted disease name along with the uploaded image preview.

### Streamlit Interface Features

-  **Upload Panel:** Accepts JPG, JPEG, and PNG leaf images
-  **Prediction Button:** Runs inference using the saved CNN model
-  **Image Preview:** Displays the uploaded image beside the prediction
-  **Sidebar Section:** Includes project summary and usage instructions
-  **Responsive Layout:** Uses st.columns for a clean user experience
-  **Fast Inference:** Supports real-time predictions with minimal latency

## Conclusion

The plant disease classification system using CNN provides an efficient and scalable method to detect diseases in crops. By integrating a custom-trained model into a web interface, we demonstrate the practical use of deep learning in agriculture. The high accuracy, low resource requirements, and intuitive interface make this solution suitable for farmers, agronomists, and researchers.

### Future Enhancements:

- Incorporating transfer learning with models like MobileNet or ResNet
- Adding support for mobile deployment via TensorFlow Lite
- Expanding dataset to include real-world noisy images from farms

## References

- TensorFlow Documentation: <https://www.tensorflow.org>
- Streamlit Official Site: <https://www.streamlit.io>
- PlantVillage Dataset (Kaggle): <https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset>
- Keras API Documentation: <https://keras.io>