IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 43, No. 1, February, 2026, pp. 491-504
Journal homepage: http://iieta.org/journals/ts

LeafInsightNet: A Deep Learning-Based Betel Leaf Disease Classification Model with Real-Time Robotic Detection

Dhilipkumar P.[1*], Duraichi N.[2], P. Tamilselvi[3], Muhammadu Sathik Raja[4]

[1] Department of Electronics and Communication Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore 641062, India
[2] Department of Electronics and Communication Engineering, Saveetha School of Engineering, Chennai 600072, India
[3] Department of Electronics and Communication Engineering, Sree Sakthi Engineering College, Coimbatore 641104, India
[4] Department of Biomedical Engineering, Dhanalakshimi Srinivasan University, Perambalur 621212, India

Corresponding Author Email: dhilipkumarp88@gmail.com

https://doi.org/10.18280/ts.430135

Received: 30 November 2025
Revised: 12 January 2026
Accepted: 15 February 2026
Available online: 28 February 2026

Keywords:
deep learning models, LeafInsightNet, dual-stage attention, betel leaves, real-time detection robot

ABSTRACT

The need for agricultural innovation arises from the growing demand for efficient methods to detect and manage crop diseases. The existing methods for leaf disease are time-consuming and labour-intensive. Recently, the arrival of artificial intelligence like machine learning (ML) and deep learning (DL)models considerably improved the agriculture field. It supports precise agriculture, real-time detection systems, and automated solutions. In this work, a new DL model called LeafInsightNet is proposed for disease classification in betel leaves. LeafInsightNet involves different multistage processes like a pre-trained ResNet-18 encoder, DynamicConvolutionBlock and Dual-Stage Attention mechanisms. This integration allowed for the extraction of relevant features from the leaf. Then, the metaheuristic algorithm of Starfish Optimization is used to select optimal features from extracted features. Finally, the parameter-tuned random forest (RF) classifier is applied for disease classification. Additionally, the system is integrated with a low-cost real-time detection robot. This robot is equipped with an ESP controller and Wi-Fi module which allows remote monitoring and disease detection. The robot's interface is designed with a Flask server for real-time interaction and control. The architecture and system are optimized for ease of use, cost-effectiveness, and real-time applicability. To further evaluate the model's practicality in energy-constrained and real-time environments, LeafInsightNet is tested on three hardware platforms: GPU (NVIDIA RTX 3060), ESP32 microcontroller, and Field-Programmable Gate Array (FPGA) (Xilinx Zynq-7000). The FPGA implementation achieves a good trade-off between inference speed, power consumption and accuracy. The implementation results indicate that LeafInsightNet is suitable for low-power and real-time agricultural robotic disease detection with scalable field deployment.

1. INTRODUCTION

Betel leaf (*Piper betel*) holds significant cultural, medicinal, and economic value, especially in countries across Asia where it is widely cultivated [1]. Betel leaves are used in various forms, including in the preparation of betel quid, which is chewed for its stimulating effects. Beyond its cultural importance, betel leaf is also a key agricultural commodity with its cultivation contributing to the livelihoods of many farmers [2]. However, the health and productivity of betel leaf plants are severely impacted by diseases. It is affected by the various diseases of downy mildew, leaf spot, and blight. The cause of this disease leads to a drop in both the quality and yield of the crop. These diseases not only threaten farmers' incomes but also affect the marketability of betel leaves.

Timely detection and effective monitoring of leaf diseases are needed to control the spread of plant infections [3]. The conventional methods use manual inspection for disease detection. It consumes more time and is subject to human error. As a result, these methods failed to detect diseases at an early stage and led to significant losses. There is a need for more accurate and automated disease monitoring systems.

Image-based detection systems have emerged as an effective solution for disease classification in plants [4]. This system applies machine learning (ML) and deep learning (DL) models for the accurate detection of leaf detection and classification. It includes different stages of preprocessing, feature extraction and classification. The performance of a system depends on the filter types used for preprocessing and the types of architecture used for feature extraction. The different ML and DL models used for classification like support vector machine, AdaBoost, and Convolutional Neural Network (CNN) [5]. This model can automatically analyze betel leaf images and detect early signs of disease with its type of infection [6].

But still, the current methods for crop disease detection face several challenges. One of the major limitations is the complexity of feature extraction. The conventional feature

extraction techniques struggled to capture the intricate details and subtle variations in leaf patterns caused by diseases. Moreover, many models are based on pre-trained networks without training them to specific datasets such as betel leaves. This leads to limited accuracy in disease classification. Additionally, robotic systems used for real-time disease detection are often prohibitively expensive which makes them inaccessible to small-scale farmers. These systems use high-end controllers, sensors, and computational resources which significantly increase the overall cost. The high cost and complexity of robotic solutions limit their use for farmers to adopt scalable and affordable disease management systems.

This work proposes a cost-effective solution by combining a novel DL model and a real-time detection robot. It integrates the DL model of LeafInsightNet for accurate betel leaf disease classification and the ESP controller with Flask interface for real-time detection. The proposed LeafInsightNet introduces several unique innovations that set it apart from conventional deep learning models for plant disease classification:

- Dynamic Convolution for Adaptive Feature Extraction.
- Compared to traditional static convolution kernels, dynamic convolution in LeafInsightNet adaptively adjusts its filters based on the input features.
- Dual-Stage Attention Mechanism for Enhanced Feature Refinement.

The incorporation of both spatial and channel attention is used for the model to selectively emphasise disease-relevant regions of the leaf and suppress background noise and redundant information.

(1) Synergistic Integration of Dynamic Convolution and Attention

The combination of dynamic convolution and dual-stage attention provides a synergistic effect and enables the model to simultaneously adapt feature extraction and refine attention focus.

(2) Improved Performance with Computational Efficiency

Despite incorporating advanced modules, LeafInsightNet remains lightweight and computationally efficient.

2. RELATED WORK

Hridoyet al. [7] developed a modified CNN model based on the Swish activation function for betel leaf disease classification. The modified CNN includes three depth-wise separable convolutions and two fully connected layers to extract features from the betel leaf. Compared to a conventional CNN, the modified CNN shows an accuracy improvement from 89.53% to 93.45%. Likewise, Abd Algani et al. [8] developed an Ant Colony Optimization (ACO) combined CNN for leaf disease categorisation. The parameter of the CNN model is tuned using ACO to achieve higher accuracy.

In their study, Ullah et al. [9] presented a new DL model called the Deep Tomato Detection Network (DTomatoDNet) for tomato leaf detection. It consists of 19 learnable layers with 1×1 convolution kernels to deeply learn features from tomato leaves. The overall accuracy achieved by the model is 91.5%.

To reduce model complexity such as parameters and calculations, an Asymmetric Dilated Convolution-based Leaf Attention Design (LAD) Inception model is proposed by Zhu et al. [10]. The fully connected layer is changed by global average pooling to further decrease complexity. Results show that the model uses only 1.25MB of space for implementation.

A hybrid model of Inception-V3 CNN with transfer learning is proposed by Liu et al. [11] for leaf disease detection. The inception-V3 model uses auxiliary classifiers to overcome the vanishing gradient problem in the model.

Cap et al. [12] constructed a new type of Generative Adversarial Network (GAN) model called LeafGAN. It uses its own attention mechanism to learn features from augmented images. Compared to CycleGAN, LeafGAN achieves 6.5% higher accuracy in detection.

Nnamdi et al. [13] proposed a self-controlled MobileNetV2 model for bean leaf disease classification. This control is used to improve the training time of the model. Experimental results on 1,368 bean leaf images show that MobileNetV2 achieves an accuracy of 89.5%. Similarly, Begum et al. [14] introduced a MobileNetV3 model based on a gated self-attention strategy called the Gated Self-Attentive Convoluted MobileNetV3 (GSAtt-CMNetV3) model for pepper leaf disease classification. Initially, the images are pre-processed with histogram equalisation. Then, clustering is applied for segmentation. Finally, Osprey optimisation is used to tune GSAtt-CMNetV3 for classification. Results on the pepper leaf dataset show an accuracy of 96.5%.

A hybrid model of the restructured residual dense network is introduced by Kanda et al. [15] for tomato leaf disease detection. The residual connection of the model is used for faster convergence. The dense connections are used to retain important features at each layer by concatenating the inputs and outputs of each layer.

Roshni Polly [16] proposed a different type of DL model for leaf disease classification. The first model is a modified version of YOLOv7 with a Rat Swarm Optimizer called YR2S. The second model is a Red Fox Optimization-based ShuffleNetv2 called RFO-ShuffleNetv2. The optimizers are used to tune the hyperparameters of the models. Likewise, S. Chouhan et al. [17] proposed a bacterial foraging optimizer combined with a radial basis function neural network called BRBFNN, for leaf disease detection. The optimizer is used to tune the weights of the radial basis function to achieve higher accuracy.

Fu et al. [18] presented the CycleGAN model with a new attention mechanism. The proposed CycleGAN is applied to the mango leaf disease data set and compared with other DL models.

The hybrid ResNet and CapsNet model is proposed by Madhavi et al. [19] for leaf disease classification. ResNet is used for strong feature extraction, and CapsNet improves spatial understanding of image transformations. The hybrid model achieves an accuracy of 93%.

Zhang and Zhang [20] proposed a new DL model called Cascading Autoencoder with Attention Residual U-Net (CAAR-UNet) for plant disease classification. The U-Net with an attention mechanism between the encoder and decoder is used for accurate segmentation. The performance of the model is compared using accuracy and intersection over union metrics.

Xiao et al. [21] proposed a new lightweight model called SE-VRNet. It combines a deep variant residual network (VRNet) and a squeeze-and-excitation (SE) module with an attention strategy. This SE-VRNet model accurately classifies the disease in the presence of dispersed locations of lesions.

Aggarwal et al. [22] proposed a Zero-Shot Transfer Learning (ZSTL) for leaf disease detection. A key component of ZSTL is creating an embedding space where both seen and unseen classes share a common representation. It uses

additional information about the unseen classes, such as semantic attributes or textual descriptions.

Prakash et al. [23] used a DL model of EfficientNet B4 for paddy disease classification. It shows an accuracy of 93%. In a similar way, Wani et al. [24] applied EfficientNet B4 for apple leaf disease classification.

Deepa, et al. [25] introduced a DL model to improve the accuracy of maize crop leaf disease detection. This model is constructed using an enhanced Faster-RCNN architecture. It uses ResNet-50 with spatial-channel attention to compute deep key points which are then localized and categorised into distinct classes. Results on the maize crop leaf disease dataset show that RCNN achieves an average accuracy of 94% and a mean Average Precision (mAP) of 0.94.

Rodríguez-Lira et al. [26] proposed an enhanced target detection model based on YOLOv7 for tomato leaves disease classification. The YOLOv7 model is modified with a new detection mechanism. For image segmentation, the Scale-Invariant Feature Transform technique is used to identify key regions.

Kotwal et al. [27] proposed a cross-layered U-Net model for segmenting affected portions in leaves. The residual connections are introduced in the U-Net to achieve better dice score rates. The extracted features are classified using a support vector machine.

Mohanty et al. [28] proposed an ensemble model for leaf health condition classification. It includes different DL models such as ResNet50, VGG19, and ConvexNet for classification. The accuracy is improved by 4.3% when compared to the sole model. Kusuma and Jothi [29] proposed a Vision Transformer-based leaf disease detection model. The Vision Transformer (ViT) applies transformer architecture to computer vision by treating an image as a sequence of patches. It learns global dependencies through self-attention. Similarly, Chang et al. [30] proposed a modified ViT-based leaf disease classification model.

Dubey et al. [31] proposed a two-fold leaf disease classification model. Initially, the features of leaves are extracted using a CNN architecture. Then, a modified bi-long short-term memory (MBi-LSTM) classifier is used for disease detection based on the extracted features. The accuracy reached by the MBi-LSTM model is 93.5%.

Bathe et al. [32] proposed ConvDepthTransEnsembleNet, a weighted deep ensemble learning architecture to improve the accuracy of rice crop leaf disease detection. The ConvDepthTransEnsembleNet showed a precision rate of 95.4% on a limited and unbalanced dataset.

3. PROPOSED MODEL

The proposed betel leaf disease classification framework consists of four key stages: data augmentation, feature extraction, feature enhancement, and classification. To increase the number of images and address data imbalance, data augmentation is performed. It generates multiple images based on noise, geometric variations, and color variations, including brightness. In the feature extraction stage, the deep learning model LeafInsightNet is used to extract more relevant features from betel leaves. After feature extraction, Starfish Optimization (SO) is used to select the optimal features. In the classification stage, the refined feature vector is fed into a parameter-tuned Random Forest (RF) classifier. The parameters of the RF classifier are tuned using SO to achieve minimal classification error. The overall workflow is given in Figure 1.

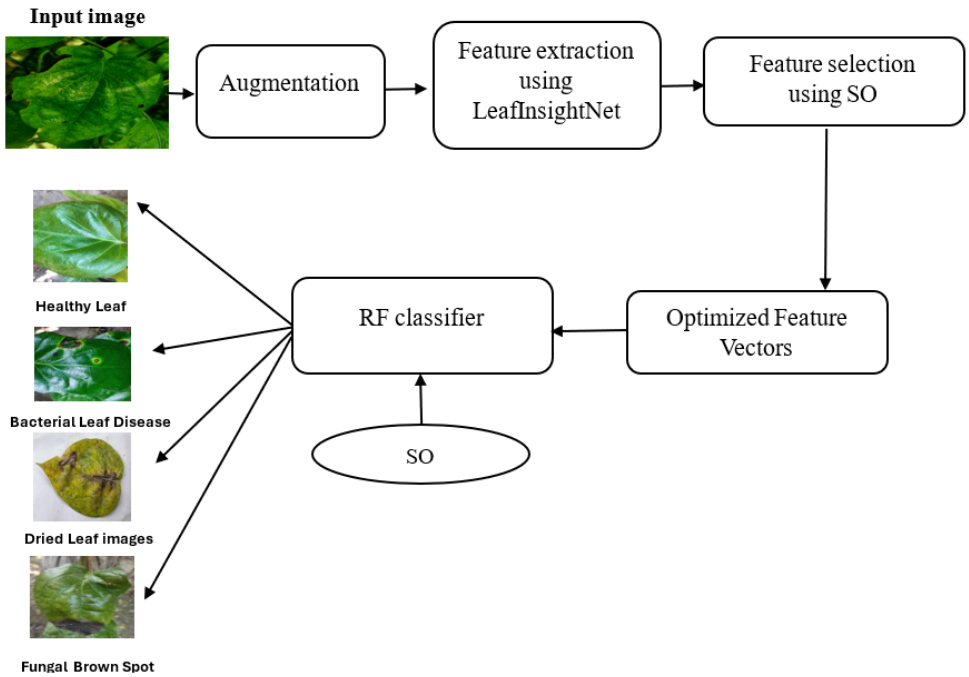

Figure 1. Overall workflow

3.1 LeafInsightNet architecture

LeafInsightNet is a feature extraction model which is composed of several advanced components to extract hierarchical and discriminative features from leaf images. These components include a pre-trained encoder (ResNet-18), dynamic convolutions, and dual-stage attention blocks. These blocks are used to focus on important regions and channels of the leaf images. The connections of the LeafInsightNet are given in Figure 2.

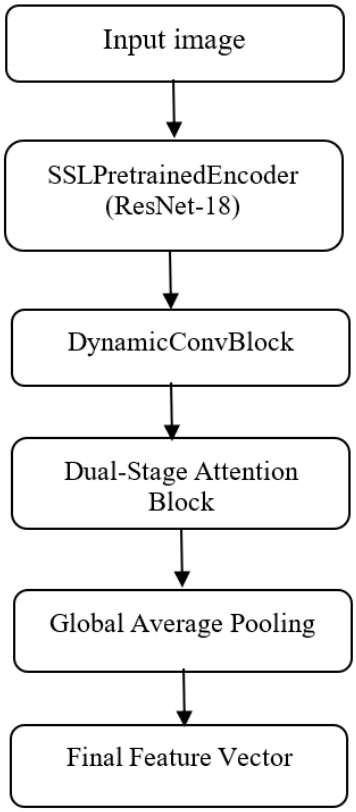

Figure 2. LeafInsightNet model

3.1.1 Input layer

The input layer is responsible for accepting the raw image data. The input image is RGB, and its size is typically 224 × 224 pixels. It can be expressed as follows:

$$X_{input} = Image \in \mathbb{R}^{N \times C \times H \times W} \quad (1)$$

where, N & C denotes the batch size and the number of channels, H and W are the height and width of the input image.

3.1.2 Self-Supervised Learning PretrainedEncoder

The ResNet-18 encoder is a pre-trained CNN model. It extracts low- and mid-level features from the leaf image. ResNet-18 uses residual connections which help mitigate the vanishing gradient problem by enabling deeper networks to learn effectively [33]. ResNet introduces residual connections where each block learns a residual function F(x), i.e., the difference between the output and input of the block as follows:

$$\mathcal{H}(x) = F\left(x, \{W_i\}\right) + x \quad (2)$$

where, $\mathcal{H}(x)$ is the output, F(x) is the learned residual function, and W_i are the weights of the layers. The ResNet-18 layers learn hierarchies of features and reduce the spatial resolution as it move deeper into the network. The last residual block output features are reduced to a vector representation using Global Average Pooling (GAP). The feature extraction from ResNet-18 is given as:

$$F_{ResNet} = ResNet - 18(X_{input}) \in \mathbb{R}^{N \times 512} \quad (3)$$

where, FResNet is the feature map of size [batch_size, 512].

3.1.3 DynamicConvBlock

Dynamic convolution uses learnable filters that are dynamically adapted based on the leaf image. This is used for the model to learn more complex features by adapting to each specific input. This block refines the feature map learned by ResNet and allows it to focus more on relevant patterns in the input data. The dynamic convolution block applies 3 × 3 convolution to the feature map with dynamically learned filters. These filters are not fixed; they adjust based on the input data. Let $F_{ResNet} \in \mathbb{R}^{N \times 512}$ be the feature map from ResNet-18, and D be the learnable dynamic filter. The output of this block is:

$$F_{DynConv} = Conv3 \times 3(F_{ResNet}, D) \quad (4)$$

where $F_{DynConv}$ is the refined feature map with dynamic convolutions and D is a dynamically learned filter of size 3 × 3.

3.1.4 Dual-stage attention block

The dual-stage attention block aims to refine the feature maps by focusing on the most informative regions and channels in the feature maps. It uses two attention strategies: Channel Attention and Spatial Attention.

3.1.5 Channel attention

This attention focuses on the most important channels in the feature map, thereby allowing the model to weigh the relevance of each feature map. It applies GAP to the feature map. The channel-wise average is passed through two fully connected layers, followed by a sigmoid activation to generate a channel attention map. It can be expressed as follows:

$$A_{channel} = \sigma(W_2.ReLU(W_1.GAP(F_{DynConv}))) \quad (5)$$

where, W_1, W_2 are the learnable weights, and σ represents the sigmoid function. $A_{channel}$ is the channel attention map.

3.1.6 Spatial attention

This attention focuses on important spatial regions (pixels) within the feature map. A 7 × 7 convolution is applied to the feature map to model spatial dependencies. The output is passed through a sigmoid function to generate a spatial attention map. It can be expressed as follows:

$$A_{spatial} = \sigma(Conv7 \times 7(F_{DynConv})) \quad (6)$$

where, $A_{spatial}$ is the spatial attention map. The refined feature map after processing both channel and spatial attention is:

$$F_{Att} = F_{DynConv} \times A_{Channel} \times A_{spatial} \quad (7)$$

This final feature map is weighted by both channel-wise and spatial-wise attention.

3.1.7 GAP

GAP is used to minimise the spatial dimensions of the feature map into a 1D vector for each input image. It computes the average over each channel and discards the spatial dimensions (height and width). It can be expressed as follows:

$$F_{GAP} = GAP(F_{Att}) \in \mathbb{R}^{N \times 256} \quad (8)$$

where, F_{GAP} is the global average pooled feature map.

3.1.8 Final feature vector

After global average pooling, the result is a compact feature vector of size 256 for each image. This vector contains high-level, discriminative features that represent the leaf image. It can be denoted as follows:

$$F_{final} = F_{GAP} \in \mathbb{R}^{N\times 256} \tag{9}$$

The final output is a discriminative feature vector of the input image, which reduces the dimensionality and retains important information for classification tasks.

The choice of 256 features is based on the fact that it allows the model to retain enough discriminative information and reduces the risk of overfitting. This dimensionality is chosen as a balance between capturing sufficient high-level information for classification and maintaining computational efficiency. Before applying SO, this feature vector was used as a rich representation of the leaf image.

LeafInsightNet is proposed to adaptively enhance feature extraction for plant disease classification. Dynamic convolution is used to learn filters that adjust based on the specific input data. It increases the model's ability to capture complex and varied patterns. This dynamic adaptation of filters contrasts with static convolutions in traditional models.

Further, the dual-stage attention mechanism refines this process by focusing the model's attention on both relevant spatial regions and important channels in the feature map. Spatial attention is used for the model to emphasise disease-relevant regions. The channel attention selectively highlights informative features across different channels. This integrated approach is used for the model to focus on the most discriminative features.

3.2 Starfish Optimization algorithm

The optimisation techniques are used to find a solution for engineering problems [34]. SO is motivated by the biological behaviors of starfish (sea stars). Starfish are marine organisms known for their remarkable abilities. It has unique locomotion, sensory systems, and regenerative capabilities [35]. The starfish's ability to explore its environment through multiple arms and its preying and regeneration mechanisms form the basis of the algorithm. Starfish move using tube feet located on the underside of their arms. They have the ability to move spontaneously without a brain or a heart. It is guided by nervous signals and hydraulic pressure. The eyes are positioned at the tips of the arms which allows them to detect brightness, shadows, heat, and environmental variations. Also, they have a unique ability to regenerate lost limbs and, in some cases, regenerate an entirely new organism to create a single arm. It is a cooperative hunting strategy where they work together to catch prey, such as clams and oysters. This unique behaviour is mathematically modelled to solve real-world problems. The SO mimics the starfish's behavioral phases—exploration and exploitation—to solve optimization challenges, with the following steps:

3.2.1 Initialization phase

The initialization phase sets up the population of starfish at random positions within the design variable space. The positions of starfish are denoted by a matrix X, where each row corresponds to a starfish's location in the multi-dimensional design space. It can be given as follows:

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots X_{1D} \\ X_{21} & X_{22} & \dots X_{2D} \\ \vdots & \vdots & \vdots \\ X_{N1} & X_{N2} & \dots X_{ND} \end{bmatrix} \tag{10}$$

where N is the number of starfish (population size), D is the number of dimensions of the optimization problem. Each position X_{ij} of the i-th starfish in the j-th dimension is generated randomly within the upper and lower limits of the design variables.

$$X_{ij} = l_j + r(u_j - l_j) \tag{11}$$

where, l_j and u_j are the lower and upper limits of the j-th dimension, r is a random coefficient between zero and one. The fitness values of all starfish are then evaluated using the objective function.

3.2.2 Exploration phase

In the exploration phase, the algorithm simulates the starfish's behavior of moving using its multiple arms (when the dimension D > 5) and a single directional search pattern when D ≤ 5. For D > 5 (five-dimensional search), the update of the starfish's position follows the equation:

$$Y_{ip} = X_{ip} + a\left(c - X_{ip}\right)r \tag{12}$$

where, Y_{ip} is the updated location of the i-th starfish in the p-th dimension, X_{ip} is the current location of the i-th starfish in the p-th dimension, a is a random factor that varies from zero to one, X_{ip} is the best position in the p-th dimension. If the updated location falls outside the boundary of the design variables, the starfish tends to remain in the previous position:

$$X_{ip} = \begin{cases} Y_{ip} \; if \; Y_{ip} \in [l_p . u_p] \\ X_{ip} \; otherwise \end{cases} \tag{13}$$

For D ≤ 5 (unidimensional search), only one arm moves at a time, and the position update follows:

$$Y_{ip} = X_{ip} + h_1.\left(X_{k1p} - X_{ip}\right) + h_2.(X_{k2p} - X_{ip}) \tag{14}$$

where, X_{k1p} and X_{k2p} are randomly selected positions of other starfish, h_1 and h_2 are random coefficients between −1 to +1, The position update depends on the energy E_t of the starfish which evolves with iterations as follows:

$$E_t = \cos\left(\frac{\pi T}{T_{max}}\right) \tag{15}$$

where, T is the current iteration, and T_{max} is the maximum number of iterations.

3.2.3 Exploitation phase

The exploitation phase focuses on using the preying and regeneration attitude of starfish. The preying behavior involves a two-directional search, where the starfish use messages from other starfish and their own best positions. The regeneration phase is focused on improving global convergence by altering the position of the last starfish. The

position update can be stated as follows:

$$Y_{ip} = X_{best,p} + Direction_1.\left(X_{ip} - X_{best,p}\right) \quad (16)$$

where, $Direction_1.\left(X_{ip} - X_{best,p}\right)$ is a random factor and $X_{best,p}$ is the best position known. The regeneration phase occurs when the starfish loses an arm. It denotes a new optimization process. This ensures better global exploration for the algorithm.

3.3 Starfish Optimization-based feature optimization

In this work, SO is used for the optimal selection of features. The quality of features majorly impact on the classification results. For optimisation, the starfish optimizer is initialised with a population of candidate solutions. Each solution denotes the feature set extracted from the LeafInsightNet model. The proposed SO-based feature optimization pseudocode is explained below:

Input: Dataset D with feature set F and labels Y, population size N, maximum number of iterations Max_Iter
Initialization
Generate an initial population of N candidate solutions with random feature subsets.
Assess the fitness of each candidate using the classification error obtained from a Random Forest classifier.
Iterative Optimization
Repeat until the stopping criterion (Max_Iter) is met:
Exploration Stage:
For each solution i:
- Identify a group of better-performing solutions CP_i.
- Randomly choose one solution SP_i from this group.
- Update the position of solution i by referencing SP_i.
- If the updated position achieves a lower classification error, replace the old position.

Exploitation Stage:
For each solution i:
- Refine its position using both the best solution found so far and the population mean.
- Accept the updated solution if it provides improved fitness.

Global Update:
- Refresh the best global solution Xbest whenever a superior candidate is found.

Output
Return the optimal feature subset Xbest along with its associated classification error.

The fitness function is calculated based on the classification error RF classifier. The SO algorithm iteratively performs exploration and exploitation stages to find optimal feature sets. A random solution (SP_i) is selected from CP_i and the current solution's feature set is altered based on SP_i. If the update improves the fitness (classifier error), the current feature subset is replaced by the new one. After the global search, the algorithm shifts focus to local search. Each solution updates its position (feature subset) based on the best solution found so far (X_{best}) and the average solution of the population. This phase refines the feature subset further, concentrating on promising regions of the feature space. Again, if the update increases the classifier's accuracy, the solution is accepted. The process repeats for a predefined number of iterations or until convergence. Finally, the optimal features are returned to the model for classification.

3.4 Random Forest classifier

The RF is a supervised ML model that works by constructing a multitude of decision trees during training and outputs either the mode of the classes. It uses the power of ensemble learning to improve prediction accuracy and reduce overfitting. The key strategies of RF are Decision Trees, Bagging, Random Subspace and Ensemble Prediction**.** The structure of the RF classifier is given in Figure 3. RF consists of multiple decision trees and each tree is built using a random subset of features and data samples. Each decision tree predicts a class for the input features. The trees are trained using the principle of the Gini impurity or entropy to split data at each node. During tree construction, RF selects a random subset of features at each split to minimize the correlation among the trees and increase generalization. After tree construction, RF uses bagging or bootstrap aggregating. In bagging, random sampling with replacement is used to create multiple training datasets from the original dataset. This process ensures diversity among the trees. The final output (accuracy) is estimated by the majority voting mechanism where the class predicted by most trees is selected as the model's output.

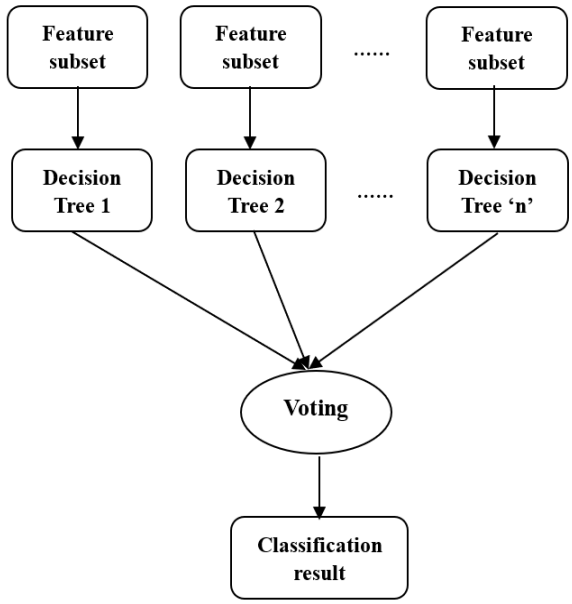

Figure 3. Random forest (RF) classifier structure

For classification tasks, splits are based on minimizing the Gini impurity. Gini impurity for a node is defined as follows:

$$G = 1 - \sum_{i=1}^{C} p_i^2 \quad (17)$$

where, C denotes the number of classes, p_i is the ratio of samples belonging to class i. A split is selected to reduce the weighted average of the Gini impurity across child nodes.

Alternatively, entropy can be used as a split criterion as follows:

$$H = -\sum_{i=1}^{C} p_i \log\ 2(p_i) \tag{18}$$

The final output of the RF classifier is determined by majority voting among T decision trees. For input X, the output class $\hat{y}$ can be computed as follows:

$$\hat{y} = model_1(X), h_2(X) \ldots . h_t(X) \tag{19}$$

where, $h_t(X)$ is the output of the t-th decision tree.

3.5 Starfish Optimization-based parameter tuning

RF classifier performance is mainly influenced by its hyperparameters, particularly the number of estimators ($n_{estimators}$) and maximum tree depth (max_{depth}). These parameters determine the complexity and generalizability of the model. In this work, the Starfish Optimization is applied to discover the optimal configuration for these hyperparameters, accelerating the optimization procedure and enhancing the overall accuracy of the RF model. The algorithm for the SO-based parameter updation is provided below:

Pseudocode: SO for RF model parameter adjustment

Input:

- Dataset D with feature set F and labels Y
- Population size N
- Maximum number of generations Max_Generations
- Search ranges for Random Forest hyperparameters (n_estimators, max_depth)

Population Initialization

- Randomly generate N candidate solutions within the specified hyperparameter bounds.
- Each candidate X_i=[n_estimators,max_depth]
- For every candidate solution:
  - Train a Random Forest classifier using X_i.
  - Evaluate its performance using classification accuracy.
  - Define the fitness function as model accuracy (lower error → higher fitness).

Optimization Process

Repeat until Max_Generations is reached:

Exploration Phase:

- For each candidate i:
  - Identify a subset of solutions with better fitness (CP_i).
  - Randomly pick one solution (SPi) from CP_i.
  - Update X_i by shifting towards the hyperparameters of SP_i.
  - If the updated X_i achieves higher accuracy, replace the old one.

Exploitation Phase:

- For each candidate i:
  - Adjust its position using both the best-known solution (X_{best}) and the mean of the population (X_{mean}).
  - Apply a fine-tuning step around the new hyperparameters.
  - Accept the refined solution if fitness improves.

Global Best Update:

- If any candidate outperforms the current best (X_{best}), update X_{best} accordingly.

Output

- The optimized Random Forest hyperparameters Xbest.
- Their corresponding evaluation metric (classification accuracy or error).

The hyperparameters $n_{estimators}$ and max_{depth} are initialized randomly within predefined ranges. Each solution (starfish) in the population represents a combination of these values. Each solution's fitness is evaluated by training an RF model with the given hyperparameters and calculating its classification error. The algorithm performs a global search during this phase. Each starfish identifies the best solutions (CP_i) and selects a random solution (SP_i) from these candidates. The position (hyperparameters) is updated based on SP_i. If the updated hyperparameters improve the fitness, the current values are replaced. The algorithm focuses on refining the hyperparameters locally by updating them based on the best result obtained so far (X_{best}) and the mean position (X_{mean}) of the population. This ensures that the optimization procedure converges to the best values.

The process is repeated until the convergence condition is reached. At the end, the best set of hyperparameters (X_{best}) is output, minimising classification error and maximising model performance.

4. RESULT AND DISCUSSION

The dataset is downloaded from the Mendeley website (https://data.mendeley.com/datasets/g7fpgj57wc/1). It consists of labeled betel leaf images. Furthermore, the images were collected from Sholavandan village, Madurai district, Tamil Nadu, India. Betel leaves are largely cultivated as a major cash crop in Sholavandan, where they are grown on approximately 150 acres. The total image count is 3,560. The distribution of images across disease categories in the original dataset is as follows: Healthy Leaf (HL) – 900 images, Bacterial Leaf Disease (BLD) – 890 images, Dried Leaf (DL) – 885 images, and Fungal Brown Spot Blight (FBSB) – 885 images. The inclusion of images from the Sholavandan region adds natural variability in terms of lighting, leaf orientation, and background conditions, which are often absent in curated datasets. This diversity enhances the representativeness of the dataset and improves the robustness of the proposed LeafInsightNet model. Consequently, the model is better equipped to generalize to real-world agricultural environments. The collected images are further increased by data augmentation. In this work, three augmentation techniques are used: noise, geometric variations, and color variations in brightness. The geometric variation includes rotations and flips. The color variations include brightness, contrast, and hue shifts. After augmentation, the total image count increases to 4,360. The entire dataset is divided into training and test sets. The test set contains 145 HL images, 142

BLD images, 143 DL images, and 143 FBSB disease images. The visualisation of the dataset images with augmentations is given in Figure 4.

The proposed LeafInsightNet is coded in Python and simulated using IDLE 3.12 environtment. The performance of the model is assesed using following metrics:

$$\text{Accuracy} = \frac{\text{TP+TN}}{(\text{TP+TN+FP+FN'})} \tag{20}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP+FN'})} \tag{21}$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP+FP'})} \tag{22}$$

$$F1\ score = 2.\frac{Precision\ .Recall}{(Precision + Recall)} \tag{23}$$

where, *TN* is True Negatives, *TP* is True Positives, *FN* is False Negatives and *FP* is False Positives.

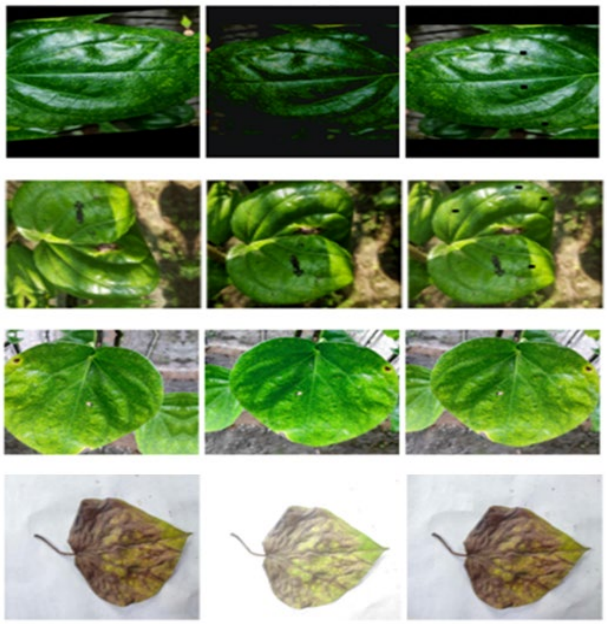

Figure 4. Input images

The hardware prototype was implemented using ESP32 modules. The two modules of ESP32 for the motor and camera interface. The L298 driver is used to control the gear motors for robot movements. The collected image is sent to the cloud through the Wi-Fi module. The images are processed in the Flask server environment. The proposed model classifies the images based on stored feature files and sends the results to the ESP32 module again. The indication given to the ESP32 modules. The hardware prototype is given in Figure 5. The corresponding processing results are displayed in a server system.

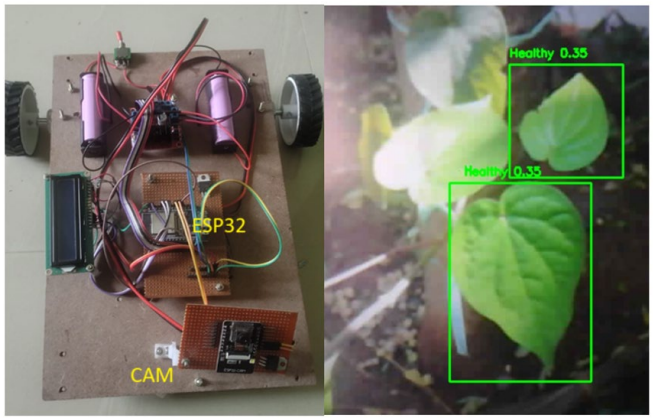

Figure 5. Implementation results

The performance evaluations for the hardware setup are conducted. The ESP32 successfully processed input images at an average rate of 12–15 frames per second which is sufficient for near real-time disease detection. The power consumption measurements indicated that the ESP32 operates at ~160 mA during active processing and ~10 mA in idle mode. It increases the operation on battery-powered deployments. These specifications prove that the proposed robotic system is both efficient and energy-conscious.

Figure 6 shows the training and validation performance of a proposed model over 30 epochs. In the accuracy plot, both training and validation accuracy improve steadily which denotes the model's effectiveness in learning and convergence. Similarly, the loss plot shows a sharp decline in both training and validation which denotes the successful error minimization. Overall, the model exhibits strong performance with balanced accuracy and low loss. Further improvements could include hyperparameter tuning or regularization to minimize validation fluctuations.

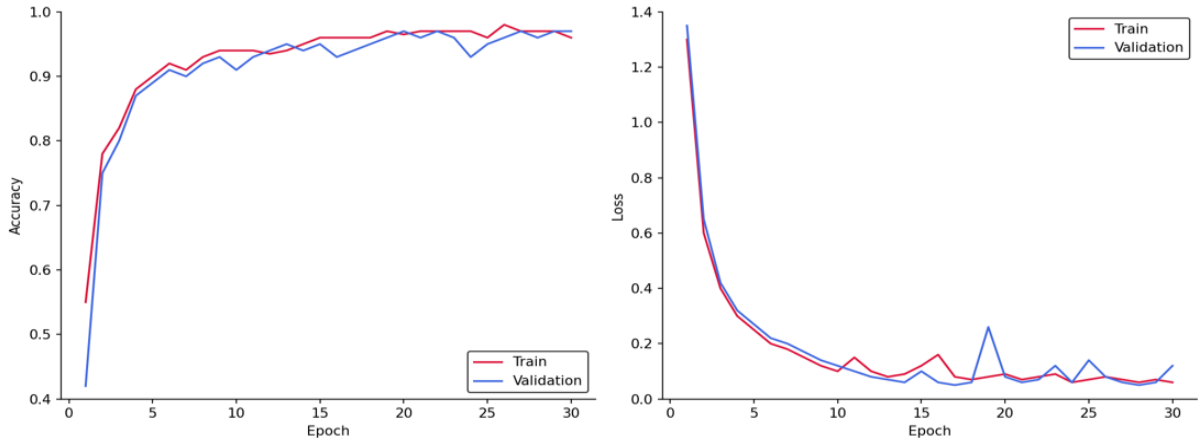

Figure 6. Model accuracy and loss validation curve

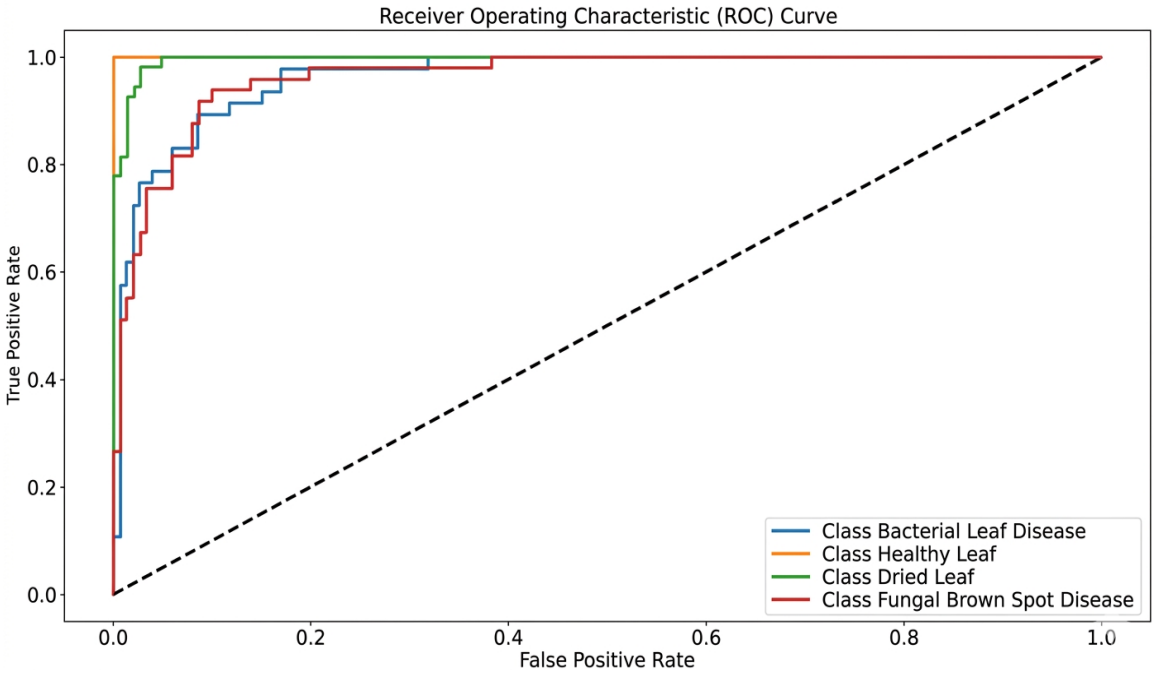

Figure 7. Receiver Operating Characteristic (ROC) of the model

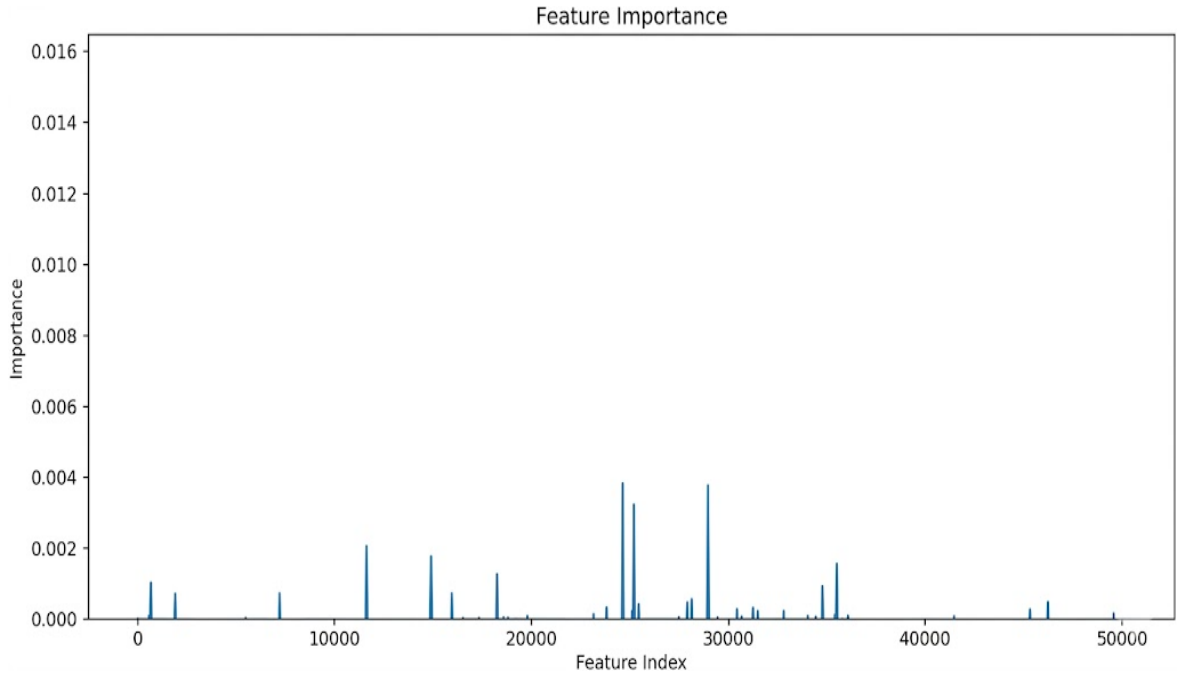

Figure 8. Feature importance plot

The Receiver Operating Characteristic (ROC) curve of the LeafInsightNet model is given in Figure 7. The ROC curves suggest that the classification models perform very well in distinguishing between these leaf conditions. The models for "Healthy Leaf" and "Dried Leaf" are particularly strong with the Area Under the Curve (AUC) scores of 1 & 0.99. The models for "Bacterial Leaf Disease" and "Fungal Brown Spot Disease" are also good, though with a slightly higher potential for misclassification with the 0.978 & 0.983 AUC scores.

The feature importance plot of LeafInsightNet model is visualised in Figure 8. It denotes that only a few features contribute significantly to the model's predictions and most have very low importance. The RF classifier parameter tuning convergence curve is given in Figure 9. The initial fluctuations show that the fish are exploring the parameter space and trying different combinations of parameters. The rapid decrease shows that the SO algorithm is effectively guiding the population towards better regions of the parameter space. The algorithm effectively explores the parameter space and converges to a solution with a low classification error. In the RF classifier, the parameter of $n_{estimators}$ varies from 10 to 200 and max_{depth} varies from 2 to 50 for optimization to achieve low error. The SFO finds $n_{estimators}$ as 95 and max_{depth} as 44 for better classification results.

The confusion matrix of the proposed model for both augmented and non-augmented data sets is given in Figure 10. The obtained results are given in Table 1. The performance metrics highlight the impact of data augmentation on model

performance. The model without augmentation achieves higher precision, recall, and F1 scores across all classes with an overall accuracy of 95.4%. However, with augmentation, there is a significant improvement with the accuracy rising to 98.7% and all metrics showing considerable improvement across classes. This proves that data augmentation improves model performance for this work.

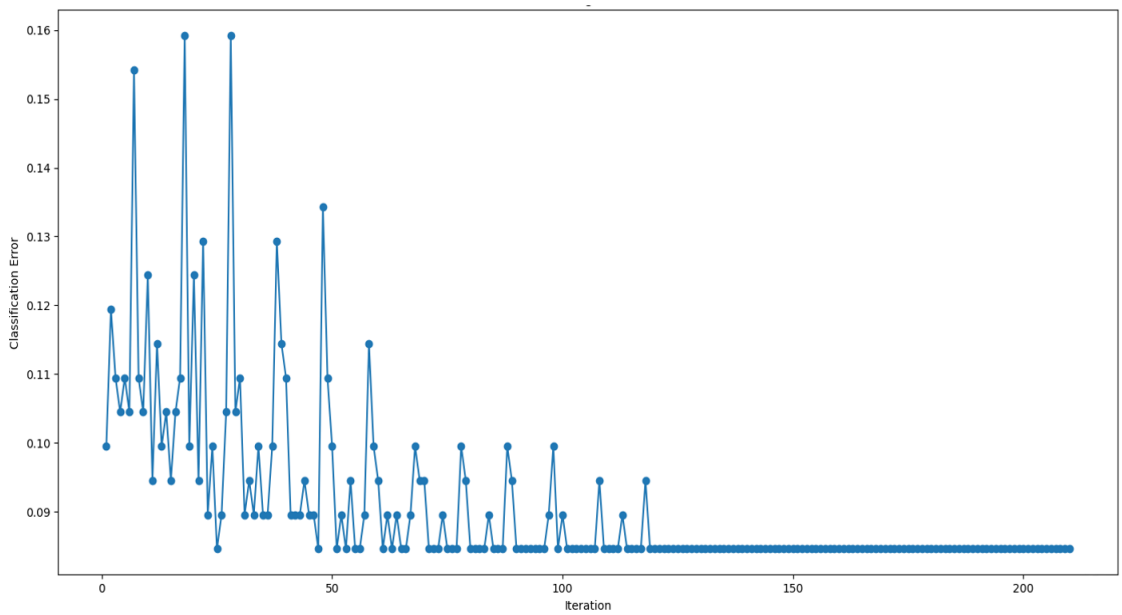

Figure 9. Starfish Optimization (SO) convergence curve

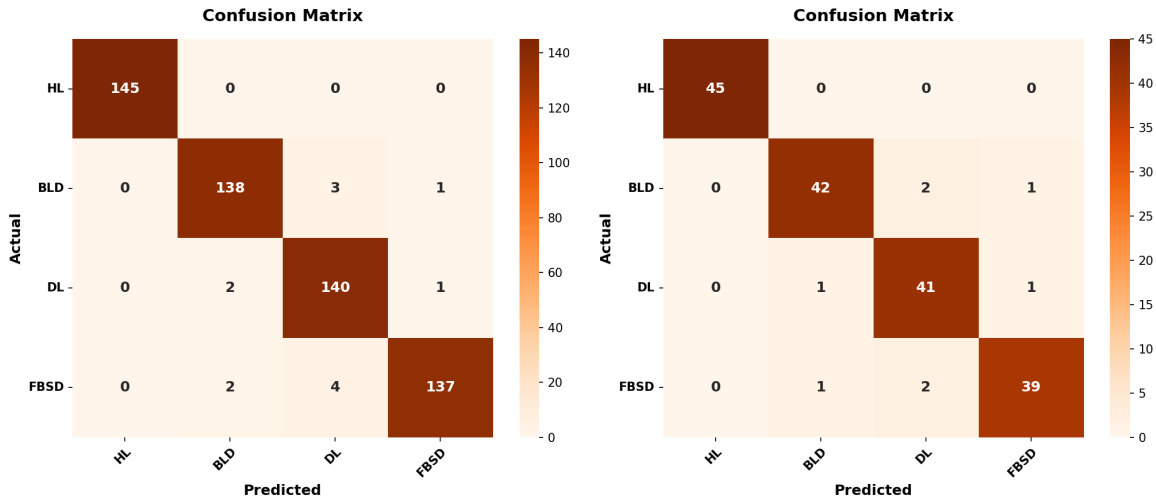

Figure 10. Confusion matrix (a) with augmentation; (b) without augmentation

Table 1. Evaluation of the model based on data augmentation

Model	Class	Precision	Recall	F1-Score	Accuracy (%)
Without augmentation	HL	1	1	1	95.4
	BLD	0.933	0.933	0.933	
	DL	0.932	0.953	0.942	
	FBSB	0.951	0.929	0.940	
With augmentation	HL	1	1	1	98.7
	BLD	0.971	0.971	0.971	
	DL	0.986	0.993	0.979	
	FBSB	0.971	0.990	0.982	

The ablation study of proposed disease classification models using different variations of the LeafInsightNet architecture is given in Table 2. The Model 1 is the LeafInsightNet with RF classification. The feature and model parameters are not optimized. It achieves an accuracy of 96.86%. This Model 2 configuration removes the dynamic block and attention mechanism which results in a slight reduction in performance. It achieves an overall accuracy of 95.6%. Model 3 is a CNN-based model with an RF classifier that achieves an accuracy of 93.7%. The outcome of the ablation study indicates that both the dynamic block and attention mechanisms in the LeafInsightNet architecture

significantly enhance the model's classification capabilities, especially for more complex or less prominent disease categories (DL and FBSB). The confusion matrix is obtained for the models given in Figure 11.

The performance comparison of LeafInsightNet model with existing models is shown in Table 3. The LeafInsightNet model achieves the highest accuracy of 97.8% and outperforms all other models in the comparison. Other models like those based on Improved YOLOv7 and ConvDepth TransEnsembleNet with accuracies of 96.3% and 95.4%, respectively, also show strong performance, but the proposed model surpasses them, proving its superior effectiveness in disease classification.

Table 2. Ablation study of proposed disease classification models

Model	Class	Precision	Recall	F1-Score	Accuracy (%)
LeafInsightNet+ RF without Feature & Parameter Optimization (Model-1)	HL	1	1	1	96.86
	BLD	0.965	0.965	0.965	
	DL	0.927	0.972	0.949	
	FBSB	0.985	0.944	0.964	
LeafInsightNet without Dynamic Block + without Attention +RF (Model-2)	HL	1	1	1	95.6
	BLD	0.937	0.944	0.940	
	DL	0.937	0.944	0.940	
	FBSB	0.971	0.985	0.978	
CNN+ RF (Model-3)	HL	1	0.993	0.997	93.7
	BLD	0.923	0.930	0.927	
	DL	0.909	0.916	0.912	
	FBSB	0.963	0.963	0.963	

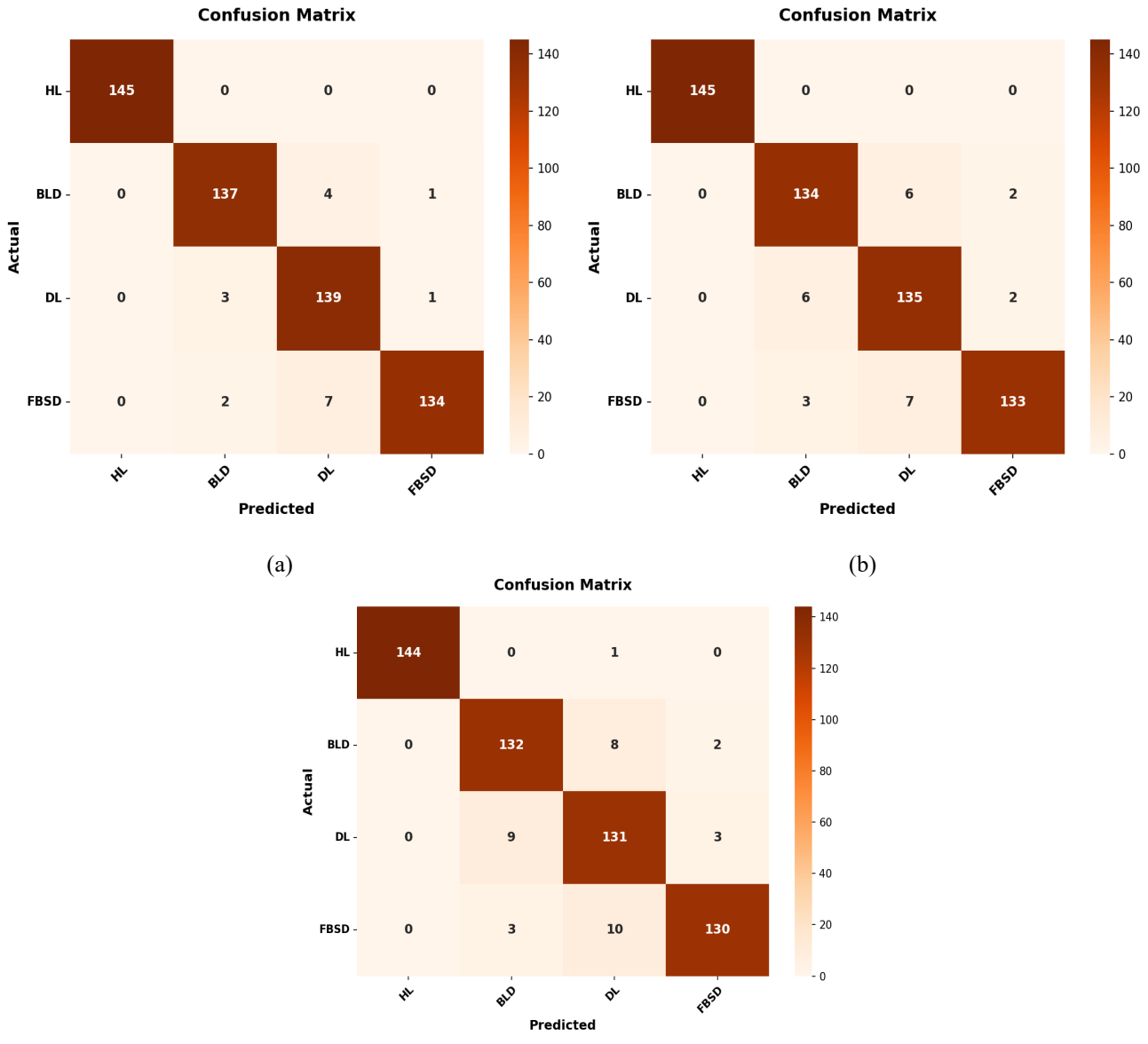

Figure 11. Confusion matrix (a) Model 1 (b) Model 2 (c) Model 3

Table 3. Performance comparison with other models

Method	Accuracy (%)
Modified CNN	92.5
ACO-CNN	90.0
DTomatoDNet	91.5
V3 CNN+ transfer learning	93.0
PiTLiD, based on pre-trained Inception-V3 CNN	94.0
LeafGAN	95.4
MobileNetV2	89.5
Gated Self-Attentive Convoluted MobileNetV3 (GSAtt-CMNetV3)	92.0
A restructured residual dense network	93.0
YR2S & RFO-ShuffleNetv2	94.8
BRBFNN	88.0
CycleGAN	92.6
ResNet + CapsNet	93.0
CAAR-UNet	91.5
SE-VRNet	90.0
Zero-Shot Transfer Learning	91.0
EfficientNet B4	93.0
Improved Faster-RCNN with ResNet-50 and spatial-channel attention	94.0
Improved YOLOv7	96.3
MICL-UNet	92.8
SUNet	92.0
Hybrid model combining ResNet50 and VGG16	93.4
Vision Transformer model	94.5
MBi-LSTM model	93.5
ConvDepth TransEnsembleNet	95.4

Table 4. Real-Time performance comparison of SO with other optimizers

Metric	SO	GA	PSO
Inference time per image (ms)	185	210	200
Classification accuracy (%)	98.7	96.5	97.3
Feature selection time (ms)	250	400	350
Parameter tuning time (ms)	350	500	450
Power consumption (W)	1.8	2.0	2.1
Energy efficiency (mJ/operation)	333	420	420

Note: SO: Starfish Optimization; GA: Genetic Algorithm; PSO: Particle Swarm Optimization

Table 5. Combined ablation study

Component Removed	Accuracy (%)
ResNet-18 (Replaced with VGG16/DenseNet)	95.2 - 96.5
Dynamic Convolution (Replaced with Static Convolutions)	96.3
Dual-Stage Attention (Replaced with Single Attention)	95.6

The performance of SO is compared with other standard optimizers like Genetic Algorithm (GA) & Particle Swarm Optimization (PSO). The comparison results are given in Table 4. The SO model shows the fastest inference time and reduced feature selection time. The SO model achieved the highest classification accuracy with minimal power consumption. Overall, the SO is identified as most feasible for real-time applications.

The ablation study-based component replacement is given in Table 5. It is observed that removal of any component from LeafInsightNet leads to a noticeable drop in performance.

In addition to the laboratory setup, the proposed prototype is evaluated under varied environmental conditions. These conditions include Varying lighting conditions, Field obstructions and Varying distances between the camera and the leaf. To access the scalability for large-scale agricultural deployment, the prototype is deployed as multiple ESP32 units across different parts of a larger area. The proposed system shows successful synchronisation and low-latency communication via Wi-Fi. The results are given in Table 6. The FPGA implementation provided better scalability and reduced power consumption to ~12 W compared to ESP32 and maintained real-time performance in the field. The accuracy of disease classification varied under different field conditions, as given in Table 7. The model shows stable performance for varying conditions.

Using cloud-based servers, the capability of protype is tested to process data from multiple robots simultaneously and analyze large-scale farm data. The system handled up to 500 images per minute with average processing times per image staying under 200 ms. It is observed that the ESP32 is suitable for small-scale and low-cost deployments. The FPGA version is better suited for large-scale farms due to its low power and high scalability.

Table 6. Updated results for realistic field conditions

Test Scenario	Inference Time (ESP32)	Inference Time (FPGA)	Power Consumption (ESP32)	Power Consumption (FPGA)
Ideal lab conditions	185 ms	35 ms	1.8 W	12 W
Field with varying light	200-250 ms	40-50 ms	1.9 W	12 W
Field with obstructions	210 ms	50 ms	2.1 W	12 W
Multiple robots in farm (Scalable test)	190 ms (avg)	38 ms (avg)	1.9 W (avg)	12 W (avg)

Table 7. Impact of environmental conditions on performance

Test Condition	Accuracy (%)	Precision	Recall	F1-Score
Original field test set	98.7	0.982	0.983	0.982
Varying lighting conditions	97.1	0.970	0.973	0.971
Field obstructions	96.3	0.963	0.961	0.962
Multiple robot deployment	97.5	0.975	0.976	0.975
Data aggregation in cloud	98.2	0.981	0.982	0.981

5. CONCLUSION

In this work, a new system is proposed for a low-cost leaf betel leaf detection robot. The data augmentation technique is used to produce a greater number of images and to increase accuracy. For leaf disease classification, a new feature extraction and classification model is proposed. It includes different modules to extract more distinctive features from betel leaf. The Starfish Optimization is used for both feature optimization and parameter tuning. The use of a robust feature extraction pipeline combined with an optimized Random Forest classifier ensures high accuracy and efficiency in

disease recognition. Furthermore, the incorporation of a low-cost, real-time detection robot improves the system's practicality and enables farmers to monitor crops remotely and take timely actions. Future work will focus on optimizing LeafInsightNet using advanced hyperparameter tuning and expanding the dataset with different betel leaf images. In addition, plan to enhance real-time detection under challenging conditions and explore low-power FPGA deployment for efficient field operation.

REFERENCES

[1] Salka, T.D., Hanafi, M.B., Rahman, S.M.S.A.A., Zulperi, D.B.M., Omar, Z. (2025). Plant leaf disease detection and classification using convolution neural networks model: A review. Artificial Intelligence Review, 58(10): 322. https://doi.org/10.1007/s10462-025-11234-6

[2] Maiti, S., Sen, C. (1979). Fungal diseases of betel vine. Pans, 25(2): 150-157. https://doi.org/10.1080/09670877909411690

[3] Zhang, H., Yang, J., Lv, C., Wei, X., Han, H., Liu, B. (2024). Incremental RPN: Hierarchical region proposal network for apple leaf disease detection in natural environments. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 21(6): 2418-2431. https://doi.org/10.1109/TCBB.2024.3469178

[4] Mohammed, A.S., Sakthivel, B., Selvi, T., Duraichi, N. (2025). Advancing agriculture 6.0: A novel residual recurrent unit for crop recommendation using IoT data. Romanian Agricultural Research, 42: 991-1004.

[5] Krishna, M.S., Machado, P., Otuka, R.I., Yahaya, S.W., Neves dos Santos, F., Ihianle, I.K. (2025). Plant leaf disease detection using deep learning: A multi-dataset approach. J, 8(1): 4. https://doi.org/10.3390/j8010004

[6] Andal, P., Thangaraj, M. (2025). Comprehensive review of methods for leaf disease identification. Discover Artificial Intelligence, 5(1): 222. https://doi.org/10.1007/s44163-025-00491-7

[7] Hridoy, R.H., Habib, T., Jabiullah, I., Rahman, R., Ahmed, F. (2021). Early recognition of betel leaf disease using deep learning with depth-wise separable convolutions. In 2021 IEEE region 10 symposium (TENSYMP), Jeju, Korea, Republic of, pp. 1-7. https://doi.org/10.1109/TENSYMP52854.2021.9551009

[8] Abd Algani, Y.M., Caro, O.J.M., Bravo, L.M.R., Kaur, C., Al Ansari, M.S., Bala, B.K. (2023). Leaf disease identification and classification using optimized deep learning. Measurement: Sensors, 25: 100643. https://doi.org/10.1016/j.measen.2022.100643

[9] Ullah, N., Khan, J.A., Almakdi, S., Alshehri, M.S., Qathrady, M.A., Aldakheel, E.A., Khafaga, D.S. (2023). A lightweight deep learning-based model for tomato leaf disease classification. Computers, Materials & Continua, 77(3): 3969-3992. https://doi.org/10.32604/cmc.2023.041819

[10] Zhu, X., Li, J., Jia, R., Liu, B., et al. (2022). LAD-Net: A novel light weight model for early apple leaf pests and diseases classification. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 20(2): 1156-1169. https://doi.org/10.1109/TCBB.2022.3191854

[11] Liu, K., Zhang, X. (2022). PiTLiD: Identification of plant disease from leaf images based on convolutional neural network. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 20(2): 1278-1288. https://doi.org/10.1109/TCBB.2022.3195291

[12] Cap, Q.H., Uga, H., Kagiwada, S., Iyatomi, H. (2020). Leafgan: An effective data augmentation method for practical plant disease diagnosis. IEEE Transactions on Automation Science and Engineering, 19(2): 1258-1267. https://doi.org/10.1109/TASE.2020.3041499

[13] Nnamdi, U.V., Abolghasemi, V. (2025). Optimised MobileNet for very lightweight and accurate plant leaf disease detection. Scientific Reports, 15(1): 43690. https://doi.org/10.1038/s41598-025-27393-z

[14] Begum, S.S.A., Syed, H. (2024). GSAtt-CMNetV3: Pepper leaf disease classification using osprey optimization. IEEE Access, 12: 32493-32506. https://doi.org/10.1109/ACCESS.2024.3358833

[15] Kanda, P.S., Xia, K., Kyslytysna, A., Owoola, E.O. (2022). Tomato leaf disease recognition on leaf images based on fine-tuned residual neural networks. Plants, 11(21): 2935. https://doi.org/10.3390/plants11212935

[16] Polly, R., Devi, E.A. (2024). Semantic segmentation for plant leaf disease classification and damage detection: A deep learning approach. Smart Agricultural Technology, 9: 100526. https://doi.org/10.1016/j.atech.2024.100526

[17] Chouhan, S.S., Kaul, A., Singh, U.P., Jain, S. (2018). Bacterial foraging optimization based radial basis function neural network (BRBFNN) for identification and classification of plant leaf diseases: An automatic approach towards plant pathology. IEEE Access, 6: 8852-8863. https://doi.org/10.1109/ACCESS.2018.2800685

[18] Fu, J., Zhao, Y., Wu, G. (2023). Potato leaf disease segmentation method based on improved UNet. Applied Sciences, 13(20): 11179. https://doi.org/10.3390/app132011179

[19] Madhavi, Yadav, D.K., Singh, T.P. (2026). CapsNet-XAI: Exploring the fusion of capsule networks and XAI for transparent leaf disease identification & classification. Evolutionary Intelligence, 19(2): 43. https://doi.org/10.1007/s12065-026-01158-7

[20] Zhang, S., Zhang, C. (2023). Modified U-Net for plant diseased leaf image segmentation. Computers and Electronics in Agriculture, 204: 107511. https://doi.org/10.1016/j.compag.2022.107511

[21] Xiao, Z., Shi, Y., Zhu, G., Xiong, J., Wu, J. (2023). Leaf disease detection based on lightweight deep residual network and attention mechanism. IEEE Access, 11: 48248-48258. https://doi.org/10.1109/ACCESS.2023.3272985

[22] Aggarwal, M., Khullar, V., Goyal, N., Gautam, R., Alblehai, F., Elghatwary, M., Singh, A. (2023). Federated transfer learning for rice-leaf disease classification across multiclient cross-silo datasets. Agronomy, 13(10): 2483. https://doi.org/10.3390/agronomy13102483

[23] Senthil Prakash, T., CP, V., Dhumale, R.B., Kiran, A. (2023). Auto-metric graph neural network for paddy leaf disease classification. Archives of Phytopathology and Plant Protection, 56(19): 1487-1508. https://doi.org/10.1080/03235408.2023.2289219

[24] Wani, O.A., Zahoor, U., Shah, S.Z.A., Khan, R. (2025). Apple leaf disease detection using transfer learning. Annals of Data Science, 12(1): 213-222. https://doi.org/10.1007/s40745-024-00555-y

[25] Deepa, R., Sharmila, S.T., Niruban, R. (2026). Integrating 3D convolutional neural network and Bidirectional Encoder Representations from Transformers for effective video summarization. Journal of Experimental & Theoretical Artificial Intelligence, 38: 261-287.
[26] Rodríguez-Lira, D.C., Córdova-Esparza, D.M., Álvarez-Alvarado, J.M., Romero-González, J.A., Terven, J., Rodríguez-Reséndiz, J. (2024). Comparative analysis of YOLO models for bean leaf disease detection in natural environments. AgriEngineering, 6(4): 4585-4603. https://doi.org/10.3390/agriengineering6040262
[27] Kotwal, J., Kashyap, R., Shafi, P.M., Kimbahune, V. (2024). Enhanced leaf disease detection: UNet for segmentation and optimized EfficientNet for disease classification. Software Impacts, 22: 100701. https://doi.org/10.1016/j.simpa.2024.100701
[28] Mohanty, S.N., Ghosh, H., Rahat, I.S., Reddy, C.V.R. (2023). Advanced deep learning models for corn leaf disease classification: A field study in Bangladesh. Engineering Proceedings, 59(1): 69. https://doi.org/10.3390/engproc2023059069
[29] Kusuma, S., Jothi, K.R. (2024). Early betel leaf disease detection using vision transformer and deep learning algorithms. International Journal of Information Technology, 16(1): 169-180. https://doi.org/10.1007/s41870-023-01647-3
[30] Chang, B., Wang, Y., Zhao, X., Li, G., Yuan, P. (2024). A general-purpose edge-feature guidance module to enhance vision transformers for plant disease identification. Expert Systems with Applications, 237: 121638. https://doi.org/10.1016/j.eswa.2023.121638
[31] Dubey, R.K., Choubey, D.K. (2024). Adaptive feature selection with deep learning MBi-LSTM model based paddy plant leaf disease classification. Multimedia Tools and Applications, 83(9): 25543-25571. https://doi.org/10.1007/s11042-023-16475-7
[32] Bathe, K., Patil, N., Patil, S., Bathe, D., Kumar, K. (2024). ConvDepthTransEnsembleNet: An improved deep learning approach for rice crop leaf disease classification. SN Computer Science, 5(4): 436. https://doi.org/10.1007/s42979-024-02783-8
[33] Kalaivani, S., Tharini, C., Viswa, T.S., Sara, K.F., Abinaya, S.T. (2025). ResNet-based classification for leaf disease detection. Journal of The Institution of Engineers (India): Series B, 106(1): 1-14. https://doi.org/10.1007/s40031-024-01062-7
[34] Ganesamoorthy, N., Sakthivel, B., Subbramania, D., Balasubadra, K. (2026). Hen maternalcare inspired optimization framework for attack detection in wireless smart grid network. International Journal of Informatics and Communication Technology, 13(1): 123-130. https://doi.org/10.11591/ijict.v13i1.pp123-130
[35] Zhong, C., Li, G., Meng, Z., Li, H., Yildiz, A.R., Mirjalili, S. (2025). Starfish optimization algorithm (SFOA): A bio-inspired metaheuristic algorithm for global optimization compared with 100 optimizers. Neural Computing and Applications, 37(5): 3641-3683. https://doi.org/10.1007/s00521-024-10694-1