AgroVision : A deepLearning based plant disease detection model

Github repository : https://github.com/saiteja9078/PlantDiseasePrediction

WebSite : https://agro-vision-kxqnjv6x-718212-cgbhaq8b37kes.streamlit.app

TeamMate 1: Sai Teja E23CSEU0574

TeamMate 2: Khushal E23CSEU0599

TeamMate 3: Srikhar E23CSEU0591

Lab instructor : Prashant Kapil

Abstract:

This project presents a deep learning-based model using Convolutional Neural Networks (CNNs) to detect plant diseases from images. By allowing users to upload a plant image, the model identifies the disease and provides a brief diagnosis, empowering timely intervention. Motivated by the need to reduce crop losses and enhance agricultural productivity, this system aims to offer a practical, accessible tool for farmers and agricultural workers. Key challenges include building a diverse dataset and achieving high accuracy despite environmental variations.

1.Introduction:

Plant diseases significantly impact crop yields and food security, yet early detection methods are often inaccessible to many farmers. This project aims to develop a deep learning model using Convolutional Neural Networks (CNNs) to identify plant diseases from images. Users can upload an image, and the model will provide the disease name and a brief diagnosis. This tool offers a practical, automated solution to support early intervention and reduce crop loss. Key challenges include building a diverse dataset and ensuring high model accuracy in varied environments. Ultimately, this project promotes sustainable agriculture by empowering users with accessible disease detection.

1.1 Motivation

Plant diseases are a major threat to agriculture, often resulting in reduced crop yields, lower quality produce, and economic losses for farmers. Traditional disease detection methods require specialized knowledge, which can be costly and difficult to access, particularly in rural or under-resourced areas. With the growing accessibility of smartphones and the potential of artificial intelligence, a reliable, automated solution for plant disease detection can empower farmers and help maintain crop health. This project aims to bridge the gap by creating an accessible, easy-to-use tool that enables users to diagnose plant diseases quickly and accurately.

1.2 Challenges

Developing an effective plant disease detection model requires a comprehensive dataset across various diseases and conditions, which can be resource-intensive. The model must handle real-world variability and maintain high accuracy. To avoid overshooting the loss function, use a small learning rate (e.g., 0.0001). If underfitting occurs, increasing the number of neurons can help. Additionally, adding more convolutional layers will enable the model to extract more relevant

features from the images, addressing confusion due to insufficient feature extraction and improving overall performance.

1.3 Contribution

This project introduces a deep learning-based model using Convolutional Neural Networks (CNNs) for plant disease detection, which provides both a disease diagnosis and a brief description based on a simple image upload. The model is designed for ease of use by farmers and agricultural workers, making it a practical tool for early intervention and disease management. By automating disease detection, this project contributes to sustainable agriculture by promoting early disease identification and supporting informed decision-making, potentially leading to reduced crop losses and increased productivity.

2. Related Survey

2.1 References and studies

[1] Arivazhagan, S., R. Newlin Shebiah, S. Ananthi, and S. Vishnu Varthin. Studied. "Detection of unhealthy regions of plant leaves and classification of plant leaf diseases using texture features." *Agricultural Engineering International: CIGR Journal*, 15, No.1 (2013): 211-217.

[2] Kiran R. Gavhale and Ujwalla Gawande. Studied. "An Overview of the Research on Plant Leaves Disease Detection using Image Processing Techniques." *IOSR Journal of Computer Engineering*, January 2014.

[3] Sachin D. Khirade and A. B. Patil. Studied. "Plant Disease Detection Using Image Processing." *International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2015, pp. 768-771. IEEE, 2015.

[4] Sannakki, Sanjeev S., Vijay S. Rajpurohit, V. B. Nargund, and Parag Kulkarni. Studied. "Diagnosis and classification of grape leaf diseases using neural networks." *In Computing, Communications, and Networking Technologies (ICCCNT), 2013 Fourth International Conference on*, pp. 1-5. IEEE, 2013.

[5] Kutty, Suhaili Beeran, Noor Ezan Abdullah, Habibah Hashim, and Aida Sulinda. Studied. "Classification of Watermelon Leaf Diseases Using Neural Network Analysis." *In Business Engineering and Industrial Applications Colloquium (BELAC), 2013 IEEE*, pp. 459-464. IEEE, 2013.

[6] Rothe, P. R., and R. V. Kshirsagar. Studied. "Cotton Leaf Disease Identification using Pattern Recognition Techniques." *In Pervasive Computing (ICPC), 2015 International Conference on*, pp. 1-6. IEEE, 2015.

[7] Pearson, Roger C., and Austin C. Goheen. Studied. *Compendium of Leaf Diseases*, American Phytopathological Society, 1988.

[8] I. El Massi, Y. Es-saady, M. El Yassa, D. Mammass, and A. Benazoun. Studied. "Automatic recognition of plant leaf diseases based on serial combination of two SVM classifiers." *2nd International Conference on Electrical and Information Technologies (ICEIT)*, IEEE, 2016.

[9] Aakanksha Rastogi, Ritika Arora, and Shanu Sharma. Studied. "Leaf Disease Detection and Grading using Computer Vision Technology & Fuzzy Logic." *2nd International Conference on Signal Processing & Integrated Network (SPIN)*, IEEE, 2015.

[10] Garima Tripathi. Studied. "Review on Color and Texture Feature Extraction Techniques." *International Journal of Enhanced Research in Management & Computer Applications*, Vol. 3, Issue 5, pp. 77-81, May 2014.

2.2 Factors causing Plant diseases

Agricultural diseases, arising at various plant development stages, can severely affect crop production. These diseases are caused by **biotic factors** (viruses, fungi, bacteria) and **abiotic factors** (water, temperature, nutrient deficiencies). This study utilizes plant leaf images from datasets like PlantVillage, which include both healthy and diseased leaves. The research highlights the application of computer vision techniques for disease detection, including dataset preparation, image preprocessing, and performance evaluation methods. These insights directly support the development of an automated plant disease detection system using deep learning models like CNNs

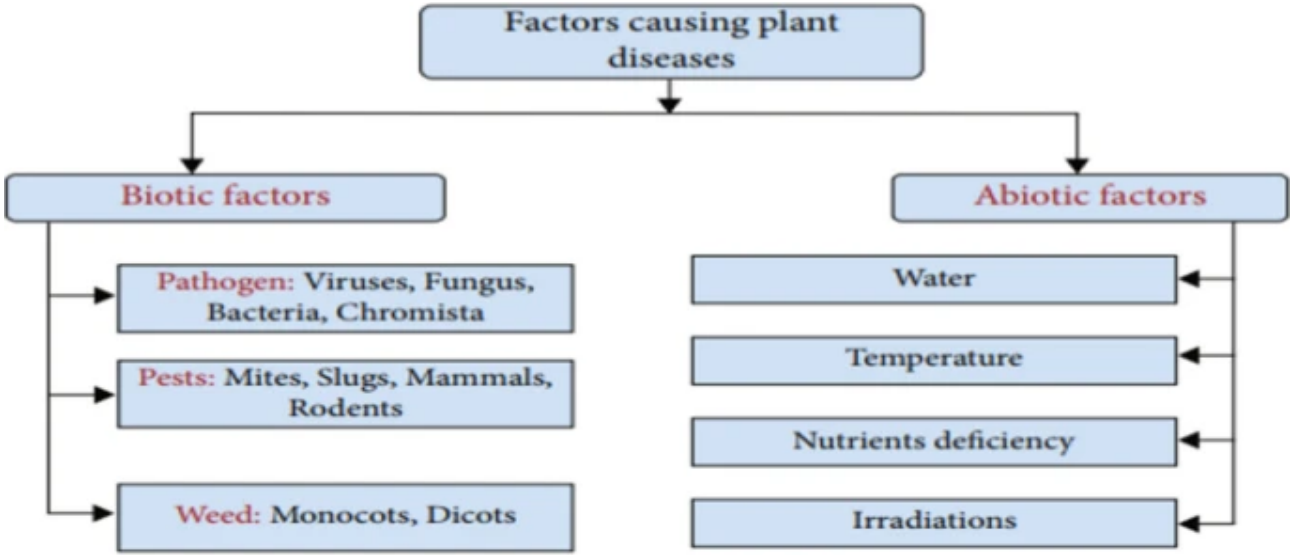


FIGURE 1

Some sample plant leaf images showing both healthy and diseased leaves from the PlantVillage dataset, along with images from other datasets, have been included for reference.

The research also provides detailed computer vision-based techniques for plant disease detection and classification, covering aspects like image acquisition, preprocessing, and data splitting, all aimed at supporting accurate and efficient disease detection. These techniques are crucial for developing automated models like CNNs to assist with plant health monitoring and crop protection.

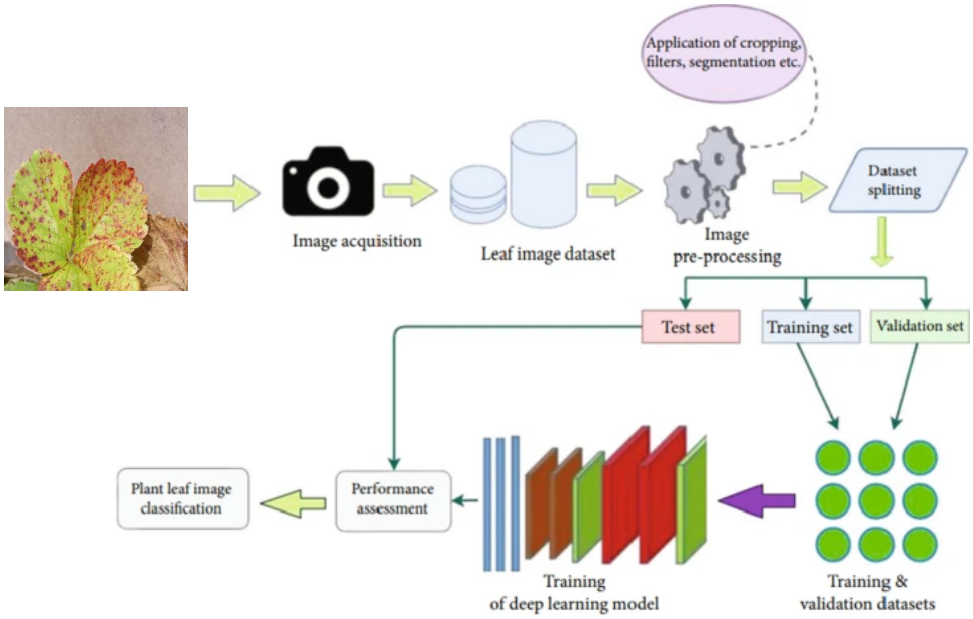


FIGURE 2: MODEL CREATION

3. Datasets and Preprocessing

3.1 Dataset Overview

Our plant disease detection model was trained on a comprehensive dataset comprising high-quality images of both healthy and diseased plant leaves across various crop species. This dataset includes sufficient diversity and volume, allowing for accurate and robust disease classification. Each crop in the dataset includes a variety of conditions, with images carefully labeled by plant type, disease type, and health status. The dataset details are as follows:

- **Apple**: Contains images of healthy apple leaves as well as leaves affected by diseases such as Apple Scab, Black Rot, and Cedar Apple Rust.
- **Blueberry**: Consists exclusively of healthy blueberry leaf images.
- **Cherry (including sour)**: Includes images of healthy leaves and those affected by Powdery Mildew.

- **Corn (maize)**: Comprises healthy corn leaf images alongside images showing Cercospora Leaf Spot, Common Rust, and Northern Leaf Blight.
- **Tomato**: Contains a wide range of conditions, including healthy leaves and leaves affected by Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Septoria Leaf Spot, Spider Mites (Two-spotted Spider Mite), Target Spot, Tomato Yellow Leaf Curl Virus, and Tomato Mosaic Virus.

The dataset's extensive variety across crop types and diseases enables the model to identify a broad spectrum of plant health conditions. With a large enough sample size and high-resolution images, this dataset supports effective training, validation, and testing, ensuring the model's applicability to real-world agricultural scenarios.

3.2 Data Preprocessing:

Image Resizing:

All images are resized to `(128, 128)` pixels to ensure consistency in input dimensions. This step is essential for neural networks, which require uniform input sizes. Resizing reduces variability and allows the model to process images efficiently.

Batching:

The dataset is divided into smaller batches, with each batch containing 32 images. Batching helps manage memory usage and speeds up training by processing multiple images simultaneously. It also allows the model to make more granular updates to its parameters.

Shuffling:

Shuffling randomizes the order of images in the dataset before each epoch. This prevents the model from learning patterns based on the order of images, which could introduce bias. It helps the model generalize better and learn robust features.

Automatic Label Inference:

Labels are automatically assigned based on the folder structure, eliminating the need for manual labeling. This technique assigns labels by reading folder names, which simplifies the dataset organization. It ensures that each image is correctly labeled based on its folder.

Categorical Label Encoding:

Labels are one-hot encoded, transforming them into binary vectors. This encoding method makes the labels compatible with the model's output layer. It ensures the model can output a probability distribution for multi-class classification tasks.

RGB Conversion:

Images are loaded in the RGB color space to standardize color representation. RGB is the most common color model used in image processing. This ensures that the model processes color information consistently across all images.

Interpolation for Resizing:

Bilinear interpolation is used when resizing images to maintain image quality. This technique adjusts pixel values by considering the surrounding pixels. It results in smoother images and reduces pixelation or distortion during resizing.

4 Methodologies

4.1 Model Architecture

The model is built using TensorFlow's Sequential API, consisting of several layers stacked one after the other. The model architecture includes:

- **Convolutional Layers (Conv2D):**
  - The initial convolutional layers use 32 filters to extract basic features such as edges and textures.
  - As the model deepens, the number of filters increases (64, 128, 256, and 512) to learn more complex features of the images.
- **Max-Pooling Layers (MaxPool2D):**
  - After each set of convolutional layers, max-pooling layers are applied to reduce the spatial dimensions of the feature maps, which helps in reducing computation and focusing on the most important features.
- **Dropout Layers:**
  - Dropout layers are added to prevent overfitting. The rate of dropout is set to 25% after the convolution layers and 40% after the fully connected layer. This helps the model generalize better by randomly dropping units during training.
- **Flatten Layer:**
  - The 2D feature maps are flattened into a 1D vector so that they can be fed into fully connected (Dense) layers for classification.
- **Dense Layers:**
  - A dense layer with 1500 units and ReLU activation is used to introduce non-linearity and learn high-level features.
  - The final dense layer has 38 units, corresponding to the 38 classes in the dataset, and uses softmax activation to output class probabilities.

4.2 Hardware Requirements and Software Requirements

- **GPU**: NVIDIA GTX 1660 or higher for efficient training.
- **CPU**: Multi-core processor (Intel i5/i7 or AMD Ryzen 5/7).
- **Memory**: 8 GB minimum; 16 GB recommended.
- **Storage**: 256 GB SSD and 50–100 GB for datasets.
- **Camera**: High-resolution (8 MP or higher) for deployment.
- **OS**: Windows 10/11, macOS, or Linux (Ubuntu preferred).
- **Programming**: Python 3.8+ with TensorFlow, NumPy, etc.
- **Tools**: PyCharm/VS Code, Git, Jupyter Notebook.
- **Deployment**: Flask/Django for web; TensorFlow Lite for mobile.
- **Database**: MySQL or Firebase for data storage.

4.3 Model Compilation and evaluation

Once the model architecture is defined, it is compiled with the following settings:

- Optimizer
- Loss Function
- Metrics

The model is evaluated using the training dataset. However, the code does not specify a validation dataset, which is typically used to assess the model's performance on unseen data. A validation dataset helps in identifying if the model is overfitting to the training data. Evaluation is done to measure the training accuracy, which shows how well the model is learning from the data.

5. Result Analysis

This project aims to classify plant diseases based on images, with a dataset containing 37 categories of plant disease labels. The model was trained using a Convolutional Neural Network (CNN) to perform multi-class classification. Below is a detailed analysis of the results.

5.1 Training and Validation Accuracy

- **Training Accuracy**: The model's training accuracy shows a steady increase over the epochs. Starting from around 59% in the first epoch, it steadily improves, reaching around 97% by the 10th epoch.
- **Test Accuracy**: The test accuracy also shows significant improvement, starting at approximately 85% in the first epoch and gradually rising to about 94.5% by the 10th epoch. The small gap between the training and validation accuracy suggests that the model is **not overfitting** and is generalizing well to unseen data.

5.2 Training and Validation Loss

- The training loss starts at 1.39 in the first epoch and decreases consistently to 0.11 by the final epoch. A decreasing loss indicates that the model's predictions are becoming more accurate with each epoch.

- The validation loss follows a similar pattern, starting at 0.47 and stabilizing around 0.25 in the later epochs. The model performs well on unseen validation data, though the loss isslightlyhigher than the training loss, which is typical due to the inherent difficulty of generalizing to new data.

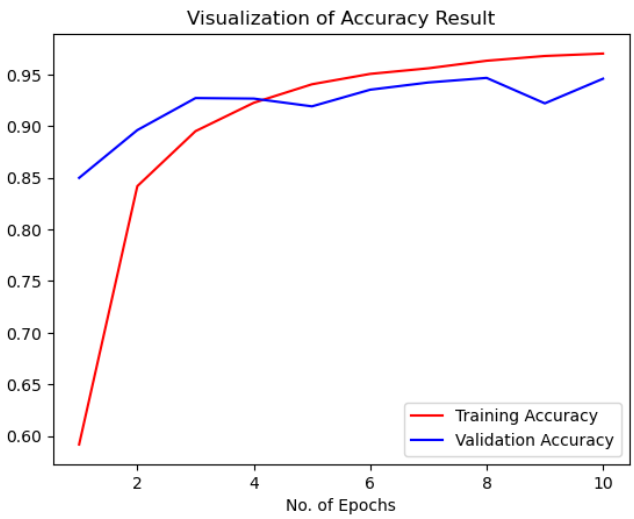
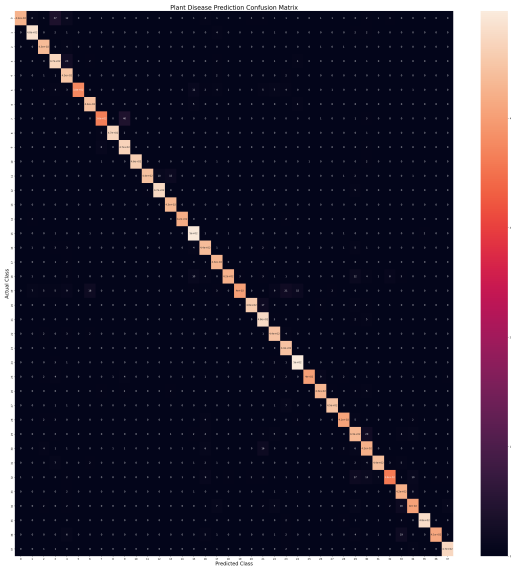


FIGURE 3 : TRAINING AND VALIDATION ACCURACY VS EPOCHS

5.3 Confusion Matrix

In the confusion matrix, correctly classified instances are represented by the diagonal entries, where the model made accurate predictions for specific classes. For example, Class 0 and Class 5 are correctly predicted, as seen in the diagonal entries at [0, 0] and [5, 5]. Misclassifications, on the other hand, are shown in the off-diagonal entries. For instance, if the value at [0, 1] is high, it indicates that instances from Class 0 were frequently misclassified as Class 1, suggesting the model confuses these two classes. Similarly, if there's a noticeable value at [3, 2], it means the model often predicted Class 2 when it should have predicted Class 3, leading to a misclassification between these two classes.

FIGURE 4 : CONFUSION MATRIX



5.4 Final Model Performance

Based on the results and metrics:

- **High Accuracy**: The model achieves high accuracy on both the training and validation datasets, with training accuracy nearing 97% and validation accuracy around 94.5%. This indicates that the model is performing well overall.
- **Generalization**: The small gap between training and validation accuracy suggests that the model is not overfitting and generalizes well to new, unseen data.
- **Model Strengths**: The model shows strong performance across most categories, correctly classifying many of the disease classes with high precision and recall.

Conclusion

The CNN model has demonstrated excellent performance in classifying plant diseases based on images, achieving high accuracy and low loss on both training and validation datasets. The model generalizes well to new data, and while further improvements could be made, it is ready for real-world applications such as plant disease diagnosis and management.

Future Work:

Future improvements for the plant disease detection system include model optimization through different architectures and transfer learning, real-time data integration from mobile apps or sensors for instant diagnoses, and a multimodal approach incorporating environmental factors. The system could be deployed on the cloud for scalability, with an enhanced user interface providing detailed explanations and treatment suggestions. Expanding the dataset to include multiple plant species and deploying the system as a mobile app would further improve accessibility and efficiency for farmers.

References:

Here is the list with the format you requested:

[1] Arivazhagan, S., R. Newlin Shebiah, S. Ananthi, and S. Vishnu Varthin: Link

[2]Ms Kiran ,R. Gavhale and Ujwalla Gawande: Link

[3] Sanjeev Sannakki, S. Vijay S. Rajpurohit, V. B. Nargund, and Pallavi Kulkarni: Link

[4] Suhaili Beeran Kutty, Noor Ezan Abdullah, Habibah Hashim, and Aida Sulinda: Link

[5] Prashanth Rothe, and R. V. Kshirsagar: Link

[6] I. El Massi, Y. Es-saady, M. El Yassa, D. Mammass, and A. Benazoun: Link

[7] Aakanksha Rastogi, Ritika Arora, and Shanu Sharma: Link