Journal of Agricultural Sciences
(Tarim Bilimleri Dergisi)

J Agr Sci-Tarim Bili
e-ISSN: 2148-9297
jas.ankara.edu.tr

JAS

A Hybrid Lightweight Deep Neural Network Approach for Plant Disease Classification Using Self-Attention Mechanism and Transfer Learning

Thaer Sultan Darweesh Alramli[a,b], Adem Tekerek[c*]

[a]*Graduate School of Natural and Applied Sciences, Gazi University, 06570 Ankara, TURKEY*

[b]*Computer Engineering Department, University of Mosul, Mosul 41002, IRAQ*

[c]*Computer Engineering Department, Faculty of Technology, Gazi University, 06570 Ankara, TURKEY*

ARTICLE INFO

Research Article

Corresponding Author: Adem Tekerek, E-mail: atekerek@gazi.edu.tr

Received: 22 August 2024 / Revised: 29 November 2024 / Accepted: 18 November 2024 / Online: 25 March 2025

Cite this article

Alramli T S D, Tekerek A (2025). A Hybrid Lightweight Deep Neural Network Approach for Plant Disease Classification Using Self-Attention Mechanism and Transfer Learning. *Journal of Agricultural Sciences (Tarim Bilimleri Dergisi),* 31(2):392-412. DOI: 10.15832/ankutbd.1537267

ABSTRACT

Classification of plant diseases is crucial for overall food security and agricultural economies in the world. However, classification has long been challenging primarily due to the various diseases it encompasses and the different environmental factors influencing them. One of the main challenges in developing an accurate classification model is obtaining high-quality, multiclass datasets. At the same time, deep learning methods like CNN may be considered state-of-the-art in detecting complex image patterns in various applications for correct diagnoses. Still, they involve poor parameter optimizations and overfitting and have very high resource requirements. This paper introduces a combined model of classifying plant diseases in an imaging approach, which incorporates the Efficient Neural Network (ENN) with a Squeeze and Excitation Network (SEN). The architecture follows high-density feature extraction by the networks, late fusion of features, and using a cross-channel attention mechanism to boost feature representation. This work uses transfer learning to design the hyperparameter optimization scheme and early stopping scheme to avoid overfitting. We tested our model on the Plant Village Dataset and the Leaf Rose Disease Dataset with an accuracy of 96.40% for the Plant Village Dataset and 97.15% accuracy on the Leaf Rose Disease Dataset. Our model achieved higher accuracy than the traditional DNNs VGG16, Inception V3, and RESNET-50 by approximately 21.04%, 9.40%, and 4.33% on the Plant Village Dataset. It improved the classification accuracy compared to VGG16, Inception V3, and RESNET-50 by 15.80%, 11.20%, and 6.02% on the Rose leaf disease dataset, respectively. Moreover, it has the lowest times as well as space complexity: 45 minutes and 150 MB, which are less than VGG16 (50 minutes, 180 MB), Inception Net (55 minutes, 170 MB), and RESNET-50 (75 minutes, 190 MB). The global results show that our approach is superior, demonstrating enhanced performance and efficiency, which makes it well-suited for real-time applications.

Keywords: Leaf classification, Convolutional neural network, Efficient Neural Network, Squeeze and Excitation Network, Classification accuracy

1. Introduction

Plant diseases threaten global food security and sustainable agriculture, reducing crop yield, quality, and economic viability (Strange et al. 2005). Rapid and accurate diagnosis of these diseases is essential for effective prevention and management strategies. Traditional diagnostic methods rely on expert analysis, which can be time-consuming, subjective, and prone to human error (Bock et al. 2020). Deep learning methods integrated with AI have emerged as a powerful solution for automating plant disease classification, enhancing efficiency and Accuracy (Li et al. 2021). Distinguishing objects in images based solely on visual traits is complex, and traditional machine-learning approaches focused on feature extraction often struggle with Accuracy (Zhao et al. 2019). The advent of deep learning has transformed this landscape, as these methods excel at recognizing intricate data patterns and can effectively handle large datasets. They offer scalability, flexibility, and end-to-end learning capabilities (Kamilaris & Prenafeta-Boldú 2018). Deep learning has become crucial in agriculture, addressing challenges in crop monitoring, predictive modeling, harvesting, and disease detection.

Previous work has proposed several plant leaf detection frameworks based on shape, texture, and vein characteristics (Agarwal et al. 2006), along with various classification models for identifying healthy and diseased plants (Liu et al. 2016; Yanikoglu et al. 2014). Chai et al. (2010) studied four diseases of tomato leaf, including early blight and late blight leaf mildew and leaf spot, extracted 18 characteristic parameters like color and texture, and shape information on the tomato leaf spot images, using stepwise discriminant and Bayesian discriminant principal component analysis PCA, respectively. The characteristic parameters and discriminant models of methods principal component analysis and fisher discriminant were used. The precisions of the two methods were 94.71% and 98.32%, respectively. Guan et al. (2010) extracted 63 parameters, including morphology, color and texture features of spots, and diseases of the rice leaf, classified and recognized with step-based discriminant analysis

and Bayesian discriminant method, three rice diseases (blast, stripe blight, and bacterial leaf blight) into the highest recognition accuracy of 97.2%. Mohanty et al. (2016) trained a deep-learning model to recognize 14 crop species and 26 crop diseases. The trained model reached an accuracy of 99.35% on the test set. Ma et al. (2018) used a deep CNN to conduct symptom-wise recognition of four cucumber diseases, including downy mildew, anthracnose, powdery mildew, and target leaf spots. The recognition accuracy reached 93.4%. Kawasaki et al. (2015) proposed a CNN-based system for detecting cucumber leaf disease, achieving an accuracy of 94.9%. Ahmad et al. (2020) employed four pretraining convolution neural networks, VGG19, VGG16, ResNet, and Inception V3, and the models were trained by fine-tuning parameters. The experimental results demonstrated that the Inception V3 performed the best on the two datasets (the laboratory dataset and the field dataset). The average performance is superior to 10% to 15% on the laboratory dataset compared with the field dataset. Bi et al. (2022) demonstrated that the accuracy rates of recognition for two apple leaf spot and rust models obtained by agriculture experts are 77.65%, 75.59%, and 73.50% with ResNet152, Inception V3, and MobileNet, respectively.

DL approaches have outperformed classical classification accuracy for several leaf-based plant disease datasets (Reyes et al. 2015; Mehdipour-Ghazi et al. 2017). Sun et al. (2017) proposed an advanced leaf classification model using RESNET-50 architecture, which enabled the residual connection and was validated on the BJFU100 dataset. This model mitigated information loss and enhanced feature density by ensuring proper gradient flow. However, this approach was still limited by computational and memory complexities constraints. Ferentinos (2018) designed a transfer learning-based CNN model that classified 87,484 plant leave images from 58 classes with an accuracy of 99.53%. Tiwari et al. (2021) reported a mean validation accuracy of 99.58% and a mean test accuracy of 99.199% through five-fold cross-validation after training a Dense Convolutional Neural Network (DCNN) on a diverged dataset of plant leaves from 26 countries. Sai Reddy and Neeraja (2022) designed a binary classifier for distinguishing between healthy and diseased leaves that achieved 96% to 99% accuracy on the New Plant Diseases (Augmented) and Rice Leaf datasets. Most recently, Ashwinkumar et al. (2022) designed an automated model based on an optimal MobileNet-based convolutional neural network that can optimize hyperparameters via the Emperor Penguin Optimizer (EPO) and relies on an Extreme Learning Machine (ELM) classifier for final classification.

The baseline models proposed various novel DL architectures and realized high classification accuracies in leaf-based plant classification tasks. However, these approaches still have problems, such as dependency on many hyperparameters, computationally expensive, poor convergence rate, and limited accuracy. In this research, we proposed a novel hybrid classification learning model by integrating two lightweight but highly accurate DL architectures, Efficient Neural Network (ENN) with a Squeeze and Excitation Network (SEN), to resolve the mentioned challenges. Efficient Neural Network (ENN) is a popular and efficient CNN architecture that has gained equal or even better performance than other traditional CNN models on various computer vision tasks (Koonce, 2021). In addition, ENN showed exceptional skill levels on image classification tasks using fewer parameters and resources than alternative architectures. Therefore, they are beneficial when it comes to being deployed in resource-restricted situations, including mobile devices or so-called edge devices. Similarly, Squeeze and Excitation Network (SEN) architectures also come from the family of light architectures that integrate effectively with other architectures and are thus compatible with several CNN models and gain top performance across many computer vision tasks (Hu et al. 2018). In addition, these models can design adaptive recalibrating feature maps when merged with other architectures and improve feature representation.

Further, we employed transfer learning (Weiss et al. 2016) to speed up the training procedure by adopting pre-trained weights and other related hyperparameters. The proposed methodology emphasizes a process of feature extraction from the ENN and SEN models, followed by a combination of self-attention blocks (Zhao et al. 2020) and a late feature fusion technique (Akilan et al. 2017) to merge them efficiently. The performance of the proposed methodology was cross-validated using a five-fold cross-validation technique (Wong et al. 2017) on two publicly available (1) Plant Village (Hughes et al. 2015) and (2) Rose Leaf Disease (Sazzad et al. 2022) Datasets. Finally, the testing results are compared with five traditional CNNs, i.e., Inception V3, RESNET-50, and VGG16 (Theckedath et al. 2020), Conventional Inception-Visual Geometry Group Network (Pal & Kumar 2023), EffecientNetV2 (Devi et al. 2023) in terms of various performance measures and recapitulated the global findings. Our contribution can be summarised as follows:

1. Design a novel lightweight hybrid CNN architecture that integrates the ENN and SEN frameworks for multiclass plant-leaf classification while enhancing performance through self-attention blocks and a late fusion technique.
2. Performance evaluation was conducted using a five-fold cross-validation technique on two publicly available (1) Plant Village and (2) Rose Leaf Disease Datasets.
3. We compared our results with five state-of-the-art methods: (1) Inception V3, (2) RESNET-50, (3) VGG16, (4) Conventional Inception-Visual Geometry Group Network, and (5) EfficientNetV2 models, evaluating them based on classification accuracy, precision, recall, F-score, ROC, and both time and space complexity.

The structure of the rest of the paper is as follows: Section 2 includes dataset details and fundamentals of the employed DL models. In Section 3, the proposed methodology is explained. The experimental results and discussion are covered in Section 4. Finally, the conclusions and the future scope of this work are presented in Section 5.

2. Material and DL Fundamentals

2.1. Dataset description

Our study used two popular and open sources: (1) PlantVillage and (2) Rose Leaf Disease datasets for training, validation, and testing of the proposed leaf classification framework. The details of both datasets are given below:

(1) PlantVillage Dataset

The PlantVillage database is one of the most extensive and publicly available standard datasets for plant disease classification, which is often used to compare the results obtained by various methods (Hughes et al. 2015). This dataset contains 54,306 plant leaves with 12 normal and 26 disease categories that can be applied to 14 different types of plants, such as tomatoes, potatoes, apples, and Grapes. The sample collection of Plantvillage varies according to angle, light, size, noise, and so on, which is why it is among the perfect plant disease recognition databases. The samples from the Plantvillage dataset are shown in Figure 1.

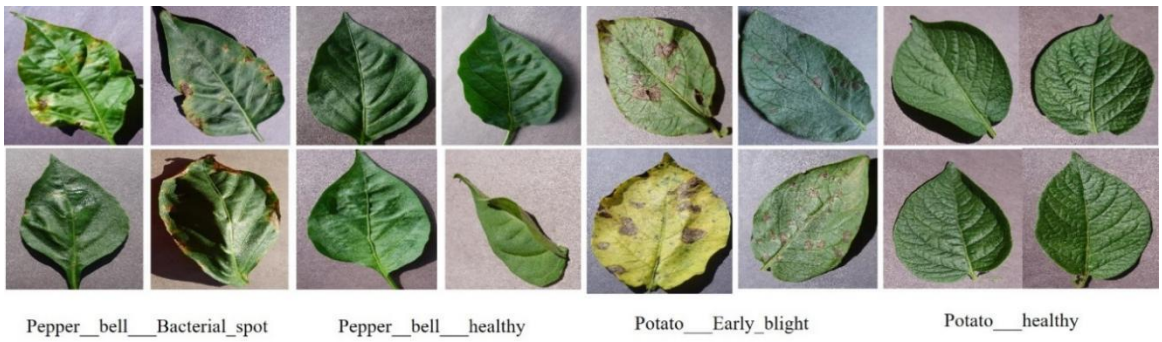


Figure 1- Visualization of leaf samples with respective classes of the PlantVillage dataset

(2) Rose Leaf Disease Dataset

The rose-leaf disease dataset is another popular and openly accessible dataset for the rose-leaf classification (Sazzad et al. 2022). It consists of 14,910 samples of rose leaves across three classes—Healthy Leaf Rose, Rose Rust, and Rose Sawfly/Rose Slug. The samples from the Rose Leaf dataset are shown in Figure 2.

Figure 2- Visualization of leaf samples with respective classes of the Rose leaf dataset

2.2. Deep learning paradigms and baseline architecture

In recent years, artificial intelligence based on deep learning has dramatically improved the most challenging tasks in computer vision, such as classification, object detection, segmentation, and generation. Deep learning models depend on artificial neural networks, which are composed of multiple layers; because of this, these models can learn complex hierarchical features independently from raw image data. Convolutional neural networks have gained widely held appeal concerning image classification; these models often outperform human performance. Due to their structure, convolutional layers may filter local patches of the input image. Thus, spatial information is not lost from these neural networks, and CNNs can capture small-scale

events and progressively learn more complex features produced by deeper networks. Recent advanced techniques such as attention mechanisms, transfer learning, generative modeling, and supervised learning have significantly improved the performance of existing CNN architectures. In our work, we employed the improvement strategies of self-attention mechanisms (Zhao et al. 2020) and transfer learning (Ghazi et al. 2017). Our proposed approach outperforms five baseline CNN architectures presented by Theckedath et al. (2020): (1) VGG-Net, (2) Inception-Net, (3) RES-Net, (4) Conventional Inception-Visual Geometry Group Network, and (5) EfficientNetV2. We describe these strategies further in the subsequent sections.

2.2.1. Self-Attention in convolution neural network

Self-attention in CNN is a mechanism that enables neural networks to detect relationships between pixels in an image. It calculates the weighted sum of each element in the sequence, with the weights changing dynamically according to how similar each element is to the others. This mechanism implies a multi-step operation including Query (Q), Key (K), and value (V) metrics to evaluate linearly transformed feature maps of input images using Eq. 1.

$$\begin{aligned} Q&=XW_Q, \\ K&=XW_K, \\ V&=XW_V \end{aligned} \quad (1)$$

Where; W_Q, W_K, and W_V are learnable weight matrices. Further, attention scores (A) are computed by considering the dot product between the Query (Q) and Key matrices (K) using Eq. 2.

$$A=\text{softmax}\left(\frac{Q*K^T}{\sqrt[2]{d}}\right) \quad (2)$$

Where; d is the dimensionality of the Key vectors. Next, Softmax normalization performs the necessary weight scaling, allowing the network to highlight contextual pixels using Eq. 3.

$$A=\text{softmax}(A) \quad (3)$$

The normalized attention ratings are utilized to generate a weighted sum of the Value matrix: $Y=AV$ represents the attended feature map. The extracted feature map is finally used in the next step.

2.2.2. Transfer learning

Transfer learning in computer vision refers to applying the experience learned by previously trained models to new vision challenges in which data is incorrectly labeled. The pre-trained models offer capabilities that are very handy for different tasks trained on massive datasets, such as ImageNet. The learned weights of the pre-trained model can be deployed as features or fine-tuned on the new dataset using intermediate layers. This way, the computational resources and time taken for training are minimized. At the same time, good performance is also realized with small or similar datasets. Let X be the input dataset with N samples, and W be the pre-trained model's weights:

The pre-trained model works as a feature extractor, and the output of one of its intermediate layers is designated as $F_{Pre}(X, W_{Pre})$. Further, the loss function L computes the discrepancy between the model's predictions and the ground truth labels (Y) using Eq. 4.

$$L=f(F_{Pre}(X,\ W_{Pre}),\ Y) \quad (4)$$

Finally, Gradient descent or its derivatives are used to optimize the model parameters (pre-trained or fine-tuned weights) to minimize the loss function. The optimization phase updates the model parameters to fit the new dataset better, leveraging the pre-trained model's knowledge.

2.2.3. Baseline CNN architectures

(1) VGG19-Net

The Visual Geometry Group at the University of Oxford proposed the VGG-Net convolutional neural network architecture. Karen Simonyan and Andrew Zisserman introduced it in their paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" in 2014 (Simonyan & Zisserman, 2014). The input of the VGG neural Network is a 224x224-pixel RGB image. It gives uniform sample sizes for the ImageNet competition, and the authors subtracted 224 x 224 from the middle of every picture. Convolutional Layers use the 3×3 receptive field to scan the entire image and generate spatial patterns. The second step is the linear transformation of the output, which happens before it is ReLU-activated. The generalized VGG structure is shown in Figure 3.

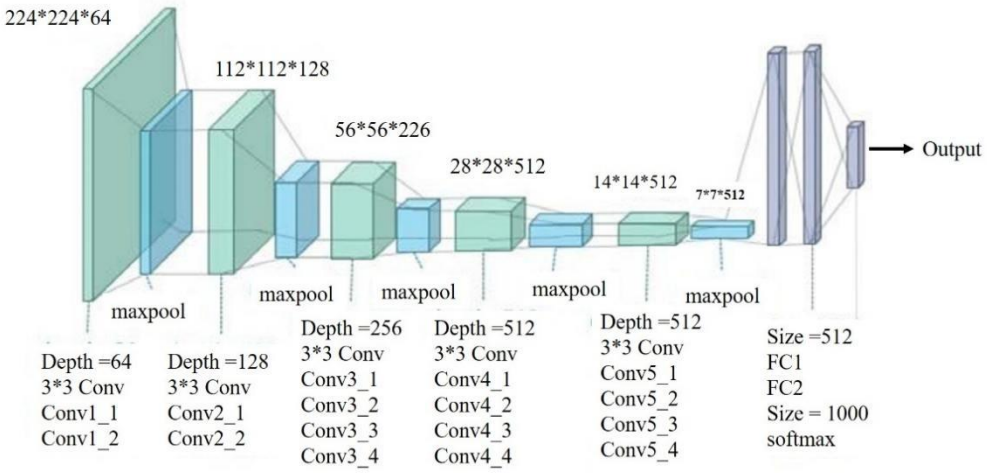


Figure 3- The generalized architecture of VGG-Net (Simonyan and Zisserman, 2014)

(2) Inception-Net

Inception-Net, also known as Google-Net, is an advanced CNN architecture that tries to overcome some of the weaknesses of VGG-Net, including overfitting and high computational cost due to its large number of hyperparameters (Szegedy et al. 2016). The convolutional network consists of 27 convolutional layers that, for instance, use a mix of kernel operations, including 1×1, 3×3, and 5×5 kernels, with the results being fed forward into the subsequent stage as feature vectors. In addition, maximum pooling is performed and concatenated before being passed on to the next inception module. These inception modules are stacked linearly to remove unnecessary and redundant image patterns for the reduction in dimension. It has 22 layers deep when not counting pooling layers and uses global average pooling right after the final inception module. Figure 4 presents Inception-Net's architecture.

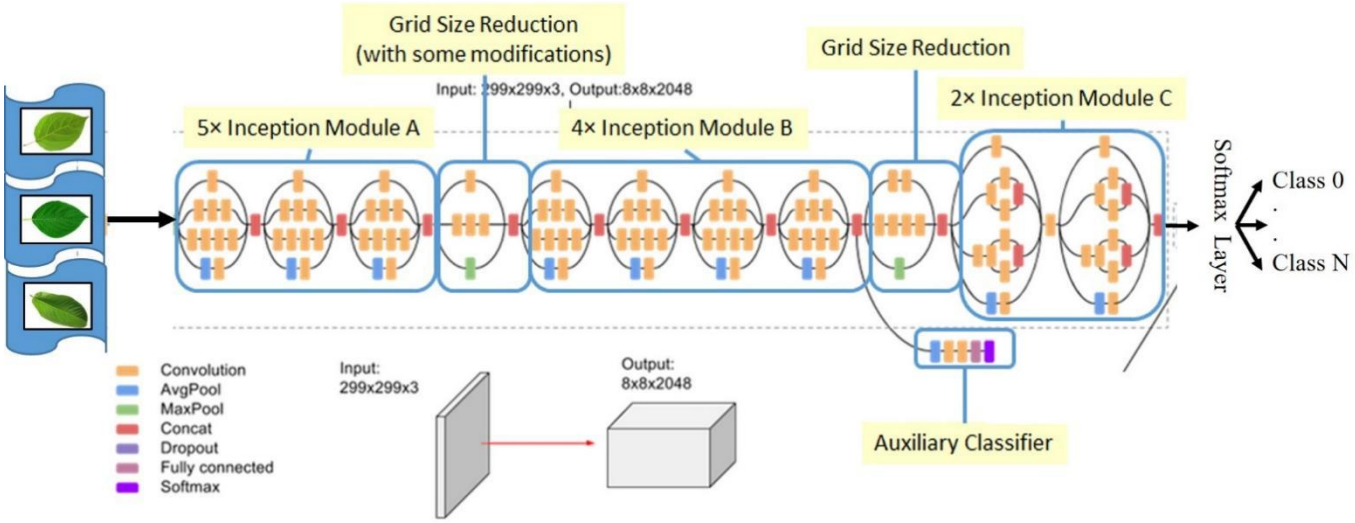


Figure 4- Inception-Net Architecture proposed in the original article (Szegedy et al. 2016)

(3) RESNET-50 Architecture

RESNET-50 is a deep neural network with 50 convolution layers within 16 blocks (He et al. 2016). It introduced skip connection to improve against the vanishing gradient problem, thus facilitating the possibility of training large deep networks. The RESNET-50 network appears as a simple bottleneck design with 1x1, 3x3, and 1x1 convolutions, which reduces the complexity of computation while preserving high representational power; instead of using fully connected layers followed by the SoftMax classifier and ReLU activation functions, it uses global average pooling (Figure 5).

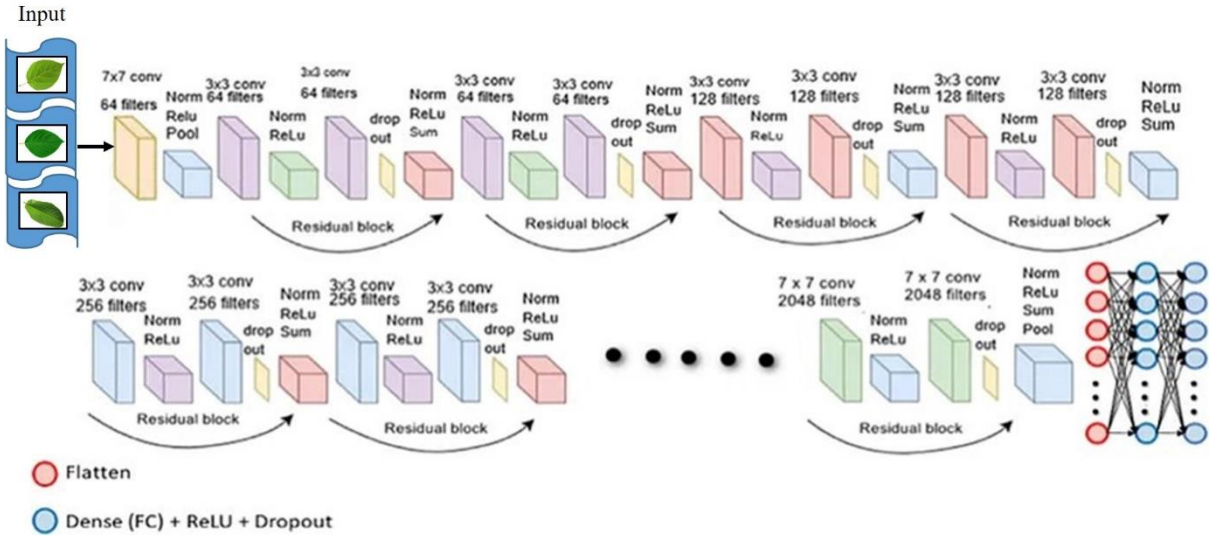


Figure 5- Original RESNET Architecture (He et al. 2016)

3. Proposed Methodology

The proposed hybrid classification framework comprises four main phases: (1) the preprocessing stage, (2) feature extraction, (3) late feature fusion, and (4) model validation, as depicted in Figure 6. In the preprocessing phase, annotations would be made to recognize the region of interest in plant images. Deep features shall be extracted from RoI using two lightweight architectures, CNNs, (1) ENN (version 1) and SEN. A self-attentive mechanism is applied on both feature sets to capture dependencies among different sub-regions in the image. These optimized features are merged using a late feature fusion technique. Finally, two dense layers are followed by a SoftMax layer, which classifies the leaves based on their imaging patterns. Five-fold cross-validation is implemented to measure the framework's performance, and the results are recorded further for analysis. The workings of each phase are discussed in detail in the subsequent subsections.

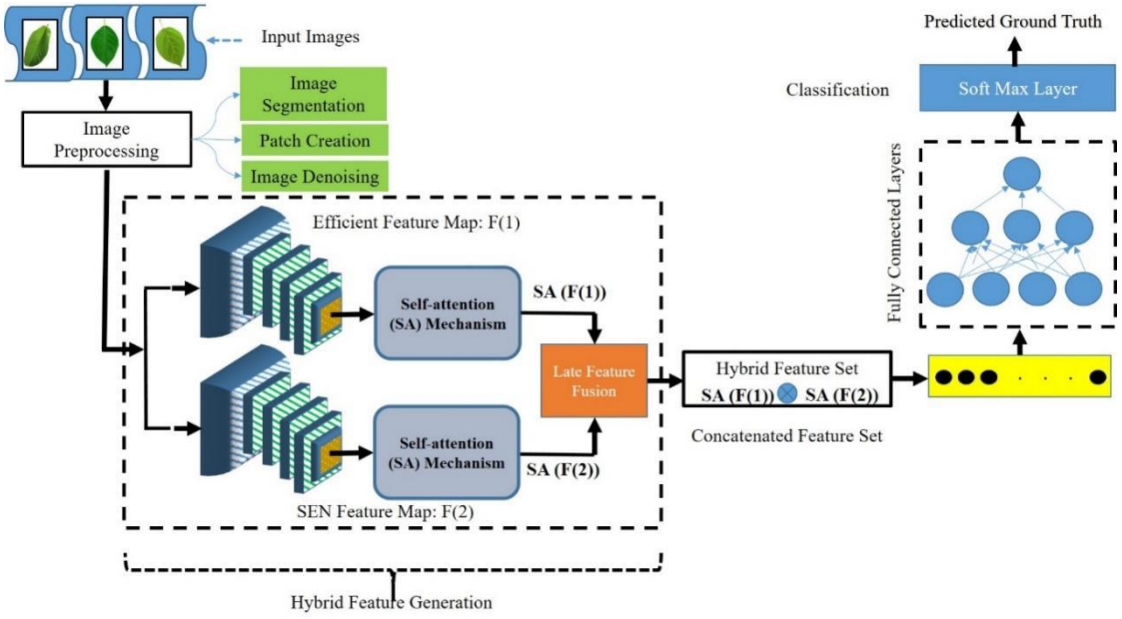


Figure 6- The proposed Leaf classification network

3.1. Data preprocessing

For the first approach, we employed the standard architecture of U-Net developed for image segmentation (Ronneberger et al. 2015) (Figure 7). Further, we fixed a suitable patch size and applied the customized BM3D filtering technique to reduce the noise level in the input leaf images (Chen et al. 2010). The method entails partitioning the input image into overlapping small blocks and continuing to scan for similar blocks within a localized neighborhood. Metrics of Euclidean distance or structural similarity index estimate the disparity of the block. In our work, we set the patch size at 224 x 224 and utilized the Gradient-Based Structural Similarity (Chen et al. 2006) Index (G-SSIM) to compute the block similarities. The G-SSIM

formula is found by calculating the similarity between the gradient distributions of the two image blocks. The mathematical definition of G-SSIM is given in Eq. 5.

$$G\text{-}SSIM\ (x,y) = \frac{2*\mu_x*\mu_y+ C_1}{\mu_x{}^2+ \mu_y{}^2+C_1} * \frac{2*\sigma_{xy}+ C_2}{\sigma_x{}^2+ \sigma_y{}^2+C_2} \quad (5)$$

Where:

- μ_x and μ_y are the means of the gradient distributions of image blocks x and y respectively.
- σ_x and σ_y are the standard deviations of the gradient distributions of image blocks x and y, respectively.
- σ_{xy} is the covariance of the gradient distributions of image blocks x and y, respectively,
- $C1$ and $C2$ are small constants added to avoid instability when the denominator is near zero.

The mean, variance, and covariance can be computed with the derivatives of the image blocks. After the process of gradient distribution, they are calculated using the formula for similarity measurement using gradients. The subsequent blocks are grouped to generate a clean waveform for the noisy signals and suppress noise. In the BM3D collaborative filtering technique, similar blocks are jointly processed, which increases denoising performance. The pseudocode of the customized BM3D algorithm is given in Algorithm 1.

(A): Original Samples

(B): Segmented Samples

Figure 7- Samples of original (A) and segmented leaves (B) using U-NET

Algorithm 1: Pseudocode for image denoising using the customized BM3D algorithm

1. **Parameter declaration**
   - PATCH_SIZE = (224, 224), C1 = 0.01, C2 = 0.03, THRESHOLD = 0.8
2. Define Functions:
   - **Noise_reduction(input_image):**
     - Apply BM3D filter to input_image.
     - Return the denoised image.
   - **Divide_into_blocks(input_image):**
     - Divide the input image into blocks of size PATCH_SIZE.
     - Return the blocks.
   - **Calculate**_similarity_index(block1, block2):
     - Calculate the Gradient-based Structural Similarity Index (G-SSIM) between block1 and block2.
     - Return the similarity index.
   - **collaborative_filtering(block1, block2):**
     - Implement BM3D Collaborative Filtering between block1 and block2.
     - Return the denoised block.
3. **Main Function:**
   - **main():**
     - Read the input image.
     - Perform noise reduction using the noise_reduction function.
     - Divide the denoised image into blocks using the divide_into_blocks function.
     - For each block in the blocks:
       - Create an empty list of similar_blocks.
       - For each other_block in the blocks:
         - If other_block is not equal to block:
           - Calculate the similarity index between block and other_block using calculate_similarity_index.
           - If the similarity index is greater than THRESHOLD:
             - Append other_block to similar_blocks.
       - For each similar_block in similar_blocks:
         - Perform collaborative filtering between block and similar_block using the collaborative_filtering function.

3.2. Hybrid feature generation

Feature hybridization is one of the most important parts of machine learning in which one looks to combine existing features or create new ones to enhance model performance. This process helps identify complex, non-linear relationships between input features and the target variable for better predictive accuracy. We have developed two lightweight yet efficient deep neural network architectures: The Efficient Neural Network (ENN) and the Squeeze and Excitation Network (SEN). These architectures correctly identify complex leaf patterns and reduce the risk of overfitting when used in applications with large feature spaces, such as the Plant Village dataset with 38 classes. Detailed descriptions for both architectures are provided in the following subsections.

3.2.1. EfficientNet (B0) architecture

EfficientNet is a state-of-the-art convolutional neural network that attempts to balance the parameters of accuracy and computational cost using compound scaling, which adjusts all dimensions-depth, width, and resolution-simultaneously (Tan et al. (2019). To achieve that, this balancing act with a compound coefficient resulted in great performance at low computational costs. The depth and complex features captured by stacked convolutional blocks include the Efficient Neural Network Version 1, which is slightly larger and better for the optimal floating-point operation per second. B0 has been chosen for its reduced computational cost and faster processing, giving it an edge in making it suitable for real-time applications. The core component of it is the use of squeeze and excitation optimization within the MBConv layer. MBConv is roughly analogous to the inverted residual blocks in MobileNet v2, allowing skip connections between convolutional blocks' input and output. It begins with a 1x1 convolution to double the number of feature maps, then depthwise and pointwise convolutions to reduce the channel numbers to 3x3. Such skip connections connect the thinner layers, and thicker layers are stacked between skips. Its architecture is shown in Figure 8.

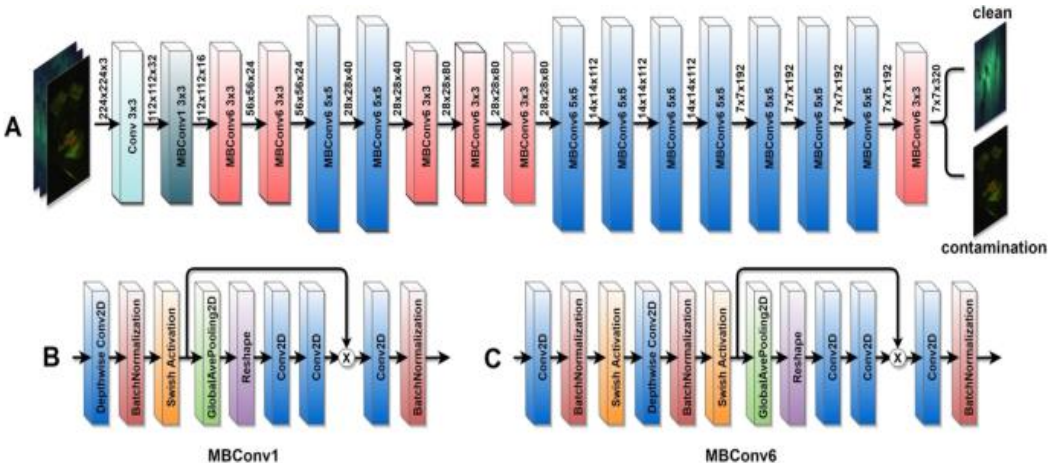


Figure 8- The concise representation of the EfficientNet-B0 model

3.2.2. Squeeze and excitation network

The SE networks introduce a novel architectural component enhancing the representative capacity of convolutional neural networks by adaptively recalibrating channel-wise feature responses (Hu et al. 2018). The SE block works under two steps: squeeze and excitation. In the squeeze step, global average pooling is applied to the feature maps; this condenses spatial information into channel descriptors, capturing the global context of feature activation. Then is the excitation step, where the squeezed outputs are passed through a fully connected layer to create channel-wise scaling factors like those of the sigmoid activation function. These scaling factors are then used to modulate the original feature maps, thus amplifying the informative features and suppressing the less relevant ones. This has enabled better learning of complex patterns by improving feature discrimination and achieved performance benefits in a large class of tasks, such as image classification, object detection, and semantic segmentation. It has been demonstrated that SE networks indeed enhance the accuracy of the model while at the same time retaining computational efficiency, making them a breakthrough improvement over deep learning methodologies. The architecture of the SE block is shown in Figure 9.

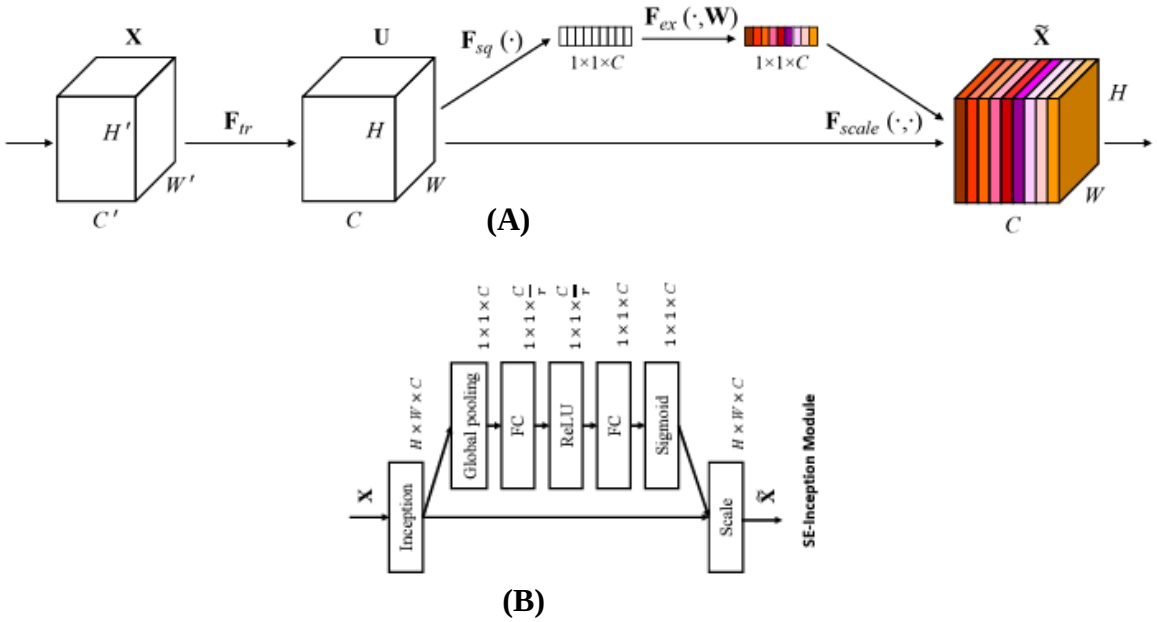


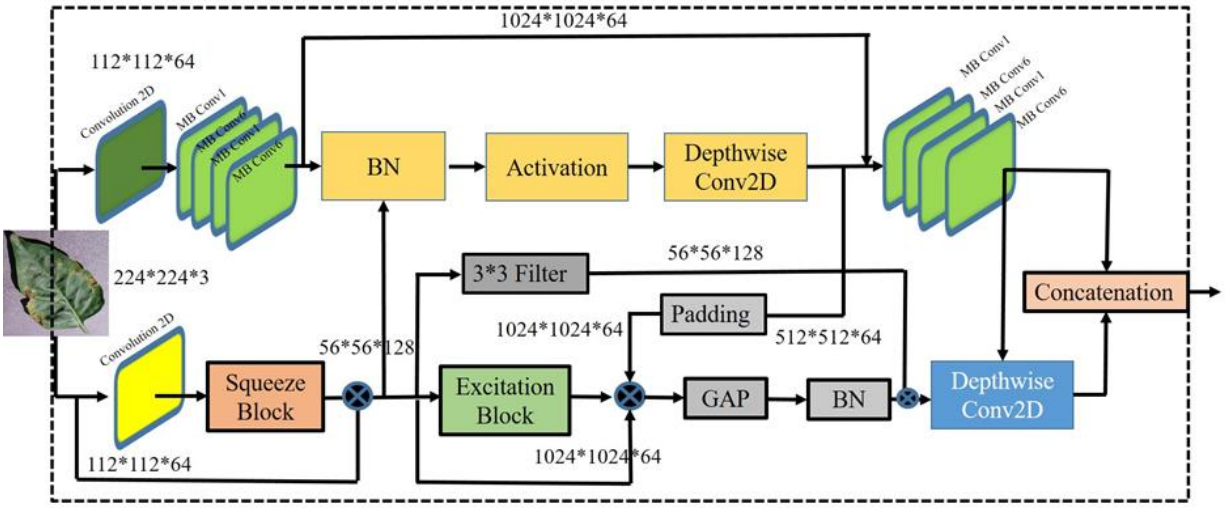
Figure 9- A Squeeze-and-Excitation block (A) and SE-Inception module (B). Both images are directly taken from the original research paper published in 2018 (Hu et al. 2018)

3.2.3. Feature hybridization

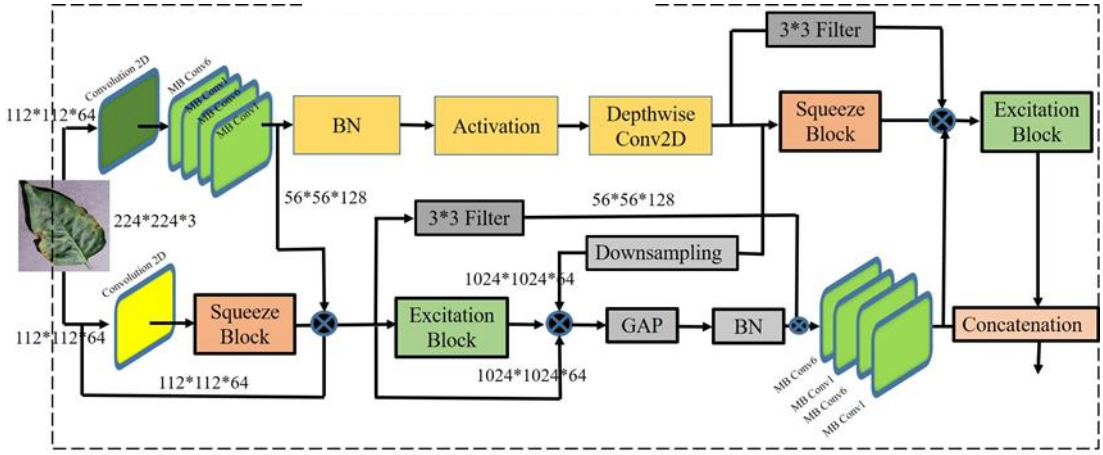
Feature hybridization forms the central module in our classification framework, utilizing two newly developed deep, dense modules: Hybrid Block 1 and Hybrid Block 2, as depicted in Figures 10 (A) and 10 (B). These modules amalgamate differently selected modules from both the Squeeze-and-Excitation architecture and EfficientNetB0 architecture to grasp the spatial locality of both diseased and healthy patterns of the leaves. This hybrid block 1 takes an input 3D tensor: 224 × 224 × 3, which interprets as an RGB image. The working of this module is initiated by using a 2D convolution layer with the kernel size 112 × 112 and 64 output channels. Then, normalization and ReLU of the non-linearity activation function apply. The spatial attention mechanism recalibrates the feature maps as added above. It gives a combined output of size 1064 ×1024×64 resulting from the Squeeze Block and the Weighted Bottleneck (WB) Conv1 block, elaborated as a parallel path of many convolution layers and concatenation.

Output from the Excitation Block is tensor size 1024 × 1024 × 64, which is processed through a 3 × 3 convolutional filter by batch normalization and followed by a non-linear activation function. It adds zeros to one side and one corner for spatial dimensions, followed by a GAP layer that reduces spatial dimensions but pools features across depth. After a BN step, a Depthwise Conv2D layer further operates on feature maps. This Depthwise Conv2D output is added to the input feature maps of size 112 × 112 × 64 and fed into another Depthwise Conv2D operation. In Hybrid Block 2, two input tensors are employed. The first is an RGB image of 224 × 224 × 3. The second is a size 112 × 112 × 64 feature map from the previous processing stage.

The main path takes an input of 224 × 224 × 3 and processes it with a 2D convolution with a 112 × 112 kernel and 64 output channels. The weighted bottleneck WB conv blocks are then utilized after batch normalization, and a ReLU activation function is used for dimensionality reduction. Finally, 56 × 56 × 128 is used, and the output from these WB blocks goes through a convolution operation of size 3 × 3. Outputs of the primary and auxiliary paths are concatenated along the channel dimension. The resulting tensor is passed through a 3 × 3 convolutional layer with a Filter Block. Another mechanism is applied, such as an attention mechanism for feature recalibration. Downsampling is done with a pooling convolution layer so that the output comes out to be a tensor of 56 × 56 × 128. GAP operation reduces the spatial dimensions while keeping the channel dimensions, followed by a batch normalization layer on GAP output. This output from the above norm-0 tensor is then merged with the 64 x 112 x 112 channel input tensor through concatenation along the channel dimension. The merged tensor is then forwarded to subsequent layers or blocks in the neural network's architecture.



(A). Hybrid Block 1



(B). Hybrid Block 2

Figure 10- The architecture of hybrid blocks 1 (A) and 2 (B). Here, BN indicates "Batch Normalization," and GAP refers to "Global Average Pooling."

3.3. Ladder-shaped CNN classification network

Our proposed multiclass classification architecture is a ladder-shaped CNN architecture with an alternating sequence of 14 modules with Hybrid Block 1 and Hybrid Block 2, as shown in Figure 11. As mentioned, these blocks comprise convolutional layers, batch normalization, activation functions, squeeze-and-excitation blocks, downsampling, and concatenations. The initial layers process the input, a set of leaf images, in parallel to be fed through the 14-module sequence: hybrid Block 1 and 2 alternates to allow multi-scale feature extraction and hierarchical representation within this architecture. The architecture alternates between consecutive Hybrid Block 2 modules with Hybrid Block 1 in between and repeats this pattern throughout all 14 modules. It improves multi-scale feature extraction and hierarchical learning, enhancing overall architecture's abilities. It follows the final module by passing information to a SoftMax layer that returns a probability distribution for multiclass classification. Hybrid blocks based on attention mechanisms, residual connections, and multi-scale processing enable critical feature usage in improving the classification of leaves as diseased or healthy. Algorithm 2 describes the pseudocode of the proposed classification algorithm.

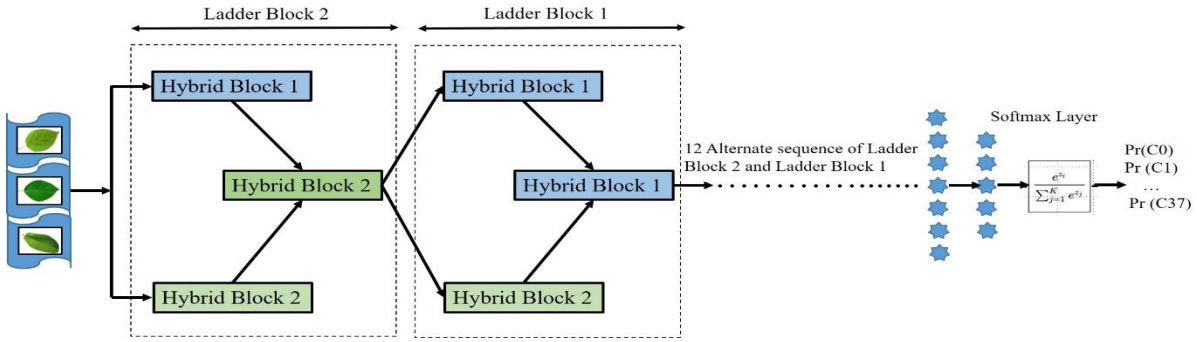


Figure 11- Proposed Classification Framework Consisting of Hybrid Blocks 1 and 2.

Algorithm 2: Pseudocode of proposed leaf classification approach

#Input: Set of leaf images Output:
#Output: Type of leaf
Comment 1: # Load and preprocess the leaf images
1. leaf_images = load_images(image_paths); leaf_images = preprocess_images(leaf_images)
Comment 2: # Initialize the CNN model
2. model = initialize_cnn_model()
***Comment 3*:** # Pass the input images through the initial layers/blocks
3. conv_1 = conv2d(*leaf_images*, *kernel_size*=3, *filters*=64, *padding*='same', *activation*='relu')
4. bn_1 = batch_normalization(conv_1)
5. max_pool_1 = max_pooling2d(*bn_1*, *pool_size*=2, *strides*=2)
6. initial_features = max_pool_1
Comment 4: # Define the sequence of hybrid block modules
7. ***for*** module_idx in range(14):
8. ***if*** module_idx % 3 == 0:
Comment 5: :# Two consecutive Hybrid Block 2 modules
9. features = hybrid_block_2(features); features = hybrid_block_1(features)
Comment 6: # Followed by a Hybrid Block 1 module
10. features = hybrid_block_1(*features*)
else:
Comment 6: # Single Hybrid Block 2 module
11. features = hybrid_block_2(*features*)
Comment 7: # Followed by a Hybrid Block 1 module
12. features = hybrid_block_1(*features*)
Comment 8: # Pass the final features through the SoftMax layer
13. flatten = flatten_layer(*features*)
14. dense = fully_connected_layer(*flatten*, *MB*=128, *activation*='relu')
15. dropout = dropout_layer(*dense*, *rate*=0.5)
16. logits = fully_connected_layer(*dropout*, *MB*=2, *activation*='softmax')
Comment 9: # Compute the predictions (Diseased or Healthy)
17. predictions = *argmax*(logits, axis=-1)
18. ***return*** predictions
Comment 10: # Define the Hybrid Block 1 function
Function 1: def hybrid_block_1(input_tensor):
19. conv_1 = conv2d(input_tensor, kernel_size=3, filters=64, padding='same', activation='relu')
20. bn_1 = batch_normalization(conv_1); act_1 = activation_layer(bn_1, activation='relu')
21. depth_conv_1 = depthwise_conv2d(*act_1*, *kernel_size*=3, *depth_multiplier*=1, padding='same', activation='relu')
22. squeeze_1 = squeeze_block(*depth_conv_1*); excite_1 = excitation_block(*squeeze_1*)
23. filter_1 = conv2d(*excite_1*, *kernel_size*=3, *filters*=64, *padding*='same')
24. bn_2 = batch_normalization(filter_1); act_2 = activation_layer(*bn_2*, *activation*='relu')
25. output_tensor = act_2
26. ***return*** output_tensor
Comment 11: # Define the Hybrid Block 2 function
Function 2: def hybrid_block_2(*input_tensor*):
27. conv_1 = conv2d(*input_tensor*, *kernel_size*=3, *filters*=128, *padding*='same', *activation*='relu')
28. bn_1 = batch_normalization(conv_1); act_1 = activation_layer(bn_1, activation='relu')
29. depth_conv_1 = depthwise_conv2d(*act_1*, *kernel_size*=3, *depth_multiplier*=1, *padding*='same', *activation*='relu')
30. squeeze_1 = squeeze_block(*depth_conv_1*)
31. filter_1 = conv2d(*squeeze_1*, *kernel_size*=3, *filters*=64, *padding*='same')
32. excite_1 = excitation_block(filter_1); downsample_1 = max_pooling2d(*excite_1*, *pool_size*=2, *strides*=2)
33. gap_1 = global_average_pooling2d(*downsample_1*); bn_2 = batch_normalization(gap_1)
34. concat_1 = concatenate(*bn_2*, *input_tensor*, *axis*=-1)
35. output_tensor = concat_1
36. ***return*** output_tensor

4. Experimental Setup and Results

This section presents the experimental analysis and demonstrates the effectiveness of the proposed solution. All experiments, including feature extraction, were conducted using Python on a PC with Windows 10, Intel i5-3.6 GHz CPU, and 8 GB of RAM. We evaluated the performance of our algorithm against state-of-the-art methods based on five criteria: (1) Average Classification Accuracy (ACA), (2) Precision, (3) Recall, (4) F-score, and (5) Time/Space complexity. ACA reflects the model's ability to classify various leaf types across multiple classes correctly. Precision measures the reliability of the model's positive predictions, defined as the ratio of True Positives (TP) to total positive predictions. It can be calculated using Equation 6.

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

True Positive (TP) and False Positive (FP) refer to how many cases the model correctly classifies as positive and the number of occurrences the model misclassifies as positive when they are negative.

Recall or sensitivity is an evaluation metric that depicts the system's ability to classify positive instances correctly. It refers to the ratio of the correct positive predictions made by the model to the total number of positive cases in the dataset. Mathematically, recall is calculated as:

$$Recall = \frac{TP}{TP+FN} \tag{7}$$

F-score or F1 score is the harmonic mean of precision and recall. It is one of the most efficient and widely used evaluation metrics among other evaluation metrics. These measures of precision and recall balance each other out in some cases. The F-measure is applied significantly when the number of positive examples far outweighs the number of negative examples in the dataset. Mathematically, the F-score is calculated as:

$$F = \frac{2*Precison*Recall}{Precison+Recall} \tag{8}$$

In machine learning, time complexity refers to how much a model scales with computational resources in performing tasks such as prediction or other operations. Generally, this is assessed regarding the computational cost incurred during training and the predictive cost incurred while inferring. Our work utilized a five-fold cross-validation methodology that split the respective datasets into training, validation, and testing sets for evaluation. The model was trained on the training set and fine-tuned using hyperparameters on the validation set. The trained model was tested against those performances for proper classification accuracy and other measures. The performance results on both datasets are summarized in the following subsections.

4.1. Hyperparameter Tuning Using Bayesian Optimization

Bayesian optimization is used to tune the hyper-parameters of the proposed deep learning model (Victoria et al. 2021). A Gaussian process was used as the surrogate model to estimate the performance of different hyperparameter configurations. In the Expected Improvement (EI) acquisition function, exploration and exploitation of new and known promising regions are balanced. The primary objective was to achieve maximum accuracy, and other performance metrics such as validation loss, precision, recall, and F1-score were also monitored during training. This ensured that the model performed well in accuracy and showed robustness and generalization when used with other performance indicators. The list of optimal hyperparameters is given in Table 1.

Table 1- Detail of hyperparameter search space and their optimal values during training

Hyperparameter	*Search Space*	*Optimal Value*
Learning rate (α)	0.0001 – 0.01	0.001
Batch Size	16 – 128	64
Dropout Rate	0.1 – 0.5	0.3
Number of Layers	2 – 10	6
Number of Neurons	64 – 512	256
Weight Decay	0.00001 – 0.001	0.0001

4.2. Experiment 1: Results on plant village dataset

The performance of the proposed model is validated on the Plant Village Dataset using a five-fold cross-validation technique. The multiclass classification results, demonstrating the model's effectiveness across various plant disease categories, are presented in Table 2. The classification results on the Plant Village dataset point out excellent performance in most categories, which can then be generalized into three accuracy levels: above 99%, between 90% and 99%, and below 90%. Classes with accuracy Above 99% classify high performance by almost perfect differentiation. For instance, Cherry (Powdery Mildew) and

Cherry (healthy) attain 100% precision, recall, and an F-score value of 1.00. Grape (Leaf Blight) and Peach (healthy) were no exceptions in accomplishing the 100% accuracy line that guarantees no false negatives or false positives would be counted. The Orange class reached an Accuracy of 99.16% with a precision of 1.00, recall of 0.99, and F-score of 0.99; more evidence that the model indeed makes good predictions regarding this disease.

Classes with accuracy Between 90% and 99% dominate the results. Here, the model performs very well for the entire range of plant conditions. Apple (healthy) had an accuracy of 98.24%, with 0.98 precision, 0.99 recall, and an F-score of 0.99. This means that the model classifies healthy apples as being robust. Corn Common Rust achieved 96.54% accuracy at precisions and recalls of 0.98 and 0.97, respectively. The F-score for the model is 0.97, meaning it is very reliable for this condition. Grape, Black Rot results came back with 93.18% accuracy, 0.92 precision, 0.97 recall, and 0.95 F-score, showing somewhat lower performance but still quite good. The accuracies are not as high as those of the top-performing classes. Classes with accuracy below 90% represent a small portion of the dataset where the model faces challenges. For instance, Apple (Black Rot) showed an accuracy of 90.00%, with 0.92 precision, 0.90 recall, and a 0.91 F-score, indicating that while the model performs respectably, overlapping features with other classes might lead to slight misclassifications. Similarly, Tomato (healthy) had an accuracy of 88.42%, with 0.75 precision, 1.00 recall, and a 0.98 F-score, suggesting that the model struggles slightly with precision but compensates with strong recall. These lower accuracy classes point to areas where additional tuning or training data might be needed to improve model performance.

Table 2- Different performance measures for the PlantVillage dataset

Class	*Precision*	*Recall*	*F-Score*	*Support*	*Accuracy*
Apple___Apple_scab	0.97	0.92	0.95	126	96.20
Apple___Black_rot	0.92	0.90	0.91	108	97.40
Apple___Cedar_apple_rust	0.98	0.95	0.96	55	97.23
Apple___healthy	0.98	0.99	0.99	392	98.24
Blueberry___healthy	0.98	0.98	0.98	300	98.02
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	100	100
Cherry_(including_sour)___healthy	0.99	1.00	0.99	264	99.66
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.80	0.84	0.82	184	84.34
Corn_(maize)__Common_rust	0.98	0.97	0.97	292	96.54
Corn_(maize)___Northern_Leaf_Blight	0.93	0.96	0.95	318	93.41
Corn_(maize)___healthy	0.98	0.99	0.98	341	97.66
Grape___Black_rot	0.92	0.97	0.95	234	93.18
Grape___Esca_(Black_Measles)	0.98	1.00	0.99	138	98.44
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	215	100
Grape___healthy	0.99	1.00	0.99	85	99.16
Orange___Haunglongbing_(Citrus_greening)	1.00	0.99	0.99	1102	99.16
Peach___Bacterial_spot	0.90	1.00	0.95	462	92.14
Peach___healthy	1.00	1.00	1.00	72,	100
Pepper,_bell___Bacterial_spot	0.99	1.00	0.99	147	99.14
Pepper,_bell___healthy	0.95	0.95	0.95	388,	98.23
Potato___Early_blight	0.94	0.98	0.96	200	95.54
Potato___Late_blight	0.94	0.98	0.96	207	93.14
Potato___healthy	0.98	1.00	0.99	192	96.66
Raspberry___healthy	0.97	0.99	0.98	78	93.34
Soybean___healthy	1.00	1.00	1.00	508	100
Squash___Powdery_mildew	0.99	0.99	0.99	330	98.74
Strawberry___Leaf_scorch	0.90	0.99	0.95	110	92.14
Strawberry___healthy	0.96	0.94	0.95	234	89.13
Tomato___Bacterial_spot	0.97	0.96	0.92	425	91.33
Tomato___Early_blight	0.90	0.94	0.95	98	95.33
Tomato___Late_blight	0.95	0.96	0.97	192	90.47
Tomato___Leaf_Mold	0.97	0.98	0.97	154	91.66
Tomato___Septoria_leaf_spot	0.97	0.98	0.93	190	92.34
Tomato___Spider_mites Two-spotted_spider_mite	0.94	0.93	0.98	318	98.21
Tomato___Target_Spot	0.99	0.97	0.94	869	93.54
Tomato___Tomato_Yellow_Leaf_Curl_Virus	0.94	0.93	0.99	854	91.12
Tomato___Tomato_mosaic_virus	0.99	0.99	0.86	373	91.24
Tomato___healthy	0.75	1.00	0.98	93	88.42

The accuracy vs. epochs graph shown in Figure 12 represents the performance of the proposed approach during training and validation in 35 epochs, showing critical behaviors in the model's learning process. In the early epochs, both training and validation accuracies sharply rise, which is expected since, at that point, the model is just starting its training and sensing patterns from the data. Rapid increases in training accuracy suggest that a model fits quite rapidly with the training data. Validation accuracy also learns much faster than the training set, suggesting generalization to unseen data by the model. After epoch 5, the

validation accuracy starts to drop steeply while the training accuracy continues to climb increasingly smoothly. That is a sign of overfitting, where the model begins to memorize specific patterns unique to the training data, reducing its ability to generalize to new data. The validation accuracy does see this drop, but the oscillations indicate instability in the model's generalization capability at this stage. As the training progresses, both curves stabilize, especially after epoch 15. The training accuracy tracks into a plateau around a high value, suggesting that the model had nearly maximized its performance on the training set. Validated accuracy stabilizes, too, but suffers more variability from the training curve of the model, indicating that although the model's generalization ability is enhanced, it remains sensitive to the fluctuations within the validation data set. It seems more of a converging model but sometimes overfits or displays extreme sensitivity precisely to specific validation samples on later epochs.

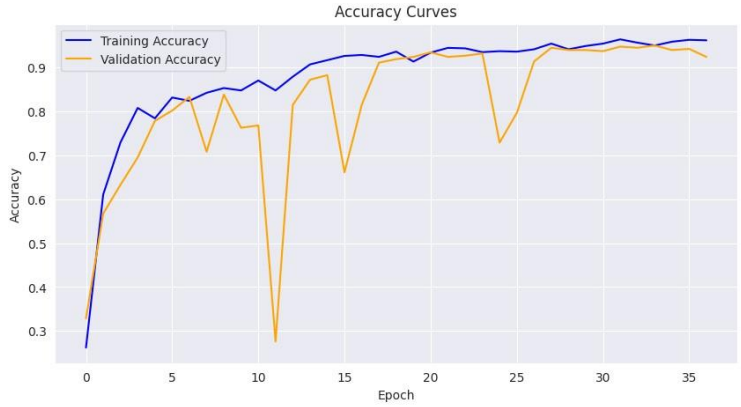


Figure 12- The Accuracy vs Epoch curve for training and testing of the proposed approach

The training and validation loss for 35 epochs are shown in **Figure 13**. The primary movement in the early epochs in the training data is the drastic drop in the training and validation loss curves, showing that the model learned well and substantially reduced its errors. While the training loss decreases as the training progresses, it indicates better generalization by the model on the training data. Instead, the validation loss often fluctuates and displays instability in generalization. It peaks at a significant validation loss around epoch 10, indicating overfitting. That means the training set error is correctly lowered at this epoch but fails to generalize when attracted to the unseen validation data. Later, after epoch 15, the training loss drops to almost zero, marking an indicator of convergence towards the model's training data. The validation loss also keeps reducing in value gradually, though with occasional small spikes. These spikes, particularly around epochs 20 and 25, suggest that even though the model performs well, it is still sensitive to specific validation batches.

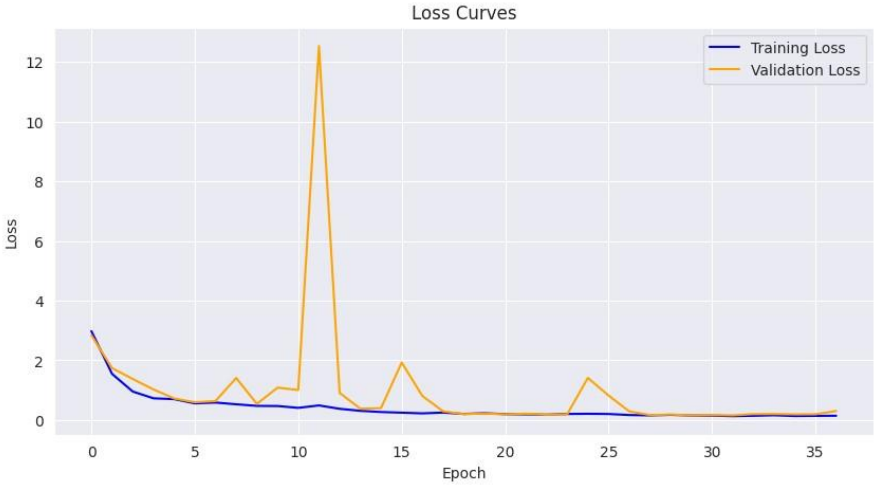


Figure 13- The Loss vs Epoch curve for training and testing of the proposed approach

The Receiver Operating Characteristics (ROC) curve for the proposed classification is shown in Figure 14. Each line in the plot is the ROC curve for a given class and is plotted against the false positive rate versus the true positive rate for a range of classification thresholds. The curves indicate how well the classifier can distinguish between classes. In this curve, almost all individual class curves reach the top-left corner nearly perfectly, except for fewer than five classes, while most classes have AUC values close to 1.00. There are three classes with AUC values below 1.00: Class 37, with an AUC of 0.98, and Class 36 and Class 35, each with an AUC of 0.99. All of them give perfect classification performance; however, their AUC values indicate some degradation in distinguishing a positive instance from a negative one compared to other classes that scored ideally and had an AUC of 1.00. In other words, the model has a perfect discrimination ability for those classes.

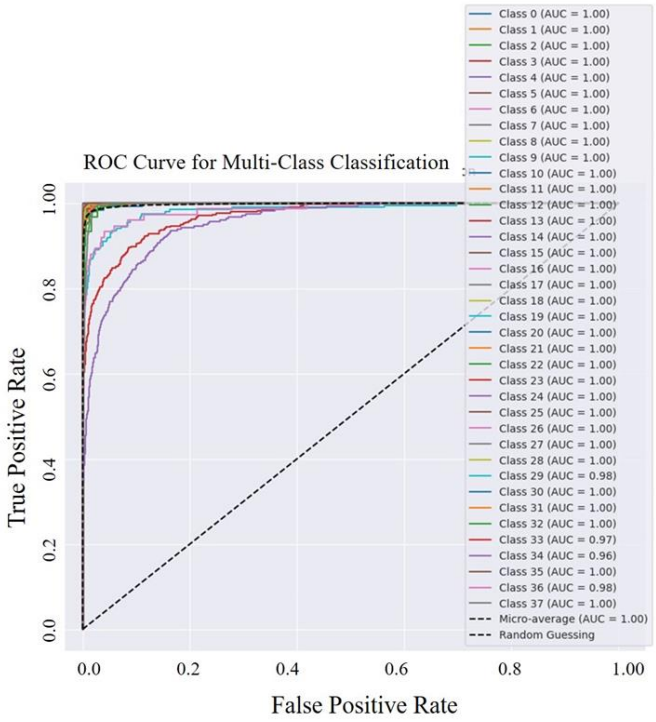


Figure 14- ROC curve for Multiclass Leaf Classification Model

4.3. Experiment 2: Results on rose leaf disease dataset

Similar to the PlantVillage dataset, the Rose leaf disease dataset is also a multiclass dataset having three classes: (1) Healthy_Leaf_Rose, (2) Rose_Rust, and (3) Rose_swafly_Rose_slug. The classification results for all three classes are shown in Table 3. For Healthy_Leaf_Rose, the model has a value of 0.99 precision, which means 99% of all the predicted instances for this class are correct. However, the recall of this class is a little low, with just 5% being missed. As a result, the class F1-score is 0.97, balancing strength in precision and slightly lower recall from showing a very high classification accuracy for this class. For Healthy_Leaf_Rose, 995 samples in the dataset give this model an accuracy of 95.38%, high though a bit less strong than the performance observed for the other two classes.

For Rose_Rust (Class 1), the model achieves a precision of 0.98, indicating that almost all predicted instances for this class are correct. The recall is also strong at 0.97, meaning that 97% of the true Rose_Rust instances are correctly identified. The F1-score, at 0.98, indicates excellent overall performance for this class, with a strong balance between precision and recall. Out of 991 samples in the dataset, the model achieves an accuracy of 97.47%, slightly better than Healthy_Leaf_Rose. This model performs very well for Rose_swafly_Rose_slug with Class 2, having recall at 0.98, or almost all true class instances are correctly detected. Precision is slightly lower than others at 0.94, showing more false positives for this class. F1-score is still very good at 0.96, showing a good balance between precision and recall. For this model, with 996 samples, class-specific accuracy shows the maximum at 98.61%, indicating that it is a strong classifier for detecting this class. The model's average accuracy for the classes is 97.15%. The model's effectiveness in classifying each category is high, though its performance for all classes is excellent. Despite a minor reduction in precision, it is best with its highest accuracy in the case of Rose_swafly_Rose_slug (Class 2). This means the model is pretty good at distinguishing between classes and is only sometimes confused, primarily in the case of Rose_swafly_Rose_slug and other categories.

Table 3- Different performance measures for the Rose Leaf dataset

Class	*Precision*	*Recall*	*F1-Score*	*Support*	*Accuracy*
Healthy_Leaf_Rose (0)	0.99	0.95	0.97	995	95.38%
Rose_Rust (1)	0.98	0.97	0.98	991	97.47%
Rose_swafly_Rose_slug (2)	0.94	0.98	0.96	996	98.61%
Average Accuracy	-	-	-	2982	97.15%

The confusion matrix shown in Figure 15 demonstrates how the model performs on the three classes. In class 0, it classified 947 correctly and mislabeled 48 instances as class 1, showing a moderate level of confusion between these two classes. The confusion was minimal since there were no misclassifications from class 0 to class 1, indicating a clear distinction between the two categories. For class 1, 962 cases were correctly classified, with only ten misclassifications as class 0 and nineteen misclassifications as class 2. It shows that although the model is excellent, there is interference between class 1 and the remaining

classes but with the slightest interference with class 2. The model accurately classified 976 instances and misclassified 20 of class 2 as class 1. For class 2, the model correctly classified 976 cases, with 20 misclassifications as class 1. However, no instances of class 2 were misclassified as class 0, indicating a strong ability to distinguish class 2 from class 0.

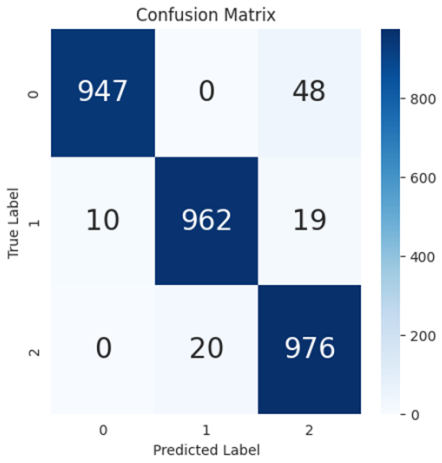


Figure 15- Confusion matrix for the Rose Leaf dataset

The graph shown in Figure 16 indicates the performance of the proposed model in training the Roseleaf dataset over 100 epochs and compares it with the testing accuracy. During training, the model steadily increases in both metrics and reaches a plateau after about 80 epochs. The testing accuracy remains stable at about 97.15%, implying the model's generalization capability is high. Here, a similar trajectory was seen between the training and testing curves, with negligible divergence between them. This indicates that the model did not suffer from an overfitting problem as the gap between training and testing accuracy continued to narrow throughout the process. In terms of convergence, the model achieves smooth progression, with significant improvements in accuracy observed during the first 40 to 50 epochs. After reaching around 60 epochs, the learning rate appears to slow, stabilizing accuracy between 95% and 97%. This trend indicates diminishing returns from further training. The final test accuracy of 97.15% shows that the model performs exceptionally well in its classification performance, demonstrating good robustness and reliability on the Roseleaf dataset. The near identity of the training and testing accuracy also indicates that the model generalizes well to unseen data without glaring signs of overfitting, as the two accuracies converge closely towards the last epochs.

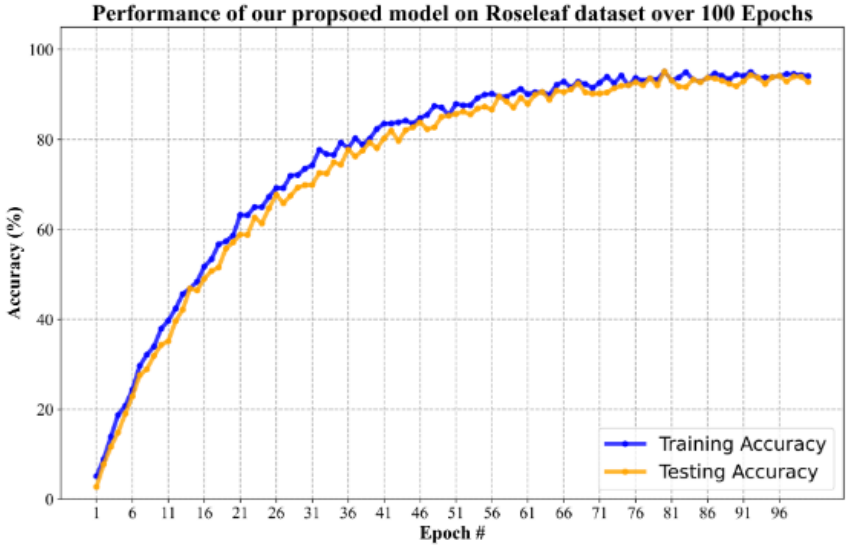


Figure 16- The Accuracy vs Epoch curve for training and testing of the proposed approach

Figure 17 represents the training and validation loss for a range of epochs in the training of our model. The blue curve shows the training loss, and the orange curve refers to the validation loss. During the early epochs in the learning process, the training loss and validation loss are relatively high, and the model hasn't yet entirely fitted the data well. Gradually, as the number of epochs increases, the training loss decreases continuously, showing the model is learning and minimizing errors in the training data. However, the validation loss has a higher variance. Between epochs 1 and 4, the validation loss drops steeply, indicating that the model first generalizes well to the unseen data. After that point, the validation loss peaks around epoch 6, rising sharply before dropping again. Such fluctuations demonstrate the likelihood of overfitting: the model is doing very well in training but

fails with the validation set. From approximately epoch ten onwards, the validation loss increases again- the increase could indicate divergence in training loss. It might be due to overfitting because the model performs worse on unseen data (validation set) but keeps improving on training data. The upward drift in both curves at epochs later than 12 indicates the requirement of early stopping or even some other regularization technique that would prevent overfitting to the training data.

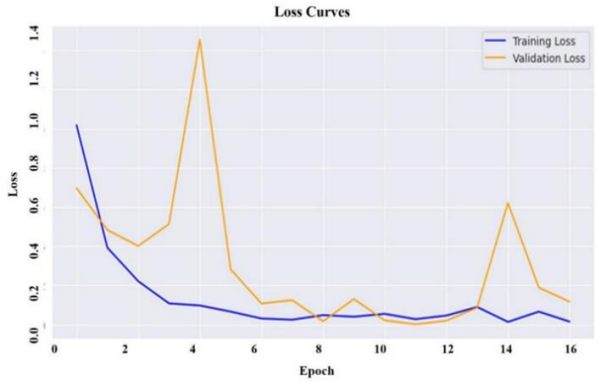


Figure 17- The Loss vs Epoch curve for training and testing of the proposed approach

The ROC for the proposed multiclass classification model is shown in Figure 18. Class 0's AUC value reaches 0.999; thus, the classifier almost perfectly differentiates instances of this particular class from others. Class 1 stands out with an AUC of 1.000, signifying flawless classification performance, meaning the model can perfectly distinguish Class 1 instances. Similarly, Class 2 has an AUC of 0.999, reflecting near-perfect classification. Also, the micro-average AUC of 0.999 aggregates performance across classes, which is another facet that proves the model can predict with high accuracy for many classes. The curves are tightly concentrated in the top-left corner, demonstrating high TPR with very low FPR, meaning the model achieves a strong balance between sensitivity and specificity.

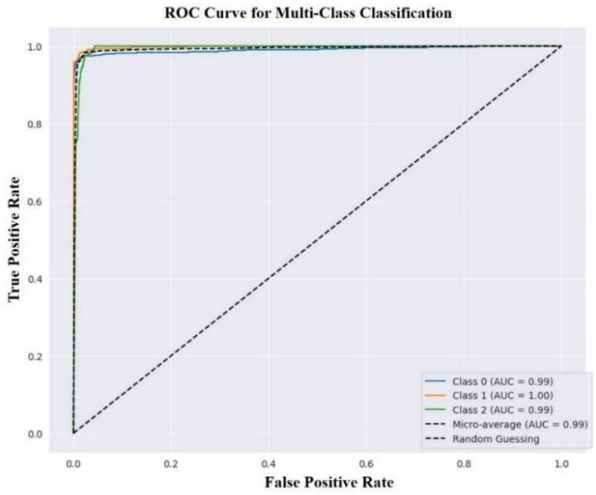


Figure 18- ROC curve for Multiclass Leaf Classification Model

4.4. Results Comparison with State-of-the-Art Methods

Its performance was compared against five baseline models: (1) the VGG16 Net, (2) Inception Net, (3) RESNET-50, (4) Conventional Inception-Visual Geometry Group Network (CIVGGN), and (5) EfficientNetV2. All the above networks were pre-trained on the Imagenet data set and have widespread medical imaging and leaf disease detection applications. The comparison on the PlantVillage dataset focused on computational efficiency and key metrics like accuracy, precision, recall, and F1-score. Our suggested approach outperforms VGG16 Net (93.5%), Inception Net (94.2%), RESNET-50 (95.1%), conventional Inception-Visual Geometry Group Network (CIVGGN) (95.64%), and EffecientNetV2 (95.18%) with a classification accuracy of 96.4% (Table 4). In Figure 22, we compare the relative performance with our approach. The results show a relative improvement of 3.1% over VGG16 Net, 2.2% over Inception Net, 1.3% over RESNET-50, 0.8% over CIVGGN, and 1.23% over EfficientNetV2.

Here, the VGG16 Net model demonstrates a poor performance mainly due to its extensive number of parameters, which process computationally very slowly and multiply memory consumption considerably. It makes it inefficient at training and

inference periods and less desirable for high-performance applications, requiring faster speed and information processing. The Inception Net architecture, though known for its parallel convolutional layers and efficient handling of features, may not perform as well in capturing complex hierarchical features compared to newer models. While it offers better accuracy than VGG16, its ability to scale for highly intricate tasks is somewhat limited. The third baseline model, RESNET-50, is known to have robust architectures. Using residual connections mitigates the vanishing gradient problem; however, it is intense and computationally expensive. Though increased depth add more accuracy to the model, its overhead on its efficiency has been a problem since real-time applications require quick response times. The CIVGGN model, which combines elements from both Inception and VGG architectures, introduces a level of complexity that doesn't always translate to superior performance. The hybrid nature of the model can result in suboptimal handling of specific tasks, particularly when compared to models designed explicitly for efficient feature extraction and classification. Although compound scaling optimizes parameters, depth, and resolution for EfficientNetV2, sometimes the trade-off between complexity and performance may be slightly required.

To validate the superiority of our proposed approach, we performed a *p*-value analysis to compare its classification accuracy with that of widely used models. The accuracy of classification in our proposed method is significantly more significant compared to VGG16Net (93.5%) with a *p*-value of 0.002, Inception Net (94.2%) with $p = 0.008$, and RESNET (95.1%) with $p = 0.015$. Even with the latest models in the EfficientNetV2 (95.18%) and Conventional Inception-VGG (95.64%), improvements boast statistical significance with *p*-values of 0.012 and 0.021, respectively. Since all the *p*-values are less than the threshold of 0.05, results confirm that the improvement inaccuracies are not by chance alone, indicating our proposed approach's robustness. This implies our method provides not only greater accuracy but also statistically significant gains over the models already existing.

The proposed approach shows the best execution time performance, with an execution time of 45 MB, much faster than all other models. EfficientNetV2 indicates a relatively lower execution time of 53 MB, thus making it more efficient than conventional models. VGG16Net and Inception Net slightly increase in some time MB, for instance, 50 and 55 MB, respectively. On the other hand, RESNET and Conventional Inception-Visual Geometry Group Network have an execution time of 75 and 83 MB, respectively, which is way higher than the proposed approach, indicating that it requires more computational power and less time-efficient as compared to this newly recommended approach. Regarding memory consumption, it also reflects that the proposed approach is the most efficient due to the utilization of 150 MB, reflecting the lowest memory consumption among all the other models. EfficientNetV2 has 167 MB, balancing accuracy and memory efficiency well. Inception Net and VGG16Net require moderate memory consumption of 170 and 180 MB, respectively. On the other hand, RESNET and Conventional Inception-Visual Geometry Group Network consume the largest MB in memory, which are 190 and 240 MB, respectively, translating to a more significant computer resource requirement and larger space complexity.

Table 4- Performance measures for different baseline methods on the PlantVillage dataset

Classification Framework	*Classification Accuracy*	*Execution Time*	*Memory Consumption (In MB)*
VGG16Net (Simonyan & Zisserman 2014)	93.5%	50	180
Inception Net (Szegedy et al. 2016)	94.2%	55	170
RESNET (He et al. 2016)	95.1%	75	190
Conventional Inception-Visual Geometry Group Network (Pal & Kumar 2023)	95.64%	83	240
EffecientNetV2 (Devi et al. 2023)	95.18%	53	167
Proposed Approach	96.40%	45	150

The results achieved on the Rose Leaf dataset show significant improvement in the ability to classify images and computational efficiency compared to the baseline models (Table 5). The VGG16Net model achieved an accuracy of 93.5% but executes in 48 milliseconds, and its memory consumption is around 121 MB. Although VGG16Net seems relatively accurate, it has many hyperparameters, making it slow for real-time applications as it is computationally expensive. The Inception Net outperformed VGG16Net with an accuracy of 94.2% and reduced running time to 19 milliseconds and memory consumption to 107 MB. This result proves that Inception Net, due to its usage of parallel convolutional layers, made this more efficient by enhancing speed and lower memory usage. However, it attains lower precision compared to more sophisticated models. Here, the testing accuracy is 95.1%, with a memory consumption of 141 MB and an execution time of 33 milliseconds. Its deeper architecture with residual connections enhanced feature extraction but consume more resources and be slower than faster models. The Conventional Inception-Visual Geometry Group Network can achieve a higher accuracy at 96.33% but at the cost of execution time (83 milliseconds) and memory usage (203 MB). Hybrid design combines features from both the Inception and the VGG architectures, making for a heavier load computationally. Therefore, the application needed to be fast and work on fewer resources simultaneously, such as edge and other terminal devices. The EfficientNetV2 balances accuracy with computation efficiency; with both parameters, classification performance is 95.24%, and execution time is 63 milliseconds while using up to 151 MB of memory. This architecture looks good at providing a good performance, but model scaling is still lacking compared with other, even more efficient models. The proposed approach outperforms the other models with the highest classification accuracy of 97.15%, lowest execution time of 13 milliseconds, and moderate memory consumption of 117 MB. This framework manages the complexities of the Rose Leaf dataset well while maintaining a high level of accuracy with fast

processing and optimized usage of resources. Therefore, the supremacy of the proposed model in classification performance and computational efficiency results in its suitability for applications like real-time plant disease detection tasks.

The proposed approach attained 97.15% classification accuracy with a *p*-value of 0.001, statistically significantly better than VGG16Net of 93.5%, Inception Net of 94.2% with $p = 0.004$, and RESNET with $p = 0.012$. Comparing the proposed approach with even more recent models in the literature like EfficientNetV2, which attained 95.24%, and Conventional Inception-VGG, which gained 96.33%, the improvements are still statistically significant with *p*-values of 0.009 and 0.017, respectively. These *p*-values -- all <0.05 -- confirm that the increase in classification accuracy brought by our proposed method cannot be attributed to chance and that, thus, it's robust and performs far better across different models.

Table 5- Performance measures for different baseline methods on the Rose Leaf dataset

Classification Framework	Classification Accuracy	Execution Time	Memory Consumption
VGG16Net (Simonyan & Zisserman 2014)	93.5%	48	121
Inception Net (Szegedy et al. 2016)	94.2%	19	107
RESNET (He et al. 2016)	95.1%	33	141
Conventional Inception-Visual Geometry Group Network (Pal & Kumar 2023)	96.33%	83	203
EffecientNetV2 (Devi et al. 2023)	95.24%	63	151
Proposed Approach	97.15%	13	117

4.5. Research Limitations

Although the proposed leaf classification network performed well on two public datasets, several limitations must be addressed. A list of a few research limitations are discussed below:

1) The alternating hybrid blocks (EfficientNet and Squeeze-and-Excitation networks) complicate the model's design. This complexity can make it harder to fine-tune and balance accuracy with the time taken to process the data.
2) Although SE blocks enhance quality feature extraction, they introduce more calculations. The result is an increase in the training and prediction time by the model, primarily when large databases are run with a low level of computing hardware.
3) The customized BM3D filtering technique removes image noise but takes more processing time than the other methods. It becomes a problem when large images are considered because this method might slow down the entire system's performance.
4) The model is validated only on two datasets used in the study, so it may not perform as well when applied to different datasets without significant modifications. It could limit its flexibility for other types of image classification problems.
5) Adding EfficientNet along with SE blocks adds up the model's memory requirement. It makes it less useful for memory-constrained devices such as cell phones or embedded systems, where excessive memory usage is impossible.

5. Conclusion and Future Scope

In this article, we proposed a new hybrid classification framework that uses the Efficient Neural Network with Squeeze and Excitation Network for leaf classification. Our architecture uses a self-attention mechanism focused on the input scans' most relevant regions of interest. We designed a dense CNN architecture with two hybrid blocks comprising SEN and EfficientNetB0 modules. Such hybrid modules were alternatively utilized in our architectures of CNNs over the 14 modules to achieve better performance. The proposed classification model achieved significant margins of 96.40% on the PlantVillage dataset and 97.15% on the Rose Leaf Disease dataset over VGG16, Inception V3, RESNET-50, Conventional Inception-Visual Geometry Group Network, and EfficientNetV2. The proposed model depicted better accuracy, recall, F-score, memory usage, and execution time in all the compared states. The primary limitation of the proposed CNN model is its "black box" nature due to the complex interconnections among layers and blocks. To address this, techniques such as saliency maps, Grad-CAM, Layer-wise Relevance Propagation (LRP), SHAP, and LIME can be explored to improve interpretability. These methods can help visualize feature importance and enhance transparency.

Furthermore, researchers should attempt to generalize the model by testing it on diverse datasets, including medical imaging and satellite image analysis. Incorporating novel interconnections like Highway Networks, Interleaved Connections, and Residual Dense Connections may improve the model's performance. Various search space optimization methods can also be a practical approach to enhance the overall performance of the proposed model (Tiwari et al. 2021). A few novel interconnections, such as Highway Networks (Srivastava et al. 2015), Interleaved Connections (Li et al. 2020), and Residual Dense Connections (Zhang et al. 2018) can also be used among different layers to enhance the performance of the proposed model.

Credit Authorship Contribution Statement

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

Many data, models, and codes in this work are open source. However, the final outputs are not available to the public, but can be sent to interested researchers.

Funding Declaration

No sponsor or funder supported this study.

Reference

Agarwal G, Belhumeur P, Feiner S, Jacobs D, Kress W J, Ramamoorthi R & White S (2006). First steps toward an electronic field guide for plants. *Taxon* 55(3): 597-610 https://doi.org/10.2307/25065637

Ahmad I, Hamid M, Yousaf S, Shah S T & Ahmad M O (2020). Optimizing pretrained convolutional neural networks for tomato leaf disease detection. *Complexity* 2020(1): 8812019 https://doi.org/10.1155/2020/8812019

Akilan T, Wu Q J, Safaei A & Jiang W (2017). A late fusion approach for harnessing multi-CNN model high-level features. *In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* 566-571. https://doi.org/10.1109/smc.2017.8122666

Ashwinkumar S, Rajagopal S, Manimaran V & Jegajothi B (2022). Automated plant leaf disease detection and classification using optimal MobileNet based convolutional neural networks, *Materials Today: Proceedings* 51(1): 480-487, https://doi.org/10.1016/j.matpr.2021.05.584

Bi C, Wang J, Duan Y, Fu B, Kang J R & Shi Y (2022). MobileNet based apple leaf diseases identification. *Mobile Networks and Applications*, 1-9. https://doi.org/10.1007/s11036-020-01640-1

Bock C H, Barbedo J G, Del Ponte E M, Bohnenkamp D & Mahlein A K (2020). From visual estimates to fully automated sensor-based measurements of plant disease severity: status and challenges for improving accuracy. *Phytopathology Research*, 2, 1-30. https://doi.org/10.1186/s42483-020-00049-8

Chai Ali, C A, Li BaoJu L B, Shi YanXia S Y, Cen ZheXin C Z, Huang HaiYang H H & Liu Jun L J (2010). Recognition of tomato foliage disease based on computer vision technology. *Acta Horticulturae Sinica*, 37(9), 1423-1430

Chen G H, Yang L & Xie S L (2006). Gradient-based structural similarity for image quality assessment. *In 2006 international conference on image processing* 2929-2932. IEEE. http://doi.org/10.1109/ICIP.2006.313132

Chen Q & Wu D (2010). Image denoising by bounded block matching and 3D filtering. *Signal Processing* 90(9): 2778-2783. https://doi.org/10.1016/j.sigpro.2010.03.016

Devi R S, Kumar V R & Sivakumar P (2023). EfficientNetV2 Model for Plant Disease Classification and Pest Recognition. *Computer Systems Science & Engineering* 45(2). https://doi.org/10.32604/csse.2023.032231

Ferentinos K P (2018). Deep learning models for plant disease detection and diagnosis. *Computers and electronics in agriculture*, 145, 311-318. https://doi.org/10.1016/j.compag.2018.01.009

Ghazi M M, Yanikoglu B & Aptoula E (2017). Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing* 235: 228-235. https://doi.org/10.1016/j.neucom.2017.01.018

Guan Z X, Tang J, Yang B J, Zhou Y F, Fan D Y & Yao Q (2010). Study on recognition method of rice disease based on image. *Chinese Journal of Rice Science* 24(5): 497

He K, Zhang X, Ren S & Sun J (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* 770-778. https://doi.org/10.1109/cvpr.2016.90

Hu J, Shen L & Sun G (2018). Squeeze-and-excitation networks. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7132-7141). https://doi.org/10.1109/cvpr.2018.00745

Hu J, Shen L & Sun G (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition 7132-7141. https://doi.org/10.1109/cvpr.2018.00745

Hughes D & Salathé M (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060

Kamilaris A & Prenafeta-Boldú F X (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture, 147, 70-90*. https://doi.org/10.1016/j.compag.2018.02.016

Kawasaki Y, Uga H, Kagiwada S & Iyatomi H (2015). Basic study of automated diagnosis of viral plant diseases using convolutional neural networks. *In Advances in Visual Computing: 11th International Symposium, ISVC 2015*, Las Vegas, NV, USA, December 14-16, 2015, Proceedings, Part II 11 (pp. 638-645). Springer International Publishing. https://doi.org/10.1007/978-3-319-27863-6_59

Koonce B (2021). Convolutional neural networks with swift for tensorflow: Image recognition and dataset categorization 109-123). New York, NY, USA: Apress.

Li F, Cong R, Bai H & He Y (2020). Deep interleaved network for image super-resolution with asymmetric co-attention. arXiv preprint arXiv:2004.11814.

Li L, Zhang S & Wang B (2021). Plant disease detection and classification by deep learning—a review. *IEEE Access*, 9, 56683-56698. https://doi.org/10.1109/access.2021.3069646

Liu N & Kan J M (2016). Improved deep belief networks and multi-feature fusion for leaf identification. *Neurocomputing*, 216, 460-467. https://doi.org/10.1016/j.neucom.2016.08.005

Ma J, Du K, Zheng F, Zhang L, Gong Z & Sun Z (2018). A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. *Computers and electronics in agriculture*, 154, 18-24. https://doi.org/10.1016/j.compag.2018.08.048

Mohanty S P, Hughes D P & Salathé M (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 1419. https://doi.org/10.3389/fpls.2016.01419

Pal A & Kumar V (2023). AgriDet: Plant Leaf Disease severity classification using agriculture detection framework. *Engineering Applications of Artificial Intelligence* 119: 105754. https://doi.org/10.1016/j.engappai.2022.105754

Reyes A K, Caicedo J C & Camargo J E (2015). Fine-tuning Deep Convolutional Networks for Plant Recognition. CLEF (Working Notes) 1391: 467-475

Ronneberger O, Fischer P & Brox T (2015). U-Net: Convolutional networks for biomedical image segmentation. *In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference*, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer International Publishing. https://doi.org/10.1007/978-3-662-54345-0_3

Sai Reddy B & Neeraja S (2022). Plant leaf disease classification and damage detection system using deep learning models. *Multimedia tools and applications* 81(17): 24021-24040. https://doi.org/10.1007/s11042-022-12147-0

Sazzad S, Rajbongshi A, Shakil R, Akter B & Kaiser M S (2022). RoseNet: Rose leave dataset for the development of an automation system to recognize the diseases of rose. Data in Brief, 44, 108497. https://doi.org/10.1016/j.dib.2022.108497

Simonyan K & Zisserman A (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Srivastava R K, Greff K & Schmidhuber J (2015). Highway networks. arXiv preprint arXiv:1505.00387.

Strange R N & Scott P R (2005). Plant disease: a threat to global food security. *Annual Review of Phytopathology*. 43: 83-116

Sun Y, Liu Y, Wang G & Zhang H (2017). Deep learning for plant identification in natural environment. *Computational intelligence and neuroscience*, 2017. https://doi.org/10.1155/2017/7361042

Szegedy C, Vanhoucke V, Ioffe S, Shlens J & Wojna Z (2016). Rethinking the inception architecture for computer vision. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818-2826. https://doi.org/10.1109/cvpr.2016.308

Tan M & Le Q (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning 6105-6114. PMLR.

Theckedath D & Sedamkar R R (2020). Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. SN Computer Science 1(2): 79. https://doi.org/10.1007/s42979-020-0114-9

Tiwari A (2023). A hybrid feature selection method using an improved binary butterfly optimization algorithm and adaptive β–hill climbing. *IEEE Access* 11: 93511-93537. https://doi.org/10.1109/access.2023.3274469

Tiwari A & Chaturvedi A (2021). A novel channel selection method for BCI classification using dynamic channel relevance. *IEEE Access*, 9: 126698-126716. https://doi.org/10.1109/access.2021.3110882

Tiwari A & Chaturvedi, A. (2022). A hybrid feature selection approach based on information theory and dynamic butterfly optimization algorithm for data classification. *Expert Systems with Applications*, 196, 116621. https://doi.org/10.1016/j.eswa.2022.116621

Tiwari A & Chaturvedi A (2023). Automatic EEG channel selection for multiclass brain-computer interface classification using multiobjective improved firefly algorithm. Multimedia Tools and Applications, 82(4), 5405-5433. https://doi.org/10.1007/s11042-022-12795-2

Tiwari V, Joshi R C & Dutta M K (2021). Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images. *Ecological Informatics* 63: 101289. https://doi.org/10.1016/j.ecoinf.2021.101289

Victoria A H & Maragatham G (2021). Automatic tuning of hyperparameters using Bayesian optimization. Evolving Systems, 12(1), 217-223. https://doi.org/10.1007/s12530-020-09345-2

Weiss K, Khoshgoftaar T M & Wang D (2016). A survey of transfer learning. *Journal of Big data* 3: 1-40

Wong T T, & Yang N Y (2017). Dependency analysis of accuracy estimates in k-fold cross validation. *IEEE Transactions on Knowledge and Data Engineering* 29(11): 2417-2427. https://doi.org/10.1109/tkde.2017.2740926

Yanikoglu B, Aptoula E & Tirkaz C. (2014). Automatic plant identification from photographs. Machine vision and applications 25: 1369-1383. https://doi.org/10.1007/s00138-014-0612-7

Zhang Y, Tian Y, Kong Y, Zhong B & Fu Y (2018). Residual dense network for image super-resolution. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2472-2481).

Zhao H, Jia J & Koltun V (2020). Exploring self-attention for image recognition. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* 10076-10085. https://doi.org/10.1109/cvpr42600.2020.01009

Zhao Z Q, Zheng P, Xu S T & Wu X (2019). Object detection with deep Learning: A review. *IEEE transactions on neural networks and learning systems* 30(11): 3212-3232. https://doi.org/10.1109/tnnls.2018.2876865