International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

EFFICIENT DEEP LEARNING: HYPERPARAMETER TUNING OF DENSENET-CNN USING CUCKOO SEARCH FOR ENHANCED ARECANUT DISEASE IMAGE CLASSIFICATION

Dinesh S[1] ,P. Sridhara Acharya[2] , Ravikiran H K[3]

[1] Research Scholar, Srinivas University Institute of Engineering and Technology, Srinivas University, Mangaluru-575001, India (e-mail: dineshnceh@gmail.com).

[2] Associate Professor, Institute of Computer and Information sciences, Srinivas University, Mangaluru-575001, India. (e-mail: sridharaacharya@gmail.com).

[3] Professor, Department of Electronics & Communication Engineering, Malnad College of Engineering, Hassan-573202, India. (e-mail: ; ravikiranhsn@gmail.com).

Abstract

Arecanut (Areca catechu), a significant commercial crop in tropical region, is vital to the livelihoods of millions of farmers and contributes significantly to the agricultural economy. However, arecanut farming is challenged by several diseases, including Stem Cracking, Yellow Leaf Disease, Mahali (Koleroga), and Stem Bleeding, which negatively affect crop yield and quality. Conventional disease detection methods primarily depend on visual inspection, making the process labour-intensive, time-consuming, and susceptible to errors. To address these challenges, this study aims to develop a robust system for classifying arecanut diseases using a DenseNet-inspired Convolutional Neural Network (CNN) architecture with bottleneck blocks, optimized via Cuckoo search for hyperparameter tuning. Standard CNNs often struggle with feature redundancy, vanishing gradients, and poor generalization, particularly for minority classes. Our approach mitigates these issues by leveraging DenseNet's feature reuse and an optimized architecture through hyperparameter tuning that enhances gradient flow while reducing overfitting. Compared to a standard CNN, the proposed model balances complexity and accuracy. While CNN achieves a 93.74% test accuracy with lower complexity, it struggles with minority classes. The base DenseNet-inspired model with growth rate 16 underperforms at 87.87% accuracy due to overfitting, whereas the growth rate 32 variant improves to 92.19%. The hyperparameter-tuned model achieves the highest test accuracy of 95.54%, demonstrating superior generalization. However, this comes at the cost of increased model complexity due to feature concatenation and additional trainable parameters. Despite this, the optimized architecture prevents overfitting while enhancing classification, particularly for difficult cases like Stem Bleeding (F1-score 0.84 vs. CNN's 0.51). This balance makes it the most effective model for disease detection.

Index Terms— Arecanut, CNN, DenseNet, Cuckoo Search, Hyperparameter tuning, Classification

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Introduction

Arecanut widely known as betel nut, is an important commercial crop predominantly grown in tropical regions. India is the leading producer and consumer, with the crop serving as a vital component of the nation's agricultural economy. It supports the livelihoods of millions of farmers and plays a crucial role in rural development. Arecanut's versatile applications span from traditional medicinal uses and Ayurveda formulations to industrial purpose such as cosmetics, furniture, and textile manufacturing, underscoring its economic and cultural importance [1][2].

Despite its substantial contributions, arecanut cultivation is fraught with challenges, particularly diseases that cause significant losses in yield and quality. These diseases arise from a variety of factors, including climatic conditions, soil health, and poor agronomic practices, all of which make the crop vulnerable to infections. Some of the most common diseases affecting arecanut include Yellow Leaf Disease, Stem Bleeding, Stem Cracking, bud borer, and Mahali Koleroga, all of which are characterized by distinct symptoms that are often difficult to identify in the early stages. Traditionally, disease detection has been carried out through visual inspection by experts, a process that is not only labor-intensive and time-consuming but also prone to inaccuracies, especially when conducted over large plantations [3][4][5].

Yellow Leaf Disease is one of the most prevalent diseases in arecanut, typically affecting young palms under ten years of age. This disease is characterized by the appearance of small brown or black spots with yellow halos, which eventually form larger blighted patches. In severe cases, the disease can lead to drying, drooping, and shedding of leaves, significantly reducing the plant's photosynthetic capacity and overall productivity. Stem Bleeding, another critical disease, manifests as lesions on the trunk that exude a reddish-brown liquid, eventually causing the affected area to decay. Stem Cracking, often associated with environmental stress and poor drainage, leads to splits in the stem, compromising the structural integrity of the plant. Mahali/Koleroga, a fungal disease, primarily affects the nut and is most prevalent during the rainy season [6][4]. This disease causes the nuts to rot and fall prematurely, leading to substantial economic losses. Additionally, the Bud Borer disease, caused by the larval stage of a pest, affects the growing tip of the palm, where the larvae feed on the tender bud. This results in stunted growth, distorted fronds, and, if left untreated, can lead to the death or severely damage the plant's overall vigor and yield [5].

With advancements in technology, the demand for automated disease detection systems in agriculture has grown. Machine learning (ML) and deep learning (DL) have emerged as promising solutions for accurate and efficient disease detection. In particular, Convolutional Neural Networks (CNNs) have demonstrated remarkable capabilities in analyzing and classifying plant diseases through image-based recognition. Their ability to extract meaningful features from diseased plant images makes them a suitable approach for arecanut disease identification.

This study proposes a CNN-based approach for detecting arecanut diseases, leveraging an enhanced model architecture for improved classification accuracy. The proposed method

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

aims to overcome challenges associated with existing techniques, such as dependency on high-quality datasets and computational resource constraints. By integrating feature optimization techniques and a robust classification framework, this approach enhances disease detection efficiency, aiding in effective disease management and improved crop production.

The paper is organized as follows: Section II presents the Literature Review, which discusses relevant studies and identifies gaps in existing research. Section III outlines the Materials and Methods, detailing the data set and methodology. The Results are presented in Section IV, where key findings are discussed and illustrated with data. Finally, the paper concludes with a summary of the findings and suggestions for future work.

Literature Review

The increasing adoption of deep learning techniques has significantly advanced the field of automated plant disease detection. Convolutional Neural Networks (CNNs) have demonstrated remarkable accuracy in identifying and classifying plant diseases, particularly in commercial crops such as arecanut. A notable study by Mallikarjuna et al. utilized AlexNet, a well-known CNN architecture, to classify arecanut images affected by diseases such as rot, split, and rot-split, significantly improving classification accuracy using multi-gradient image enhancement techniques [1]. Another study proposed a CNN-based method for disease detection in arecanut leaves, underscoring deep learning's potential in automating disease management practices [3]. The performance of these methods heavily relies on the quality and variety of the datasets used for training.

Another study aimed at detecting arecanut diseases in India used a CNN and SVM for classification. A dataset of over 250 images of both healthy and diseased areca nuts was created. Textural features were extracted using techniques like Wavelet, Gabor, GLDM, and GLCM. The CNN model performed exceptionally well, achieving an accuracy of 97.5%, with a precision, recall, and F1-score of 98.6%, 96.4%, and 97.5% respectively. The SVM model showed slightly lower performance with an accuracy of 95%. The study highlighted that while CNN excels in feature extraction and complex pattern recognition, it requires more computational resources, whereas SVM can be a viable alternative in resource-constrained settings [4]. A similar study developed an Android-based disease detection application using CNN for arecanut diseases in Indonesia, a leading exporter of the crop. The system achieved 98.7% average accuracy in disease detection, with specific diseases achieving accuracy rates ranging from 94% (Fruit Rot) to 100% (Yellowing Leaf Spot, Foot Rot, Shoot Rot, and Leaf Blight). This mobile application provides farmers with a user-friendly interface, offering quick and reliable disease detection and contributing to improved crop management and yield. The study also noted that future developments could focus on expanding the dataset and integrating other machine learning methods for enhanced real-time detection [7]. The study [8] proposed a CNN-based system for arecanut disease detection, trained on 620 images and tested with an 80:20 training-test split. The system achieved 88.46% accuracy, identifying diseases like Mahali, Stem Bleeding, and Yellow Leaf Spot. The model not only predicted diseases but also recommended remedies, promoting smart farming and helping

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

farmers make informed decisions to enhance crop yield and minimize disease impact [8]. Further, an innovative approach combining multi-gradient-directional (MGD) images with ResNet deep learning models to detect diseases in arecanut is proposed. Diseases like rot, split, and rot-split were considered, with images affected by these diseases often losing important edge details. To restore these features, multi-Sobel directional masks were applied to enhance the image details. The model achieved superior results in disease identification, outperforming traditional methods by significantly improving classification rates, recall, precision, and F-measure. It also emphasized the importance of image augmentation to prevent overfitting and underfitting, demonstrating that MGD images play a crucial role in improving disease identification accuracy [9].

The limitations of existing studies on arecanut disease detection include small datasets, a limited number of disease classes, and reliance on high-quality data, all of which affect model generalization. The computational resources required by CNNs make them less feasible in resource-constrained environments, while image quality and environmental factors influence classification accuracy. Overfitting and underfitting remain concerns with limited or unbalanced datasets. Additionally, previous work has not focused on low-complexity models and hyperparameter tuning. These issues can be mitigated through dataset expansion, the use of computationally efficient models, and proper hyperparameter tuning. However, the proposed DenseNet-inspired CNN model with bottleneck blocks and feature concatenation overcomes these challenges by improving classification accuracy through better feature extraction and reduced complexity, requiring fewer computational resources. Feature concatenation enhances the model's ability to combine relevant features from different layers, leading to a more robust representation of the data. By leveraging expanded and enhanced datasets, the model mitigates issues related to dataset quality while offering superior performance compared to traditional CNN architectures.

Materials and Methods

A. Dataset Description

The dataset utilized for Arecanut disease detection and classification comprises images of different parts of the Arecanut plant, including leaves, nuts, trunks, and roots. The images are categorized into both healthy and diseased conditions to facilitate an accurate classification of various diseases affecting the Arecanut crop [20]. The dataset is structured to facilitate the development of deep learning-based models for automated disease diagnosis. Sample images are presented in Figure 1. The dataset consists of 11,063 labeled images categorized into eight distinct classes, as detailed below, and the dataset distribution for CNN Model Training, Validation, and Testing is tabulated in Table 1.

TABLE I DATASET DISTRIBUTION OF ARECANUT IMAGES FOR CNN MODEL TRAINING, VALIDATION, AND TESTING

Samples	No. of Training images	No. Validation images	No. of Testing images
Healthy	604	46	106

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Leaf			
Healthy Nut	1886	142	330
Healthy Trunk	1432	107	252
Mahali Koleroga	2563	192	449
Stem Bleeding	151	12	26
Bud Borer	144	11	25
Healthy Foot	50	04	09
Stem Cracking	540	41	25
Yellow Leaf Disease	1477	111	259

Healthy Leaf

Healthy Nut

Bud Borer

Yellow Leaf Disease

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Healthy Foot

Healthy Trunk

Stem Bleeding

Stem Cracking

Mahali Koleroga

Fig. 1. Sample images from Dataset

B. Implementation

Bottleneck block

A DenseNet-inspired bottleneck block is a computationally efficient building block in convolutional neural networks (CNNs) designed to enhance feature learning by reducing redundancy and encouraging feature reuse through feature concatenation. It begins with a 1×1 convolution layer, which projects the input feature maps into a lower-dimensional space, reducing the number of channels and thereby decreasing computational cost while retaining essential information. Mathematically, the dimensionality reduction can be expressed as [12][13]:

$$F_{reduce} = W_{1x1} * X \qquad (1)$$

where F_{reduce} is the reduced feature map, X is the input feature map, W_{1x1} represents the weights of the 1×1 convolution, and $*$ denotes the convolution operation.

This is followed by a 3×3 convolution layer applied to the compressed feature maps to extract richer spatial features:

$$F_{expand} = W_{3x3} * F_{reduce} \qquad (2)$$

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Where F_{expand} represents the output after feature extraction and W_{3x3} denotes the weights of the 3×3 convolution. Finally, the output of this 3×3 convolution is concatenated with the original input feature maps, ensuring that previously learned features are preserved and reused in subsequent layers:

$$F_{output} = Concatenate\ (X * F_{expanded}) \quad (3)$$

This mechanism offers several advantages, including efficient feature reuse, as the concatenation operation preserves earlier features, reduced parameter complexity due to the dimensionality reduction step, enhanced gradient flow by directly connecting all layers, and improved representation learning through complementary feature extraction rather than redundant patterns. However, it has limitations such as increased memory usage from the concatenation of feature maps, implementation complexity due to dense connectivity, and a higher risk of overfitting when the increase in feature map size is not regularized appropriately. Despite these challenges, the DenseNet-inspired bottleneck block is widely adopted in deep learning due to its ability to balance computational efficiency and superior feature learning.

Methodology

The methodology of the DenseNet-inspired CNN model involves a series of detailed processes and architectural choices designed to extract meaningful features, improve classification performance, and ensure computational efficiency. Feature extraction begins at the input layer, where the image data, consisting of 128×128×3 pixel RGB images, undergoes convolutional operations. The first layer applies a 5×5 convolution filter with 64 kernels and a stride of 2, which is designed to detect low-level features such as edges, corners, and textures. By using a relatively large kernel size and stride, the receptive field is increased early in the network, enabling the model to capture broader spatial information while reducing the resolution of the feature maps. The 'same' padding ensures that the spatial dimensions of the output remain consistent, preserving edge information. Following this, batch normalization stabilizes the distribution of activations, speeding up training and reducing the likelihood of vanishing gradients. Max pooling with a 3×3 filter and a stride of 2 is then applied, which further down-samples the spatial dimensions while retaining the most prominent features in localized regions. This step reduces computational complexity and ensures that only the most relevant information is carried forward [10][11].

The model then progresses to the DenseNet-inspired bottleneck blocks, which are pivotal for efficient feature extraction and model performance. In each bottleneck block, a 1×1 convolution reduces the number of input feature maps, effectively compressing the data and reducing computational overhead. This bottleneck layer serves as a feature selector, identifying the most informative patterns from the input. The compressed data is then passed through a 3×3 convolution, which focuses on spatial feature extraction. These features include mid-level patterns like shapes, textures, and object parts. Both layers use ReLU activation, encouraging sparse activations that highlight the most prominent patterns while suppressing less significant ones. Batch normalization in both stages ensures stable training and reduces internal covariate shift.

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A key characteristic of the bottleneck blocks is the concatenation of the original input with the newly extracted features. This dense connectivity fosters feature reuse, where information from earlier layers is directly propagated to later ones. As a result, the network can efficiently learn both low-level and high-level features, enhancing gradient flow and mitigating the vanishing gradient problem. Additionally, this connectivity allows the model to retain important contextual information that might otherwise be lost in deeper layers.

Global average pooling is then applied to the feature maps, compressing them into a single vector by averaging the spatial dimensions. This operation retains the global context of the features, such as overall object structure, and prevents overfitting by avoiding fully connected layers with a large number of parameters. The resulting feature vector represents high-level abstractions, such as object categories or distinguishing characteristics, learned from the input images.

The fully connected layer will further process these high-level features; refine their representation and combine them in ways that are more meaningful for classification. Dropout is applied at this stage to randomly deactivate certain number of neurons during training, which prevents over-reliance on specific features and improves generalization to unseen data. The final dense layer, with C neurons and Softmax activation, outputs the probabilities for each class, enabling multi-class classification.

The architecture improves classification performance by leveraging several mechanisms. The bottleneck blocks enhance gradient flow and encourage feature reuse, and the dense connectivity prevents information loss. The model extracts features at multiple levels, ranging from low-level edges and textures to high-level object parts and global structures, ensuring robust and comprehensive feature learning. Furthermore, batch normalization and dropout enhance stability and generalization. Despite these strengths, the model is not without limitations. The dense connectivity, while effective for gradient flow, increases memory usage, which could limit scalability for extremely deep networks. Additionally, the model's reliance on global average pooling might lead to a loss of spatial details, which could affect tasks requiring precise localization.

Hyperparameter tuning

The goal of the Cuckoo Search Optimized CNN for Image Classification algorithm is to enhance the performance of Convolutional Neural Networks (CNNs) by optimizing crucial hyperparameters using the Cuckoo Search algorithm. Cuckoo Search is a nature-inspired optimization technique based on the reproductive behaviour of cuckoo birds. These birds lay their eggs in the nests of other birds, and through Lévy flights, they explore the search space to find better nests (solutions). This methodology allows for an efficient search of the hyperparameter space, which is essential to improving the CNN model's accuracy and generalization performance for image classification tasks. The hyperparameters optimized in this process include the learning rate, batch size, the number of dense units, and the dropout rate, all of which significantly influence the model’s ability to learn from the data [14][15][16].

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The CNN model architecture used in this process is a proposed DenseNet-inspired CNN. The Cuckoo Search algorithm begins by initializing a population of nests, where each nest represents a potential solution with a set of hyperparameters. The hyperparameters include the learning rate, batch size, the number of units in the dense layers, and the dropout rate. Each nest's fitness is evaluated by training the CNN model using its corresponding hyperparameter set and measuring the validation loss. This process requires training the model for a predefined number of epochs to obtain reliable performance metrics.

Once the initial nests are evaluated, Levy flight exploration is used to generate new candidate solutions by perturbing the current solutions. Levy flights introduce randomness to the search, enabling the algorithm to avoid local optima and explore the search space more effectively. If a new solution leads to a lower validation loss than an existing one, the new solution replaces the previous one. In addition, some of the worst-performing nests are abandoned with a probability defined by pa, and new random solutions are generated in their place. This mechanism helps in maintaining diversity in the population and prevents stagnation during the search process. The Cuckoo Search proceeds iteratively, with the algorithm evaluating and updating the population for a set number of generations. After these iterations, the best-performing set of hyperparameters is selected for training the final CNN model.

After the Cuckoo Search algorithm concludes, the best set of hyperparameters is extracted. These values are then used to build the final CNN model, which is trained on the training dataset. The training process involves adjusting the weights of the network through backpropagation**,** based on the loss function and the optimizer. Once the model is fully trained, its performance is evaluated on the test dataset**.** This provides an unbiased measure of the model's ability to generalize to unseen data.

Algorithm: Cuckoo Search Optimized DenseNet-inspired CNN for Image Classification

Input:

1. Training dataset: Initialize images for training.
2. Validation dataset: Initialize images for validation.
3. Testing dataset: Initialize images for final evaluation.
4. Hyperparameter search space: Learning rate, batch size, dense units, dropout rate.
5. Number of nests (num_nests): Population size of the Cuckoo Search algorithm.
6. Number of generations (num_generations): Iterations for optimization.
7. Probability of abandonment (pa): Probability of abandoning nests.
8. Number of epochs: The total number of iterations over the training dataset.
9. Number of epochs for each individual hyperparameter tuning iteration: Set dynamically based on validation loss monitoring.
10. Step size scaling factor: Controls the magnitude of the steps in the Levy flight process, influencing exploration (larger α) or exploitation (smaller α) in the Cuckoo Search algorithm.

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

11. Levy flight exponent: Defines the power-law distribution of step sizes, with smaller β (closer to 1) promoting large steps for exploration, and larger β (closer to 3) enabling smaller steps for local exploitation

Output:

- Optimized hyperparameters for the CNN model.
- Final trained CNN model.
- Classification metrics: Accuracy, confusion matrix, loss curves.

Step 1: Data Preprocessing

1. Load image datasets from the directory.
2. Apply data augmentation.
3. Create train, validation, and test generators.

Step 2: Define DenseNet-inspired CNN Model

2.1 Input Layer:

- The input to the model is an image of size (128, 128, 3).

2.2 Feature Extraction:

- Apply an initial convolutional layer with 64 filters of size (5x5) with a stride of (2,2), followed by Batch Normalization and MaxPooling.
- Implement bottleneck blocks that include 1x1 convolutions followed by 3x3 convolutions, inspired by DenseNet.
- Utilize feature concatenation at each bottleneck block to facilitate information flow across layers.

2.3 Global Feature Reduction:

- Apply Global Average Pooling (GAP) to reduce spatial dimensions before passing data to dense layers.

2.4 Fully Connected Layers:

- Use a dense layer with hyperparameter-optimized units (determined via Cuckoo Search).
- Apply dropout (also optimized) for regularization to prevent overfitting.

2.5 Output Layer:

- Use a Softmax activation function for multi-class classification with two output neurons for binary classification.

2.6 Compilation:

- The model is compiled using categorical cross-entropy as the loss function and the Adam optimizer.

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Step 3: Implement Cuckoo Search Algorithm for Hyperparameter Tuning

3.1 Initialization:

• Generate num_nests initial solutions randomly, where each solution represents a set of hyperparameters: learning rate, batch size, dense units, dropout rate.

3.2 Fitness Evaluation:

• Each nest (solution) is evaluated by training a CNN model with the corresponding hyperparameters and measuring validation loss.

• Each individual hyperparameter set is trained for a fixed number of epochs (e.g., 10-20 epochs) to determine fitness.

3.3 Lévy Flight Exploration:

• New solutions are generated using Levy flight-based perturbations to introduce randomness while searching for optimal values.

Levy Flight Equation (Global Search):

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \cdot \text{Levy}(\beta) \quad (4)$$

where: $x_i^{(t)}$ is the current solution (nest position), $x_i^{(t+1)}$ is the updated solution, α is the step size (scaling factor), and $\text{Levy}(\beta)$ represents a Levy flight step.

Nest Abandonment (Exploration):

$$x_i^{(t+1)} = \begin{cases} Random\ Solution, & if\ r < P_a \\ x_i^{(t)} & otherwise \end{cases} \quad (5)$$

where: $x_i^{(t)}$ is the current solution, α is the step size scaling factor, P_a is the discovery rate (abandonment probability), r is a uniformly random number in [0,1].

Step sizes calculation:

$$Levy\ (\beta) \sim u = t^{-\beta}, (\ 1 < \beta \leq 3) \quad (6)$$

Where β is the exponent that controls the shape of the distribution.

3.4 Selection:

• If a new solution yields a lower validation loss, it replaces the existing solution.

3.5 Abandonment Mechanism:

• With probability pa, some of the worst solutions are replaced by new randomly generated solutions.

3.6 Iteration:

• Steps 2 to 5 are repeated for num_generations until convergence is achieved.

3.7 Final Selection:

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- The best-performing hyperparameter set is chosen for final model training.

Step 4: Train the CNN Model with Optimized Hyperparameters

- Extract the best hyperparameters from Cuckoo Search.
- Build the final CNN model using optimized hyperparameters.
- Train & Validate the model using the training dataset and validate validation dataset.

Step 5: Model Evaluation and Performance Analysis

- Evaluate the final trained model on the test dataset to measure its generalization performance, and compute classification report.

Results

The accurate and efficient identification of arecanut diseases is crucial for maintaining crop yield and quality. Deep learning-based models, particularly CNNs, have demonstrated significant potential in automating disease detection by learning hierarchical representations from image data. However, traditional CNN architectures suffer from issues such as vanishing gradients, excessive computational redundancy, and overfitting [18]. The DenseNet-inspired CNN model proposed in this study addresses these challenges by leveraging dense connectivity, bottleneck layers, and global average pooling, resulting in a more compact, efficient, and robust classification model. The performance of the proposed DenseNet-inspired CNN model for arecanut disease classification is evaluated based on accuracy, precision, recall, and F1-score. The results demonstrate the effectiveness of the model in identifying different arecanut diseases compared to a traditional CNN and MobileNetV3Small model. Table 2 presents the comparison of model parameters for different architectures.

The comparison of model parameters highlights the efficiency of the proposed DenseNet-inspired CNN architectures compared to traditional CNN, MobileNetV3Small, and EfficientNetB0 models. The baseline CNN model, with over 5.05 million parameters, is computationally expensive and susceptible to overfitting due to its large fully connected layers. MobileNetV3Small significantly reduces parameters to 22.6 million, providing a lightweight transfer learning solution. However, its performance is limited in domain-specific fine-tuning scenarios, often leading to lower class-specific detection accuracy. EfficientNetB0 offers a balanced trade-off between accuracy and efficiency with 4.31 million parameters, but it remains heavier than the proposed models. In contrast, the DenseNet-inspired architectures drastically reduce the parameter count while maintaining competitive accuracy.

The Proposed Model with a growth rate of 16 requires only 262k parameters, while the growth rate 32 variant has 513k parameters, both significantly lower than conventional architectures. The hyperparameter-tuned version (growth rate 32) slightly increases parameters to 611k, improving feature extraction and classification performance while keeping computational and memory requirements minimal. This compact design leverages dense connectivity and bottleneck layers to enhance gradient flow and feature reuse, resulting

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

in reduced overfitting risk, lower training costs, and suitability for real-time agricultural applications on resource-constrained devices. Overall, the proposed models provide a practical alternative to standard CNN and MobileNetV3Small approaches, offering high efficiency without compromising accuracy.

Hyperparameter tuning plays a crucial role in optimizing the performance of machine learning and deep learning models. The optimization process was carried out using 7 nests over 5 generations, optimizing 4 parameters such as learning rate, batch size, dense units, and dropout rate—within defined search spaces. The lower bounds were set at 1e-5, 16, 256, and 0.2, respectively, while the upper bounds were set at 1e-3, 128, 1024, and 0.6, respectively, with each Cuckoo search iteration conducted for 15 epochs to ensure effective exploration of the hyperparameter space. In this study, we analyzed different hyperparameter values for varying probability of abandonment (Pa) and learning parameters **(**α and β**)** to identify the best configuration. The parameter ranges considered are Pa values ranging from 0.1 to 0.5, with an optimal selection of Pa = 0.4**,** Alpha (α) values ranging from 0.01 to 1.0**,** and Beta (β) values ranging from 1.0 to 1.5**.** A systematic evaluation was performed across multiple generations, varying the learning rate, batch size, number of dense units, and dropout rate**.** The results were analyzed to determine the best hyperparameter combination that yields the highest performance, where the best configuration obtained includes a learning rate of 0.0001**,** batch size of 49**,** dense units of 748**,** and a dropout rate of 31**%** for Pa = 0.4, α = 0.01, β = 1.5 as shown in Table 3.

Table 4 & Table 5 illustrates the performance of different models in terms of classification metrics. The proposed model with hyperparameter tuning achieves the highest accuracy, precision, recall, and F1-score across different disease classes. The confusion matrices in Figures 2-6 visually illustrate the classification performance of different models.

TABLE II COMPARISON OF MODEL PARAMETERS FOR DIFFERENT ARCHITECTURES

Feature	**CNN Model**	**MobileNetV3Small**	**EfficientNet B0**	**Proposed Model (Growth Rate = 16)**	**Proposed Model (Growth Rate = 32)**	**Proposed Hyperparameter tunned Model (Growth Rate = 32)**
Base Concept	Standard CNN with increasing filter sizes	Transfer Learning using MobileNetV3Small	Transfer Learning using EfficientNetB0	Inspired by DenseNet with bottleneck layers	Inspired by DenseNet with bottleneck layers	Inspired by DenseNet with bottleneck layers
Input Shape	(128, 128, 3)	224, 224, 3	224, 224, 3	(128, 128, 3)	(128, 128, 3)	(128, 128, 3)
Initial Conv	64 filters,	Conv2D (3x3, 16	Conv2D layer	64 filters,	64 filters,	64 filters, (5x5)

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Layer	(3x3) kernel, stride (2,2)	filters, stride 2) followed by BatchNorm + H-Swish activation	(3x3 convolution with stride 2, 32 filters)	(5x5) kernel, stride (2,2)	(5x5) kernel, stride (2,2)	kernel, stride (2,2)
Batch Normalization	After every major conv layer	Built-in as part of the default layer structure of	Built-in BatchNorm layers (applied after most convolutions)	After every major conv layer	After every major conv layer	After every major conv layer
Pooling Layers	MaxPooling (2x2)	Stride-based downsampling and depthwise pooling via inverted residuals; no MaxPooling	Global Average Pooling	MaxPooling (3x3, stride 2)	MaxPooling (3x3, stride 2)	MaxPooling (3x3, stride 2)
Feature Extraction Layers	Standard Conv2D layers	All convolutional blocks in MobileNetV3Small (up to last conv layer)	16 Mobile Inverted Bottleneck Convolution (MBConv) blocks	Bottleneck blocks (1x1 and 3x3 conv layers)	Bottleneck blocks (1x1 and 3x3 conv layers)	Bottleneck blocks (1x1 and 3x3 conv layers)
Growth Rate	N/A	N/A	Implicit scaling by EfficientNetB0 width and depth multipliers (base growth rate = 1.0)	16 (dense connectivity)	32 (dense connectivity)	32 (dense connectivity)
Number of Convolution Blocks	2	11	16	3 (1 initial + 2 bottleneck blocks)	3 (1 initial + 2 bottleneck blocks)	3 (1 initial + 2 bottleneck blocks)
Total Convolution Layers	2	28	82	5	5	5
Global Average Pooling	No	No GAP layer used explicitly (Flatten used instead)	Yes	Yes	Yes	Yes

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

(GAP)						
Fully Connected Layers	512 neurons, Dropout (40%), Output	Dense(256, ReLU)	Dense(256, ReLU) + Dropout(0.3)	512 neurons, Dropout (40%), Output	512 neurons, Dropout (40%), Output	748 neurons, Dropout (31%), Output
Output Layer	Softmax (9 classes)	Softmax (9 classes)	Softmax (9 classes)	Softmax (9 classes)	Softmax (9 classes)	Softmax (9 classes)
Total Parameters	50,53,085	2,26,22,859	43,12,741	2,62,653	5,13,245	6,10,949
Optimizer	Adam (lr = 0.001)	Adam (lr = 0.01)	Adam (lr = 0.001)	Adam (lr = 0.001)	Adam (lr = 0.001)	Adam (lr = 0.0001)
Batch Size	32	32	32	32	32	49

TABLE III HYPERPARAMETER TUNING RESULTS FOR $P_A = 0.4$, $_A = 0.01$, $_B = 1.5$

Generation	**Nest**	**Learning Rate**	**Batch Size**	**Dense Units**	**Dropout Rate (%)**	**Best Hyperparameter**
1	**1**	**0.04046**	**32**	**975**	**23**	**Learning Rate: 0.0001** **Batch Size: 49** **Dense Units: 748** **Dropout Rate: 0.31**
	2	**0.01377**	**19**	**803**	**24**	
	3	**0.07088**	**37**	**309**	**26**	
	4	**0.08596**	**22**	**571**	**53**	
	5	**0.05702**	**27**	**336**	**25**	
	6	**0.05987**	**26**	**465**	**60**	
	7	**0.05036**	**36**	**1022**	**38**	
2	**1**	**0.01610**	**28**	**504**	**25**	
	2	**0.08682**	**32**	**263**	**30**	
	3	**0.0001**	**37**	**309**	**27**	
	4	**0.0001**	**24**	**571**	**57**	
	5	**0.0001**	**27**	**336**	**20**	
	6	**0.07289**	**49**	**748**	**31**	
	7	**0.0001**	**37**	**1023**	**38**	
3	**1**	**0.01610**	**28**	**504**	**25**	
	2	**0.0001**	**32**	**256**	**30**	

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	3	**0.01899**	**19**	**787**	**34**	
	4	**0.02322**	**19**	**514**	**39**	
	5	**0.0001**	**27**	**336**	**20**	
	6	**0.0001**	**49**	**748**	**31**	
	7	**0.01481**	**55**	**636**	**49**	
4	**1**	**0.09591**	**51**	**497**	**49**	
	2	**0.00334**	**53**	**425**	**27**	
	3	**0.01899**	**19**	**787**	**34**	
	4	**0.02322**	**19**	**514**	**39**	
	5	**0.04144**	**18**	**878**	**52**	
	6	**0.0001**	**49**	**748**	**31**	
	7	**0.04527**	**42**	**406**	**44**	
5	**1**	**0.02469**	**57**	**710**	**29**	
	2	**0.07016**	**42**	**682**	**39**	
	3	**0.0001**	**19**	**787**	**34**	
	4	**0.0001**	**16**	**514**	**39**	
	5	**0.04795**	**50**	**311**	**40**	
	6	**0.0001**	**49**	**748**	**31**	
	7	**0.04527**	**42**	**406**	**44**	

TABLE IV

COMPARISON OF MODEL PERFORMANCE FOR ARECANUT DISEASE DETECTION AND CLASSIFICATION

Methods	**Classes**	**Precision rate**	**Recall rate**	**F1-score**	**Support**
CNN	**Healthy_Leaf**	1.00	0.85	0.92	106
	Healthy_Nut	0.99	1.00	0.99	330
	Healthy_Trunk	0.90	0.96	0.93	252
	Mahali_Koleroga	0.95	1.00	0.97	449
	Stem_bleeding	1.00	0.35	0.51	26

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	Bud borer	0.88	0.56	0.68	25
	Healthy_foot	1.00	0.78	0.88	9
	Stem cracking	0.96	0.68	0.80	94
	Yellow leaf disease	0.87	0.95	0.91	259
	Accuracy			0.94	1550
	Macro avg	0.95	0.79	0.84	1550
	Weighted avg	0.94	0.94	0.93	1550
MobileNetV3Small	**Healthy_Leaf**	1.00	0.51	0.68	106
	Healthy_Nut	1.00	1.00	1.00	330
	Healthy_Trunk	1.00	0.57	0.73	252
	Mahali_Koleroga	1.00	0.99	1.00	449
	Stem_bleeding	1.00	0.77	0.87	26
	Bud borer	0.00	0.00	0.00	25
	Healthy_foot	0.00	0.00	0.00	9
	Stem cracking	1.00	0.51	0.68	94
	Yellow leaf disease	0.51	1.00	0.67	259
	Accuracy			0.84	1550
	Macro avg	0.72	0.59	0.62	1550
	Weighted avg	0.90	0.84	0.83	1550
EfficientNet B0	**Healthy_Leaf**	0.32	0.12	0.18	106
	Healthy_Nut	0.66	0.64	0.65	330
	Healthy_Trunk	0.59	0.54	0.56	252
	Mahali_Koleroga	0.67	0.85	0.75	449
	Stem_bleeding	0.00	0.00	0.00	26
	Bud borer	0.00	0.00	0.00	25
	Healthy_foot	0.00	0.00	0.00	9
	Stem cracking	0.28	0.50	0.36	94
	Yellow leaf disease	0.51	0.44	0.47	259
	Accuracy			0.58	1550
	Macro avg	0.34	0.34	0.33	1550

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	Weighted avg	0.56	0.58	0.56	1550
Proposed Model with Growth Rate = 16 (Without Hyperparameter Tuning), Dropout 0.4	**Healthy_Leaf**	0.61	1.00	0.76	106
	Healthy_Nut	1.00	0.97	0.99	330
	Healthy_Trunk	0.81	0.92	0.86	252
	Mahali_Koleroga	0.99	0.95	0.97	449
	Stem_bleeding	0.74	0.65	0.69	26
	Bud borer	0.60	0.24	0.34	25
	Healthy_foot	1.00	0.11	0.20	9
	Stem cracking	0.64	0.91	0.75	94
	Yellow leaf disease	0.99	0.64	0.78	259
	Accuracy			0.88	1550
	Macro avg	0.82	0.71	0.70	1550
	Weighted avg	0.90	0.88	0.88	1550
Proposed Model with Growth Rate = 32 (Without Hyperparameter Tuning), Dropout 0.4	**Healthy_Leaf**	0.75	0.97	0.85	106
	Healthy_Nut	0.98	1.00	0.99	330
	Healthy_Trunk	0.82	0.98	0.89	252
	Mahali_Koleroga	1.00	0.94	0.97	449
	Stem_bleeding	0.83	0.73	0.78	26
	Bud borer	0.81	0.52	0.63	25
	Healthy_foot	1.00	0.67	0.80	9
	Stem cracking	0.85	0.66	0.74	94
	Yellow leaf disease	0.97	0.87	0.92	259
	Accuracy			0.92	1550
	Macro avg	0.89	0.82	0.84	1550
	Weighted avg	0.93	0.92	0.92	1550
Hyperparameter tunned proposed Model with Growth Rate = 32	**Healthy_Leaf**	0.94	0.96	0.95	106
	Healthy_Nut	1.00	0.98	0.99	330
	Healthy_Trunk	0.99	0.91	0.95	252
	Mahali_Koleroga	0.92	1.00	0.96	449

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

	Stem_bleeding	1.00	0.73	0.84	26
	Bud borer	1.00	0.92	0.96	25
	Healthy_foot	1.00	1.00	1.00	9
	Stem cracking	0.91	0.91	0.91	94
	Yellow leaf disease	0.95	0.92	0.94	259
	Accuracy			0.96	1550
	Macro avg	0.97	0.93	0.94	1550
	Weighted avg	0.96	0.96	0.96	1550

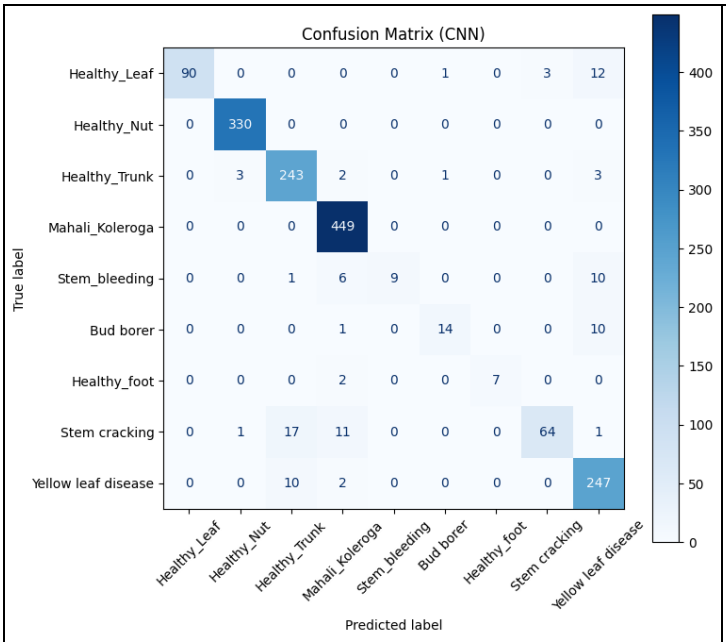

Fig. 2. Confusion matrix for classification using CNN

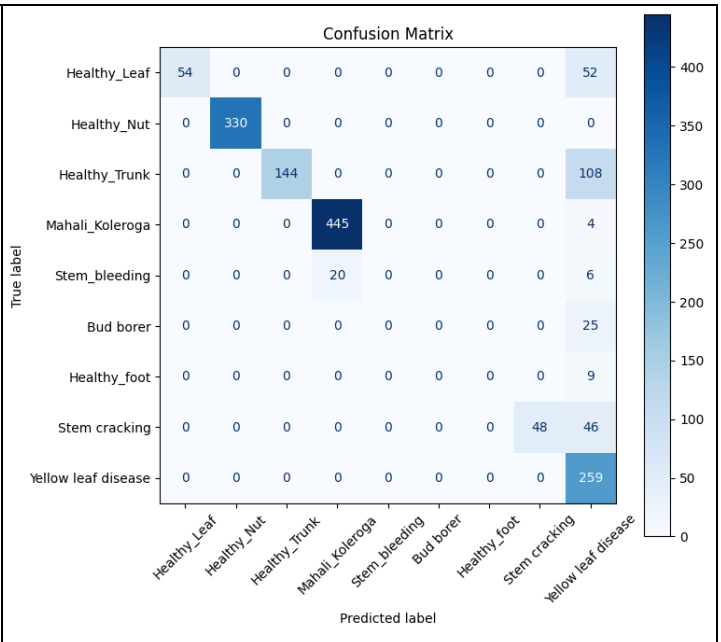

Fig.3. Confusion matrix for MobileNetV3Small Model

Received: July 25, 2025

International Journal of Applied Mathematics
Volume 38 No. 3s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

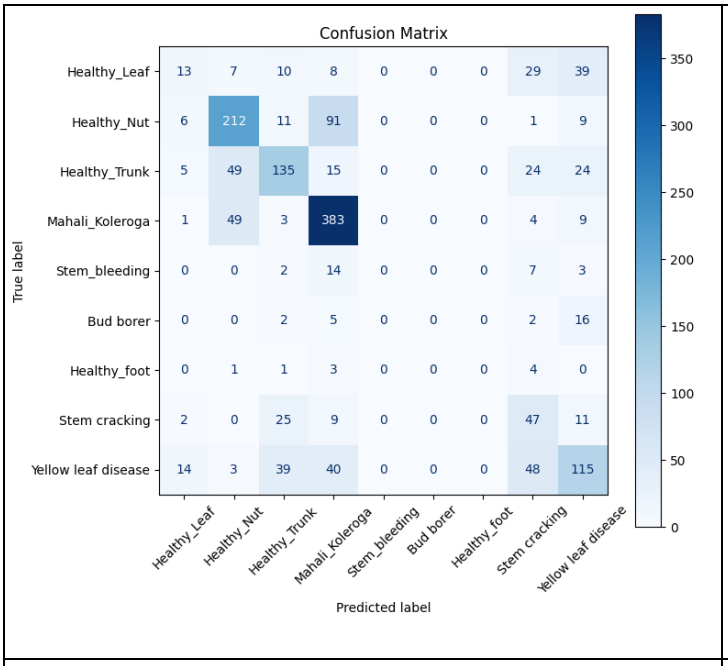

Fig.4. EfficientNet B0

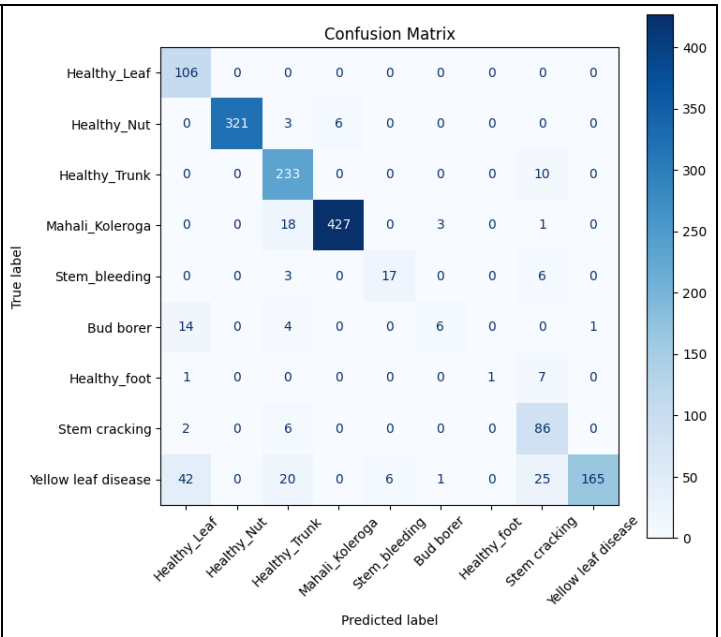

Fig.4. Proposed Model with Growth Rate = 16 (Without hyper parameter tuning)

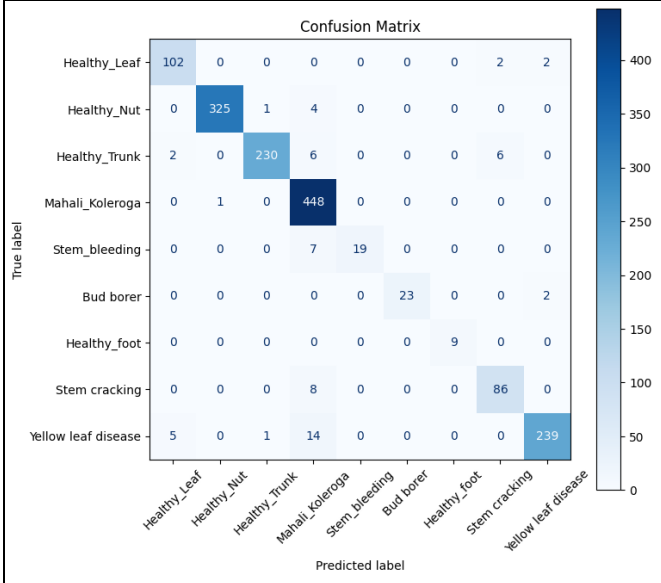

Fig.5. Proposed Model with Growth Rate = 32 (Without hyper parameter tuning)

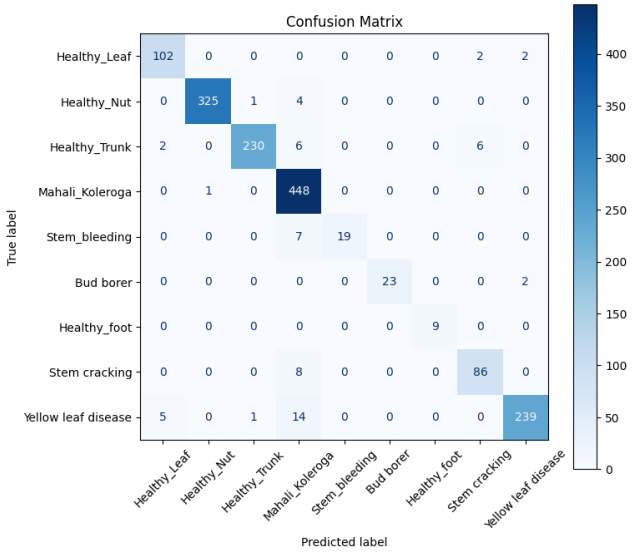

Fig. 6. Hyperparameter tuned proposed Model with Growth Rate = 32

Classification of arecanut diseases is a crucial task for enabling early detection and effective disease management. This study compares the classification performance of five deep learning models: a standard CNN, MobileNetV3Small, EfficientNet B0, and three variations of the proposed DenseNet-inspired architecture with different growth rates and hyperparameter tuning. The models are evaluated using precision, recall, F1-score, accuracy, and loss metrics across training, validation, and test datasets to analyze their strengths and weaknesses for each disease class.

Received: July 25, 2025

Results reveal varying performance across models. For Healthy Leaf, the hyperparameter-tuned proposed model with growth rate 32 achieves the highest F1-score (0.95), followed by CNN (0.92). MobileNetV3Small (0.68) and EfficientNet B0 (0.18) underperform significantly, with EfficientNet B0 struggling to recall this class correctly. In Healthy Nut, CNN and the hyperparameter-tuned proposed model perform similarly (0.99 F1-score), while EfficientNet B0 lags at 0.65 due to lower recall. For Healthy Trunk, the proposed model (0.95) slightly outperforms CNN (0.93), whereas MobileNetV3Small (0.73) and EfficientNet B0 (0.56) fail to achieve competitive results.

Detection of Mahali Koleroga is strong across most models, with the proposed model (0.96) and MobileNetV3Small (1.00) achieving near-perfect classification. However, EfficientNet B0 performs moderately (0.75). In Stem Bleeding, the hyperparameter-tuned proposed model improves detection with an F1-score of 0.84, outperforming CNN (0.51) and EfficientNet B0 (0.00). MobileNetV3Small achieves 0.87 here but with slightly lower precision.

For minority classes like Bud Borer and Healthy Foot, the hyperparameter-tuned proposed model demonstrates significant advantages, scoring 0.96 and 1.00 respectively, where other models struggle—CNN scores 0.68 and 0.88, MobileNetV3Small fails completely (0.00 for both), and EfficientNet B0 also fails to detect these classes (0.00). Detection of Stem Cracking is relatively balanced between CNN (0.80) and the proposed model (0.91), while EfficientNet B0 achieves only 0.36. In Yellow Leaf Disease, the proposed model again performs best (0.94), while CNN (0.91), MobileNetV3Small (0.67), and EfficientNet B0 (0.47) lag behind.

Overall, the hyperparameter-tuned proposed model with growth rate 32 achieves the highest performance across nearly all disease classes, particularly excelling in minority class detection, with an overall accuracy of 96%. The standard CNN performs well but struggles in minority classes, MobileNetV3Small shows inconsistencies across classes despite strengths in Mahali Koleroga detection, and EfficientNet B0 exhibits the weakest performance overall (58% accuracy), failing to detect several classes effectively. The proposed model with growth rate 16 performs reasonably well but underperforms compared to the growth rate 32 variant, indicating the benefits of higher growth rates and optimized hyperparameters in improving disease classification accuracy.

The results in Table 5 indicate that the hyperparameter-tuned DenseNet-inspired model (growth rate 32) delivers the highest overall classification performance, achieving a 96% test accuracy along with superior precision, recall, and F1-scores for most disease classes. This model particularly excels in minority class detection, perfectly classifying Healthy Foot (F1 = 1.00) and achieving strong results for Bud Borer detection (F1 = 0.96), where other models, such as CNN (F1 = 0.68) and MobileNetV3Small (F1 = 0.00), exhibit significant limitations. The untuned growth rate 32 model also performs competitively (test accuracy 92.19%), outperforming CNN on minority classes and demonstrating better generalization than the growth rate 16 variant (87.87%).

The baseline CNN achieves good overall accuracy (93.74%) but shows signs of overfitting, with high training accuracy (97.86%) and a notable validation accuracy drop (93.09%). It also struggles with Stem Bleeding and Bud Borer detection. MobileNetV3Small

Received: July 25, 2025

underperforms (83.87% test accuracy), exhibiting high validation loss (0.86) and severe class imbalance issues, failing to correctly classify both Healthy Foot and Bud Borer. EfficientNet B0 performs the worst (58.38% test accuracy), failing to converge during training (51.60% accuracy, loss = 1.29) and demonstrating poor detection capabilities across multiple disease classes.

Additionally, the comparison of model parameters indicates that the proposed DenseNet-inspired CNN models significantly reduce trainable and total parameters compared to traditional CNNs and MobileNetV3Small, which are computationally expensive and prone to overfitting. With a growth rate of 32, the models achieve lower memory usage and computational costs, making them more practical for real-time agricultural applications. Furthermore, hyperparameter tuning, optimized using the Cuckoo Search algorithm, enhances performance while keeping parameter overhead minimal.

Overall, the proposed DenseNet-inspired architecture, especially the hyperparameter-tuned growth rate 32 model, provides a more compact yet highly accurate solution for multi-class arecanut disease classification. Its robust handling of minority classes, superior generalization ability, reduced parameter count, and optimized computational efficiency make it significantly more suitable for real-time, resource-constrained agricultural applications compared to CNN, MobileNetV3Small, and EfficientNet B0.

TABLE V

COMPARISON OF CLASSIFICATION METRICS OF DIFFERENT MODELS WITH PROPOSED MODEL FOR ARECANUT DISEASE DETECTION AND CLASSIFICATION

Models	**Train Accuracy**	**Train Loss**	**Validation Accuracy**	**Validation Loss**	**Test Accuracy**	**Test Loss**
CNN	0.9786	0.0707	0.9309	0.2278	0.9374	0.2201
MobileNetV3Small	0.9155	0.3313	0.7913	0.8635	0.8387	0.4303
EfficientNet B0	0.5160	1.2975	0.5465	1.2386	0.5838	1.1577
Proposed Model with Growth	0.9693	0.0861	0.8544	0.6269	0.8787	0.3593

Received: July 25, 2025

Rate = 16						
Proposed Model with Growth Rate = 32	0.9517	0.13374	0.8814	0.3324	0.9219	0.2110
Hyperparameter tuned proposed Model with Growth Rate = 32	0.9774	0.0834	0.9715	0.1019	0.9554	0.2248

TABLE V I

COMPARISON OF CLASSIFICATION METHODS AND ACCURACY

SL No.	Method	Dataset Description	Accuracy (%)
1	CNN [8]	Dataset consists of 620 images: 200 healthy and 420 unhealthy. Classes: Yellow Leaf Disease, Mahali/Koleroga, Yellow Spot, and Stem Bleeding Disease.	88.46%
2	ResNet [9]	Dataset consists of 281 images, augmented to 12,124. Four classes: Healthy, Rot, Split Rot, and Split.	88.1%

Received: July 25, 2025

3	Convolutional Neural Network [6]	Dataset consists of 1,100 images: Four classes: Yellow Leaf, Healthy Leaf, Nut Split.	93.05%
4	DenseNet201-SVM+RBF Kernel [17]	Dataset consists of 6748 images and 11 classes of Sugarcane Leaf Disease images	96.74 %
5	KNN, SVM, RF, AlexNet, VGG16, ResNet34, EfeicientNet, MobileNetV2 [19]	Ten class labels of PlantVillage dataset with 1591 healthy and 5357 infected tomato images.	KNN=82.10% SVM=91.00% RF=82.70% AlexNet=92.70% VGG16=98.90% ResNet34=99.70% EfcientNet=98.90% MobileNetV2=91.20%
6	EfficientNetB6 [20]	Dataset consist of 1295 Beans Leaf Diseases images with three classes	91.74%
7	Proposed	Dataset consists of 11,063 labeled images categorized into eight distinct classes, as described in the Dataset section	95.54%

Table 6 compares the performance of various methods in classifying plant diseases based on image datasets. It highlights the accuracy of models like CNN, EfficientNetB6, ResNet, DenseNet and a custom convolutional neural network, using datasets of varying sizes and classes. The proposed method, with the largest dataset of 11,063 images, achieves the best accuracy, demonstrating the effectiveness of using diverse, well-labeled image data for disease classification.

Received: July 25, 2025

Conclusion

In conclusion, the comparative analysis of CNN, MobileNetV3Small and the proposed DenseNet-inspired models for arecanut disease detection demonstrates that the hyperparameter-tuned model with a growth rate of 32 provides the best classification performance. This model achieves superior accuracy, precision, recall, and F1-score across various disease categories while maintaining an optimal balance between trainable parameters and computational efficiency. The study confirms that hyperparameter tuning plays a crucial role in enhancing model generalization, and a well-optimized growth rate significantly impacts feature extraction quality. The Cuckoo search algorithm was utilized for hyperparameter tuning, enabling efficient exploration of the search space to optimize model parameters. While this approach significantly improves model performance, it also comes with certain challenges, such as increased computational overhead and sensitivity to initial parameter settings. Despite these drawbacks, the proposed model demonstrates superior performance compared to the standard CNN and MobileNetV3Small model, particularly in accurately classifying minority classes, resulting in a more reliable disease classification.

Future work can focus on incorporating more advanced deep learning architectures, such as attention mechanisms, transformer-based networks, or hybrid models, to further improve classification accuracy. Additionally, integrating domain adaptation techniques and semi-supervised learning can help address challenges posed by limited labeled datasets, making the model more robust in real-world agricultural applications.

References

[1] Mallikarjuna, S. B., Palaiahnakote Shivakumara, Vijeta Khare, Vinay Kumar, M. Basavanna, Umapada Pal, and B. Poornima. "CNN based method for multi-type diseased arecanut image classification." Malaysian Journal of Computer Science 34, no. 3 (2021): 255-265. https://doi.org/10.22452/mjcs.vol34no3.3

[2] Siddesha, S., S. K. Niranjan, and VN Manjunath Aradhya. "Color features and KNN in classification of raw arecanut images." In 2018 second international conference on green computing and internet of things (ICGCIoT), pp. 504-509. IEEE, 2018. 10.1109/ICGCIoT.2018.8753075

[3] Karthik, V. Arun, Jatin Shivaprakash, and D. Rajeswari. "Disease Detection in Arecanut using Convolutional Neural Network." In 2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), pp. 1-6. IEEE, 2024. 10.1109/ACCAI61061.2024.10602152

[4] Hakim, Arief Rachman, Budi Warsito, Muhammad Rionando Divangga, Diah Safitri, and Ardiana Alifatus Sa'adah. "Performance convolutional neural network (CNN) and support vector machine (SVM) on tuberculosis disease classification based on X-ray image." Commun. Math. Biol. Neurosci. 2025 (2025): Article-ID. https://www.doi.org/10.56726/IRJMETS41103

[5] Puneeth, B. R., and P. S. Nethravathi. "A literature review of the detection and categorization of various arecanut diseases using image processing and machine learning

Received: July 25, 2025

approaches." International Journal of Applied Engineering and Management Letters (IJAEML) 5, no. 2 (2021): 183-204. https://doi.org/10.47992/IJAEML.2581.7000.0112

[6] Hegde, Ajit, Vijaya Shetty Sadanand, Chinmay Ganapati Hegde, Krishnamurthy Manjunath Naik, and Kanaad Deepak Shastri. "Identification and categorization of diseases in arecanut: a machine learning approach." Indonesian Journal of Electrical Engineering and Computer Science 31, no. 3 (2023): 1803-1810. 10.11591/ijeecs.v31.i3.pp1803-1810

[7] Riza, Ferdy. "Android-Based Areca Plant Disease Detection Using Convolutional Neural Network (Cnn) Algorithm." Instal: Jurnal Komputer 16, no. 03 (2024): 277-288. Doi. 10.54209/jurnalinstall.v16i03.213

[8] Anilkumar, M. G., T. G. Karibasaveshwara, H. K. Pavan, and D. Sainath Urankar. "Detection of diseases in arecanut using convolutional neural networks." International Research Journal of Engineering and Technology (IRJET) 8, no. 5 (2021): 4282-4286.

[9] Mallikarjuna, S. B., Palaiahnakote Shivakumara, Vijeta Khare, M. Basavanna, Umapada Pal, and B. Poornima. "Multi-gradient-direction based deep learning model for arecanut disease identification." CAAI Transactions on Intelligence Technology 7, no. 2 (2022): 156-166. https://doi.org/10.1049/cit2.12088

[10] Chen, Leiyu, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. "Review of image classification algorithms based on convolutional neural networks." Remote Sensing 13, no. 22 (2021): 4712. https://doi.org/10.3390/rs13224712

[11] Ravikiran, H. K., J. Jayanth, M. S. Sathisha, and K. Bindu. "Optimizing Sheep Breed Classification with Bat Algorithm-Tuned CNN Hyperparameters." SN Computer Science 5, no. 2 (2024): 219. https://doi.org/10.1007/s42979-023-02544-z

[12] Hasan, Najmul, Yukun Bao, Ashadullah Shawon, and Yanmei Huang. "DenseNet convolutional neural networks application for predicting COVID-19 using CT image." SN computer science 2, no. 5 (2021): 389. https://doi.org/10.1007/s42979-021-00782-7

[13] Scharnagl, Bastian, and Christian Groth. "Evaluation of different deep learning approaches for EEG classification." In 2022 5th International Conference on Artificial Intelligence for Industries (AI4I), pp. 42-47. IEEE, 2022. 10.1109/AI4I54798.2022.00018

[14] Guerrero-Luis, Maribel, Fevrier Valdez, and Oscar Castillo. "A review on the cuckoo search algorithm." Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications (2021): 113-124. https://doi.org/10.1007/978-3-030-68776-2_7

[15] Tong, Yu, and Bo Yu. "Research on hyper-parameter optimization of activity recognition algorithm based on improved cuckoo search." Entropy 24, no. 6 (2022): 845. https://doi.org/10.3390/e24060845

[16] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." In 2009 World congress on nature & biologically inspired computing (NaBIC), pp. 210-214. IEEE, 2009. 10.1109/NABIC.2009.5393690

Received: July 25, 2025

[17] Kurniawan, Rudi, Erna Siaga, Samsuryadi Samsuryadi, Andrianto Tri Susilo, and Lukman Sunardi. "Hybrid DCNN-SVM Architecture for Optimizing Sugarcane Leaf Disease Classification." IAENG International Journal of Computer Science 52, no. 4 (2025).

[18] Guo-Xi, Ren. "Research on a convolutional neural network method for modulation waveform classification." IAENG International Journal of Computer Science 50, no. 3 (2023).

[19] Tan, Lijuan, Jinzhu Lu, and Huanyu Jiang. "Tomato leaf diseases classification based on leaf images: a comparison between classical machine learning and deep learning methods." AgriEngineering 3, no. 3 (2021): 542-558.

[20] Singh, Vimal, Anuradha Chug, and Amit Prakash Singh. "Classification of beans leaf diseases using fine tuned cnn model." Procedia Computer Science 218 (2023): 348-356.

[21] https://www.kaggle.com/datasets/tejpaviraj/arecanut

Received: July 25, 2025