

# PLANT DISEASE DETECTION

## Using Image Processing And Machine Learning

Poornima N

Assistant Professor

Mohammed Fahad, Raakin Ahmed, Gowrishankar R, Yatish R

*The National Institute of Engineering*

Mysuru, India

[4ni19cs069\\_a@nie.ac.in](mailto:4ni19cs069_a@nie.ac.in) [4ni19cs088\\_a@nie.ac.in](mailto:4ni19cs088_a@nie.ac.in) [4ni19cs043\\_a@nie.ac.in](mailto:4ni19cs043_a@nie.ac.in) [4ni16cs124\\_a@nie.ac.in](mailto:4ni16cs124_a@nie.ac.in)

**Abstract** — Our Plant disease detection project presents a Convolutional Neural Network (CNN) model for the classification of plant diseases based on image data. The dataset comprises images of various plant diseases and healthy plants obtained from the "PlantVillage" database. The images are preprocessed by resizing them to a standard size and applying augmentation techniques. The CNN model is built using the Keras library and consists of multiple convolutional layers followed by pooling, batch normalization, and dropout layers. The model is trained using the Adam optimizer and evaluated on a test set. The training and validation accuracy and loss are plotted over the epochs to analyze the model's performance. The trained model achieves a certain accuracy on the test set, indicating its potential for accurately identifying plant diseases. The saved model can be utilized for real-world applications in plant disease detection and management, providing valuable assistance to farmers and researchers.

### INTRODUCTION

Plant diseases have a significant impact on agricultural productivity and can lead to severe economic losses. Early detection and accurate identification of plant diseases are crucial for implementing timely interventions and preventing further spread. In recent years, advancements in computer vision and machine learning techniques have provided promising solutions for automated plant disease diagnosis.

This project focuses on the development of a Convolutional Neural Network (CNN) model for the classification of plant diseases using image data. By leveraging the power of deep learning, the CNN model can learn intricate patterns and features in plant images, enabling accurate disease classification.

The dataset used in this project consists of a diverse collection of plant disease images obtained from the "PlantVillage" database. These images encompass a wide range of plant species and various diseases affecting them. The dataset includes both diseased plants and healthy plants, providing a comprehensive training and evaluation set.

To prepare the data for model training, the images are preprocessed by resizing them to a standard size and applying augmentation techniques to increase the dataset's diversity. The CNN model is constructed using the Keras library, a high-level neural networks API

After training, the model's efficacy is evaluated on a separate test set, assessing its ability to generalize and accurately classify plant diseases. The evaluation results provide insights into the model's performance and its potential for practical application.

Overall, this project demonstrates the effectiveness of CNNs in plant disease classification and highlights their role in advancing automated plant disease diagnosis. By accurately identifying plant diseases from images, this technology can aid farmers, researchers, and agricultural experts in making informed decisions for disease management, enhancing crop health, and improving overall agricultural productivity.

The provided code demonstrates the implementation of a Convolutional Neural Network (CNN) for image classification using the Keras library with TensorFlow backend. Image classification is a fundamental task in computer vision that involves assigning predefined labels to images based on their content. CNNs have proven to be highly effective in handling such tasks, as they can automatically learn and extract relevant features from images.

The code begins by importing the necessary libraries such as numpy, pickle, cv2, and others. It also imports modules from Keras and scikit-learn for model creation, data preprocessing, and evaluation. Additionally, matplotlib is imported for visualizing the training process.

The main functionality of the code involves loading images from a specified directory, resizing them to a default size of 256x256 pixels, and converting them into arrays. The images are then split into training and testing sets using the `train_test_split` function from scikit-learn.

To improve the model's performance and generalization, data augmentation is applied to the training set using the `ImageDataGenerator` class from Keras. This augmentation involves randomly applying transformations such as rotation, shifting, shearing, zooming, and horizontal flipping to the training images. These augmented images are then used for training the CNN model.

The CNN architecture consists of several layers, including convolutional, activation, normalization, pooling, and dense layers. The model is compiled with the Adam optimizer and binary cross-entropy loss function. The training process is initiated using the `fit_generator` function, which trains the model on the augmented training data and validates it using the testing set.

During training, the code plots the training and validation accuracy/loss curves using matplotlib to visualize the model's

performance. After training, the model's accuracy is evaluated on the test set. Finally, the trained model and label binarizer are saved for future use

## RELATED WORK

In 2015, S. Khirade et Al. tackled the problem of plant disease detection using digital image processing techniques and back propagation neural network (BPNN). Authors have elaborated different techniques for the detection of plant disease using the images of leaves. They have implemented Otsu's thresholding followed by boundary detection and spot detection algorithm to segment the infected part in leaf. After that, they have extracted the features such as color, texture, morphology, edges etc. for classification of plant disease. BPNN is used for classification i.e., to detect the plant disease. The data set used here is Plant Pathology Challenge. Shiroop Madiwalar and Medha Wyawahare analyzed different image processing approaches for plant disease detection in their research. Authors analyzed the color and texture features for the detection of plant disease. They have experimented their algorithms on the dataset of 110 RGB images. The features extracted for classification were mean and standard deviation of RGB and YCbCr channels, grey level co-occurrence matrix (GLCM) features, the mean and standard deviation of the image convolved with Gabor filter. Photo images of the leaves were taken using a digital camera with a white background. A total of 110 images were acquired from which 86 images were used for training process and 24 images used for testing process. The confusion matrices for the 24 test images with 8 of each type. Peyman Moghadam et Al. demonstrated the application of hyperspectral imaging in plant disease detection task [3]. visible and near-infrared (VNIR) and short-wave infrared (SWIR) spectrums were used in this research. Authors have used k-means clustering algorithm in spectral domain for the segmentation of leaf. They have proposed a novel grid removal algorithm to remove the grid from hyperspectral images. Authors have achieved the accuracy of 83% with vegetation indices in VNIR spectral range and 93% accuracy with full spectrum. Though the proposed method achieved higher accuracy, it requires the hyperspectral camera with 324 spectral bands so the solution becomes too costly. 30 plants were inoculated with Tomato Spotted Wilt Virus (TSWV) when they were 8 weeks old. The disease progression of TSWV was observed for 21 days after inoculation. A further 30 plants were not inoculated and served as control of healthy plants for this experiment. Sharath D. M. et Al. developed the Bacterial Blight detection system for Pomegranate plant by using features such as color, mean, homogeneity, SD, variance, correlation, entropy, edges etc. Authors have implemented grab cut segmentation for segmenting the region of interest in the image [4]. A canny edge detector was used to extract the edges from the images. Authors have successfully developed a system which can predict the infection level in the fruit. We checked the result for nearly 450 bacterial blight infected s, among those nearly 35 images showed the infection level

11

below 20%, it indicates the infection level is at initial stage. Further about 72 images showed infection of about 40%

which is about to get infection by bacterial blight and rest all the images, 343 shows the infection level of more than 80% Garima Shrestha et Al. deployed the convolutional neural network to detect the plant disease [5]. Authors have successfully classified 12 plant diseases with 88.80% accuracy. The dataset of 3000 high resolution RGB images were used for experimentation. The network has 3 blocks of convolution and pooling layers. This makes the network computationally expensive. Also, the F1 score of the model is 0.12 which is exceptionally low because of higher number of false negative predictions. 200 images of each of the 15 classes, a total of 3000 images have been provided. The dataset is divided in the ratio of 80:20 for training and test set respectively.

Identification of plant disease using image processing technique, the work of Abhirami Devaraj, Karunya Rathan, Karunya Jaahnavi and K Indra deals with the identification diseases like *Alternaria Alternata*, Bacterial Blight, Anthracnose, *Cercospora Leaf Spot*. This methodology uses k-means clustering for segmentation and GLCM (Grey Level Co- occurrence Matrix) is used for feature extraction and then upon the data obtained random forest algorithm is used for identification. Here the images are loaded into MATLAB image processing system, the image is enhanced and converted from RGB to L\*a\*b (Lightness, Chromaticity, Hue) images with k-means clustering and then the Random Forest algorithm is applied to identify the disease. Finally, this project helps in Identification of disease and provides solution for it.

## I. PROPOSED WORK

The proposed system aims to develop a robust plant disease classification system using a Convolutional Neural Network (CNN) approach. This system leverages the power of deep learning to automatically learn and extract relevant features from plant images, enabling accurate and efficient disease diagnosis. The key components and steps involved in the proposed system are as follows:

**Dataset Collection:** A diverse dataset of plant disease images is collected from the "PlantVillage" database. This dataset includes images of various plant species affected by different diseases, as well as images of healthy plants for comparison.

**Data Preprocessing:** The collected images are preprocessed to ensure uniformity and improve model performance. The images are resized to a standard size to facilitate consistent input dimensions for the CNN model. Additionally, data augmentation techniques are applied to increase the dataset's size and diversity, thereby enhancing the model's ability to generalize.

**CNN Model Architecture:** The proposed system employs a CNN model for disease classification. The CNN architecture consists of multiple convolutional layers, followed by activation functions batch normalization, and pooling layers. These layers enable the model to learn hierarchical representations and capture intricate patterns and disease-specific features.

**Model Training:** The CNN model is trained using the preprocessed dataset. The training process involves feeding the input images through the network, computing the loss

using a binary cross-entropy loss function, and optimizing the model parameters using the Adam optimizer. The model is trained over multiple epochs with a specified learning rate.

**Evaluation and Validation:** The trained CNN model is evaluated using a separate test set to assess its performance in classifying plant diseases. Performance metrics such as accuracy and loss are calculated to quantify the model's effectiveness. Additionally, a validation set can be used during the training process to monitor the model's performance and prevent overfitting.

**Visualization and Analysis:** The system generates visualizations such as plots of training and validation accuracy/loss over epochs. These visualizations provide insights into the model's learning progress and can aid in fine-tuning the model's hyperparameters.

**Model Deployment:** Once the CNN model achieves satisfactory performance, it is saved for future use. The saved model can be deployed in real-world applications to classify plant diseases automatically. It can be integrated into user-friendly interfaces or mobile applications to provide farmers and agricultural experts with an accessible tool for disease diagnosis.

## II. IMPLEMENTATION

### 1. Data Collection and Preprocessing:

- o The system begins with the collection of a diverse dataset of plant disease images from the "PlantVillage" database.
- o The collected images are preprocessed by resizing them to a default size and converting them to arrays using OpenCV.
- o Data augmentation techniques, such as rotation, shift, shear, zoom, and flip, are applied to increase the dataset's diversity and improve model performance.
- o The preprocessed images and corresponding labels are stored in separate lists.

XML, CSS

### 2. Model Creation and Training:

- o The system utilizes the Keras library to create a Convolutional Neural Network (CNN) model for plant disease classification.
- o The CNN model consists of several layers, including convolutional layers, activation functions (ReLU), batch normalization, max pooling, dropout, and fully connected layers. The model is compiled with the Adam optimizer, a binary cross-entropy loss function, and accuracy as the evaluation metric.
- o Data is split into training and testing sets using the train test split function from scikit-learn.
- o The model is trained using the training data and evaluated using the testing data.
- o During training, data is fed into the model in batches using the ImageDataGenerator class from Keras to handle data augmentation.

### 3. Visualization and Analysis:

- o The system generates visualizations to analyze the model's performance.

- o Training and validation accuracy and loss values are plotted over epochs using matplotlib.
- o These visualizations provide insights into the model's learning progress and potential overfitting issues.

### 4. Model Evaluation and Saving:

- o The system evaluates the model's performance by calculating the accuracy on the testing data.
- o The trained model is saved to disk using the pickle library, allowing it to be reused for future predictions or deployments.
- o Additionally, the LabelBinarizer object, used for label transformation, is also saved to disk.

### 5. Model Deployment and Usage:

- o The saved model (cnn\_model.pkl) can be deployed in various applications or systems for plant disease classification.
- o The deployed model can accept images as input and provide predictions for the respective plant diseases.
- o The model can be integrated into user interfaces, mobile apps, or web applications to facilitate easy access and usage by farmers or agricultural experts.

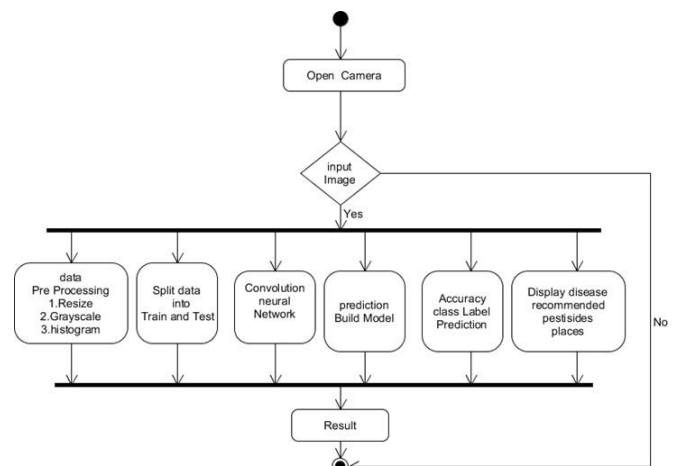


Fig 3.4

## III. RESULTS

**1. Importing Libraries:** The code begins by importing necessary libraries such as numpy, pickle, cv2, os, sklearn, keras, matplotlib, and tensorflow. These libraries provide functionalities for numerical operations, data manipulation, image processing, machine learning, visualization, and deep learning.

**2. Loading and Preprocessing Images:** The code reads images from a specified directory using the `cv2.imread` function and resizes them to a default size of 256x256 pixels using `cv2.resize`. The images are converted into arrays using `img\_to\_array` from the Keras library. This step ensures that the image data is in a suitable format for further processing.

**3. Data Preparation:** The code organizes the images into lists of image arrays and their corresponding labels. It iterates through the directory structure to collect the images,

excluding any unnecessary files such as ``.DS_Store``. The images are associated with their respective labels, representing different classes or categories.

4. **Label Binarization:** The code uses the ``LabelBinarizer`` class from sklearn to transform the categorical labels into binary vectors. This step is essential for training a multiclass classification model. The ``fit_transform`` function converts the labels into binary form, and the resulting binarizer is saved using ``pickle`` for future use.

5. **Data Splitting:** The code splits the image arrays and label vectors into training and testing sets using ``train_test_split`` from sklearn. This division ensures that the model can learn from a subset of the data (training set) and evaluate its performance on unseen data (testing set).

6. **Data Augmentation:** To enhance the model's generalization and prevent overfitting, the code applies data augmentation using the ``ImageDataGenerator`` class from Keras. Augmentation techniques such as rotation, shifting, shearing, zooming, and flipping are randomly applied to the training images. This augmentation generates additional training samples with variations, increasing the model's ability to generalize to different scenarios.

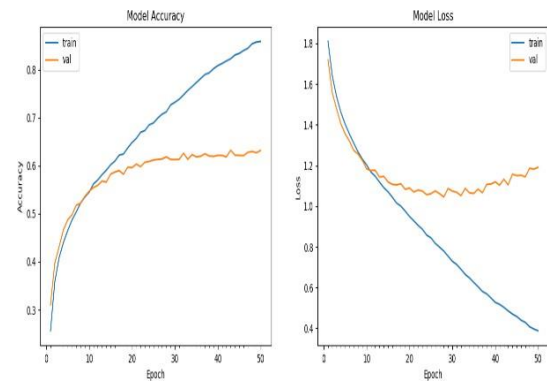
7. **CNN Model Architecture:** The code defines the architecture of the CNN model using the ``Sequential`` model from Keras. The model consists of convolutional layers, activation functions, batch normalization layers, max pooling layers, dropout layers, and fully connected (dense) layers. These layers work together to extract meaningful features from the input images and classify them into respective classes.

8. **Model Compilation and Training:** The code compiles the model by specifying the optimizer, loss function, and evaluation metrics. It uses the Adam optimizer and binary cross-entropy loss for binary classification. The ``fit_generator`` function is used to train the model on the augmented training data and validate it using the testing set. The training process occurs over a specified number of epochs.

9. **Training Visualization:** During training, the code plots the training and validation accuracy/loss curves using matplotlib. These plots provide insights into the model's performance and help analyze if the model is underfitting or overfitting.

10. **Model Evaluation:** After training, the code evaluates the trained model's accuracy on the testing set using the ``evaluate`` function. The evaluation results are printed, indicating how well the model performs on unseen data.

11. **Model Saving:** The code saves the trained model and the label binarizer using ``pickle``. This allows the model to be reused or deployed later without retraining.



**Figure 4. Accuracy Plot**

#### IV. CONCLUSION

Convolutional Neural Networks (CNNs) have proven to be highly effective for plant disease detection. CNNs are capable of automatically learning and extracting intricate patterns and features from images, making them well-suited for analyzing plant images and identifying diseases.

By training a CNN on a large dataset of labeled plant images, the model can learn to differentiate between healthy plants and those affected by various diseases. The convolutional layers in the CNN enable the network to capture local patterns and textures, while pooling layers help reduce spatial dimensions and achieve translation invariance.

The fully connected layers at the end of the CNN map the extracted features to disease classifications. Plant disease detection using CNNs offers several benefits. It enables early and accurate detection of diseases, allowing for timely interventions and minimizing crop losses. It also provides a scalable and automated approach, reducing the need for manual inspection and enabling high-throughput analysis.

Overall, the use of CNNs for plant disease detection holds great potential in revolutionizing agriculture by facilitating early disease identification and promoting efficient plant health management practices.

#### REFERENCES

- [1] S. D. Khirade and A. B. Patil, "Plant Disease Detection Using Image Processing," 2015 International Conference on Computing Communication Control and Automation, 2015, pp.768- 771, doi: 10.1109/ICCUBEA.2015.153.
- [2] S. C. Madiwalar and M. V. Wyawahare, "Plant disease identification: A comparative study," 2017 International Conference on Data Management, Analytics, and Innovation (ICDMAI), 2017, pp. 13-18, doi: 10.1109/ICDMAI.2017.8073478.
- [3] P. Moghadam, D. Ward, E. Goan, S. Jayawardena, P. Sikka and E. Hernandez, "Plant Disease Detection Using Hyperspectral Imaging," 2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2017, pp. 1-8, doi: 10.1109/DICTA.2017.8227476.
- [4] S. D.M., Akhilesh, S. A. Kumar, R. M.G. and P. C., "Image based Plant Disease Detection in Pomegranate Plant for Bacterial Blight," 2019 International

Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0645-0649, doi: 10.1109/ICCSP.2019.8698007

- [5] G. Shrestha, Deepsikha, M. Das and N. Dey, "Plant Disease Detection Using CNN," 2020 IEEE Applied Signal Processing Conference (ASPCON), 2020, pp. 109-113, doi: 10.1109/ASPCON49795.2020.9276722.
- [6] A. Devaraj, K. Rathan, S. Jaahnavi and K. Indira, "Identification of Plant Disease using Image Processing Technique," 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0749-0753, doi: 10.1109/ICCSP.2019.8698056.
- [7] J.K. Kamble, "Plant Disease Detector," 2018 International Conference on Advances in Communication and Computing Technology (ICACCT), 2018, pp. 97-101, doi: 10.1109/ICACCT.2018.8529612.