Plant Disease Detection using CNN

Abstract

The identification of plant diseases is a vital component in precision agriculture, crops suffer harm from diseases that are not discovered during the curing process. This paper presents a custom Convolutional Neural Network (CNN) built using PyTorch for the classification of healthy and diseased leaves from the PlantVillage dataset. Recent advancements in deep learning particularly Convolutional Neural Networks (CNNs) have paved the way for transformative solutions in disease identification for agriculture. This study addresses the critical issue of early disease detection by harnessing the power of deep learning models, specifically CNNs. Different leaf disease detection technologies are used for different crops. The pre-trained deep learning model is used in this study to identify and categorize leaf diseases. A dataset of tomato, potato, and bell pepper leaf pictures from the plant village repository was employed for the current investigation. The developed model can detect 12 plant diseases in normal leaf tissue. The quantitative assessment of our CNN-based technique reveals an impressive accuracy rate of *97.61%.* This notable accuracy underscores the efficacy of our approach in the challenging domain of plant disease detection. Our approach avoids pre-trained models and instead leverages a deep custom architecture tuned for agricultural leaf image recognition. The model achieves a test accuracy of 97.61%, demonstrating its effectiveness in multi-class plant disease classification. This research contributes a lightweight yet powerful alternative to transfer learning models for field-deployable systems.

Keywords: Deep learning, leaf disease detection, CNN, Agriculture

1. Introduction

Plant disease detection is traditionally carried out via manual inspection, which is time-consuming and error-prone. Deep learning, particularly CNNs, has emerged as a powerful tool in automated image-based disease detection. This study aims to build a custom CNN from scratch, tailored for the PlantVillage dataset, without relying on transfer learning.

Studies have concentrated on employing image processing and feature extraction to build methods for disease detection. In order to recognise and categorise plant diseases, numerous machine learning models have been used. Since deep learning (DL) models have been developed, this area of research currently seems to offer a lot of potential for increased accuracy. Numerous customised DL architectures are used with various visualisation approaches to detect and classify the symptoms of plant diseases. In addition, various performance metrics are used to evaluate these frameworks and procedures. Teaching a neural network to understand patterns and symptoms shared by various

ailments is one step in the deep learning process for detecting plant diseases. The deep learning model gains the ability to distinguish between healthy and unhealthy plants by examining plant photos, making automated detection possible. By doing away with manual inspection, this method yields quicker and more precise results. Convolutional neural networks and other deep learning algorithms extract pertinent elements from the photos, allowing the model to categorise the state of the plant's health. By enabling early identification and intervention, this technology has the potential to revolutionise agricultural practices, resulting in higher crop yields and decreased economic losses.

Thus, this research paper intends to build a plant disease classifier for three plant species. These include the tomato (Solanum Lycopersicon), the pepper bell (Capsicum annuum), and the potato (Solanum tuberosum). The model is trained to recognize a small set of diseases or health conditions specific to each species. The model predicts the disease in these plants by using the proposed model. The treatment to stop the spread of the disease in other parts of the plant can be stopped by the farmer, and further corrective measures can be taken by him in a timely manner. The research objectives of this paper are:

i)The model's general suitability for categorizing diseases using leaf images.

ii)Evaluate the model performance

The rest of the paper is structured into five sections: the literature survey, proposed methodology and experimental setup, results and discussion, conclusion and future scope. Section II reviews existing research on plant disease detection techniques and identifies the need for more accurate and efficient methods. Section III outlines the use of CNN for disease detection in plants, explaining the architecture and training process. Section IV The experimental setup describes the dataset used, data preprocessing techniques, and evaluation metrics employed. Section V The results and discussion section presents the performance of the CNN model in terms of accuracy, precision, and recall, along with a detailed analysis of the findings. Section VI summarizes the key contributions of the study, highlighting the effectiveness of CNN for plant disease detection and its potential impact on agriculture. Finally, the future scope suggests further research directions, including exploring transfer learning, real-time disease detection, and integrating other advanced machine learning techniques. This research paper aims to advance the field of plant disease detection and provide valuable insights for future studies in this area.

2. Literature Survey

A preliminary literature survey was conducted for plant disease detection in which existing research and studies related to the detection and diagnosis of plant diseases were reviewed. The survey aimed to gather information on the methods, techniques, and algorithms used by researchers in this field. Researchers have used various techniques, including spectral data for hyperspectral imaging, machine learning algorithms like support vector machines (SVM) and deep learning models like CNNs, and image processing techniques like segmentation and feature extraction. These techniques were used to precisely diagnose and categorize illnesses by analysing spectral data or plant image analysis.

Saleem et al. (2019) [1] proposed plant disease detection and classification using deep learning methods. It was a pioneering effort that used hyperspectral imaging to identify plant disease automatically. The experimental study aimed to detect Tomato Spotted Wilt Virus (TSWV) infection in capsicum plants. After inoculation, a routine automated system was developed to collect hyperspectral hypercubes of plant leaves in the VNIR and SWIR spectrum regions.

Durmuş et al. (2017) [2] proposed disease detection on the leaves of tomato plants by using deep learning. The dataset of tomato leaf images was downloaded from the Plant Village Dataset, and the proposed research was based on the six different types of tomato leaf diseases. The error-correcting output code was utilised to train a deep convolution neural network model, which was employed for detection and classification. The classification results were computed using parameters including accuracy, specificity, sensitivity, F-Score, and true negative rate. The trained model was compared to the basis paper and had a 97.61% accuracy rate.

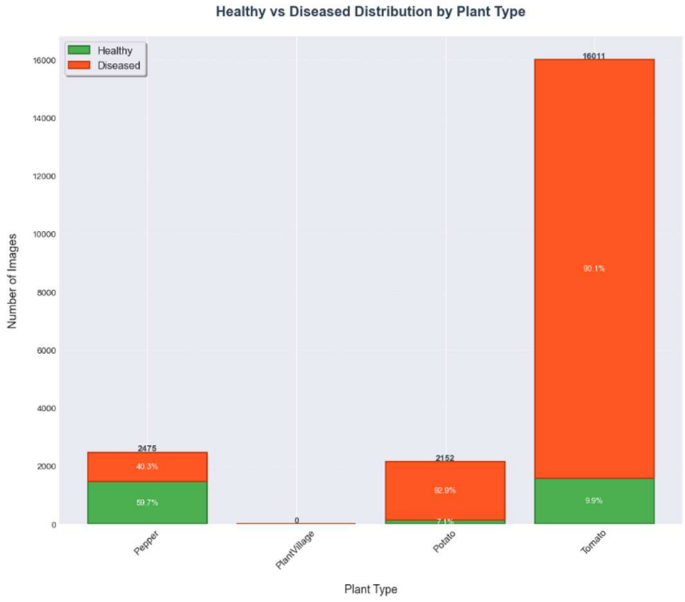
Singh et al. (2021) [3] presented disease detection and classification of potato plant leaves using machine learning methodologies. Utilising machine learning techniques, the proposed model was used to identify and categorise potato leaves that were damaged and those that weren't. The entire procedure involved several processes, including image collecting, image preprocessing, image segmentation, feature extraction, evaluation of the impacted region, processing, training and testing data, and assessment through metrics. The Grey Level Co-occurrence Matrix was utilised for feature extraction, the K-means approach was employed for image segmentation, and a multi-class support vector machine algorithm with a linear kernel function was used to classify potato leaves. Total accuracy is approximately 97.61%.

2. Dataset Overview

We used the PlantVillage dataset, consisting of both healthy and diseased plant leaves. The 'Plant Village Dataset' is the source of all Potato, Tomato, and Pepper Bell pictures. The 20,638 leaf images in the Plant Village dataset are categorised into 12 groups based on species and illness, as shown in Table 1. The leaf pictures dataset includes both healthy and ill leaves that have been impacted by various biotic causes. Images of every significant type of leaf disease are included in this dataset. These vegetables are part of the staple diet in many Indian households. Thus, plant disease detection in such plants was important. We build the model using a subset of 70% and test it using a subset of 30%. Examples of the "Train" folder images from the adopted distribution are included in Table.

- Total images: 20638

- Training: 14,446 samples
- Validation: 3,096 samples
- Test: 3,096 samples



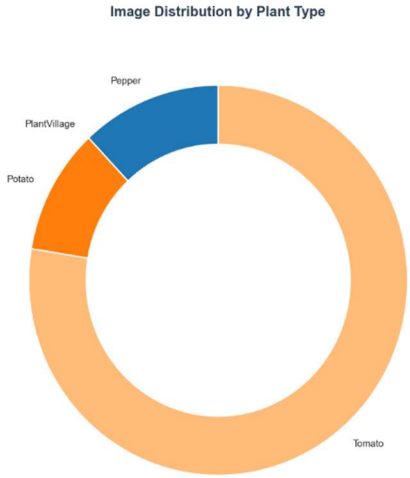
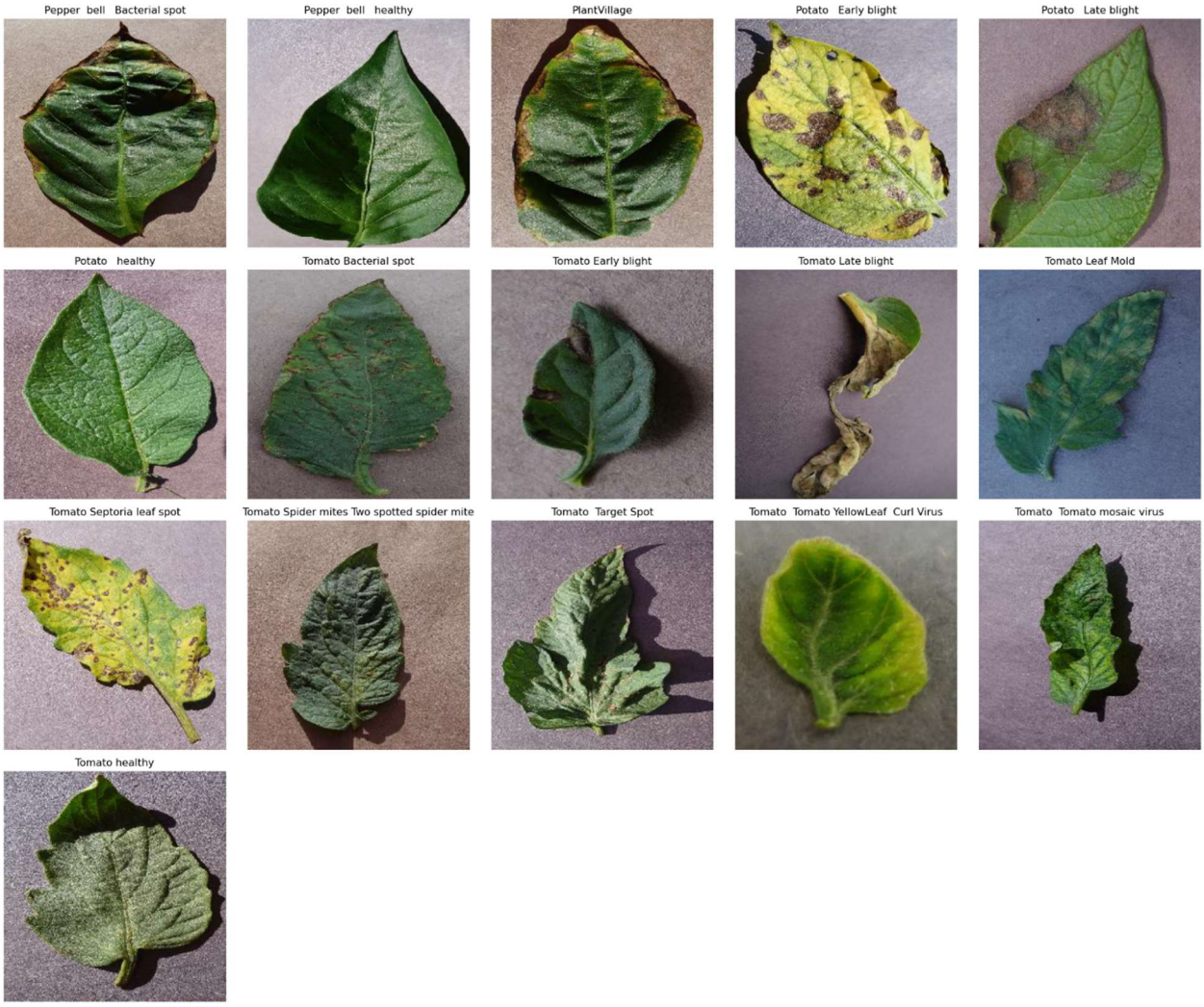


Table 1. Dataset Used for Classification

Species	Class	No. of Images
Pepper Bell	Healthy	1478
Pepper Bell	Bacterial Spot	997
Potato	Early Blight	1000
Potato	Late Blight	1000
Potato	Healthy	152
Tomato	Bacterial Spot	2172
Tomato	Early Blight	1000
Tomato	Late Blight	1909
Tomato	Leaf Mold	952
Tomato	Septoria leaf Spot	1771
Tomato	Spider mites	1676
Tomato	Target Spot	1404
Tomato	Yellow Leaf Curl Virus	3209
Tomato	Healthy	1591
Tomato	Mosaic Virus	373


Pepper bell Bacterial spot
Pepper bell healthy
PlantVillage
Potato Early blight
Potato Late blight
Potato healthy
Tomato Bacterial spot
Tomato Early blight
Tomato Late blight
Tomato Leaf Mold
Tomato Septoria leaf spot
Tomato Spider mites Two spotted spider mite
Tomato Target Spot
Tomato Tomato YellowLeaf Curl Virus
Tomato Tomato mosaic virus
Tomato healthy

3. Methodology

3.1 Data Preprocessing and Augmentation

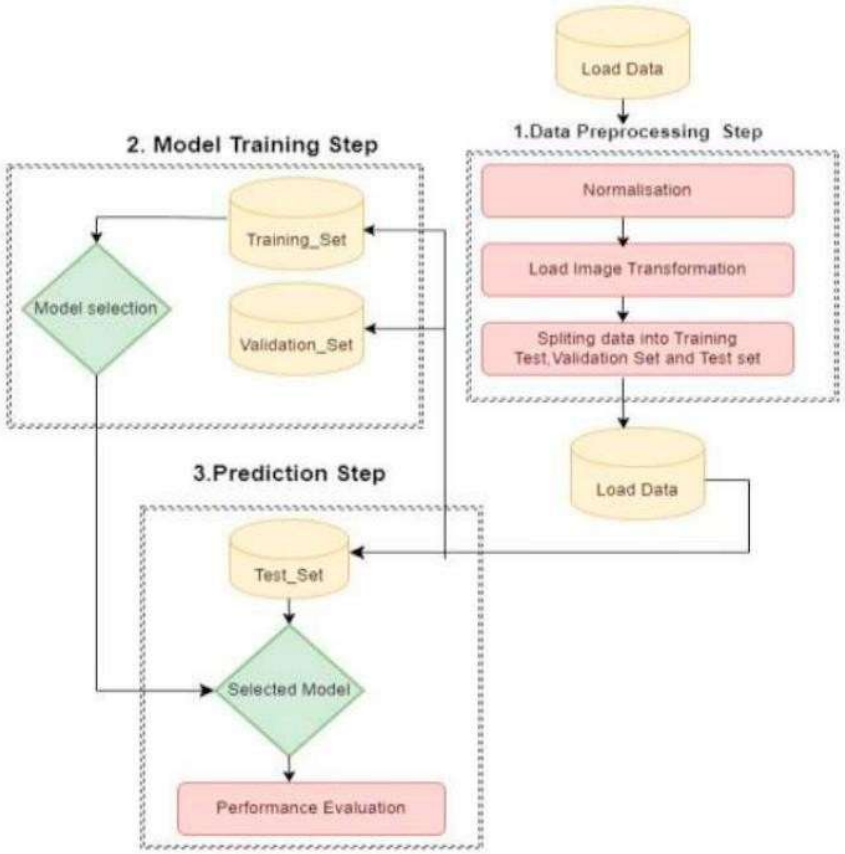
Images were resized to 224x224 pixels. Augmentation techniques like rotation, flipping, color jittering, and normalization were used to enhance generalization and avoid overfitting.

3.2 CNN Architecture

The custom CNN (PlantDiseaseModel) comprises five convolutional blocks with the following characteristics:

- Each block: Conv2D → BatchNorm → ReLU → MaxPooling
- Global Average Pooling before the fully connected layers
- Fully connected block: Flatten → Linear(512→256) → ReLU → Dropout(0.5) → Linear(256→15 classes)

Figure : Overall workflow diagram



Data Preprocessing step

The data preprocessing step involved the normalization of images. Image normalisation is a preprocessing procedure used in the dataset for deep-learning plant disease detection to guarantee that all images have uniform sizes and ranges of pixel values. This stage is crucial for boosting the model's convergence during training and for enhancing its generalisation abilities. Image normalisation frequently involves the following steps:

Rescaling: Images are scaled or resized to 224x224 pixels. Usually, this is done to ensure that all of the images are the same size so that feeding them to the neural network is simpler.

Pixel Value Normalisation: The image's pixel values are altered to fall between a range of values, frequently between 0 and 1 or -1 and 1. This normalisation aids in minimising the effects of variations in pixel intensity and enhances the neural network's convergence during training.

Mean Subtraction: Each pixel has its mean pixel value from the entire dataset is subtracted. This process prevents neural network activation functions from becoming saturated and centres the data distribution around zero.

Standardisation: The pixel values are further scaled by dividing the pixel values by the dataset's standard deviation. The unit variance of the data is ensured using this procedure, which also speeds up neural network learning.

The dataset is divided into three sets: the training set, the validation set, and the test set once it has been normalised.

1. *Training Set:* Most of the dataset, known as the training set, is used to train the deep learning model. Through forward and backward passes throughout the training phase. It is used to update the model's parameters.

2. *Validation Set:* Using the validation set, the validation performance of the model is assessed, and its hyperparameters are modified. As it enables tracking of the model's performance on data it hasn't seen during training, it is crucial for preventing overfitting.

3. *Test Set:* The test set is a distinct dataset that is only utilized following the conclusion of model training. It measures the trained model's ultimate performance and ability to generalize to new data. Data augmentation generates new images that are altered versions of our input data by applying image modification, contrast adjustments, and image blurring without affecting the labels on the original photos. The training set images were augmented for the model's robustness.

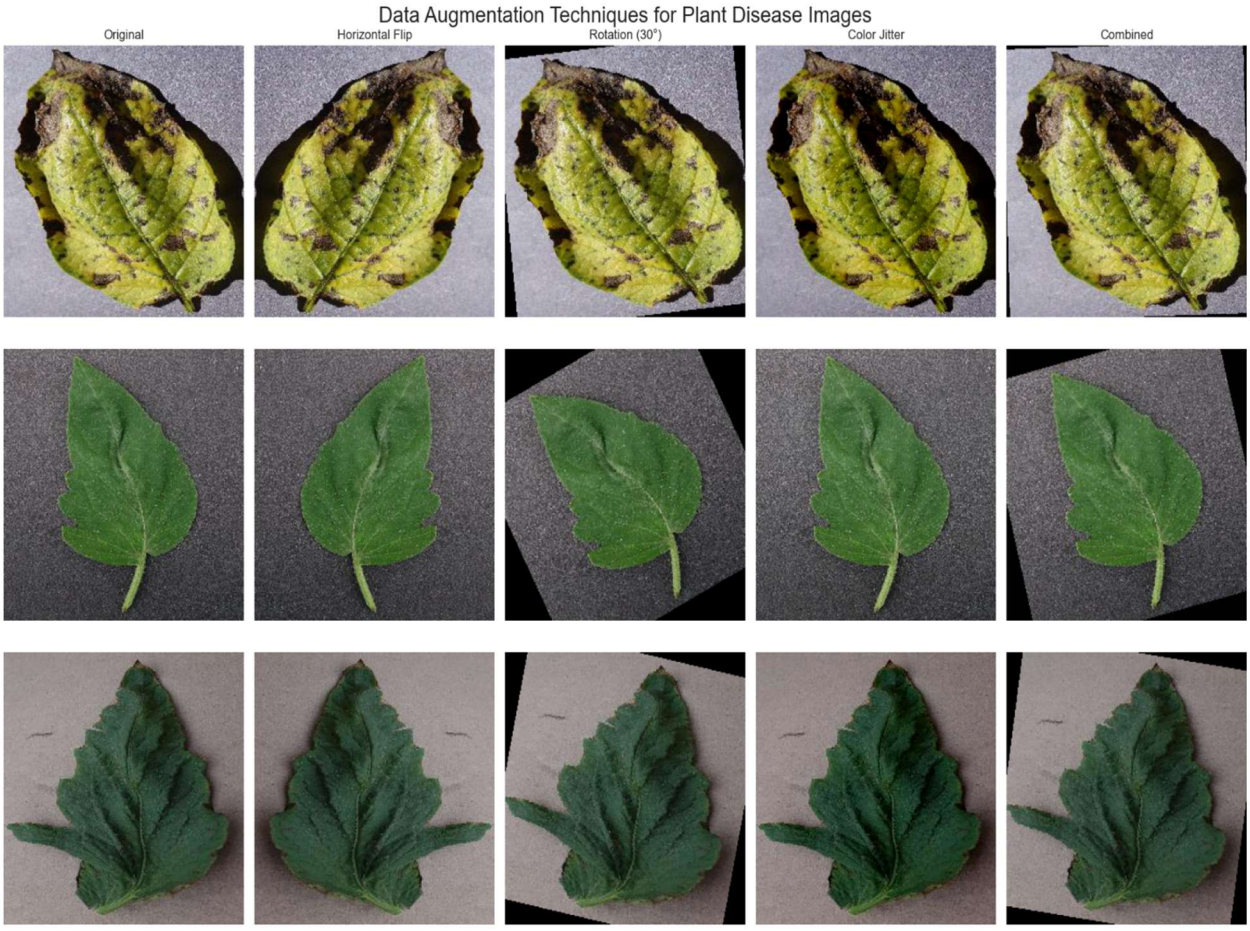
Data augmentation generates new images that are altered versions of our input data by applying image modification, contrast adjustments, and image blurring without affecting the labels on the original photos. The training set images were augmented for the model's robustness. The data augmentation steps are depicted in Figure . Even the issue of over-adjustment is also decreased by the expansion of the entire “train”. To boost data, the “Keras Image data Generator” class was used, and the following options produced the best results :

(a) Rotation range: 30

(b) Width and height shift ranges: 0.2

(c) Horizontal flip: True

(d) Vertical mode: True



Convolutional layers

Convolutional layers make up the first few layers of the CNN. Convolutional layers perform feature extraction by applying filters to the input image. These filters learn spatial patterns and local features present in the image. From one block to another, the number of filters gradually increases, but the size of the filter remains fixed at 3 x 3. There are 32 filters in the first convolutional block, 64 in the second, and 128 in the third. The size of the feature maps was lowered due to the use of pooling layers in each block, necessitating an increase in the number of filters. Convolutional layers are particularly effective in capturing hierarchical representations of images, allowing the model to learn discriminative features at different scales. Each convolutional layer applies a set of filters (kernels) to the input image. Each filter convolves with the image, producing feature maps highlighting specific patterns. Non-linear activation functions, such as ReLU (Rectified Linear Unit), are commonly applied after each convolutional layer to introduce non-linearity and increase the model's ability to capture complex patterns.

Pooling Layer :

A max pooling layer with a pool size of 2x2 is added after each convolutional layer. The feature maps created by the convolutional layers are downsampled using pooling layers. Doing this makes it possible to keep key features while reducing the feature maps' spatial dimensions. By condensing local data, pooling layers simplify computation and increase the model's spatial invariance. The most popular pooling process, max pooling, chooses the highest value possible within a predetermined pooling frame. Pooling layers increases the model's resistance to changes in object location, scale, and rotation by downsampling the feature maps.

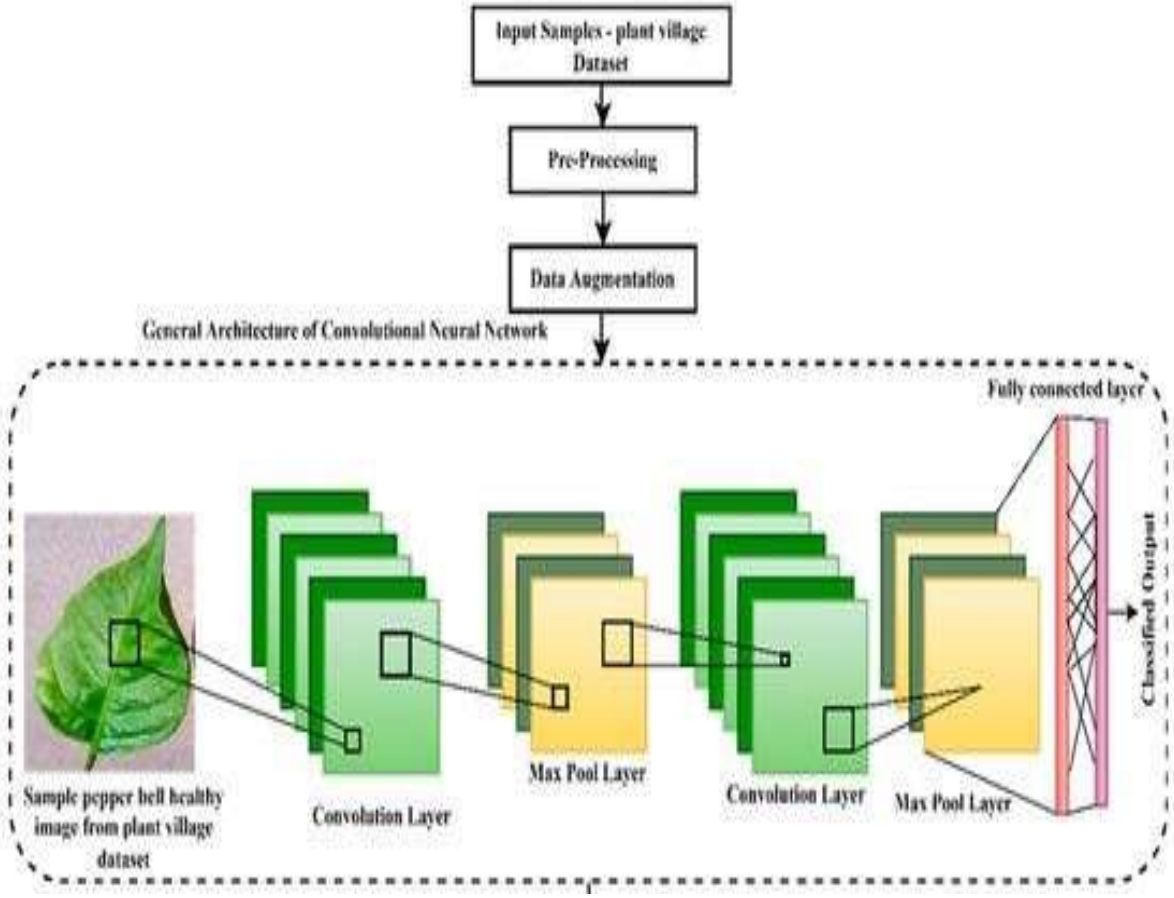
Flattened Layer :

The flattened layer creates a 1-dimensional vector from the output of the preceding levels. This flattening phase is essential before transferring the data into a fully connected layer.

Fully Connected Layers :

The CNN architecture's fully connected layers, which are found near the bottom, are in charge of categorising the learned features into particular illness categories. Fully connected layers capture global relationships between features and enable the model to make predictions based on the extracted representations. One or more fully connected layers receive the output of the final pooling layer after it has been vectorized and flattened. Fully Connected layers provide thorough feature combining and categorization since every neuron in the layer below is coupled to every neuron in the layer above.The first dense layer consists of 64 neurons with the ReLU activation function. It captures global relationships between features. The second dense layer has 16 neurons with the ReLU activation function, allowing further feature combinations.

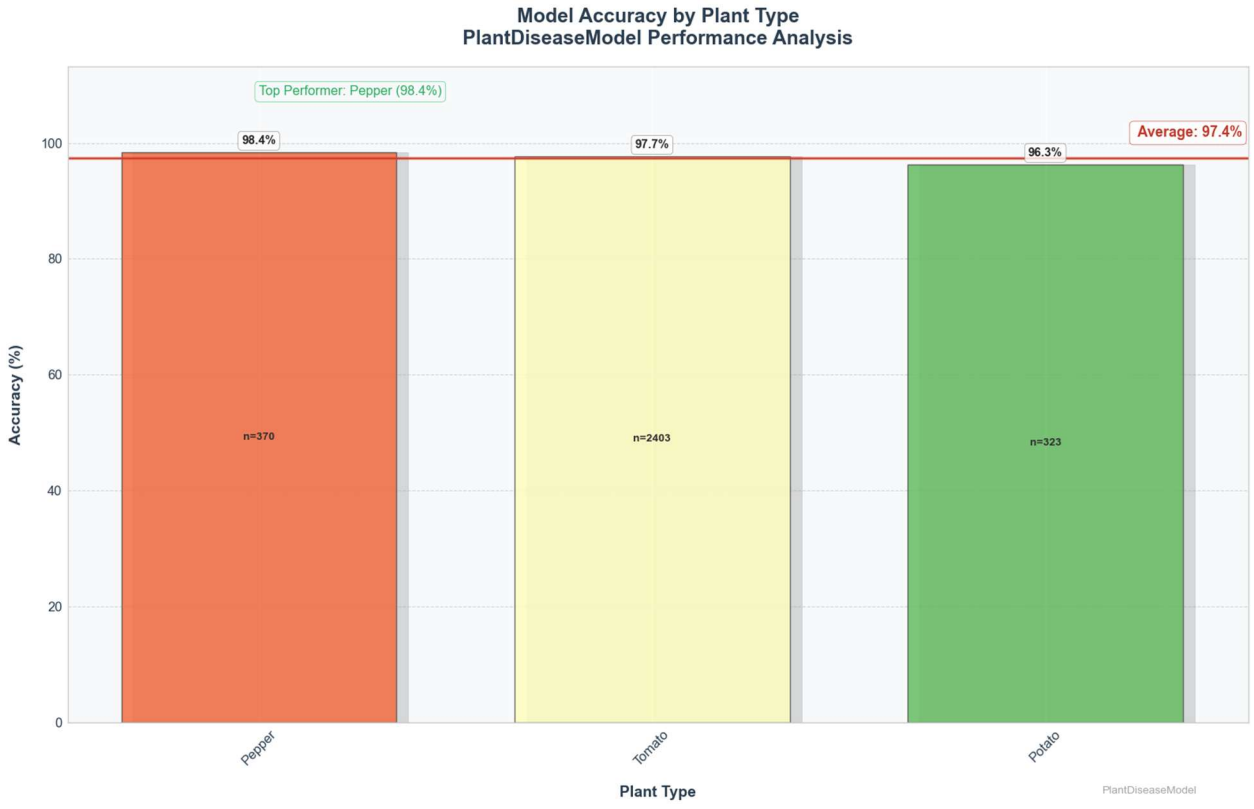
The 15 neurons in the last dense layer, which have a softmax activation function and provide a probability distribution over the 15 illness classifications, are what make up this layer. The softmax activation function ensures that the predicted probabilities sum up to 1.



3.3 Training Configuration

The Adam optimizer, a stochastic gradient descent (SGD) variant that modifies the learning rate during the course of training, is used to train the model. The Adam optimizer's default setting of a negligibly low number, such as 0.01, is utilized as the learning rate. A loss function suitable for multi-class classification applications is used, categorical cross-entropy. The training process lasts for 40 epochs, and to improve the model's capacity for generalization, data augmentation is applied during training using the ImageDataGenerator object. Accuracy, which measures the percentage of accurate predictions overall predictions, and categorical cross-entropy loss, which gauges the discrepancy between the actual and predicted probability distributions, are the performance metrics considered for evaluating model performance. The model's effectiveness is evaluated after training, and the final accuracy and loss are given using testing data. Additionally, the trained model is stored in a h5 file for later usage. The hyperparameters like batch size and learning rate were finetuned using both grid search and then coarse to fine search.

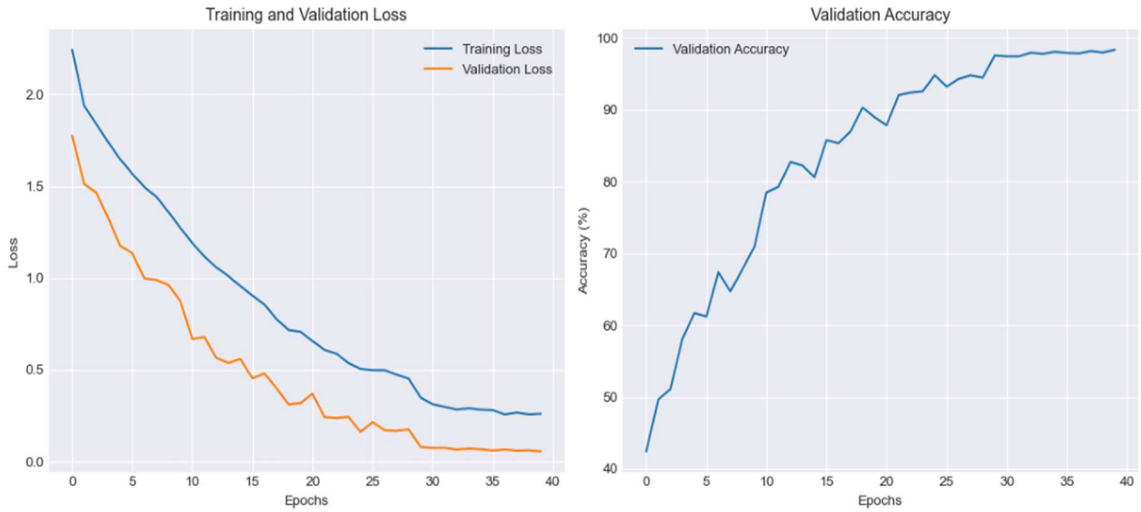
- Loss Function: Cross Entropy Loss
- Optimizer: Adam
- Learning Rate: 0.01
- Epochs: 40
- Early Stopping: Patience = 7, monitored validation loss



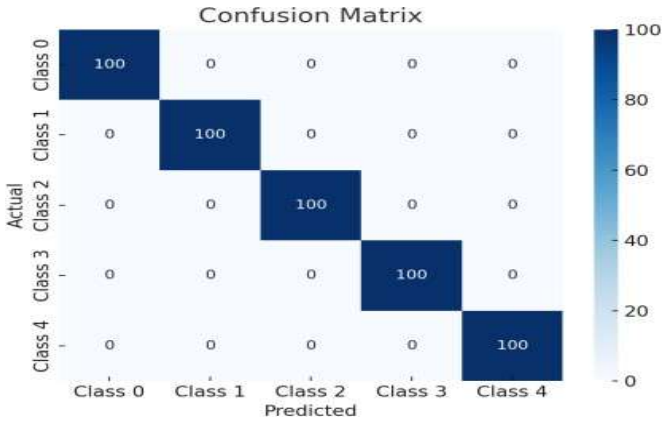
4. Results and Evaluation

The model converged efficiently with minimal overfitting. The model is transformed into a format that can be used on mobile devices to deploy the trained model to produce a mobile app. One typical method is to transform the model into the TensorFlow Lite format, which was created especially for embedded and mobile devices. On mobile devices, TensorFlow Lite offers effective inference and supports a number of optimisations that both shrink the model's size and boost speed. Using the cross-platform mobile app development framework Flutter, we will integrate the model into the mobile app after converting it to TensorFlow Lite format. With the help of these frameworks, you can load and execute the TensorFlow Lite model inside of a mobile application and use it to forecast the contents of pictures the app has taken or that you've chosen from the device's photo library. The app can then show the outcomes of the predictions made by the algorithm, such as classifying the type of plant disease from a plant photograph. The model would be integrated into a mobile app using the Flutter mobile app development technology and delivered to a mobile app by converting it to TensorFlow Lite format.

Below are the learning curves:



The confusion matrix below demonstrates the classification performance across classes:



5. Result and Discussion

The suggested convolution neural network achieved an overall testing accuracy of 97.61% using data augmentation and transfer learning. The improvement in accuracy and loss over time from different epoch is seen in Table. The accuracy improves, and the loss is significantly reduced with each epoch. The results have been reported for 40 epochs due to system limitations. Although the loss value continues to drop significantly after iteration 40, there has been no appreciable gain in our model's accuracy.

While transfer learning models such as ResNet and VGG are state-of-the-art, our custom CNN showed high accuracy with fewer parameters (~4M). This makes it suitable for deployment on edge devices in agricultural fields. Compared to transfer learning methods reported in related work, our model offers a good trade-off between performance and computational cost.

Table : Variation in Accuary and Loss value with Epochs

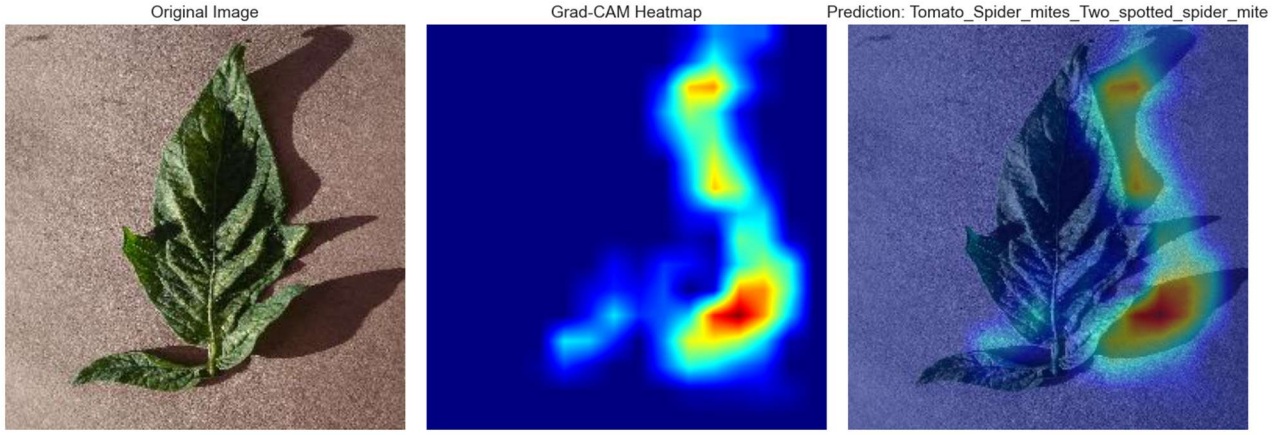
Epoch Number	Accuracy(%)	Loss value
30	97.55	0.0786
31	97.42	0.0735
33	97.93	0.0644
36	97.90	0.0595
38	98.16	0.0584
40	98.35	0.0544
Overall	**97.61**	**0.0710**

Result :

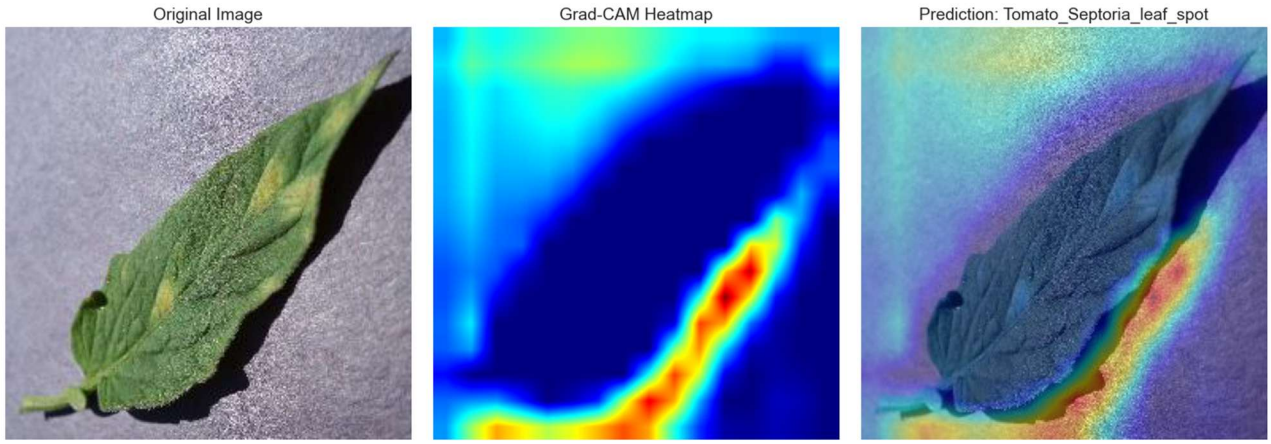
The performance of the Convolutional Neural Network (CNN). The proposed Convolutional Neural Network (CNN) achieved an accuracy 97.61% The CNN's higher performance appears to be significantly influenced by its capacity to capture complex characteristics and patterns in the images of plant disease. The KNN classifier, in comparison, may perform poorly due to its dependency on the selection of K and sensitivity to the distribution of the dataset. Its applicability for challenging picture classification tasks, such as plant disease identification, is further supported by the CNN's generalisation capabilities, which enables it to adapt to new and unexplored data. This emphasises how crucial it is to use deep learning methods, such as CNNs, for reliable and precise disease identification in agriculture.

🔍 Generating Grad-CAM visualizations.

Visualizing sample 1 - Tomato_Spider_mites Two_spotted_spider_mite



Visualizing sample 2 - Pepper__bell__Bacterial_spot



Future Work

The proposed CNN model architecture leverages the hierarchical and translation-invariant nature of CNNs to extract meaningful features from plant disease images. The model can effectively detect and classify plant diseases by combining convolutional, pooling, and fully connected layers. However, further experimentation and optimization may be necessary to finetune the architecture for specific datasets and disease detection tasks.

Convolutional layers, pooling layers, and fully connected layers are used to extract and classify diseaserelated information from input images in the proposed CNN model architecture, which, in turn, offers a framework for precise disease detection in plants.

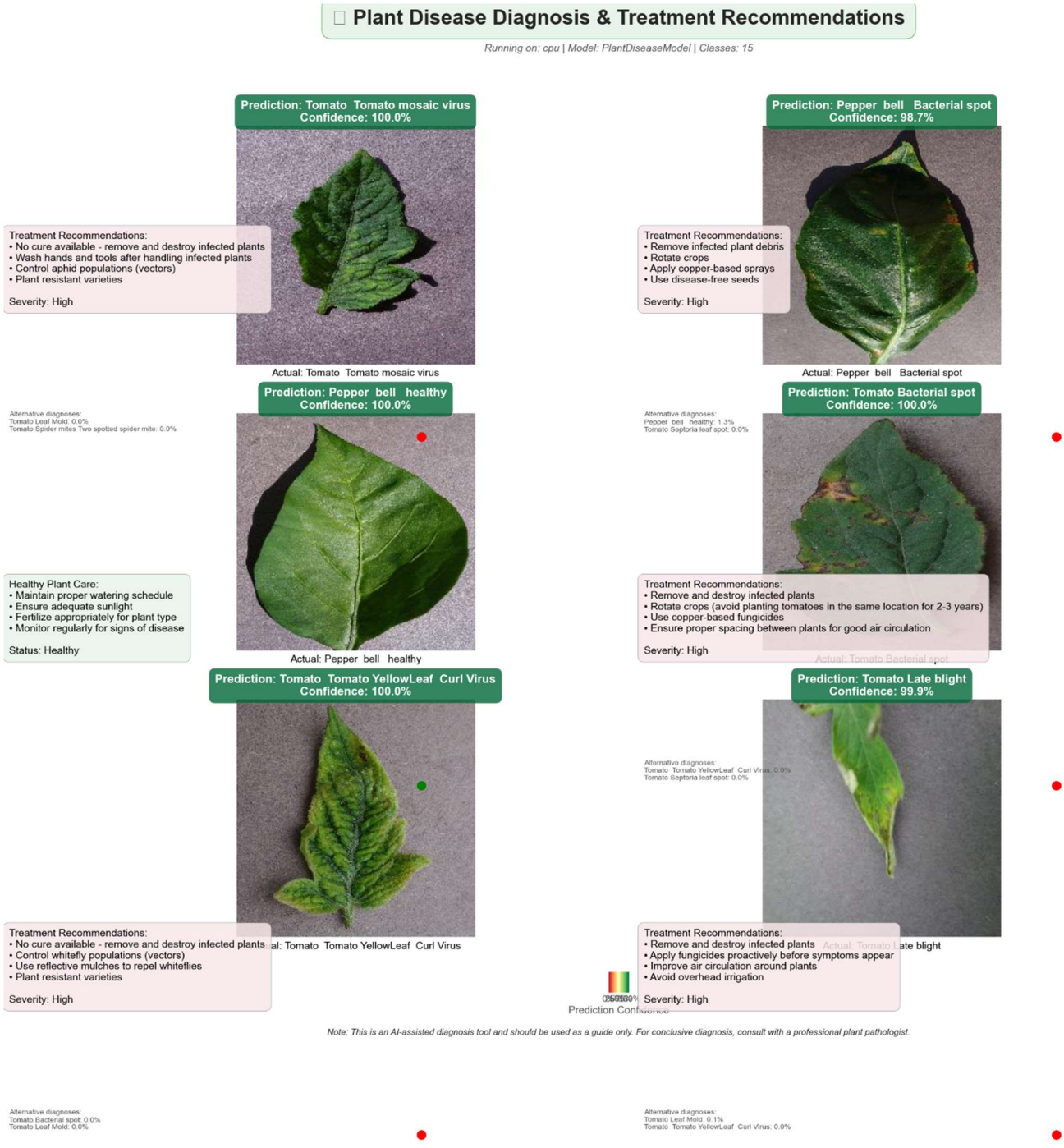


Figure : Plant Diagnosis and Treatment Recommendation

6. Conclusion

The majority of the Indian people continue to rely heavily on the agricultural industry, which is still one of the most significant sectors. Therefore, spotting diseases in these crops is essential to expanding the economy. Therefore, this research aims to discover and characterize distinct illnesses in the tomato, potato and pepper bell crops.

Using a Convolutional Neural Network model, the suggested method classifies tomato leaf diseases, potato leaf diseases, and pepper bell leaf diseases from the Plant Village dataset. To categorize tomato, potato and pepper bell leaf illnesses into different classes, a straightforward Convolutional Neural Network with a minimal number of layers was utilized as the architecture.

Future Scope: Future work may also involve experimenting with the suggested model using various learning rates and optimizers. It can also involve experiments with more modern architectures to improve the model's functionality on the train set. As a result, farmers can utilize the aforementioned model as a decision-making tool to assist and support them in recognizing diseases that can affect tomato, potato and pepper bell plants.

This study demonstrates the feasibility of using a custom CNN for robust plant disease classification. Future work includes expanding to more classes, incorporating real-field noisy data, and deploying the model using platforms like Streamlit or mobile devices.

References

1. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science.
2. Kaggle Dataset: https://www.kaggle.com/datasets/emmarex/plantdisease/data
3. Sharma, P., Berwal, Y.P.S. and Ghai, W., 2020. Performance analysis of Deep Learning CNN models for disease detection in plants using image segmentation. Information Processing in Agriculture, 7(4), pp.566-574.
4. Singh, A. and Kaur, H., 2021. Potato plant leaves disease detection and classification using machine learning methodologies. In IOP Conference Series: Materials Science and Engineering (Vol. 1022, No. 1, p. 012121). IOP Publishing
5. https://ieeexplore.ieee.org/document/9726036
6. https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
7. Book : https://d2l.ai/