Journal of Theoretical and Applied Information Technology
30st June 2026. Vol.104. No.12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

LAB-CLAHE ENHANCEMENT AND SYMPTOM-CENTERED PREPROCESSING FOR ROBUST LEAF DISEASE DETECTION

G NAGI REDDY, V VIJAYA KUMAR

[1,2]School of Engineering, Anurag University, TS, India
E-mail: gnagireddy871@gmail.com

ABSTRACT

Plant diseases represent a great challenge to agricultural production, with significant economic losses and substandard food. The early symptom of plant disease is often small, low contrast and contaminated with background noise, and the symptom is hard to differentiate from healthy leaf area under different illumination and background conditions. The key to efficient management lies in the ability to detect and apprehend in time which consequently can protect yields. The research indicates a hybrid framework of plant-disease diagnosis (integrating preprocessing based on symptoms, handcrafted feature extraction and deep learning classifier) that is presented in the study. Bilateral filtering and LAB-CLAHE are also used in preprocessing phase to reduce noise as well as increase the visibility of disease signs. Disease-specific characteristics are represented in a set of manually designed descriptors like the gradient, edge, texture, shape, and color features. The neural networks in question are combined with such features as custom CNN, YOLO, and NASNetMobile, creating a complete pipeline of detection. Better activation, which is called Enhanced Tans Network A LL (tanhReLU), is suggested to enhance nonlinear feature learning. Experiments were conducted using a dataset of 10, 000 tomato leaf images with ten disease classes, in more than 25 epochs. The end result CNN had a training, validation, and test accuracy of 93.02,78.33,78.12. Visualizations by use of feature-map and grad-cam established that the network was paying attention to infected leaf areas. Experimental results indicate that the proposed approach has better robustness and performance in detecting tomato leaf disease images, and the proposed visualization approach confirms that the proposed model provides attention to the infected regions. All in all, the hybrid solution proves to be more robust and has a greater diagnostic function than the conventional methods. In conclusion, the proposed hybrid method is found to be an effective solution in terms of efficient detection of plant diseases in complex real-world applications.

Keywords: *Hybrid Feature Extraction, Convolutional Neural Network, YOLO, Nasnetmobile, Bilateral Filtering, CLAHE, LBP, HOG, Custom Activation Function, Deep Learning In Agriculture.*

1. INTRODUCTION

Agriculture is important in the world food security and economic development, particularly in the developing nations, where many people rely on agricultural activities as a means of their livelihood [9]. Healthy farming is necessary in order to have high productivity, quality yield and sustainable agricultural principles. Nevertheless, fungi, bacteria, dental infections, and pests all cause diseases to crops that cause a great loss in yield and quality of the crop, and trigger subsequent losses incurred, and food shortages. Agricultural research indicates that much agricultural output in the world is wasted every year because of outbreak of diseases [10]. The timely and correct identification of the diseases of plants therefore plays a key role in treating and preventing them. Proven monitoring systems are strongly dependent on the manual scrutiny of the farmer or agricultural professional which may prove cumbersome and non-uniform. As computer vision and artificial intelligence develop, automated disease detectors have become possible solutions. These systems have the capacity of analyzing images of plants and detecting the diseases fast and with precision. Robotic procedures aid to minimize the effects of humans and enhance the diagnostic accuracy[11]. Consequently, smart disease detection systems are gaining relevance in the contemporary precision agriculture.

Plant diseases are difficult and prone to errors particularly when manual identification is used especially where the farmer lacks technical know-how or the farmers do not have the services of the agricultural experts [12]. Visual manifestations are similar in many diseases, so it is hard to differentiate them by examining them with naked eyes. The lighting, the orientation of the leaf, as well

as the background noise are other environmental factors which make accurate diagnosis more difficult. Common inspection techniques are time consuming and not suitable in large farms where thousands of plants are to be inspected on a regular basis. In addition, wrong or late diagnostics may cause the wrong application of pesticides, higher expenses, and grievous losses. The current computer vision algorithms tend to utilize the presence of a straightforward color segmentation or manually designed features, which are prone to noise and changes in the lighting condition [13]. The methods of deep learning have enhanced the ability to detect, although most models lack the capacity to detect disease symptoms that have low contrast, or infections at the initial stages. Moreover, there are methods that rely only on raw RGB images and do not highlight symptomatic features [14]. The problem is that there is a necessity of the system which can correctly diagnose diseases in different conditions. A system like that must not only be fast and scalable but it must also be able to be used in highlighting early symptoms. Thus, the integrated system that will enhance the performance of the plant disease detection needs a strong and hybrid solution.

1.1 Traditional Image Processing Approaches

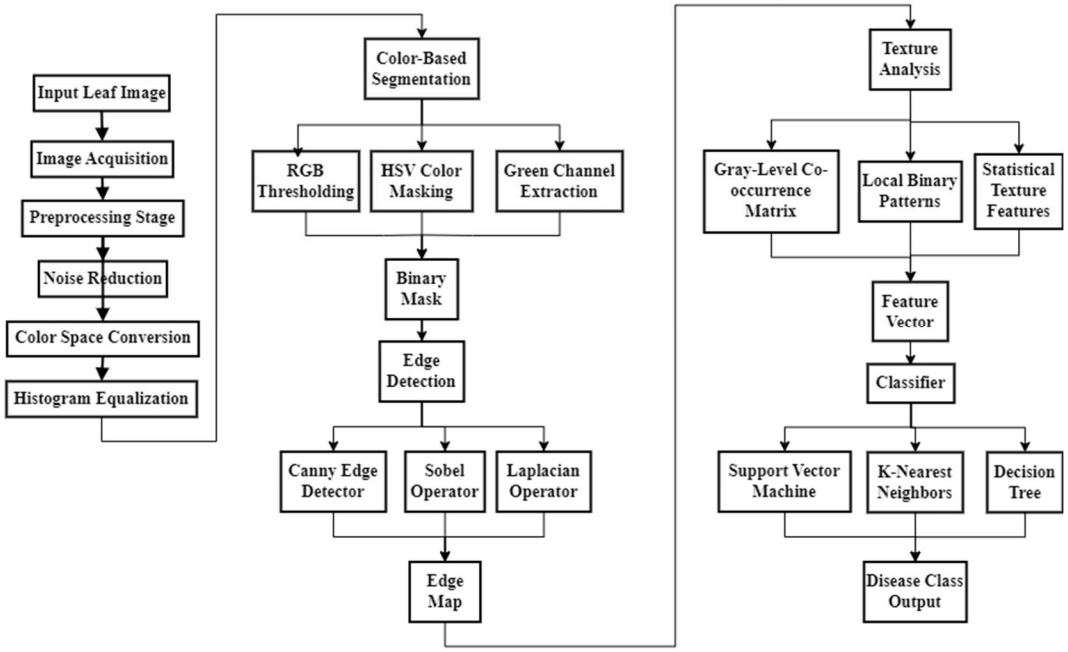

Figure 1: Traditional Plant Disease Detection Pipeline

The conventional systems that were being used to detect plant diseases mainly used classical image processing methods with basic machine learning classifiers [15]. These techniques usually started with the reception of the image, and then preprocessing was done (removal of noise, enhancement of contrast, etc.) Isolating diseased areas and healthy leaf areas by using color thresholds was a common method of isolating diseased areas by color-based segmentation. The edges of the lesions were detected using Sobel or Canny operators after segmentation. The descriptive features of the segmented regions were obtained through straightforward techniques of texture analysis, such as Local Binary Patterns or Grey-Level Co-occurrence Matrices [16]. These handcrafted features were subsequently taken to an existing disease classification classifier, such as a Support Vector machine, a Decision tree, or a K nearest neighbor, among others. Although these methods were easy to calculate as well as simple to execute, they were very sensitive to all the lighting conditions and the background noise. Minor differences in color or brightness had the potential of influencing the accuracy of segmentation heavily. Furthermore, intricate patterns of diseases could not be defined by features that were handcrafted. This made the conventional approaches unsustainable, and unsuitable to the practical agricultural application.

1.2 Machine Learning with Handcrafted Features

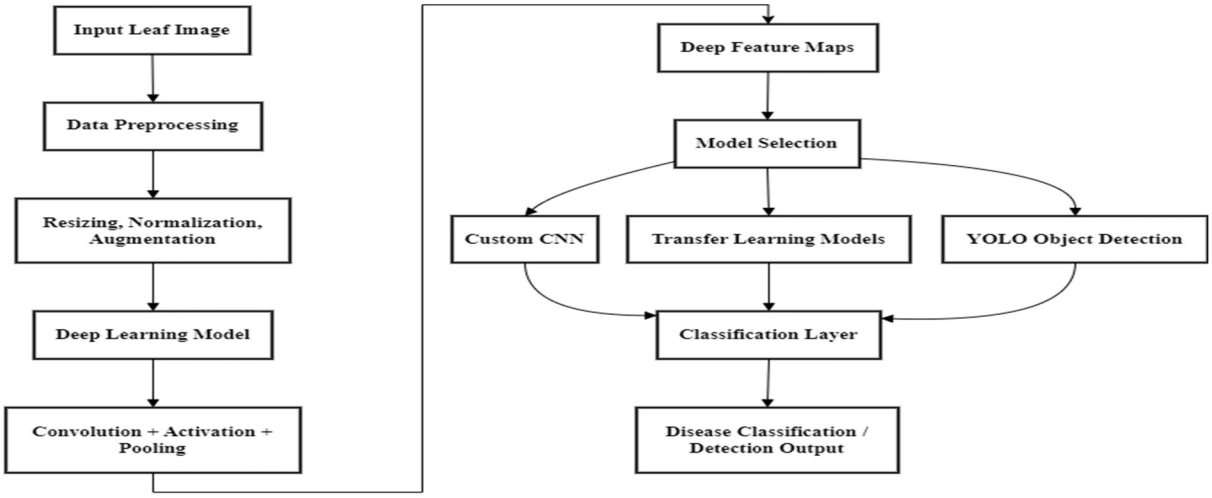

Figure 2: Handcrafted Feature-Based Disease Detection Pipeline

The techniques of machine learning based on handcrafted features were proposed in order to surmount the performance of the traditional image processing methods. In these techniques, the images are processed at first with some preprocessing stage like eliminating noise, improving contrast, converting to color space, etc. Once the area of interest is isolated, different features are obtained to explain the visual properties of the leaf. Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) are texture features that are usually adopted to describe disease patterns and surface abnormalities. Use of color histograms in RGB space, or HSV space, is useful in identifying discoloration due to infections. Lesion geometry is measured using shape features such as: contour area, and polygonal perimeter. These features are extracted and a collection of them are added into a single feature vector and a normalization procedure takes place before being classified. The performance of the Support Vector Machines, random forests and K nearest neighbor methods are used as traditional machine learning methods to recognize the disease type. Such methods tend to be more effective than threshold-based methods. Nevertheless, their work is extremely reliant on the level of handcrafted features selection. They also cannot capture complicated patterns and most of the times they fail to do so in a different light or different environmental conditions.

1.3 Deep Learning-Based Approaches

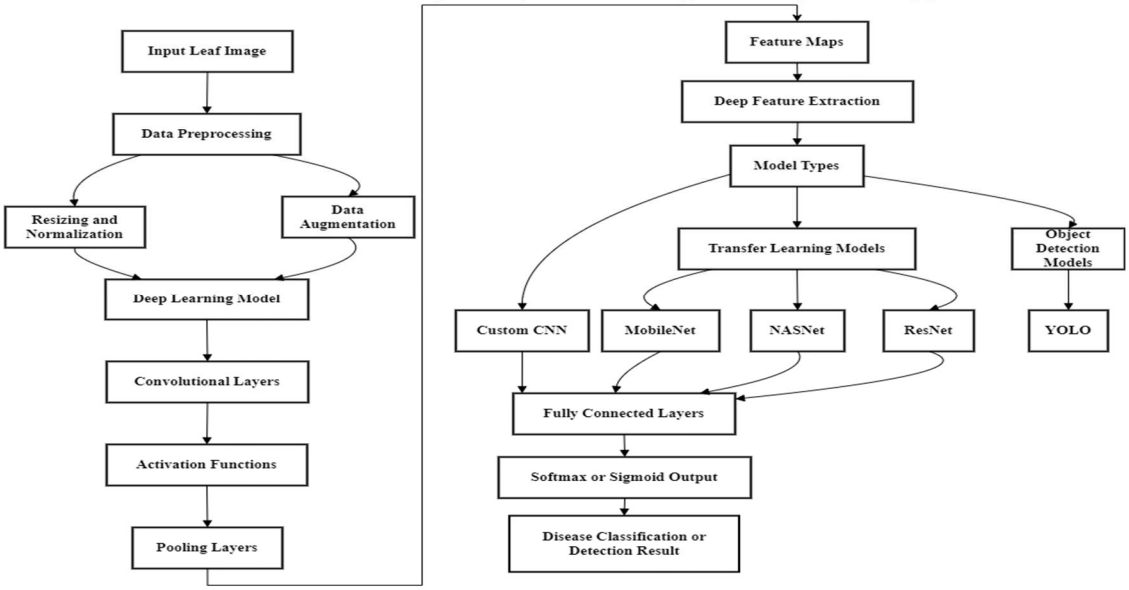

Figure 3: Deep Learning-Based Plant Disease Detection Pipeline

Deep learning methods have also gone a long way in enhancing the performance of plant disease detection systems by reducing the need to do feature extraction manually based on raw images. Convolutional Neural Networks (CNNs) are popular in the task due to their capability to penetrate intricate spatial patterns and textures. A normal deep learning pipeline will immediately size, normalize, and occasionally augment the input images in order to make the datasets more diversified. Then the images undergo several convolutional layers, and automatic hierarchical features are obtained. These activation functions are nonlinear, and pooling layers decrease the size and computational complexity of space. The deep output feature maps obtained are classified by fully connected layers and softmax output. Models of transfer learning like MobileNet, NASNet, and ResNet are commonly applied to enhance performance when working on small sets of data. These trained models take advantage of the learning that has been experienced on massive image datasets. This method is extended by object detection models such as YOLO which identify the diseased regions along with the classification. In spite of the high accuracy of deep learning methods, they usually rely on extensive datasets and cannot pay attention to the features among the symptoms.

1.4 Problem Statement

Plant disease detection is still an important research problem because plant infection can result in significant yield loss, quality degradation and economic damage if the disease is not detected in an early stage. Diseases often present with low-contrast symptoms, are influenced by random noise, irregular lighting, complex background, and leaf orientation in field conditions making the diagnosis difficult to both manual and automated systems. Current approaches often fall short in detecting early signs and handling real-world variability, thus requiring an improved detection framework. For this reason, this study proposes a hybrid method which is expected to enhance the visibility of symptoms and the reliability of the symptom classification process in the challenging agricultural environments.

1.5 Limitations of Existing Methods

Regardless of the remarkable progress in the novel technologies associated with the detection of plant diseases, there are a number of constraints to the existing methods. The conventional approach of image processing is very sensitive to changes in lighting, shadows, and the background noise that may cause the wrong segmentation and feature extraction. Numerous modalities are unable to identify the symptoms of diseases in low stages since slight colors or texture variations are usually ignored. The approaches to machine learning that use handcrafted features are heavily relying on the quality and the choice of those features, so they are less adaptable to a different disease type or environment. Deep learning models, conversely, utilize raw RGB inputs and can be trained to disregard domain specific features like the shape of a lesion or subtle pest texture. These models also make use of huge, varied data in order to generalize. Such datasets are not accessible in a large number of realistic farming environments, which results in lower performance. Moreover, the models trained on the particular crops or conditions frequently fail in use in the other environments. This generalization is not so that their practical application is restricted. Consequently, a stronger and flexible detection framework is required.

Early and accurate detection of plant diseases is essential as prompt identification of the disease can stop spread, minimize crop loss, and enhance agricultural productivity. Manual inspection is challenging in real farming environments because of the different lighting conditions, backgrounds, and the early stages of the disease are subtle, leading to slow and unreliable disease recognition. The proposed study aims to tackle this need with a hybrid system that integrates the symptom-based pre-processing, the handcrafted feature extraction and deep learning-based classification. In the preprocessing stage, certain diseased regions are enhanced and the extracted features are utilized to boost the model performance in recognizing disease accurately and effectively, while the structure of the leaf is retained.

1.6 Motivation for the Proposed Approach

The shortcomings of the current plant disease detection means also indicate that one needs a more holistic and strong solution. The combination of both can be combined to form a hybrid solution that can address the advantages of each model and leverage the potential of the approaches. Other handcrafted aspects, including texture, shape, and color descriptors, have the ability to capture the particular disease factors that deep models might not notice. Simultaneously, computer networks based on deep learning are able to learn intricate and abstract patterns among image data automatically. The other practical one is preprocessing because most of the existing models process raw images without improving the symptoms of the disease. Early disease signs can be brought out by the use of

techniques like contrast enhancement and edge-preserving filtering that involve preprocessing the images based on their symptoms. This can greatly enhance the feature extraction and the performance of the model. Also, regular activation functions are not necessarily the best to be used with disease-related image patterns. Activation functions Custom activation functions can be used to highlight the important features and reject noises. The basis of the proposed framework of hybrid disease detection is based on these motivations.

The detection of plant diseases is difficult due to the fact that many plant diseases have subtle, variable and early non-specific signs and symptoms. Conventional methods and even many deep learning algorithms are not reliable in field images due to the presence of noise, varying lighting conditions, occlusions, and complex background. Furthermore, manual diagnosis has low efficiency and reliability, and is not suitable for large-scale and timely agricultural monitoring. It is desirable to develop a practical detection system that can retain the fine lesion details, highlight the symptomatic regions and increase the robustness in actual farming environments, which is known as a hybrid detection system.

1.7 Contributions of the Proposed Work

The research approach is experimental, comparative and is based on previous research on plant disease detection, medical image analysis and agricultural image processing. These studies suggest that in the noisy field and low-contrast settings, single-stage methods tend to perform poorly, prompting the development of a hybrid method that combines preprocessing, handcrafted features and deep learning. The present study is thus a test of a proposed solution to the above mentioned limitations found in previous studies in order to establish its ability to achieve better accuracy, robustness and practical applicability.

This paper proposes an intermediate plan disease diagnosis framework with dual-purpose pre-processing, multi-feature extraction, and deep neural network system to enhance diagnostic accuracy. The initial significant contribution is a symptom-based preprocessing pipeline, which combines bilateral filtering with LAB-CLAHE contrast enhancement in order to decrease the level of noise and at the same time emphasize the disease area without destroying the leaf boundary. The second contribution is that it extracts numerous complementary handcrafted features, such as gradient, edge, textures, shape, and color features, which give a more comprehensive representation of features of the disease. The third addition is related to the introduction of custom activation functions involving a tanhReLU mixture that was created to push nonlinear learning of features and select nonrelevant information. Lastly, the paper suggests a hybrid deep learning system, which combines these handcrafted features and models, including a custom CNN, YOLO, and NASNetMobile. This synergy exploits the domain-specificities as well as automatic deep feature learning which gives elevated robustness and detection.

This proposes a hybrid method for plant disease detection that integrates preprocessing, handcrafted descriptors, and deep learning to provide early and strong detection. This work presents the feasibility of achieving better disease visibility with symptom enhancement and feature fusion for more accurate classification compared with approaches relying solely on the raw image inputs. The contribution is important as it tackles real world problems of low contrast, noise and background variation, which are still at the heart of the problems in real world agricultural monitoring. The study thus provides a valuable direction for future research on accurate and scalable crop disease detection.

2. LITERATURE REVIEW

Sadman Sadik Khan et al [1] The dataset will include 4, 252 high-resolution leaf pictures (1080x2400 pixels) taken in potato and eggplant cultivations in Bangladesh under seven classes. The methodology starts with the collection of real data in actual fields using mobile cameras which have a 48 megapixel camera mounted on a stand and then the dataset is cleansed and some blurred or overexposed data is eliminated. Preprocessing will include rescaling pixel values between 0-255 between 0-1, downsizing images to 448x448, and dividing them into training (3401 images), validation (426), and testing (425) data sets. The transfer learning uses 5 pre-trained CNN models (VGG16, VGG19, MobileNetV2, ResNet50, and InceptionV3) in which the architectural designs such as depthwise separable convolutions in MobileNetV2 utilize its search features to extract features of a leaf image. The pre-trained weights are adjusted on the custom dataset to model healthy and diseased leaves between potato and eggplant and different classes. The procedure focuses on augmentation and preprocessing to be generalized to different field conditions.

S. Abisha et al [2] The images used are of brinjal (eggplant) farms in Tamil Nadu and Kerala, India, and have 1500 high-resolution images (3840x2160

pixels) taken with a Nikon Z5 camera in natural lights. This begins with preprocessing (Wiener filtering to remove Gaussian noise) and image enhancement (spatial domain image sharpening and contrast boosting) and image resizing to 256x256 pixels. Using the FCM clustering and EM mixes Gaussian Model, diseased regions are divided. FCM and EM segmented images are then subject to DST to perform multi-resolution analysis that decomposes the image into coefficients that are joined together by use of median rule in a 3x3 window to produce a fused image that retains edges and singularities. Examples of such features found in fused images are texture, color, and structural. These combined feature vectors are used as input to classifiers: RBFNN with input layer, hidden radial basis and output layer; and eight-layer DCNN with convolutional, max-pooling, dropout, fully connected layers, ReLU activations, and SoftMax output with six-classes. The DCNN is based on AlexNet-inspired architecture but with hyperparameters such as 3x3 filter, learning rate of 0.001 and 50 epochs optimized by cross-validation by 10-folds.

W. S. Hettiarachchi et al [3] Main dataset, which is obtained at Kaggle, consists of 8320 images of the chilli leaf, which are divided into six disease classes. A secondary dataset of 3,000 images of a healthy and unhealthy leaf is distinguished, supplemented by methods and divided 80/20 to train/test after the reorganization to 256x256 pixels and normalization. The system uses two-step operation: First, a classification model binary healthy vs. unhealthy is used based on CNN, SVM or KNN and the 3k image dataset. Unhealthy leaves proceed to the multi-class classification of the image dataset of 8k to identify one of six diseases using the same algorithms. DenseNet121 is the implementation of CNN in hierarchical extraction of features via dense convolutional layers, global average pooling, ReLU-based FC layers with dropout to ensure regularization, batch normalization, Adam optimizer, and early stopping, which yields six classes by the use of the softmax. SVM identifies the most optimal hyperplanes with the greatest margins in feature space whereas KNN is associated with vote by majority of the k-nearest neighbors through distance measurements. The state-of-the-art CNN is plugged into a FastAPI server to perform inferences in real-time and a Flutter frontend mobile application that enables one to capture and upload images, diagnose an illness, and recommend a remedy rule to an expert-curated knowledge base.

Zahida Yaseen et al [4] The dataset is obtained through the work of Roboflow and consists of high-resolution images of crops of chilli that were shot in poor environmental conditions in the form of three maturity stages. Photographs are characterised by contrasting lighting and angles, uniformity of the classes following preprocessing, such as resizing, data augmentation and pixel normalisation to the 0-1 range, divided into train, validation, and test sets. The system takes the images of chilli plants through the IoT devices, preprocesses by resizing, augmentation, and normalisation, and passes them through the YOLOv11 object detection model that has been trained and combined to loss functions. YOLOv11 processes images to both classify the maturity stages and localize the chilli pods using bounding boxes through the use of improved loss functions, hybridizing attention mechanisms, and higher real-time capabilities than the predecessors. The model provides real-time classification of maturity and coordinates and labels that can be used in automated decisions of harvest. The system has been trained to work with a GPU environment and enables precise agriculture, by giving farmers detailed feedback that allows them to act on the readiness of crops in large fields with different conditions.

Marriam Nawaz et al [5] PlantVillage dataset consists of around 4 500 annotated potato leaf images which are normalized to 299 by 299 by 3 pixels and consist of three classes. PotatoGuardNet is a system that consists of an Inception-ResNetV2 backbone that is combined with Faster R-CNN to localize and simultaneously classify potato leaf diseases. Input images that are resized to 299x299x3 enter Inception-ResNetV2 and the stem (9 convolutional and 2 max-pooling layers), followed by 12 residual modules that happens to extract multi-scale deep feature maps. The RPN produces anchor boxes with various scales and aspect ratios on feature maps by passing a 3x3 sliding window over the feature maps and predicts objectness scores and refinement offsets. The best proposals are forwarded to the classification network of Fast R-CNN to make the disease category prediction and the regression network to refine bounding box. Non-maximum suppression removes redundant detections, and final results have disease class labels, confidence, and accurate bounding box coordinates of the affected regions. Optimized with Adam on Nvidia GTX1070 GPU with combined classification and regression loss.

Sarah M. Alhammad et al [6] Samples In the PlantVillage potato leaf disease data, there are 2,152 original images of three classes. To prevent overtraining, the customized VGG16 model adopts ImageNet trained weights and convolutional layers

are frozen instead of the top classification layers substituted with new fully connected layers with dropout to classify three potato leaf diseases. Images of potato leaves with the dimension of 224x224 undergo through VGG16 convolutional and 5 max-pooling layers to extract hierarchical features such as edges, to disease patterns. Global average pooling is a feature that minimizes spatial dimensions followed by Early blight, Late blight, and Healthy prediction with dense layers, ReLU activation and softmax output. The model has been trained using Adam optimizer, categorical cross-entropy loss with a batch size of 32 and 100 epochs in the Google Colab TensorFlow/Keras platform. Grad-CAM visualization The final convolutional layer gradients are used to create class activation heatmaps, which are used to identify disease-specific areas contributing to model prediction in order to make models interpretable. Training/validation split: 15% of training data is used as validation in order to check overfitting of the optimization process.

Maheen Shahzad et al [7] Another dataset, the PlantVillage tomato dataset probably has a sub-sample of 20,000 well-balanced tomato images in 10 classes. To be sure that the model is well trained and tested, images are divided in sampling (70 percent training (14,000), 15 percent validation (3,000), and 15 percent testing (3,000)). The lightweight custom CNN resorts to 224x224x3 RGB tomato leaf images that are processed using a stem layer (3x3 convolution and 32 filters, stride 2) and five bottleneck blocks consisting of depthwise separable convolution, SE blocks of channel attention, and Swish activation functions. Both bottlenecks widen channels using 1×1 convolution, utilizes 3×3 depthwise convolution to extract spatial features, recalibrates channels using SEs global average pooling and gating mechanism, and projects back using 1×1 convolution with skip convictions. The process of preprocessing the images consists of bilinear resizing to a 224x224, normalizing the pixels to the 0-255 range, and augmentation. To classify 10 classes of diseases using a single 7x7x512 feature map, global average pooling is used to reduce the size to 1x1x512 and after that, a fully connected 128-neuron layer with 0.4 dropout and softmax output is employed. The architecture is also focused on edge deployment efficiency with 5.8M parameters used in mobile/smartphone compatibility.

L. N. Swamy et al [8] The datasets used in the study of tomato leaf diseases caused by bacteria in the PlantVillage include tomato leaf disease pictures with a special emphasis on a group of two classes. The dataset is divided into 80 percent to train, 10 percent to validate, and 10 percent to test with the images being adjusted to the size of 256X 256 and increased. Preprocessing that is done on Tomato leaf images include labeling, compression to 256x256 pixel tensors, uniform dimensionality scaling of inputs and augmentation. ResNet-152 model is a model that processes the images with the use of the deep architecture that has 152 layers with a skip connection of the residual type, which allows direct feature propagation across the layers, avoiding the problem of gradient vanishing in very deep networks. Convolutional layers are used to derive hierarchy features in low level edges to high level disease patterns and max-pooling layers with 2x2 filters and a stride of 2 are used to trim spatial features and conserve features of interest. The feature maps then go through the batch normalization and activation functions with the residual blocks preserving information flow using identity shortcut connections. The ultimate convolutional output is transformed into a 1D output in the flattening process, which is then inputted into FC dense layers and AF to classify into two categories, Bacterial Spot and Bacterial Speck. VGGNet can be used as a comparison base with identical 3x3 convolutional layers with same padding, max-pooling, and fully connected layers as well as the identical preprocessed data. The architecture builds on the automatic feature learning property of CNN that takes raw pixel data and does not require any manual feature engineering.

Table 1: Analysis on Existing Approaches

Citied	Algorithm	Dataset	Merits	Demerits
[13]	CNN, MobileNetV2	Bangladesh	Simple and easy to implement.	By removing blur and other issues of images the process which may lead to data insufficiency..
[14]	DCNN, RBFNN	Plant village	By utilizing both methodologies the prediction was efficient.	Without fusion the performance hold little variation.

[15]	CNN, SVM, KNN	Kaggle,	Scalability and increased scope.	Unhealthy images are not identified efficiently.
[17]	YOLOV11	Robo-Flow	Simple and efficient for prediction.	Only particular area was derived.
[18]	PotatoGuardNet – Inception-ResNetv2 based FRCNN	Plantvillage	By utilizing combined TL methodologies the prediction and time consumption was less and efficient.	Should test on real-world applications.
[19]	XAI, DL, TL	Plantvillage, potato leaf,	Worked on huge datasets which has achieved high accuracy.	Required more techniques for clear and efficient image predictions.
[20]	SE and SAF	Plantvillage	Robust	Performance for object detection has to be improved.
[21]	CNN	Plantvillage	Not only leaves but also in tomato disease are identified.	Compared algorithm are not directly DL.

3. PROPOSED METHODOLOGY

Leaf images will be enhanced with bilateral filtering and LAB-CLAHE, followed by the analysis with a fusion of handcrafted features and deep-learning systems, in order to identify plant diseases at an early stage with increased accuracy and robustness in complex field settings.

1. Take all images of plant leaves from the chosen data set.
2. Apply bilateral filtering and LAB-CLAHE to reduce noise and improve diseased regions, by symptom-based preprocessing.
3. Obtain handcrafted features like gradient, edge, texture, shape, and color descriptors.
4. Train the hybrid learning framework with the processed images and extracted features.
5. Train and test the deep learning models (including the custom CNN, YOLO and NASNetMobile).
6. Make comparison of results by accuracy and other performance measures.
7. Evaluate the potential of the proposed framework to enhance the early detection, strength and classification of the disease.

The next system proposed is based on a hybrid pipeline that consists of symptom-based preprocessing, handcrafted features extraction, and deep learning-based classification. It starts with an input leaf image and then the noise is removed by bilateral filtering, and the contrast enhanced by LAB-CLAHE. This stage of preprocessing marks out disease areas without a requirement of leaf edges. Upon improvement, several kinds of features are obtained, such as gradient, edge, texture, shape, and color descriptors. All these features bring about a combined representation of disease characteristics. The improved RGB images and human made features are further fed to the deep learning systems including a custom CNN, YOLO, and NASNetMobile. The CNN does classification according to trained feature maps, whereas the YOLO is able to identify disease spots. To compare the performance, NASNetMobile is utilized in transfer learning. The system is then able to predict the disease class and this is the final output of the system. The system is meant to enhance the accuracy of detection that is achieved via a mixture of domain-specific features and learned features. The system aims to improve detection accuracy by combining domain-specific and learned features. The overall transformation of an input image I through preprocessing can be expressed as: The concatenation of feature maps in the detection head is represented as

$$F_{concat} = [F_1 \oplus F_2 \oplus F_3]$$

where F_1, F_2, F_3 are feature maps from different layers and $\oplus$ denotes concatenation. The bounding box regression loss is computed as

$$L_{bbox} = 1 - IoU(B_{pred}, B_{gt})$$

where B_{pred} is the predicted bounding box and B_{gt} is the ground truth box. The classification loss is defined using cross-entropy as

$$L_{cls} = -\sum_{i=1}^{C} y_i log(p_i)$$

where y_i is the ground truth label and p_i is the predicted probability for class i.

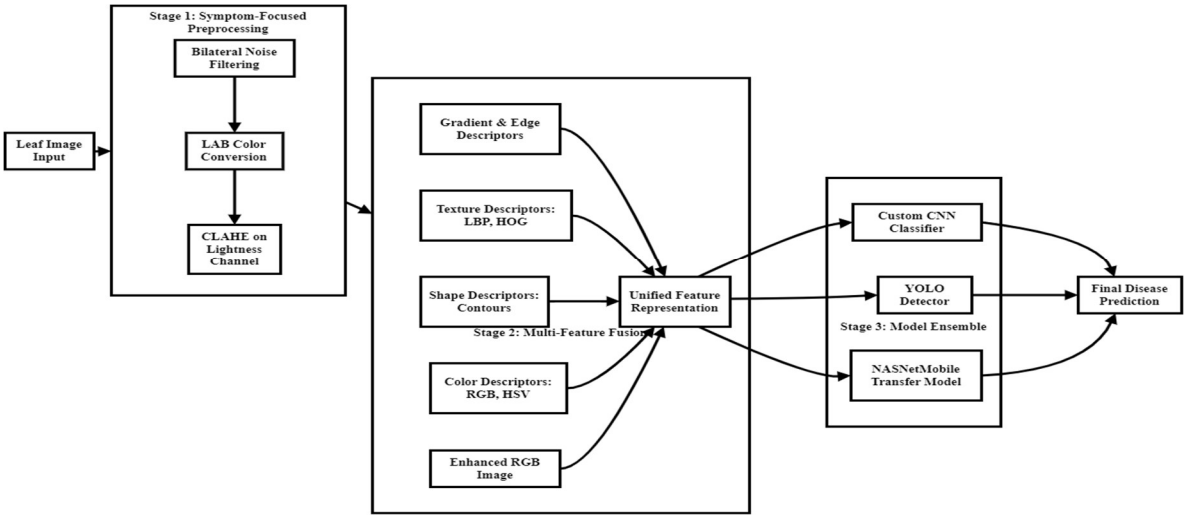

Figure 4: Proposed Hybrid Plant Disease Detection Framework

The main aim of the research is to design a powerful and precise plant disease detection system based on a hybrid methodology. The initial one is to create a new state-of-the-art pipeline of preprocessing that improves the symptoms of disease and still has valuable leaf forms. These consist of bilateral filtering and LAB-CLAHE contrast enhancement. The second goal is to depict several forms of handcrafted characteristics such as texture, form, color, gradient, and edge descriptions. These attributes assist in capturing disease specific attributes which might not have been well learned by deep networks. The third is to combine these artisan features with deep learning models like a custom CNN, YOLO, and NASNetMobile. With this integration, the resulting hybrid system will have the advantages of both domain-specific descriptors and automatic feature learning. The other objective is to add custom activation functions in order to enhance nonlinear feature representations. The last goal is to enhance the overall detection accuracy, robustness, and overallization across the disease types and conditions. All these goals will lead to the development of a more valid and feasible plant disease detection system.

Journal of Theoretical and Applied Information Technology
30st June 2026. Vol.104. No.12
© Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

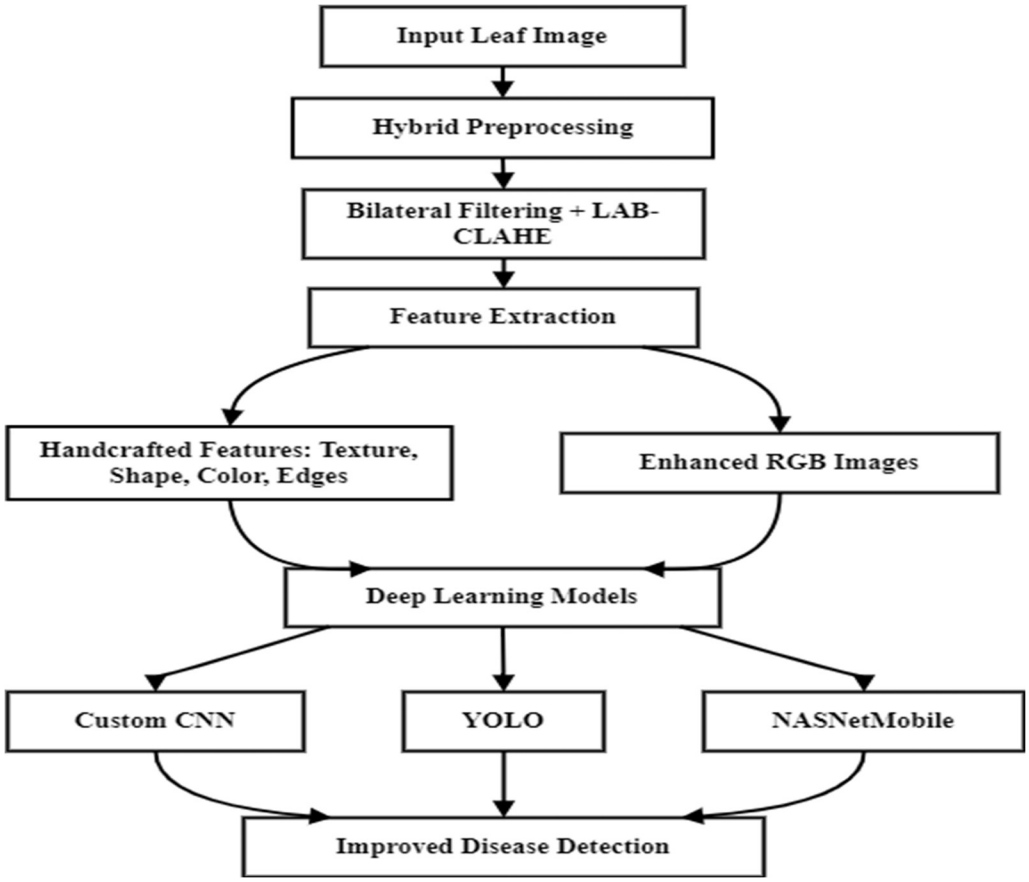

Figure 5: Workflow of the Proposed Hybrid System

3.1 Image Preprocessing

3.1.1 Bilateral Filtering

Pseudocode: Bilateral Filtering

Input:

- Original image $I(x, y)$
- Spatial standard deviation σ_s
- Intensity standard deviation σ_r

Output:

- Smoothed image $I_b(x, y)$

Start

1. Read input image $I(x, y)$.
2. For each pixel p in the image:
   - Define a neighborhood window around p.
3. For each neighboring pixel q in the window:
   - Compute spatial distance: $d_s = \| p - q \|$
   - Compute intensity difference: $d_r = |I(p) - I(q)|$
4. Compute spatial weight: $w_s = e^{-\frac{d_s^2}{2\sigma_s^2}}$
5. Compute range weight: $w_r = e^{-\frac{d_r^2}{2\sigma_r^2}}$
6. Compute combined weight: $w(p, q) = w_s \cdot w_r$
7. Normalize weights.
8. Compute filtered pixel value: $I_b(p) = \frac{1}{W_p} \sum_{q \in \Omega} I(q) w(p, q)$
9. Repeat for all pixels.

End

The first preprocessing is bilateral filtering, which is used to eliminate noise without erasing valuable structural information of the leaf. Bilateral filtering is unlike the traditional linear filters which only depend on the spatial proximity between pixels but not the intensity similarity of the pixels. This enables the filter to flatten out uniform areas and keep edges of lesions and leaves. The weighted average of neighbor pixels is calculated over the spatial and

spatial differences in case of each pixel. The spatial weight determines the effect of immediate pixels, whereas the range weight makes certain that pixels with similar intensities give a larger contribution of the outcome. The two forms of weighting render bilateral filtering as a tool appropriate in the disease detection tasks where the preservation of edges is highly important. The filtered pixel value at location p is computed as:

$$I_b(p) = \frac{1}{W_p} \sum_{q \in \Omega} I(q)\, w(p,q)$$

where W_p is the normalization factor. The spatial weight is defined as:

$$w_s(p,q) = e^{-\frac{\|p-q\|^2}{2\sigma_s^2}}$$

The range weight is defined as:

$$w_r(p,q) = e^{-\frac{(I(p)-I(q))^2}{2\sigma_r^2}}$$

The final combined weight is:

$$w(p,q) = w_s(p,q) \cdot w_r(p,q)$$

This process reduces noise without blurring lesion boundaries, making subsequent feature extraction more reliable.

3.1.2 LAB Color Space Conversion

Numerical Example: RGB to LAB Conversion

Input:

A sample pixel from a leaf image:

$$R = 120, G = 160, B = 80$$

Step 1: Normalize RGB values (0-1 range)

$$R_n = \frac{120}{255} = 0.4706$$

$$G_n = \frac{160}{255} = 0.6275$$

$$B_n = \frac{80}{255} = 0.3137$$

Step 2: Convert normalized RGB to XYZ color space Using standard transformation:

$$X = 0.4124R_n + 0.3576G_n + 0.1805B_n$$

$$Y = 0.2126R_n + 0.7152G_n + 0.0722B_n$$

$$Z = 0.0193R_n + 0.1192G_n + 0.9505B_n$$

Substituting values:

$$X = 0.4124(0.4706) + 0.3576(0.6275) + 0.1805(0.3137)$$

$$X = 0.194 + 0.224 + 0.056 = 0.474$$

$$Y = 0.2126(0.4706) + 0.7152(0.6275) + 0.0722(0.3137)$$

$$Y = 0.100 + 0.449 + 0.022 = 0.571$$

$$Z = 0.0193(0.4706) + 0.1192(0.6275) + 0.9505(0.3137)$$

$$Z = 0.009 + 0.075 + 0.298 = 0.382$$

Step 3: Normalize with reference white (D65)

$$X_r = \frac{X}{0.9505} = 0.498$$

$$Y_r = \frac{Y}{1.000} = 0.571$$

$$Z_r = \frac{Z}{1.089} = 0.351$$

Step 4: Convert to LAB components Using transformation function:

$$f(t) = t^{1/3}$$

Compute:

$$f(X_r) = 0.793, \quad f(Y_r) = 0.828, \quad f(Z_r) = 0.704$$

Now compute LAB:

$$L^* = 116f(Y_r) - 16 = 116(0.828) - 16 = 80.05$$

$$a^* = 500[f(X_r) - f(Y_r)] = 500(0.793 - 0.828) = -17.5$$

$$b^* = 200[f(Y_r) - f(Z_r)] = 200(0.828 - 0.704) = 24.8$$

Final LAB pixel:

$$L^* = 80.05, \quad a^* = -17.5, \quad b^* = 24.8$$

The LAB color space conversion is performed to separate the brightness information from the color components of the leaf image. Unlike the RGB space, where color and intensity are mixed, the LAB space consists of three independent channels: lightness L^*, green-red component a^*, and blue-yellow component b^*. The conversion begins by normalizing the RGB values and transforming them into the XYZ color space using a linear transformation such as $X = 0.4124R + 0.3576G + 0.1805B$ .The resulting XYZ values are then normalized using a reference white point, for example $X_r = X/X_{n'}$ to ensure color consistency. The LAB components are computed using nonlinear transformations, where the lightness is calculated as $L^* = 116f(Y_r) - 16$,which represents perceived brightness. The chromatic components are computed as $a^* = 500[f(X_r) - f(Y_r)]$ and $b^* = 200[f(Y_r) - f(Z_r)]$, representing color variations along red-green and blue-yellow axes. This is due to the fact that this separation enables the system to use contrast enhancement in the lightness channel without corrupting color information. Consequently, spots of disease become more evident and the natural colors of the leaves are not lost. LAB space would thus be quite appropriate in plant disease detection regarding preprocessing based on the symptoms.

3.1.3 CLAHE Contrast Enhancement

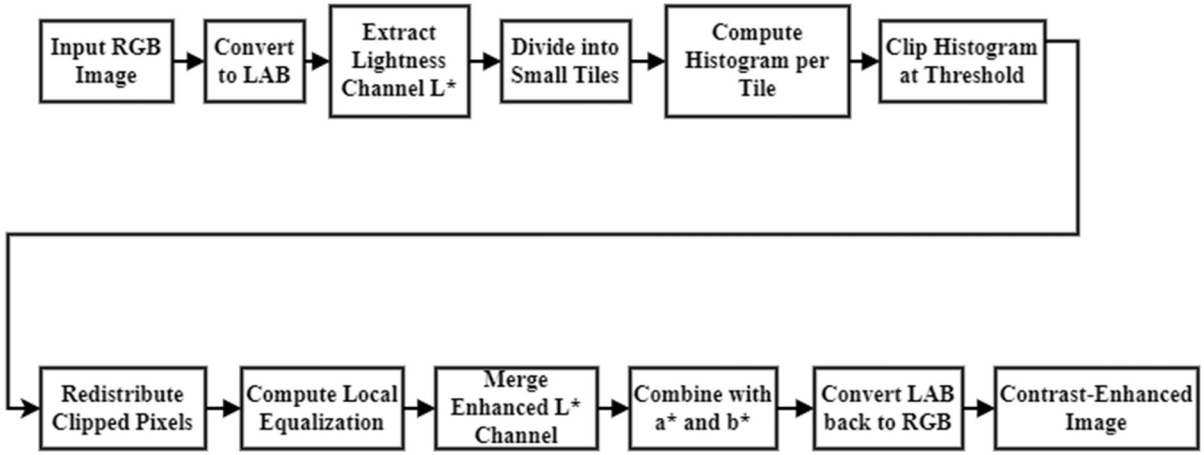

Figure 6: CLAHE Applied to Lightness Channel

Contrast Limited Adaptive Histogram Equalization (CLAHE) is used to modify lightness channel of the LAB image to bring out disease symptoms. This starts by subdivision of the channel of lightness into small nonoverlapping tiles. A local histogram of the intensity is also calculated on each tile. Alternative to global histogram equalization, CLAHE uses local enhancement with a clipping limit in order to avoid the exaggeration of noise. The clipping process ensures that the histogram values do not exceed a predefined threshold, which can be expressed as $H_c(i) = min(H(i), T)$,where $H(i)$ is the original histogram and$T$ is the clip limit. The clipped values are redistributed uniformly across the histogram bins to maintain total intensity, calculated as $H_r(i) = H_c(i) + \frac{\sum(H(i)-H_c(i))}{N}$. After redistribution, the cumulative distribution function is used to remap pixel intensities, defined as $I'(x,y) = (L_{max} - L_{min}) \cdot CDF\big(I(x,y)\big) + L_{min}$. This step is used to make local contrast among the tiles. The channel of increased light is then added back to the a 2- and b 2 -channels. Lastly, LAB image will be converted into RGB image. This selective boosting enables the disease regions to be seen more clearly whilst maintaining the colour of natural leaves and as well as avoiding potential amplification of noise.

3.2 Handcrafted Feature Extraction

3.2.1 Gradient Features (Sobel)

Pseudocode: Sobel Gradient Feature Extraction

Input:
Preprocessed grayscale image $I(x,y)$
Output:
Gradient magnitude image $G(x,y)$
Start

1. Read the grayscale image $I(x,y)$.
2. Define Sobel kernels: Horizontal kernel:
$S_x = [-1\ 0\ 1\ \ -2\ 0\ 2\ \ -1\ 0\ 1\ ]$
Vertical kernel:
$S_y = [-1\ \ -2\ \ -1\ 0\ 0\ 0\ 1\ 2\ 1\ ]$
3. For each pixel (x,y) in the image: • Extract a 3×3 neighborhood.
4. Compute horizontal gradient:
$$G_x(x,y) = I * S_x$$
5. Compute vertical gradient:
$$G_y(x,y) = I * S_y$$
6. Compute gradient magnitude:
$$G(x,y) = \sqrt{G_x^2 + G_y^2}$$
7. Normalize gradient values.
8. Store result as gradient feature map.

End

Gradient features are extracted using the Sobel operator to capture intensity variations that correspond to and vertical directions, allowing detection of abrupt intensity changes. For a given image $I(x,y)$, the horizontal gradient is calculated using convolution with the kernel $G_x = I * S_x$, while the vertical gradient is computed as $G_y = I * S_y$. These gradients represent the rate of intensity change along their respective directions. The overall edge strength at each pixel is then obtained using the gradient magnitude formula $G(x,y) = \sqrt{G_x^2 + G_y^2}$which combines both directional responses.In some cases, the gradient orientation is also computed using $\theta(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right)$, providing directional information about lesion boundaries. The The Sobel operator lays more importance on edges and less on noise as it has a weighted kerne structure that eliminates small noise.

This is what makes it appropriate in identifying disease foci, which show sharp variations in intensity. The gradient map that is generated emphasizes the areas that are infected over the generally healthy ones. Such gradient attributes are valuable features to the hybrid disease detection pipeline.

3.2.2. Edge Detection using Canny

Numerical Example: Canny Edge Detection

Input: A 3 × 3 grayscale patch:

$$I = [52\ 55\ 61\ 68\ 70\ 75\ 80\ 85\ 90\ ]$$

Step 1: Gradient Calculation Assume Sobel gradients:

$$G_x = 30, \quad G_y = 40$$

Gradient magnitude:

$$G = \sqrt{30^2 + 40^2} = \sqrt{900 + 1600} = \sqrt{2500} = 50$$

Gradient direction:

$$\theta = tan^{-1}\left(\frac{40}{30}\right) = 53.13°$$

Step 2: Non-Maximum Suppression Assume neighboring magnitudes:

$$G_{left} = 35, \quad G_{right} = 20$$

Since:

$$50 > 35\ and\ 50 > 20$$

Pixel is kept as an edge.

Step 3: Double Thresholding Assume:

$$T_{low} = 20, \quad T_{high} = 45$$

Since:

$$50 > T_{high}$$

Pixel is marked as a strong edge.

Step 4: Edge Tracking by Hysteresis Strong edges are preserved. Weak edges connected to strong edges are retained.

Final edge pixel value:

$$E(x,y) = \{1, G(x,y) \geq T_{high}\ 1, T_{low} \leq G(x,y) < T_{high}\ and\ connected\ 0, otherwise$$

Here:

$$E(x,y) = 1$$

Canny edge detection is used to identify precise lesion boundaries in the leaf image. The method begins by computing image gradients using operators such as Sobel, where the gradient magnitude is calculated as $G = \sqrt{G_x^2 + G_y^2}$to represent edge strength.The direction of the gradient is determined using $\theta = tan^{-1}(G_y/G_x)$,which helps in aligning edges correctly. Gradient computation is followed by non- maximum suppression to eliminate weak or non-edge pixels comparing each pixel of the gradient direction with its neighboring pixels. This algorithm is followed by the use of the double thresholding to classify edges as strong or weak, using thresholds T_{low} and T_{high}. This process can be expressed as $E(x,y) = 1$ if $G(x,y) \geq T_{high}$, indicating a strong edge. Weak edges are preserved only if they are connected to strong edges, following the hysteresis rule $E(x,y) = 1$ when $T_{low} \leq G(x,y) < T_{high}$ and connectivity is satisfied. It is a multi-stage algorithm that allows lesion boundaries to be properly identified in addition to noise suppression. Consequently, Canny detector generates fine and sharp edges which spread disease areas in a more accurate form. These boundaries are valuable structural features to the hybrid detection system.

3.2.3 Texture Features

a) Local Binary Patterns (LBP)

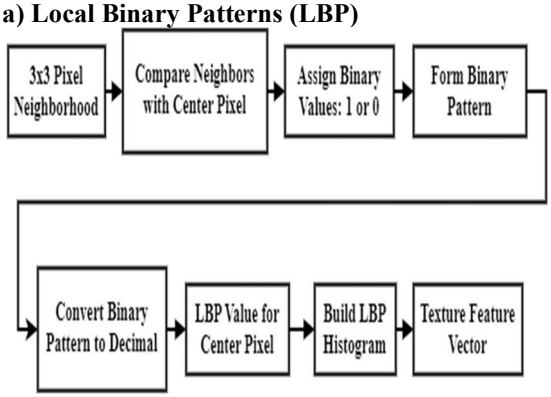

Figure 7: Local Binary Pattern Encoding

The Local Binary Patterns are made to capture small scale texture details that are related to pest damage and disease spots. The technique works on a small window of the neighborhood of each pixel which is usually 3 by 3, and the central pixel is contrasted with the neighbors of the same pixel. For each neighboring pixel, a binary value is assigned based on the condition $b_i = 1$ if $I_i \geq I_c$ and $b_i = 0$ otherwise, where I_c is the center pixel intensity. These binary values are arranged in a circular order to form a binary pattern. The LBP value is then computed using the weighted sum formula $LBP = \sum_{i=0}^{P-1} b_i \cdot 2^i$, where P represents the number o neighbors. This converts the binary pattern into a decimal value that describes local texture. The distribution of these values across the image is captured using a histogram defined as $H(k) = \sum_{x,y} \delta(LBP(x,y) - k)$,where k represents the bin index. This is the texture feature vector represented by this histogram. LBP is specifically useful in the identification of small lesions, insect marks and disease patterns of small lesions before the disease progresses. To changes in illumination, it is simple to compute and resistant to change. Consequently, it offers useful fine-text features to the hybrid disease detection system.

b) Histogram of Oriented Gradients (HOG)

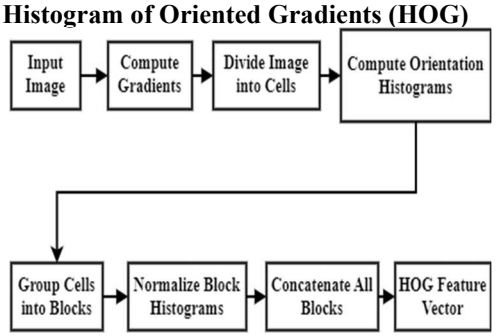

Figure 8: HOG Feature Extraction Process

Directional texture information of leaf images is captured using Histogram of Oriented Gradients which starts by calculating intensity gradients in both horizontal and vertical directions using operators like Sobel. The gradient magnitude at each pixel is calculated as $G = \sqrt{G_x^2 + G_y^2}$, representing the strength of intensity change. The gradient orientation is computed using $\theta = tan^{-1}\left(\frac{G_y}{G_x}\right)$, which indicates the direction of the edge or texture. This step is followed by partitioning the image into small cells and each cell consisting of the histogram of gradient orientations of weights proportional to gradient magnitudes. These cell histograms are stacked to bigger blocks to be normalized in order to limit illumination effects. The normalized histogram for each block is computed using $H' = \frac{H}{\sqrt{\|H\|^2 + \epsilon^2}}$, where H is the original histogram and ϵ is a small constant for stability. Normally, all the block histograms are concatenated to obtain the final HOG feature vector. This aspect is used to represent both structural and directional patterns relating to disease lesions. HOG comes in handy especially in identifying long, or organized patterns of diseases. It is a complement of LBP that gives orientation-based texture information. A combination of these features contributes to the strength of the hybrid system of detection.

3.2.4 Shape Features

Shape features are derived to measure the geometric properties of the lesions of the disease that are on the leaf surface. The contours are then detected after segmentation and edge recognition in order to describe the borders of the infected areas. Every contour is associated with a related collection of edge pixels which form the shape of a lesion. The area of a detected lesion is computed using the contour integration formula $A = \sum_{i=1}^{N} (x_i y_{i+1} - x_{i+1} y_i)/2$, where (x_i, y_i) represent contour points. The perimeter of the lesion is calculated by summing the distances between consecutive contour points, expressed as $P = \sum_{i=1}^{N} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$. These geometric properties help distinguish between different disease types that produce distinct lesion shapes.In addition, shape compactness is computed using $C = \frac{4\pi A}{P^2}$,which measures how closely the lesion resembles a circular form. Irregular lesions typically have lower compactness values, indicating disease severity or spread. These descriptors of shape give structural data that supplements the texture and color data. The system is able to measure the size and shape in lesions by measuring area of contours and perimeter. This enhances the standard of disease classification. Shape attributes are then important in the hybrid stage of feature extraction.

3.2.5 Color Features

The color properties are harvested to attain differences in leaf pigmentation due to the disease infections. Various plant diseases will tend to create common discolorations of yellowing, browning, or dark spots, which may be measured using statistical analysis of the colour channel. In the RGB color space, the mean intensity of each channel is computed as $\mu_c = \frac{1}{N}\sum_{i=1}^{N} I_c(i)$,where $c \in \{R, G, B\}$ and N is the number of pixels The standard deviation is also calculated using $\sigma_c = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (I_c(i) - \mu_c)^2}$ to measure color variation within the leaf region. These statistics help differentiate healthy and diseased tissues. The image is also converted into HSV color space to better represent perceptual color properties. The hue component, which indicates dominant color, is computed as $H = arctan\left(\frac{\sqrt{3}(G-B)}{2R-G-B}\right)$. Such transformation isolates colo information across brightness hence is more resistant to lighting variations. The intensity and brightness of the color is also rendered by saturation and value components. The system is able to combine RGB statistics and HSV descriptors to have a complete color representation. These color properties are needed to determine the disease peculiar pattern of discoloration.

3.3. Custom CNN Architecture

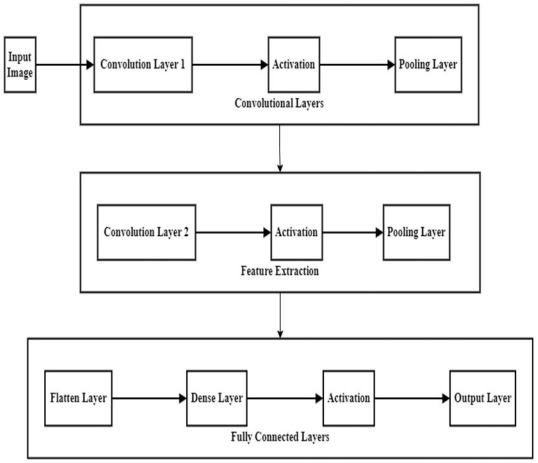

Figure 9: Custom CNN Architecture for Plant Disease Classification

The customer convolutional neural network will be made to extract disease-specific patterns among the improved leaf images. The network can take in input images with dimensions of 224x224x3 which are the height, width and RGB of the image. The first convolutional layer applies multiple filters to the input image, producing feature maps according to the convolution operation $F_{i,j,k} = \sum_{m,n,c} I_{i+m,j+n,c} \cdot W_{m,n,c,k} + b_k$,where W represents the filter weights. These feature maps are passed through activation functions to introduce nonlinearity. Max-pooling layers are used after convolution to reduce spatial dimensions, computed as $P_{i,j} = max_{(m,n)\in\Omega} F_{i+m,j+n,"}$ $where\ \Omega$ represents the pooling window. Learned hierarchical features, starting with simple edges but all the way to complex lesions, is made possible by multiple convolution-pooling blocks. The final feature maps are reduced to a one dimensional vector before being sent to a fully connected layer. The output layer uses a softmax function to produce class probabilities, defined as $p_k = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}'$ where K is the number of disease classes. The last prediction is the highest probability of classes. This structure allows automatic learning of the characteristics of disease with no human involvement. The design compromises the accuracy and computational performance to be run in practice.

3.3.1 Custom Activation Function for CNN

The proposed CNN uses a custom activation function that combines the nonlinear properties of the hyperbolic tangent function with the sparsity benefits of the ReLU function. The activation first applies the hyperbolic tangent transformation, expressed as $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$,which maps input values into the range [-1,1]. This helps in capturing subtle variations in disease patterns and improves gradient flow during training. The output of the tanh function is then passed through a rectified linear unit, defined as $f(x) = max(0, x)$,which suppresses negative responses and retains only positive activations. The combined activation function is therefore expressed as $f(x) = max\big(0, tanh(x)\big)$. This hybrid solution is to be sure that small yet significant elements are maintained and noises and useless signals are minimized. The tanh stage will have smooth gradients around zero as compared to normal ReLU. The ReLU stage does not saturate negative areas as compared to pure tanh. This leads to a better feature discrimination and convergence. The custom activation function is implemented following every convolutional layer of the CNN It is used to improve the potential of the network to find small disease signs in leaf images.

3.3.2 YOLO-Based Detection Model

Pseudocode: YOLO-Based Disease Detection

Input:

- Preprocessed leaf image I
- Trained YOLO model parameters

Output:

- Bounding boxes around diseased regions
- Disease class predictions

Start

1. Read input image I.
2. Resize image to YOLO input size.
3. Divide image into an $S \times S$ grid.
4. For each grid cell:

Predict bounding box coordinates:

$$b_x, b_y, b_w, b_h$$

Predict object confidence score:

$$C = P(object) \times IoU$$

5. For each predicted box: - Compute Intersection over Union:

$$IoU = \frac{Area\big(B_{pred} \cap B_{gt}\big)}{Area\big(B_{pred} \cup B_{gt}\big)}$$

6. Compute class probabilities:

$$P(class_i|object)$$

7. Compute final class score: $S_i = C \times P(class_i|object)$
8. Apply threshold to remove low-confidence boxes.
9. Perform Non-Maximum Suppression (NMS) to remove overlapping boxes.
10. Output final bounding boxes and disease labels. End

The detection model based on the YOLO is applied to localize the disease areas in the image of the leaf. An object detector unlike the traditional classification model uses a single forward pass to predict bounding boxes and class probabilities. The input image is broken down into a $S \times S$ grid, with each cell in the grid charged with predicting objects with centres falling into the area represented by these cells. The model predicts bounding box coordinates and scores of confidence using each grid cell. The confidence score is calculated as $C = P(object) \times IoU$, where IoU measures the overlap between predicted and ground truth boxes. The intersection over union is defined as $IoU = \frac{Area(B_{pred} \cap B_{gt})}{Area(B_{pred} \cup B_{gt})}$, which quantifies localization accuracy. Each bounding box also produces class probabilities, and the final score for a class is computed as $S_i = C \times P(class_i|object)$. A threshold is used to eliminate boxes with low confidence. It is followed by non-maximum suppression in order to remove unnecessary overlapping boxes. This is done to make sure that only the most correct detections prevail. YOLO model facilitates the online detection of diseased areas. It is not only classifying and localizing, but it is also appropriate in the field.

3.3.3 Custom Activation Function for YOLO The YOLO model integrates a custom activation function that is aimed at promoting the detectiveness of subtle disease patterns. This operation is a combination of nonlinear transformation of hyperbolic tangent and the rectification property of the function of ReLU. The first stage applies the tanh function, defined as $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$,which compresses the input values into the range [-1,1]. This helps stabilize gradients and capture fine variations in feature responses. The output is then passed through a rectified linear transformation expressed as $f(x) = max(0, x)$, which removes negative activations that may represent noise or irrelevant patterns. The combined activation can be written as $f(x) = max(0, tanh(x))$,forming a hybrid nonlinear function. This design improves feature selectivity in the detection layers. The derivative of the tanh function, given by $\frac{d}{dx} tanh(x) = 1 - tanh^2(x)$, provides smooth gradients during training. This helps the network converge more effectively. Compared to standard activations, the hybrid function emphasizes meaningful features while suppressing noise. As a result, it enhances the YOLO model's ability to localize disease regions accurately.

3.3.4 Transfer Learning with NASNetMobile

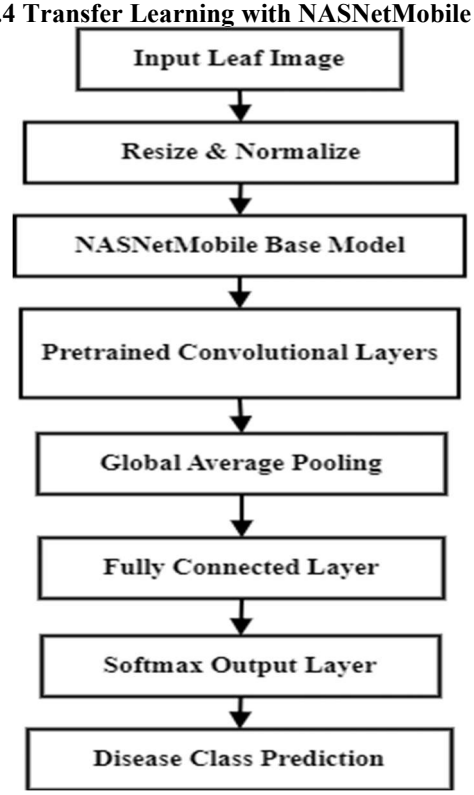

Figure 10: NASNetMobile Transfer Learning Pipeline

The transfer learning using NASNetMobile is used to use the knowledge acquired by the large scale pictures and enhance the performance of classifying the diseases. NASNetMobile represents a model that is an efficient and high-accuracy neural architecture search. The model is exposed to a large dataset like ImageNet to achieve pretrained weights used in the initiation of the model. As part of training, the input leaf image is resized and normalized and then it is fed into the initial convolutional layers. These layers extract deep hierarchical features using convolution operations such as $F_{i,j,k} = \sum I_{i+m,j+n,c} \cdot W_{m,n,c,k}$ where W represents learned filters. Instead of training the entire network from scratch, the pretrained layers are either frozen or partially fine-tuned. Global average pooling is applied to reduce feature maps into a compact representation, computed as $G_k = \frac{1}{N} \sum_{i=1}^{N} F_{i,k}$. The pooled features are then passed to a fully connected layer for classification. The final class probabilities are generated using the softmax function $p_k = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}$. Fine-tuning adjusts the higher layers of the network to adapt to plant disease patterns. The method saves on training time and enhances generalization. Subsequently, NASNetMobile as a potent base

comparison with the tailored CNN and YOLO models can be performed.

3.3.5 Hybrid Feature Integration

The system proposed combines the handcrafted features and the features of the deep learning representations to form a complete system of symptom-focused detection. Several human-created features including the gradient, edge, texture, shape and color descriptors are then obtained with the leaf image following the preprocessing stage. These features are combined into a single feature vector defined as $F_h = [F_g, F_e, F_t, F_s, F_c]$, where each component represents a specific type of handcrafted descriptor. Simultaneously, the deep learning models extract high-level representations from the enhanced images, expressed as $F_d = f_{CNN}(I_p)$, where I_p is the preprocessed image. The hybrid feature representation is formed by concatenating both feature sets using $F_{hybrid} = [F_h \oplus F_d]$, where $\oplus$ denotes feature concatenation. The joint representation enables the system to represent both the domain-specific as well as auto-learned features. The reason is that the handcrafted details give priority to symptoms associated features, including the shape and texture of lesions. The deep features give abstract features and multimodal visual relationships. This assimilation enhances the strength of the detection process. It also increases performance in both different lighting and environmental conditions. The hybrid model makes it sure that fine-grained and high-level disease features are brought into use to make a correct classification.

3.3.6 Model Training Procedure

The model training process is geared towards the performance optimization of the hybrid detection system under the supervised learning. As a measure of appropriate model evaluation, a dataset is initially split into training, validation and testing sets. In the course of training, the models are trained to recognize the map between the input features to the disease classes using a loss function. The optimization process updates model parameters using gradient descent, expressed as $\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$,where

$\eta is\ the\ learning\ rate. The\ cross - entropy\ loss\ function\ is\ used\ for_{rr}$ classification, defined as $L = -\sum_k \square = 1^{\Lambda} y_k log(p_k)$, where y_k is the true label and p_k is the predictec probability. Model performance is measured using accuracy, calculated as $Acc = \frac{TP+T}{TP+TN+FP+F}$. A learning rate is adjusted optimistically by the optimizer (Adam) to finish quicker. The training process requires several epochs which are complete passes through the training data. The batch size is used to decide how many samples are to go before model weights are updated. Validation information is the one that is used to monitor performance and eliminate overfitting. Figure shows evaluation of the final model on the test set, which is performed after training. The process will guarantee good and consistent refinement of models to various datasets.

4. RESULTS & DISCUSSIONS

Table 2: Dataset Distribution

Dataset Split	Number of Batches	Approx. Images	Purpose
Training	219	~7000	Model learning
Validation	31	~1000	Performance monitoring
Testing	63	~2000	Final evaluation
Total	313	~10,000	Complete dataset

The dataset to be used in the given research paper will be comprised of tomato leaf images that are classified as ten diseases and healthy leaves. One thousand and five hundred images (10,000) of 10 different classes were loaded that was used to make a varied model to learn. The images were downsampled to 256 x 256 pixels and were trained in 32 image batches. The entire data before splitting comprised 313 batches. The dataset was mixed up and divided into training, validation and testing parts. The final denomination gave 219 batches to train, 31 batches to validate and 63 batches to test. This distribution was adequate to make sure that the model had adequate data to learn but still with independent validation and testing sets. Testing set was maintained under an absolute disguise during the training process in order to present a pure estimation of the model performance.

```
Epoch 15/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 261s 1s/step - accuracy: 0.8680 - loss: 0.3634 - val_accuracy: 0.8065 - val_loss: 0.5318
Epoch 16/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 246s 1s/step - accuracy: 0.8836 - loss: 0.3318 - val_accuracy: 0.8165 - val_loss: 0.5347
Epoch 17/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 246s 1s/step - accuracy: 0.8812 - loss: 0.3471 - val_accuracy: 0.8105 - val_loss: 0.6299
Epoch 18/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 263s 1s/step - accuracy: 0.8914 - loss: 0.3047 - val_accuracy: 0.7591 - val_loss: 0.7480
Epoch 19/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 247s 1s/step - accuracy: 0.8952 - loss: 0.3068 - val_accuracy: 0.8004 - val_loss: 0.6644
Epoch 20/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 247s 1s/step - accuracy: 0.9072 - loss: 0.2608 - val_accuracy: 0.7913 - val_loss: 0.6486
Epoch 21/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 262s 1s/step - accuracy: 0.9062 - loss: 0.2625 - val_accuracy: 0.8962 - val_loss: 0.2906
Epoch 22/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 247s 1s/step - accuracy: 0.9134 - loss: 0.2482 - val_accuracy: 0.8901 - val_loss: 0.2924
Epoch 23/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 255s 1s/step - accuracy: 0.9158 - loss: 0.2470 - val_accuracy: 0.9032 - val_loss: 0.2687
Epoch 24/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 294s 1s/step - accuracy: 0.9194 - loss: 0.2317 - val_accuracy: 0.8921 - val_loss: 0.2651
Epoch 25/25
219/219 ━━━━━━━━━━━━━━━━━━━━ 296s 1s/step - accuracy: 0.9272 - loss: 0.1958 - val_accuracy: 0.7833 - val_loss: 0.8956
```

Figure 11a: Training Epochs

```
final_train_accuracy = history.history['accuracy'][-1]
final_val_accuracy = history.history['val_accuracy'][-1]

print(f"Final Training Accuracy: {final_train_accuracy:.4f}")
print(f"Final Validation Accuracy: {final_val_accuracy:.4f}")
```

```
Final Training Accuracy: 0.9302
Final Validation Accuracy: 0.7833
```

Figure 11b: Final Training and Validation Accuracy

The training was set up as 25 epochs with the training dataset and validation performance was checked by the end of every epoch. The initial model training was 17.83% with a loss of 2.17, which was a result of random learning. During training, accuracy increased over time and was 70.48 and 84.56 by epoch 5 and 11 respectively. The validation accuracy increased as well with increasing epochs where in epoch 1 it was at 22.68% and later it grew on reaching epoch 16 where it was above 80%. The training loss reduced steadily up to 0.19 at the last epoch. Validation accuracy, however, exhibited slight variation because of similarities and classes of diseases. The last training accuracy was 93.02, and the last validation one was 78.33.

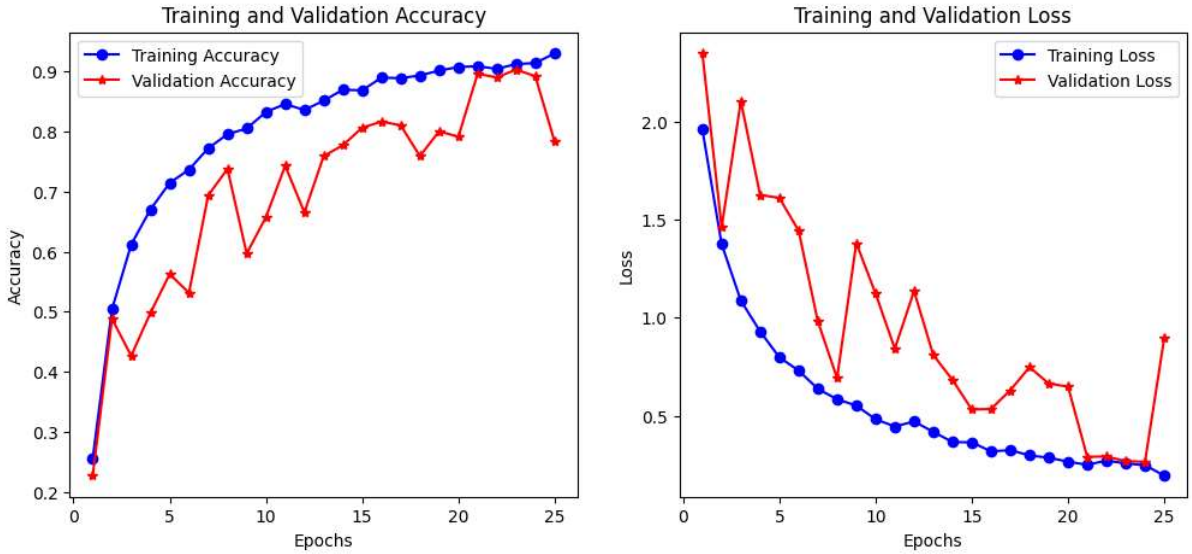

Figure 12: Training and Validation Accuracy

The trained model was tested on the testing dataset of 63 batches that were not involved in training and validation. The test accuracy and test loss obtained after the evaluation process was 78.12% and 0.89 respectively. The last training accuracy was 93.02, and the validation accuracy was 78.33 that is close to the test results. This resemblance shows that the model had a constant generalization of unobserved data. The fact that training and validation accuracy are at variance of about 15% indicate that there is a moderate overfitting but acceptable performance. The test loss rate is quite low, which proves that the predictions were quite sure. These findings have shown that the suggested preprocessing model and hybrid features model generated a well-functioning disease model. In general, the testing stage will ensure the suitability of the system in reality.

Table 3: Final Model Performance

Metric	Value
Final Training Accuracy	93.02%
Final Validation Accuracy	78.33%
Test Accuracy	78.12%
Test Loss	0.89
Training Epochs	25
Batch Size	32

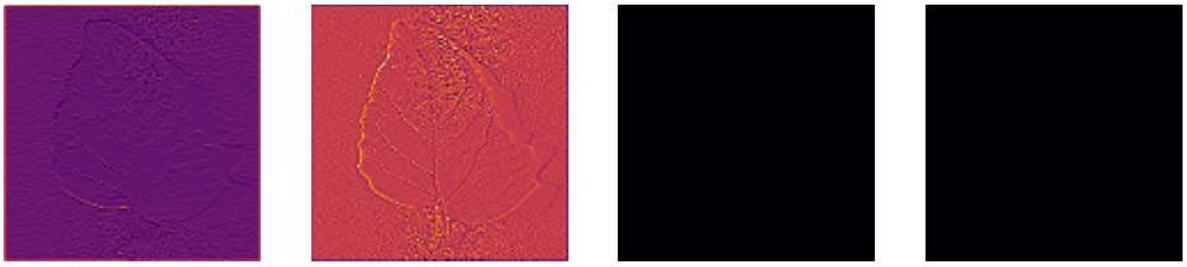

Figure 13: Feature Map

The model was analyzed with regards to how it identifies disease patterns of the leaf images based on feature visualization techniques. Sobel operator was used on 256 x 256 grayscale images to create edge maps which indicated the boundaries of the lesions. The intensity of the gradient was brought to 0- 255 to enhance the visualization of the disease areas. The structures of the resulting edge maps represented the apparent differences in the healthy and the infected regions. The analysis of CNN feature maps of the intermediate convolution layers provided the pattern of learning patterns of several channels. The various convolution outputs were also shown concurrently in the feature visualization experiment so as to see the relationship between these responses and the texture. Simple edges were represented by early layers and complex lesion shapes by deeper layers. These visualizations affirmed that the network had learned informative features of diseases. Its findings prove that the hybridization of handcrafted edge descriptors and deep learning representations is effective.

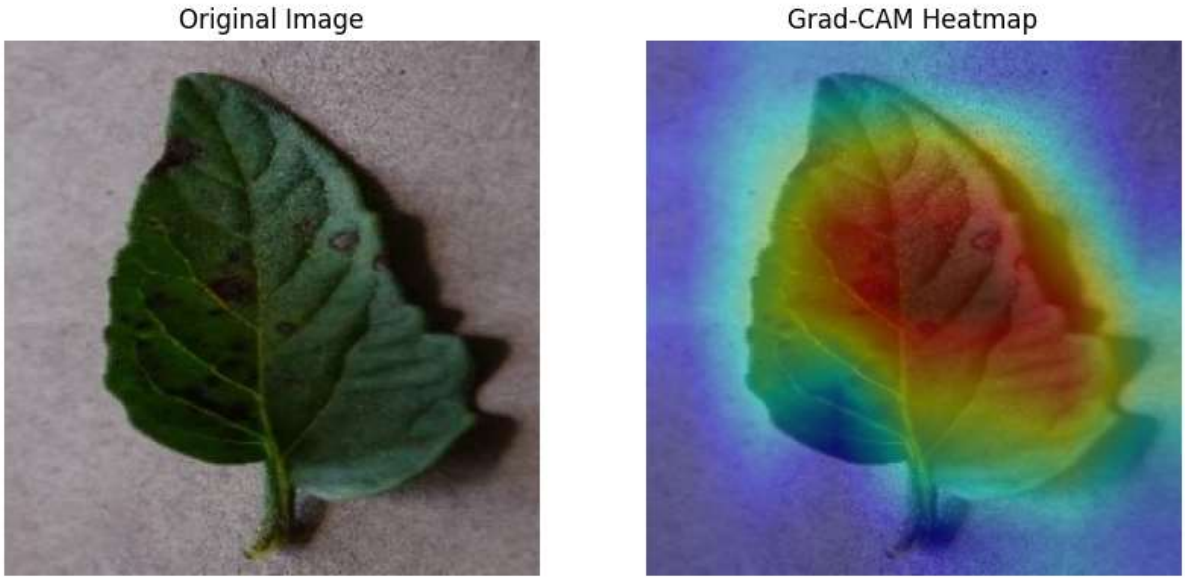

Figure 14: Original vs Grad-CaM Heatmap

The process of decision-making by the model was interpreted by grad-cam visualization. The process created heatmaps based on a trained net with a 224 224 Pixels input. The most probable prediction value of the example image was found to be 0.995 which implies that the prediction was highly confident. The Grad-CAM heatmap demonstrated the area of the leaf that had a lesion and therefore the model was concentrated on the infected parts. Normalization of the heatmap intensity was carried out and the result was transformed into a color map that would be used to visualize the result. The result obtained was the oberlay of the original image and the heatmap which was a weighted merge. This brought an illustrative explanation of the areas of attention of the model. The spots that were highlighted were similar to the spots of the disease that could be seen on the leaf. These findings affirm that the model had learnt to target useful parts of the symptoms instead of the background characteristics.

4. DISCUSSION

Evaluation on a variety of and representative datasets should support the proposed claims, since plant disease detection models can have different performances in the field as compared to curated laboratory conditions. Previous research indicates the significance of non-lab datasets like PlantDoc and in-the-wild segmentation datasets as they contain lighting variations, occlusion and background complexities that more closely mimic real agricultural settings. Furthermore, large multi-class datasets for many crops can be used to confirm the efficacy of a method for various crops and disease classes. So the current work is to be taken as a promising step, but the assertion of the work is more convincing in the light of a wider cross-dataset and cross-condition validation.

5. CONCLUSION

The framework proposed in this work tackles the first objective of improving plant disease detection by employing symptom-based pre-processing and feature fusion but the results have to be benchmarked with the latest state-of-the-art approach. In the existing literature, the performance of advanced CNN, transfer learning and hybrid deep learning approach are reported to be very high especially on the controlled datasets. Recent works further enhance the robustness with transformer-based architectures, attention mechanism, and hybrid feature-fusion approaches. The primary contribution of the present work is the disease-symptom enhancement with hybrid classification; however, further validation is still needed on a variety of real-world datasets.

In this research, a hybrid plant disease detection system based on the combination of state-of-the-art preprocessing methods, handcrafted features of the plant specimen, and deep learning was developed. Preprocessing phase was an integration of bilateral filtering with the LAB-CLAHE contrast enhancement to minimize noise, and bring out symptoms of the disease at the same time. Several

handcrafted characteristics such as gradient, edge, texture, shape, and color characteristics were voted out to extract disease-specific characteristics. The following were combined with the deep learning models like a custom CNN, YOLO, and NASNetMobile to produce a powerful detection pipeline. The CNN trained became accustomed to 10,000 tomato leaf images (with ten categories) to perform 25 epochs. The model resulted in a final training accuracy of 93.02, validation accuracy of 78.33, and a test accuracy of 78.12 which proves useful generalization and learning. Visualization of features and Grad-CAM heatmaps proved that the model paid attention to the lesion areas during prediction. Even though a bit of confusion was noted among visual similar diseases, on the whole, the performance was high in most classes. The hybrid method effectively integrated domain knowledge and deep learning representations and enhanced their robustness and detection. Future directions, however, are to have larger datasets, enhanced augmentation policy, and real-time mobile or field-of-deployment.

In this work, the leaf-based plant disease detection using the preprocessing, handcrafted features and deep learning classification is covered. No details of root or stem diseases, severity rating, treatment recommendations or deployment in actual field systems. It can only be used for image diagnosis and the data set evaluated.

REFERENCES

[1] Khan, S. S., Jim, S. S. H., Rahman, M. M., Nafis, A. A. S., Abdullah, N., & Aisharjo, F. R. (2025). Deep Neural Networks for Identifying Leaf Diseases in Potato and Eggplant Crops.

[2] Abisha, S., Mutawa, A. M., Murugappan, M., & Krishnan, S. (2023). Brinjal leaf diseases detection based on discrete Shearlet transform and Deep Convolutional Neural Network. PLoS ONE, 18(4 April). https://doi.org/10.1371/journal.pone.0284021

[3] Hettiarachchi, W. S., & Herath, H. M. D. S. (2025, August). AI-based Chilli Leaf Disease Detection and Remedy Recommendation System. In 2025 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET) (pp. 449-454). IEEE.

[4] Yaseen, Z., Lahore, U., Zarka Saeed, P., Asif, U., Khalid Masood, P., & Sajjad, M. (2025). Real-Time Image-Based Monitoring Of Chilli Plant Maturation Using Object Detection Techniques. 1.

[5] Nawaz, M., Javed, A., & Saudagar, A. K. J. (2026). PotatoGuardNet: a refined deep learning framework for potato leaf disease detection. Frontiers in Plant Science, 17, 1720276.

[6] Alhammad, S. M., Khafaga, D. S., El-Hady, W. M., Samy, F. M., & Hosny, K. M. (2025). Deep learning and explainable AI for classification of potato leaf diseases. Frontiers in Artificial Intelligence, 7, 1449329.

[7] Shahzad, M., Javed, M. A., Ashraf, E., Ahmad, H. I., & Masood, S. (2026). A deep learning model for accurate tomato leaf disease identification. Turkish Journal of Electrical Engineering and Computer Sciences, 34(1), 84-101.

[8] Swamy, L. N., Parande, P. v., Kumar, H. P. M., Rakshitha, V., Thimmaraj, S. N., & Yogesha, T. (2026). Convolutional Neural Network Architectures for Enhanced Tomato Leaf Disease Classification using ResNet-152 and VGGNet Model. Indian Journal Of Agricultural Research, Of. https://doi.org/10.18805/ijare.a-6375

[9] Padmanabhuni, S., & Gera, P. (2025). Bayesian optimized deep learning and ensemble classification approach for multiclass plant disease identification. Discover Sustainability, 6(1). https://doi.org/10.1007/s43621-025-01648-1

[10] Kaur, A., Randhawa, G. S., Farooque, A. A., Ali, M., Singh, H., Al-Mughrabi, K., Bell, D., & Singh, R. (2026). Crop disease surveillance through integration of machine and deep learning in the face of climate change. Journal of Agriculture and Food Research, 26, 102733. https://doi.org/10.1016/j.jafr.2026.102733

[11] Talaat, F.M., Ibrahim, M.A., Karim, A.A. et al. IoT-Integrated robotic system for automated plant disease detection and environmental monitoring. Sci Rep 16, 1638 (2026). https://doi.org/10.1038/s41598-025-32624-4

[12] Mallick, C., Patra, A., Dash, S., Mishra, P. K., & Paikaray, B. K. (2026). Leveraging deep learning for strategic decision-making in sustainable agriculture: enhancing plant disease detection for optimised supply chain management and ecosystem health. International Journal of Applied Management Science, 18(1), 90–110. https://doi.org/10.1504/ijams.2026.151290

[13] Rahaman, J., Paul, P., Chowdhury, A. et al. A customized MobileNetV3Large-based deep learning framework for plant disease detection. Discov Artif Intell 6, 110 (2026). https://doi.org/10.1007/s44163-025-00733-8

[14] Shahzad, M., Javed, M. A., Ashraf, E., Ahmad, H. I., & Masood, S. (2026). A deep learning model for accurate tomato leaf disease identification. Turkish Journal of Electrical Engineering and Computer Sciences, 34(1), 84–101. https://doi.org/10.55730/1300-0632.4164

[15] Deepa, R., Mathew, M. P., Baskar, S., & M, A. K. (2026). L-Net: a lightweight CNN framework for sustainable multicrop leaf disease detection and classification on edge devices. Sustainable Food Technology, 4(1), 985–1003. https://doi.org/10.1039/d5fb00191a

[16] Catal Reis, H. CropHealthNet: Potato Disease Detection Using Depthwise Separable Convolutional Neural Networks. Journal of Crop Health 78, 29 (2026). https://doi.org/10.1007/s10343-026-01294-1

[17] Prashanth, J. S., Krishna, G. B., Prasad, A. V., & Rao, P. R. (2025, March). Smart Farming Revolution: A Cutting-Edge Review of Deep Learning and IoT Innovations in Agriculture. In Operations Research Forum (Vol. 6, No. 1, pp. 1-39). Springer International Publishing

[18] Prashanth, J. Siva, Nageswara Rao Moparthi, G. Bala Krishna, "MPCSAR-AHH: A hybrid deep learning model for real-time detection of cassava leaf diseases and fertilizer recommendation" Computers and Electrical Engineering 119 (2024): 109628.

[19] Balakrishna, G., and Nageswara Rao Moparthi. "The automatic agricultural crop maintenance system using runway scheduling algorithm: fuzzyc-LR for IoT networks." Int. J. Adv. Comput. Sci. Appl. 11.11 (2020): 654-665.

[20] Balakrishna, G., et al. "Implementing Solar Power Smart Irrigation System." Innovations in Computer Science and Engineering: Proceedings of the Ninth ICICSE, 2021. Singapore: Springer Singapore, 2022. 561-567.

[21] Balakrishna, G., and Nageswara Rao Moparthi. "Study report on Indian agriculture with IoT." International Journal of Electrical and Computer Engineering 10.3 (2020): 2322.