http://dx.doi.org/10.62762/TETAI.2025.878660

RESEARCH ARTICLE

Classification of Rice Leaf Diseases Based on Lightweight Deep Learning Model

Chao-Yun Chang[1] **and Chih-Chin Lai**[1,*]

[1] Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung 811726, Taiwan

Abstract

Traditional methods for classifying plant diseases usually depend on manual observation, which is time-consuming, labor-intensive, and prone to human error. The rise of deep learning has greatly advanced this field by enabling more accurate and efficient classification techniques. In this paper, we introduce a novel lightweight deep learning framework that builds on the RegNetY convolutional neural network architecture by incorporating a modified Efficient Channel Attention module. This enhancement is specifically designed to improve the classification of various rice leaf diseases. Our experiments on a publicly available dataset show that the proposed approach not only boosts classification accuracy but also significantly reduces computational complexity and memory usage, making it ideal for deployment on resource-limited edge devices.

Keywords: rice leaf disease classification, deep learning, lightweight convolutional neural network, attention mechanism.

Submitted: 06 December 2025
Accepted: 20 April 2026
Published: 26 May 2026

Vol. 3, **No.** 2, 2026.
10.62762/TETAI.2025.878660

***Corresponding author:**
✉ Chih-Chin Lai
cclai@nuk.edu.tw

1 Introduction

Rice is one of the world's most important staple crops and plays a crucial role in global food security. However, its production is often threatened by a variety of diseases caused by viruses, fungi, bacteria, and pests. These diseases can significantly reduce both yield and crop quality, creating major challenges for farmers and the agricultural economy. Because rice diseases are so diverse and spread quickly, early detection and accurate diagnosis are essential for protecting crop health and boosting agricultural productivity.

With the rapid advancement of deep learning technologies, convolutional neural networks (CNNs) have driven significant breakthroughs in computer vision tasks, including plant disease identification [1–3]. CNNs can learn complex disease-related features directly from leaf images and, when trained on large-scale datasets, achieve high classification accuracy. However, most state-of-the-art models focus primarily on accuracy, often at the cost of computational efficiency. These models typically use deep architectures with a large number of parameters [4, 5], which limits their practical use in real-world scenarios—especially on edge devices like smartphones or single-board computers, where computational power and memory are limited.

Citation
Chang, C. Y., & Lai, C. C. (2026). Classification of Rice Leaf Diseases Based on Lightweight Deep Learning Model. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 3(2), 142–159.

To overcome these challenges, this study presents a modified RegNetY-based deep learning architecture [7] designed specifically for classifying rice leaf diseases. By refining the network structure, incorporating an optimized attention mechanism, and fine-tuning the training process, the proposed model achieves high classification accuracy while significantly reducing computational complexity and memory usage. These improvements make the model especially well-suited for use on resource-limited edge devices, providing a practical and efficient solution for real-time plant disease diagnosis in agricultural environments.

2 Related Work

2.1 Application of Lightweight Deep Learning Models for Plant Leaf Disease Image Recognition

Bhujel et al. [8] developed a lightweight CNN that integrates multiple attention mechanisms for tomato leaf disease classification. Among the attention modules tested, the Convolutional Block Attention Module (CBAM) [9] delivered the best performance, achieving an average accuracy of 99.69%. While incorporating attention modules significantly improved both precision and recall, it also introduced substantial computational overhead. Moreover, the study did not evaluate the model's feasibility for deployment on resource-constrained devices, limiting its practical applicability in edge computing scenarios.

Zhou et al. [10] proposed GE-ShuffleNet [11], which combines the Ghost module [12] with Efficient Channel Attention (ECA) [13] and removes redundant convolutions to reduce model complexity. This architecture achieved 96.6% accuracy on the Rice Leaf dataset using only 1.521 million parameters, demonstrating a favorable balance between performance and computational cost. However, its classification accuracy remained vulnerable to background noise, and the model's robustness under real-world conditions was not systematically assessed.

Beyond rice, transfer learning with lightweight architectures has also shown strong performance in other crops: Brawijaya et al. [14] applied MobileNetV1 to potato leaf disease classification and achieved 99.5% accuracy on a Kaggle dataset, outperforming both InceptionV3 and VGG16 while demonstrating faster inference speed and greater resource efficiency for edge computing. However, the study did not address class imbalance, which could affect generalization across disease categories.

Bijoy et al. [15] introduced a lightweight deep CNN (dCNN) for classifying multiple rice leaf diseases and benchmarked its performance against 21 popular deep learning architectures, including AlexNet, ResNet50, DenseNet121, and Vision Transformer. Their model achieved an impressive classification accuracy of 99.81% while using only 0.17 million parameters, highlighting its efficiency. However, the study did not analyze inference time or explore practical deployment on edge devices. Additionally, comparisons with more recent lightweight models are needed to further validate its performance advantages.

2.2 Evolution of Rice Leaf Disease Image Classification Techniques

Sethy et al. [16] proposed a deep learning-based rice disease recognition system using 5,932 field-acquired leaf images covering four major disease types: bacterial leaf blight, rice blast, brown spot, and tungro. The study evaluated the performance of 11 CNN models via transfer learning and examined the effectiveness of combining deep features with a support vector machine (SVM) classifier. Experimental results showed that ResNet50 features combined with an SVM achieved an F1-score of 98.38%. Additionally, the study found that the FC6 layer of VGG16, VGG19, and AlexNet contributed more significantly to classification performance than the FC7 and FC8 layers. While the study offered a comprehensive comparison of modern CNNs and traditional methods, the high computational cost and model complexity pose challenges for real-time deployment and practical maintenance.

Bari et al. [17] employed the Faster R-CNN framework for real-time rice disease detection by integrating a region proposal network (RPN) to generate and localize candidate regions. The model was trained on a combination of public and field-collected datasets, successfully identifying rice blast, brown spot, and rice hispa with accuracies of 98.09%, 98.85%, and 99.17%, respectively, while achieving 99.25% accuracy for healthy leaves. Although the method demonstrated superior accuracy compared to traditional approaches, its two-stage architecture introduced additional processing time, limiting its suitability for time-sensitive applications and on-field automation.

Haque et al. [18] proposed a rice leaf disease detection

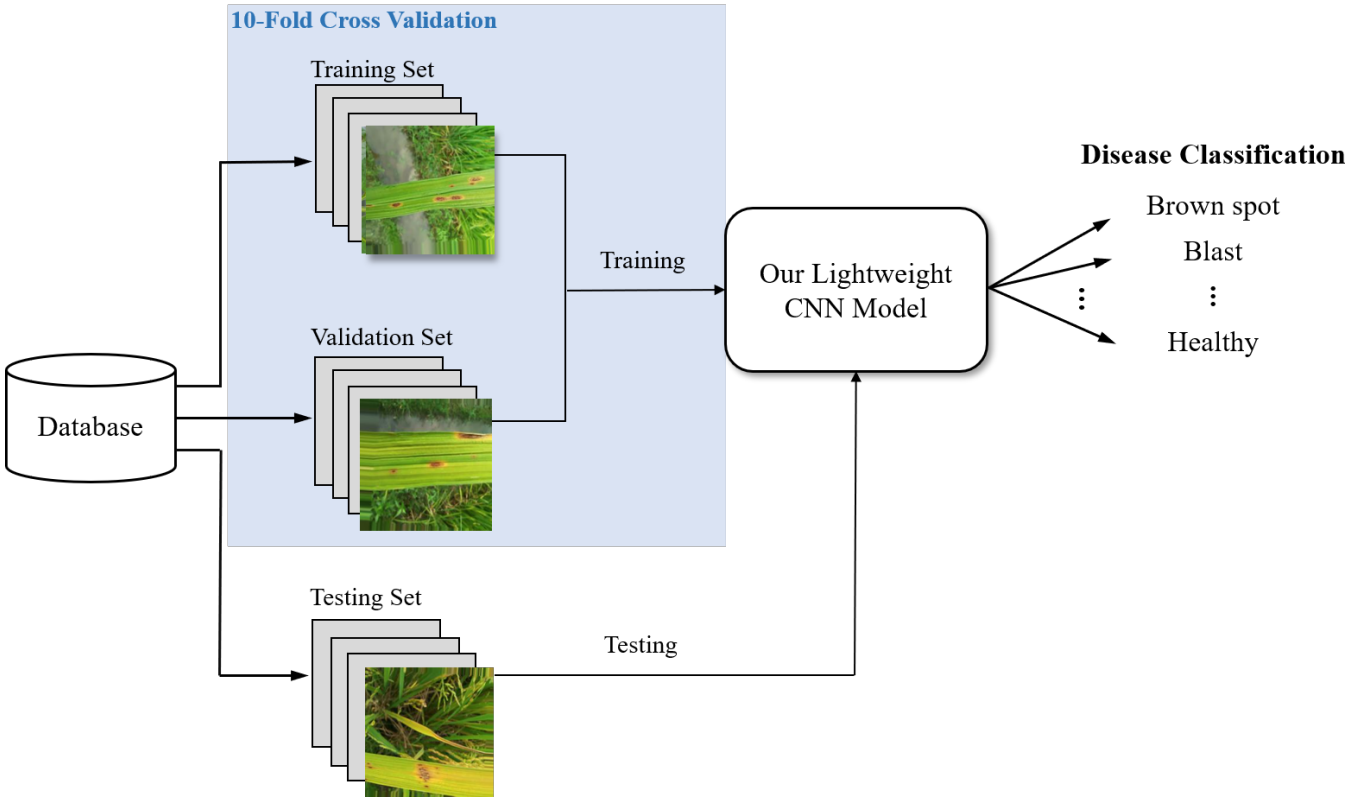

Figure 1. Training and testing flowchart for rice leaf disease classification.

and classification model based on YOLOv5, leveraging its high speed and accuracy to enhance diagnostic efficiency. The model targeted five diseases—bacterial stripe, rice blast, brown spot, leaf blight, and yellow wilt—while optimizing anchor boxes and the loss function to minimize both false positives and false negatives. Data augmentation techniques were employed to improve generalization. The model achieved a precision of 90%, with a mean average precision (mAP) surpassing that of Faster R-CNN and SSD. However, the limited dataset size and lack of diversity constrained its generalizability in diverse real-world scenarios.

Ahad et al. [19] investigated transfer learning for rice disease classification by evaluating six CNN architectures and introducing an ensemble model, DEX, which integrates DenseNet121, EfficientNetB7, and Xception. Trained on a dataset covering nine categories (eight diseases and one healthy class), the model utilized transfer learning and data augmentation to enhance convergence and generalization. The DEX model achieved a classification accuracy of 98%, outperforming the individual CNNs in precision, recall, and F1-score. Nonetheless, the authors noted that limited sample availability in some disease categories could negatively impact the model's generalization capability.

Ritharson et al. [20] proposed an efficient rice disease recognition approach combining deep learning and transfer learning techniques. They compiled a dataset of 5,932 manually labeled images, augmented with publicly available samples, covering nine classes (eight disease categories and one healthy class) with varying severity levels. After data augmentation, a custom CNN model was trained and benchmarked against VGG16, Xception, ResNet50, and DenseNet121. The customized VGG16 achieved a classification accuracy of 99.94%, demonstrating strong generalization performance. However, the study did not report inference time or computational requirements, leaving the model's feasibility for edge device deployment unassessed. The challenge of multi-class classification across diverse disease categories with subtle inter-class differences has also been demonstrated in other crop species [6], underscoring the generality of these difficulties beyond rice.

3 Methodology

3.1 Flowchart for Rice Leaf Disease Image Classification

We employ deep learning techniques to classify rice leaf diseases, as shown in Figure 1. The classification process involves two main phases: training and testing. Both phases use a rice leaf disease dataset for model

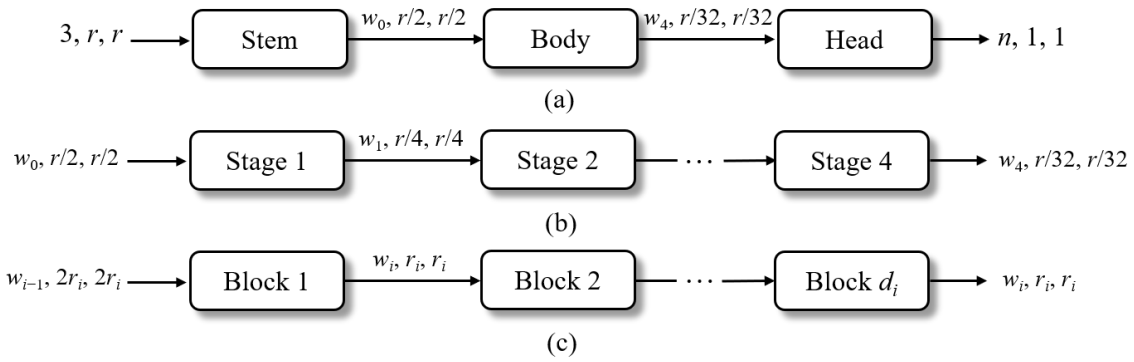

Figure 2. RegNetY model architecture. (a) Overall AnyNet framework, (b) Main body composition, and (c) Block structure.

development and performance evaluation.

During the training phase, a 10-fold cross-validation strategy is used to split the dataset into multiple subsets. In each iteration, one subset serves as the validation set, while the remaining subsets are used for training. This process is repeated across all folds to ensure robust model evaluation and validation. Hyperparameter tuning is performed during each validation cycle to identify the optimal configuration. After each training iteration, the model weights that achieve the highest validation performance are saved for later inference.

In the testing phase, the best-performing weights obtained from cross-validation are used to evaluate the model on an independent test set. This approach provides an unbiased assessment of the model's classification performance on unseen data, ensuring the reliability and generalizability of the final results.

3.1.1 RegNetY Model Architecture

The RegNetY model is built upon the AnyNet architecture, a highly efficient and modular CNN design, as illustrated in Figure 2(a). The AnyNet framework consists of three main components: the stem, body, and head. The stem performs initial feature extraction and downsampling from the input image, where r denotes the input resolution and w represents the channel width of the resulting feature maps. The head includes a global average pooling layer followed by a fully connected layer for final classification, with n corresponding to the number of output classes.

The body consists of four sequential stages, as shown in Figure 2(b), with the spatial resolution r progressively decreasing at each stage. Each stage is made up of multiple blocks, illustrated in Figure 2(c). The RegNetY architecture allows flexible adjustment of both network depth (d_i) and channel width (w_i) at each stage, where d_i and w_i represent the depth and width, respectively. This flexibility enables the creation of various RegNetY model variants that can be optimally tailored to specific application needs, facilitating efficient deployment across a wide range of computational constraints.

To enhance inter-channel feature representation within each block, the Squeeze-and-Excitation (SE) attention module [21] is integrated. Each block contains two computational paths. In the main branch, a 1×1 convolution first adjusts the number of channels, followed by a 3×3 convolution for feature extraction. The resulting output is then refined by the SE module, which adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. This is achieved through a squeeze operation that captures global channel information via global average pooling, followed by an excitation step using a small fully connected network that generates channel-wise weights. These weights are applied via element-wise multiplication to scale the feature maps, strengthening important channels and suppressing less relevant ones. Finally, another 1×1 convolution restores the required channel dimensionality.

When downsampling is needed within a block, the stride of the 3×3 convolution is set to 2, and the number of input channels is doubled. This reduces the spatial dimensions of the feature map by half, as illustrated in Figure 3(a). Conversely, to maintain the original feature map resolution, the stride is set to 1, preserving the spatial dimensions, as shown in Figure 3(b).

In both configurations, the final output is obtained by element-wise addition of the main branch and a shortcut connection, forming a residual learning structure. This design helps alleviate the vanishing gradient problem by providing a direct

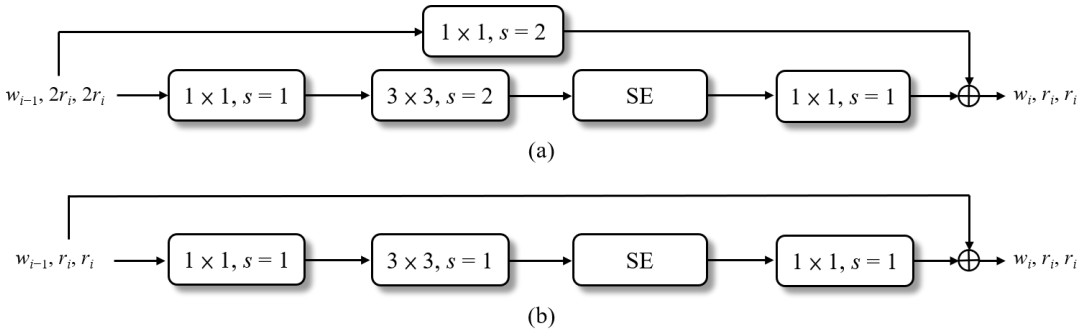

Figure 3. Internal block structure of the RegNetY model. (a) Path with Downsampling to Half Resolution and (b) Path Maintaining Original Spatial Resolution.

path for gradient flow during backpropagation, which facilitates the training of deeper networks and improves feature representation. The flexible block design enables RegNetY to effectively extract multi-scale features at different levels, while the integration of the SE module further enhances the model's focus on critical feature information. Together, these elements significantly boost the model's representational capacity, making it well-suited for complex visual tasks.

3.1.2 SE Attention Module in the RegNetY Model

The RegNetY model incorporates the SE attention module within each block to enhance its ability to selectively emphasize informative channel features, thereby improving the overall quality of its feature representations. The core principle of the SE module is its capacity to adaptively reweight channel-wise responses, enabling the model to amplify relevant feature information while suppressing redundant or non-discriminative channels.

As shown in Figure 4, the SE module offers a lightweight yet effective mechanism that boosts channel attention without altering the spatial resolution of the feature maps. By recalibrating the importance of each channel, the module allows the network to focus on salient features—an advantage that is especially valuable for tasks like rice leaf disease classification, where capturing subtle visual variations is crucial.

While the SE module effectively enhances channel-wise feature representation, its reliance on two fully connected layers results in a significant increase in parameters and floating-point operations (FLOPs). This computational overhead becomes especially problematic as the number of channels grows, making it challenging to deploy the model on resource-constrained edge devices. To address this limitation, the present study introduces a modified Efficient Channel Attention (mECA) module as a replacement for the SE module. The mECA module is specifically designed to reduce the parameter count and improve inference efficiency, all while maintaining strong representational capacity. A detailed description of its architecture is provided in Section 3.2.2.

3.2 Modified RegNetY Model Architecture

The RegNetY model series has gained significant attention for its efficient architectural design and strong empirical performance. Its flexible configuration allows optimization by adjusting key parameters such as network depth, group width, and initial channel width (w_0), enabling a balanced trade-off between computational complexity and classification accuracy across diverse applications. Although the RegNetY family includes smaller-scale variants, the integration of the SE attention module helps these models maintain robust feature extraction capabilities. As a result, RegNetY models not only deliver competitive performance among state-of-the-art deep learning architectures but also show great potential for lightweight applications, especially in resource-constrained environments.

3.2.1 Lightweight Design of Modified RegNetY Model Architecture

In deep learning applications, computational complexity and parameter count often pose significant challenges for both training and deployment, especially when targeting resource-constrained edge devices. To improve execution efficiency, this study focuses on architectural modifications to the RegNetY model.

The naming convention of the RegNetY series is based on the number of giga floating-point operations (GFLOPs), a standard metric for measuring model

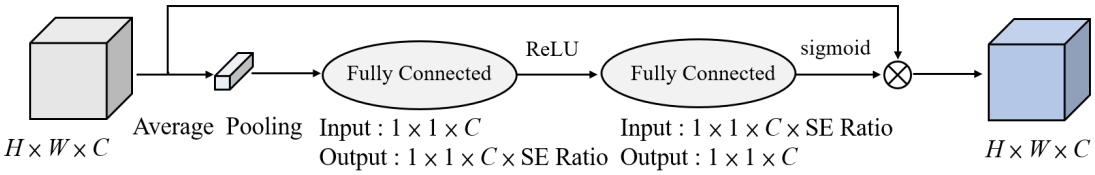

Figure 4. Architecture of the SE Channel Attention module.

complexity. The formal definition of GFLOPs is given in Eq. (1).

$$GFLOPs = \frac{\text{Total floating-point operations}}{10^9}. \quad (1)$$

Within the RegNetY series, the largest variant—RegNetY-128GF—has a computational cost of 128 GFLOPs, while the smallest variant, RegNetY-200mf, requires only 0.2 GFLOPs. The primary goal of this study is to further reduce the computational complexity to 0.1 GFLOPs, creating an ultra-lightweight version called RegNetY-100MF.

Achieving this objective requires a thorough understanding of the main contributors to computational cost in deep neural networks—namely, convolutional layers and fully connected layers. The number of FLOPs depends on factors such as input resolution, the number of channels, and network depth. Among these, convolutional layers typically account for about 90% of the total computational load. During a convolution operation, each input channel is convolved with a corresponding kernel, and the resulting values are combined to produce the output feature map [22]. The calculation of FLOPs for a convolutional layer is formally defined in Eq. (2).

$$FLOPs_{conv} = 2 \times H_{out} \times W_{out} \times C_{out} \times K_h \times K_w \times C_{in}, \quad (2)$$

where H_{out} and W_{out} represent the height and width of the output feature map, respectively, while C_{out} denotes the number of output channels, which also corresponds to the number of convolution kernels used in the layer. K_h and K_w are the height and width of each convolution kernel, and C_{in} is the number of input channels, i.e., the number of channels in the output from the preceding layer. The factor of 2 accounts for one multiplication followed by one addition per operation, following the FLOPs calculation convention used by the PyTorch library.

Although the overall computational cost of fully connected (FC) layers is relatively low, they remain a critical factor affecting model efficiency, especially during the classification stage. The FLOPs for a fully connected (FC) layer are calculated using the formula in Eq. (3):

$$FLOPs_{fc} = 2 \times N_{in} \times N_{out}, \quad (3)$$

where N_{in} and N_{out} represent the number of neurons in the previous and current layers, respectively. The multiplication by 2 accounts for one multiplication and one addition per operation, consistent with the FLOPs calculation convention. Additionally, the SE attention module integrated into the RegNetY model contains two fully connected layers, which further increase the overall parameter count and computational cost. Replacing or modifying the SE module during architecture refinement can help reduce the model's computational burden.

Based on the adjusted architecture, we calculate the FLOPs for each layer and recompute the total cost, ensuring it remains below 0.1 GFLOPs to satisfy our ultra-lightweight design requirements.

3.2.2 Modified ECA Attention Module

To reduce both the computational cost and the number of parameters, the original SE attention module in RegNetY is replaced with a novel mECA module. This change further improves the model's performance in recognizing rice leaf diseases. Compared to the original ECA module, the proposed mECA architecture incorporates an additional batch normalization (BN) layer after the 1D convolution operation [23]. This modification helps stabilize the distribution of feature responses and reduces the risk of premature overfitting during the early stages of training.

The standard ECA module extracts global channel descriptors using global average pooling (GAP) and captures local inter-channel dependencies through 1D convolution. This approach allows the model to emphasize informative features while suppressing redundant ones. In contrast to the SE module, which models channel-wise relationships using two fully connected layers, the ECA module achieves

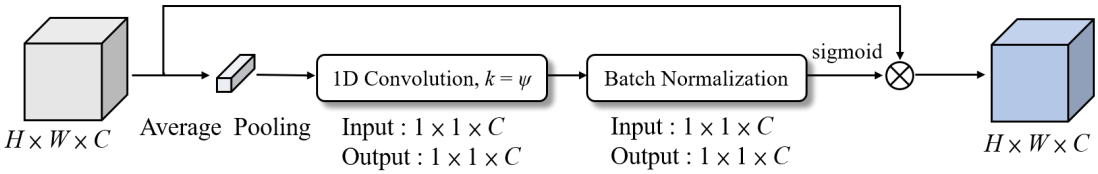

Figure 5. Modified ECA module architecture with BN integration.

comparable or superior performance with significantly lower computational overhead. As a result, it is especially well-suited for deployment in lightweight and resource-constrained environments.

The operational process of the ECA module is as follows: Given an input feature map of size $H \times W \times C$, a GAP operation is first applied to compress the spatial dimensions, generating a channel-wise descriptor vector z_c, as defined in Eq. (4).

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_c(i,j), \tag{4}$$

where H and W represent the height and width of the input image, respectively; z_c denotes the global average value of the Cth channel, and X_c refers to the original feature map of that channel.

Next, the kernel size k for the 1D convolution is adaptively determined based on the number of channels, as defined in Eq. (5).

$$z'_c = \text{Conv1D}(z_c), \text{kernel size } k = \psi(C) = \left| \frac{\log_2(C)+b}{\gamma} \right|_{\text{odd}}, \tag{5}$$

where $\gamma = 2$, $b = 1$, and the number of channels C is used to compute the nearest odd number as the kernel size for the 1D convolution.

Building on the 1D convolution operation, we propose a modified architecture that integrates a Batch Normalization (BN) layer immediately following the convolution. This normalization stabilizes the distribution of feature responses and prevents saturation when the outputs pass through the subsequent sigmoid activation, as defined in Eq. (6). Consequently, this stabilization improves the model's robustness and generalization during training.

$$\hat{z}_c = \text{BN}(z'_c) = \text{BN}(\text{Conv1D}(z_c)). \tag{6}$$

The normalized feature vector, $\hat{z}_c$, is subsequently passed through a sigmoid function to generate the attention weight s_c, as defined in Eq. (7).

$$s_c = \sigma(\hat{z}_c), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Finally, the attention weight s_c is applied to the original input feature map χ_c via channel-wise multiplication to produce the weighted output $\hat{\chi}_c$, as defined in Eq. (8).

$$\hat{\chi}_c = s_c \cdot \chi_c. \tag{8}$$

The incorporation of the BN layer is motivated by the desire to improve feature stability and learning efficiency. In the original ECA module, the output of the 1D convolution is fed directly into the sigmoid activation function, which can produce a skewed or highly variable feature distribution. This instability may cause vanishing gradients and impede effective learning, especially in the early training stages. By placing a BN layer before the sigmoid activation, the feature distribution is regularized, accelerating training convergence, stabilizing weight updates, and reducing the risk of overfitting [24]. This approach leverages BN's ability to standardize feature statistics within mini-batches, thus fostering a more stable and efficient optimization process.

Overall, the proposed mECA module improves the stability of channel attention learning and boosts recognition performance without significantly increasing parameter count or computational cost. By integrating a batch normalization layer immediately after the 1D convolution, mECA stabilizes feature distributions and reduces overfitting risks, leading to more effective training convergence. This novel design retains ECA's lightweight nature while enhancing robustness, making it highly suitable for resource-constrained applications such as rice leaf disease recognition. The detailed architectural design of the mECA module is illustrated in Figure 5, showcasing its streamlined yet effective structure.

The proposed mECA module enhances the stability of channel attention learning and boosts recognition performance with negligible overhead in terms of parameters and computational complexity.

Integrating a BN layer immediately after the 1D convolution stabilizes feature distributions and mitigates overfitting to ensure superior training convergence. This design preserves the lightweight efficiency of the original ECA while significantly enhancing its robustness. Consequently, it is ideal for resource-constrained applications such as rice leaf disease recognition. The streamlined architecture of the mECA module is illustrated in Figure 5.

3.3 Optimization of Model Training Parameters

Enhancing the performance of deep learning models while mitigating overfitting requires effective parameter optimization strategies. Overfitting can hinder a model's ability to generalize, particularly in tasks with high intra-class similarity, such as rice leaf disease classification. In this study, we employ two techniques to balance classification accuracy and training stability while minimizing unnecessary computational overhead. Specifically, we incorporate weight decay regularization [25] and the ReduceLROnPlateau learning rate scheduling strategy, which has been widely adopted in deep learning-based plant disease classification to improve training convergence and generalization [26].

3.3.1 Weight Decay Method

In rice leaf image classification, the visual features of different disease categories often show high similarity, with only subtle variations between classes. If the model relies too heavily on specific features or noise in the training data, it may achieve high accuracy on the training set but perform poorly on validation, test sets, or in real-world scenarios. To address this, weight decay is applied to limit the complexity of the learned weights, thereby improving the model's generalization ability and robustness in rice leaf disease classification tasks.

Weight decay is a widely used regularization technique in deep learning, typically implemented as L2 regularization. Its main goal is to balance the model's adaptability to training data with its ability to generalize to unseen inputs by discouraging excessively large weight values. This constraint promotes model stability, reduces the risk of overfitting during training, and ultimately improves generalization performance.

The concept of weight decay involves modifying the original loss function by adding an L2 penalty term, as illustrated in Eq. (9).

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \cdot \left(\frac{\partial L}{\partial \mathbf{w}_t} + \lambda \cdot \mathbf{w}_t \right), \tag{9}$$

where $\mathbf{w}_t$ is the weight of the tth iteration, η is the learning rate, L is the loss function, and λ is the weight decay coefficient.

3.3.2 ReduceLROnPlateau Learning Rate Adjustment Strategies

The learning rate is a crucial hyperparameter in training deep learning models, as it directly influences both the speed of convergence and the final classification performance. A learning rate that is too high can cause unstable, oscillatory behavior and prevent the model from converging properly. Conversely, a learning rate that is too low may result in slow training progress or convergence to suboptimal local minima.

ReduceLROnPlateau is an adaptive learning rate scheduling technique that monitors the validation loss during training and reduces the learning rate when no improvement is observed within a specified patience interval. When the monitored metric fails to improve after a predetermined number of epochs, the learning rate is multiplied by a predefined reduction factor (RF), effectively lowering its value. This dynamic adjustment aligns the learning rate with the training progress, enhancing stability, accelerating convergence in later stages, and ultimately improving the model's generalization performance. The conceptual formulation of this adjustment is presented in Eq. (10).

$$\eta_{t+1} = \begin{cases} \eta_t \times \mathrm{RF}, & \text{if } L_{\mathrm{val}}(t) \text{ does not improve} \\ & \text{more than } \epsilon \text{ for patience} \\ & \text{epochs;} \\ \eta_t, & \text{otherwise.} \end{cases} \tag{10}$$

In rice leaf disease image classification, where inter-class distinctions in lesion morphology, texture, and chromatic properties are frequently subtle, an excessively elevated learning rate may impede the model's capacity to capture fine-grained discriminative features, consequently constraining classification performance. The incorporation of the ReduceLROnPlateau learning rate scheduler enables adaptive modulation of the learning rate upon detecting stagnation in validation performance. This

mechanism facilitates continued model refinement under a more conservative optimization regime, reducing susceptibility to local minima entrapment and contributing to enhanced classification accuracy.

4 Experiments

4.1 Experimental Environment

The experimental environment comprised a desktop computer equipped with an 8-core AMD Ryzen 7 5800X CPU, an NVIDIA GeForce RTX 3080 GPU, and 16 GB of RAM. The system operated on Windows 11 Professional and utilized Anaconda 24.3.0 for package management and environment configuration, with Python 3.11.9 as the programming language. Model development and training were conducted using the PyTorch 2.4.1 deep learning framework, leveraging GPU acceleration through CUDA 12.4 and cuDNN 9.0.0.

For all experiments, the initial training configuration was standardized as follows: a learning rate of 0.001, a batch size of 16, and 50 training epochs. The Adam optimizer was employed for parameter optimization, while cross-entropy loss served as the objective function, selected due to its effectiveness in facilitating rapid and accurate convergence in multi-class classification tasks.

4.2 Rice Leaf Disease Database

A publicly available rice leaf disease image dataset, characterized by substantial volume and variability, was selected to support the development of deep learning models. This dataset comprises multiple rice disease categories and includes images captured under diverse conditions, encompassing both real-world agricultural fields and controlled environments. This variability enriches the dataset by introducing heterogeneity across key visual dimensions, including illumination, textural properties, and background complexity, thereby establishing the conditions necessary for robust model training, validation, and performance evaluation.

The Rice_Leaf dataset, obtained from the Kaggle platform [27], comprises a total of 11,790 labeled images, each with a spatial resolution of 224×224 pixels. The distribution of images across rice leaf disease categories is presented in Table 1. This dataset was randomly partitioned into training, validation, and testing subsets. Specifically, 90% of the data was allocated for model development, with the remaining 10% reserved as an independent test set. The development portion was further divided into training and validation sets at a 9:1 ratio, resulting in 9,548 training, 1,060 validation, and 1,182 testing images. To ensure experimental reproducibility and preserve class distribution proportions, stratified sampling was employed with a fixed random seed value of 42. Furthermore, a 10-fold cross-validation strategy was applied to the combined training and validation data. The test set remained strictly isolated throughout the process, used only for the final evaluation to ensure an unbiased assessment.

Table 1. Class Distribution in the Rice_Leaf Dataset.

Class	Number of Images
Bacterial Leaf Blight	1,197
Brown Spot	1,546
Healthy	1,085
Leaf Blast	1,748
Leaf Scald	1,332
Narrow Brown Leaf Spot	954
Neck Blast	1,000
Hispa	1,299
Sheath Blight	1,629
Total	11,790

The dataset comprises eight common categories of rice leaf diseases in addition to a healthy class. These categories include bacterial blight, brown spot, healthy, leaf blast, leaf scald, narrow brown spot, neck blast, hispa, and sheath blight, as illustrated in Figure 6. These diseases are caused by a variety of pathogens, including fungi, viruses, and insect pests, and collectively represent the morphological, color, and textural diversity and complexity characteristic of rice leaf diseases. Such diversity within the dataset supports robust deep learning model training and evaluation by capturing the subtle inter-class variations and visual heterogeneity typical of these diseases.

4.3 Experimental Results and Analysis

This section presents four experiments designed to evaluate the performance of the proposed method using the Rice_Leaf dataset. Experiment 1 explores the impact of adjusting the model architecture by varying the group width and initial width (w_0) while maintaining a fixed depth of 7. Six different architectural configurations are systematically evaluated to identify the optimal balance between model capacity and efficiency. Experiment 2 replaces the original SE attention module in the RegNetY

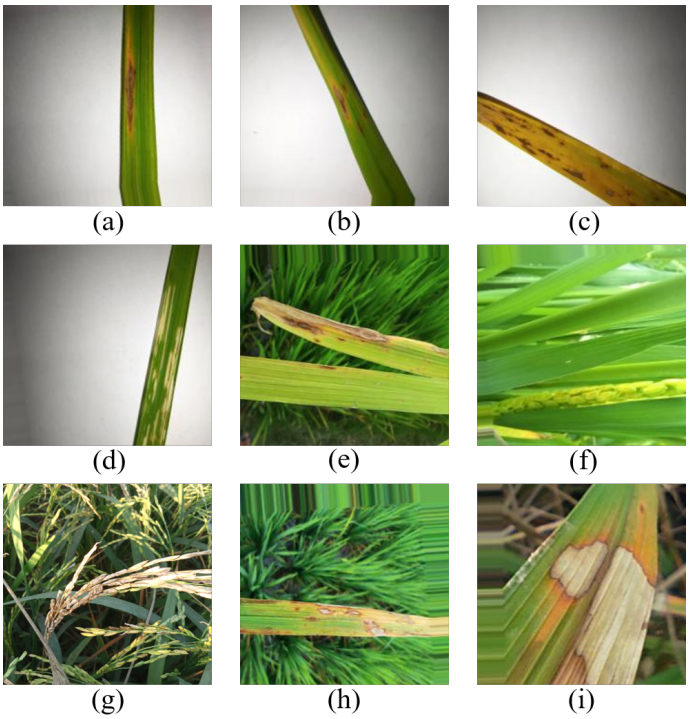

Figure 6. Sample Images from the Rice_Leaf dataset. (a) Leaf Blast, (b) Brown Spot, (c) Narrow Brown Leaf Spot, (d) Hispa, (e) Leaf Scald, (f) Healthy, (g) Neck Blast, (h) Bacterial Blight, and (i) Sheath Blight.

architecture with the proposed mECA module. The performance of the mECA-enhanced model is compared against the original SE module as well as other commonly used attention mechanisms to assess improvements in feature recalibration and classification accuracy. Experiment 3 investigates the effects of different training parameters on model performance, specifically comparing optimizers and learning rate adjustment strategies. This includes examining how parameter optimization methods and dynamic learning rate scheduling impact convergence speed, training stability, and final classification accuracy. Experiment 4 benchmarks the proposed model against other representative lightweight architectures to validate its overall effectiveness and competitiveness in rice leaf disease classification. This comparative analysis highlights the trade-offs between model complexity, accuracy, and inference efficiency. Collectively, these experiments comprehensively assess the architectural design, attention mechanism integration, training optimization, and comparative performance of the proposed method, providing insights to support its practical deployment.

To evaluate classification performance, several standard metrics are employed, including accuracy, precision, recall, and F1-score. Accuracy serves as the primary metric, reflecting the model's overall classification capability across all classes. Precision quantifies the model's ability to minimize false positives, while recall measures its effectiveness in correctly identifying all true positive instances. The F1-score provides a balanced assessment by calculating the harmonic mean of precision and recall, thereby capturing both aspects of classification performance. The formal definitions of these metrics are presented in Eqs. (11)– (14).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{14}$$

4.3.1 *Comparison of Size Reduction Performance of RegNetY Model*

In this experiment, we examine the impact of varying initial widths (w_0) and group widths (GW) on the network architecture's performance to identify the optimal configuration. Evaluation metrics include training time, model size, average accuracy obtained from 10-fold cross-validation, and inference time. The results for different combinations of initial and group widths, with a fixed network depth of 7, are summarized in Tables 2 and 3.

As illustrated in Table 2, architectures with a group width of 4 generally outperform those with a group width of 8 under the same initial width setting in terms of classification accuracy. Notably, the model configured with a group width of 4 and an initial width of 16 achieved the highest average accuracy of 99.18%, accompanied by a relatively low standard deviation. This indicates superior classification stability and overall effectiveness for this configuration.

As presented in Table 3, the model configuration with a group width of 4 and an initial width of 16 achieved the highest classification accuracy, demonstrating superior feature extraction capability and stability. Additionally, this configuration exhibited relatively low inference time and computational complexity, underscoring its suitability for deployment in resource-constrained environments.

Overall, the experimental results indicate that configurations with smaller group and initial widths tend to provide improved accuracy alongside enhanced inference efficiency. These findings highlight the importance of carefully selecting architectural parameters to optimize model performance, facilitating reliable and effective multi-class classification of rice leaf diseases.

4.3.2 *Performance Comparison of Modified RegNetY Attention Module Architecture*

In this experiment, we further investigate the impact of replacing the original SE attention module in the RegNetY architecture with several commonly used attention mechanisms, including the Block Attention Module (BAM) [28], CBAM, ECA, Simple Attention Module (SimAM) [29], and the proposed mECA module. The evaluation was conducted using 10-fold cross-validation to ensure robustness, accompanied by comprehensive performance analyses. The experimental results are summarized in Table 4, providing a comparative assessment of the effectiveness of each attention mechanism in enhancing model performance.

As shown in Table 4, the modified RegNetY model reduced training time by approximately 5.1% (from 127,824 s to 121,248 s) compared to the original RegNetY, while achieving a notable improvement in average accuracy across 10-fold cross-validation, increasing from 99.18% to 99.52%. Furthermore, the standard deviation decreased from 0.30 to 0.20, indicating enhanced classification stability. These results suggest that replacing the SE attention module with the proposed mECA module provides substantial benefits in terms of both classification accuracy and model robustness.

Further performance evaluations of the RegNetY model equipped with various attention modules are presented in Table 5. The experimental findings reveal that the proposed RegNetY variant augmented with the mECA module attained the highest test accuracy of 98.56%, exceeding the performance of the original RegNetY model (98.48%) by a margin of 0.08 percentage points. These results provide empirical evidence supporting the effectiveness of the mECA module in improving fine-grained rice leaf disease classification accuracy. The enhanced classification performance exhibited by the mECA-equipped model is principally ascribed to the incorporation of a BN layer within the mECA module. By stabilizing the distribution of intermediate feature representations, this component enables the model to more selectively attend to disease-relevant features, thereby substantially reinforcing its discriminative capability and yielding improved overall classification accuracy.

An analysis of the evaluation metrics revealed that recall was the only measure to exhibit a slight decline compared to the original model, decreasing from 98.56% to 98.47%. Investigating the class distribution within the Rice_Leaf dataset, it was found that the Leaf Blast category contains the largest number of images (1,748), while Narrow Brown Leaf Spot has the fewest (954 images). During training, classes with fewer samples contribute less to the optimization process, thereby limiting the model's ability to learn sufficiently discriminative features for these minority classes. Consequently, instances belonging to underrepresented classes are more prone to misclassification as majority classes during testing, resulting in an increased number of false negatives. This rise in false negatives adversely affects the recall metric, which measures the model's effectiveness in correctly identifying all true positive cases.

Moreover, the enhanced feature extraction capability provided by the mECA module may inadvertently increase the model's bias toward classes exhibiting stronger feature representations and having larger sample sizes. This phenomenon can further exacerbate the observed decline in recall for underrepresented classes, as the model becomes more predisposed to

Table 2. Performance and Model Sizes of Architecture Variants with Depth 7 under 10-fold Cross-validation.

Architecture	Training Time (sec.)	Size (MB)	Accuracy (%)	Std. Dev.
GW = 8, $w_0 = 16$	96,225	1.3	99.10	0.28
GW = 8, $w_0 = 24$	95,650	1.1	98.95	0.38
GW = 8, $w_0 = 32$	95,608	1.7	98.27	0.38
GW = 4, $w_0 = 16$	127,824	1.4	99.18	0.30
GW = 4, $w_0 = 24$	173,073	1.2	99.10	0.34
GW = 4, $w_0 = 32$	145,248	1.4	98.87	0.41

Table 3. Evaluation of Performance Indicators for Various Architecture Combinations at Depth 7.

Architecture	Inference Time (sec.)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
GW = 8, $w_0 = 16$	0.0070	98.22	98.20	98.43	98.31
GW = 8, $w_0 = 24$	0.0072	98.14	97.97	98.34	98.15
GW = 8, $w_0 = 32$	0.0073	94.16	94.48	93.91	94.19
GW = 4, $w_0 = 16$	0.0064	98.48	98.41	98.56	98.48
GW = 4, $w_0 = 24$	0.0067	97.46	97.63	97.48	97.55
GW = 4, $w_0 = 32$	0.0069	98.14	98.34	98.26	98.30

Table 4. Average Accuracy and Standard Deviation of RegNetY Using Various Attention Modules.

Method	Training Time (sec.)	Average Accuracy (%)	Std. Dev.
RegNetY(with SE)	127,824	99.18	0.30
RegNetY − SE + BAM	89,844	98.15	0.47
RegNetY − SE + CBAM	73,064	97.95	0.40
RegNetY − SE + ECA	146,435	99.09	0.28
RegNetY − SE + SimAM	132,834	98.32	0.64
RegNetY − SE + mECA	121,248	99.52	0.20

Table 5. Evaluation of RegNetY Performance Using Various Attention Modules.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
RegNetY(with SE)	98.48	98.41	98.56	98.48
RegNetY − SE + BAM	95.80	95.81	95.88	95.84
RegNetY − SE + CBAM	97.29	97.51	97.39	97.45
RegNetY − SE + ECA	97.71	97.84	97.73	97.78
RegNetY − SE + SimAM	96.62	96.84	96.34	96.58
RegNetY − SE + mECA	98.56	98.67	98.47	98.57

Table 6. Model Size Comparison of RegNetY Architectures Using Different Attention Modules.

Method	GFLOPs	Size (MB)	Num. of Parameters	Inference Time (sec.)
RegNetY(with SE)	0.101908	1.425	517,060	0.0064
RegNetY − SE + BAM	0.115393	1.445	520,242	0.0088
RegNetY − SE + CBAM	0.102425	1.279	479,270	0.0092
RegNetY − SE + ECA	0.101858	1.203	462,421	0.0059
RegNetY − SE + SimAM	0.101579	1.116	462,392	0.0068
RegNetY − SE + mECA	0.101862	1.215	462,435	0.0060

prioritizing well-expressed majority classes during classification. Such behavior aligns with known challenges in imbalanced datasets, where models tend to favor majority classes both due to their prevalence and the more distinguishable feature patterns they offer, thereby reducing sensitivity to minority classes. The experimental comparison of model sizes for RegNetY architectures with different attention modules is summarized in Table 6, providing additional context on the computational trade-offs associated with these mechanisms.

Inference time was a critical metric prioritized during the model refinement process. As shown in Table 6, the RegNetY model incorporating the proposed mECA module demonstrated excellent performance in terms of inference time. Although the mECA-equipped model exhibited a marginally higher inference time of 0.0060 seconds compared

Table 7. 10-fold Cross-validation Performance Comparison of Modified RegNetY with Different Weight Decay Coefficients.

Weight Decay	Training Time (sec.)	Average Accuracy (%)	Std. Dev.
1E-4	195,852	99.46	0.22
1E-5	149,217	99.43	0.16
1E-6	104,367	99.24	0.14

to the original ECA-based model (0.0059 seconds), it nonetheless demonstrated superior computational efficiency relative to the baseline RegNetY model (0.0064 seconds). Furthermore, the modified model achieved a significant improvement in classification performance compared to the original RegNetY, indicating that the mECA module effectively balances model size and classification performance.

These experimental results collectively demonstrate that replacing the original SE attention module with the proposed mECA module yields significant improvements across accuracy, inference time, and model size.

4.3.3 The Impact of Training Optimizers and Learning Rate Policies on Model Performance

In this experiment, we continued using the best-performing RegNetY model integrated with the mECA module and evaluated the effects of two optimization strategies—AdamW and ReduceLROnPlateau—on model training and performance.

Compared to the traditional Adam optimizer, we adopted AdamW and followed the recommended configuration range from the original AdamW paper. We tested commonly used weight decay coefficients for small-scale models: 1E-4, 1E-5, and 1E-6. The experimental results are summarized in Table 7.

As shown in Table 7, setting the weight decay coefficient to 1E-4 makes parameter updates overly conservative, leading to significantly slower convergence and longer training times. Conversely, a coefficient of 1E-6 speeds up the initial learning phase but risks causing the model to miss the optimal solution during training, which limits accuracy improvements. Therefore, 1E-5 proved to be the most balanced choice, effectively reducing overfitting while preserving sufficient model expressiveness.

For learning rate adjustment, we adopted the ReduceLROnPlateau strategy with a patience value of 7. Additionally, the reduction factor was set to 0.1, meaning the learning rate is reduced to one-tenth of its current value each time the adjustment is triggered. This allows for more fine-grained weight updates during the convergence phase, thereby improving classification performance.

The experimental results for RegNetY using different combinations of training parameter optimizations, evaluated through 10-fold cross-validation, are summarized in Table 8. As shown in Table 8, using the AdamW optimizer alone resulted in a slight decrease in average accuracy to 99.43%; however, the standard deviation improved to a more stable 0.16. This indicates that, despite a minor trade-off in accuracy, the training process became more stable overall.

When the ReduceLROnPlateau learning rate adjustment strategy was applied—whether alone or in combination with AdamW—both accuracy and standard deviation improved significantly. Notably, the combination of ReduceLROnPlateau and AdamW yielded the highest average accuracy of 99.78% with a standard deviation of just 0.17, representing the most balanced and effective optimization strategy among all tested configurations.

Additional experimental results for RegNetY with various training parameter optimization combinations are listed in Table 9. As shown in Table 9, the combination of the AdamW optimizer and the ReduceLROnPlateau learning rate adjustment strategy delivered the best performance across all evaluation metrics. The results from Experiment 3 demonstrate that ReduceLROnPlateau significantly improves model convergence efficiency and enhances classification stability. When used alongside AdamW, this strategy not only boosts overall classification accuracy but also substantially reduces performance variability, as evidenced by the lower standard deviation.

4.3.4 Performance Comparison between the Modified RegNetY Model and Other Lightweight Models

In this experiment, we systematically compared the performance and model size of the proposed customized RegNetY model with several

Table 8. Training Time and Performance Comparison of Modified RegNetY under Various Optimization Settings.

Method	Training Time (sec.)	Average Accuracy (%)	Std. Dev.
Adam	121,248	99.52	0.20
AdamW	149,217	99.43	0.16
Adam + ReduceLROnPlateau	69,522	99.75	0.18
AdamW + ReduceLROnPlateau	109,283	99.78	0.17

Table 9. Performance Evaluation of Modified RegNetY under Various Training Optimization Settings.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Adam	98.39	98.35	98.48	98.41
AdamW	98.39	98.41	98.42	98.42
Adam + ReduceLROnPlateau	98.81	98.78	98.95	98.86
AdamW + ReduceLROnPlateau	99.66	99.65	99.69	99.67

Table 10. Model Size Comparison between the Proposed Model and Other Lightweight Models.

Method	GFLOPs	Size (MB)	Num. of Parameters	Inference Time (sec.)
SqueezeNet 1.1	0.35	4.7	1,235,496	0.0041
MNASNet1_3	0.53	24.2	6,282,256	0.0077
ShuffleNetV2_x2_0	0.58	28.4	7,393,996	0.0092
MobileNet_V3	0.22	21.1	5,483,032	0.0093
EfficientNet_B0	0.39	20.1	5,288,548	0.0115
Ours	0.10	1.2	462,435	0.0060

representative lightweight models, such as SqueezeNet 1.1 [30], MNASNet1_3 [31], ShuffleNetV2_x2_0, MobileNet_V3 [32], and EfficientNet_B0. The corresponding experimental results are summarized in Tables 10–12.

As shown in Table 10, the proposed method achieves significantly lower computational complexity, parameter count, and model size compared to other lightweight models. Despite operating under strict computational and storage constraints, it maintains a competitive inference speed, with an average processing time of 0.0060 seconds per test image. While this is slightly slower than SqueezeNet 1.1 (0.0041 seconds), the proposed model offers a clear advantage over most lightweight architectures by delivering a better balance between efficiency and predictive performance.

As shown in Table 11, although the proposed model's training time is slightly longer than that of MNASNet1_3 and EfficientNet_B0, it achieves the highest average classification accuracy of 99.78% with a low standard deviation of 0.17. This demonstrates not only superior predictive performance but also strong consistency across different cross-validation folds. In contrast, the other lightweight models show considerably lower average accuracy and higher variance, reflecting reduced classification stability and generalization ability. A comparative evaluation of test performance between the proposed model and other lightweight architectures on the Rice_Leaf dataset is presented in Table 12.

5 Discussion

In addition to the experimental results mentioned earlier, this section presents the confusion matrix to highlight class-specific misclassification patterns. We also analyze and discuss the misclassified images identified during testing. The related results are illustrated in Figures 7 –9.

From the confusion matrix shown in Figure 7, we observed that our method occasionally misclassified Leaf Blast as Bacterial Leaf Blight. Additionally, Leaf Blast and Narrow Brown Spot were sometimes mistaken for Leaf Scald. To better understand these errors, we analyzed the misclassified images. The

Table 11. Comparison of Training Time and 10-fold Cross-validation Performance Across Various Lightweight Models.

Method	Training Time (sec.)	Average Accuracy (%)	Std. Dev.
SqueezeNet 1.1	235,308	97.62	0.52
MNASNet1_3	80,422	92.80	0.55
ShuffleNetV2_x2_0	319,235	95.73	0.75
MobileNet_V3	146,521	98.24	0.35
EfficientNet_B0	103,455	97.11	0.65
Ours	109,283	99.78	0.17

Table 12. Performance Comparison between the Proposed model and Various Lightweight Models.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
SqueezeNet 1.1	94.75	94.75	94.64	94.69
MNASNet1_3	92.47	93.37	92.61	92.99
ShuffleNetV2_x2_0	93.82	94.05	93.30	93.67
MobileNet_V3	93.82	94.90	93.27	94.07
EfficientNet_B0	94.84	94.87	94.70	94.78
Ours	99.66	99.65	99.69	99.67

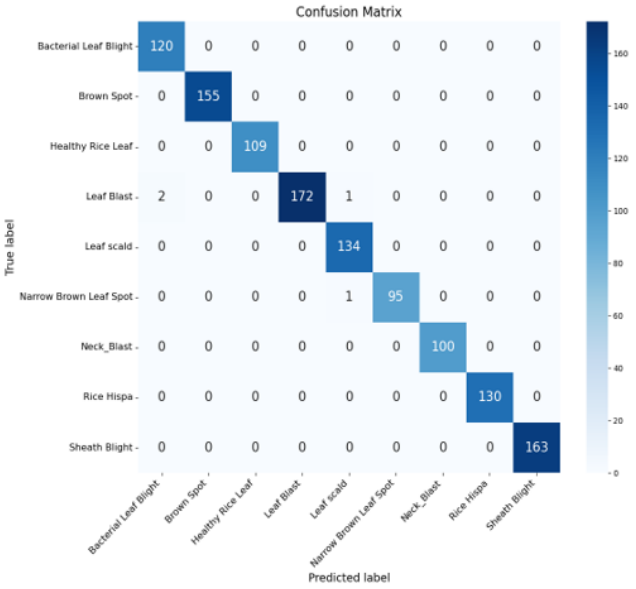

Figure 7. Confusion Matrix of Test Results for Rice Leaf Disease Classification.

(a) (b)

Figure 8. Misclassification Comparison – Part I. (a) Correct Bacterial Leaf Blight; (b) Leaf Blast Misclassified as Bacterial Leaf Blight.

comparative results are presented in Figures 8 and 9.

Figure 8 illustrates a case where our method misclassified Leaf Blast as Bacterial Leaf Blight in the Rice_Leaf dataset. In Figure 8(a), the image accurately depicts Bacterial Leaf Blight, characterized by yellowish-brown lesions with indistinct boundaries and smooth color transitions, sometimes forming a V-shaped pattern. In contrast, Figure 8(b) shows a leaf affected by Leaf Blast that the model incorrectly identified as Bacterial Leaf Blight. We believe this misclassification occurred because the lesion boundaries in the image are slightly blurred, closely resembling the visual traits of Bacterial Leaf Blight.

Figure 9 presents two types of misclassification cases observed in the Rice_Leaf dataset. As shown in Figure 9(a), the leaf is correctly labeled as Leaf Scald, which is typically characterized by alternating light and dark brown streaks appearing at the leaf tips or edges. In Figure 9(b), the leaf is actually affected by Leaf Blast, but the model misclassified it as Leaf Scald. Upon visual inspection, the leaf edge in this image also displays banded lesions similar to those found in Leaf Scald. We therefore speculate that this leaf may exhibit symptoms of both diseases, causing the misclassification due to overlapping visual features.

Figures 9(c) and 9(d) illustrate another type of misclassification. Figure 9(c) correctly shows a sample

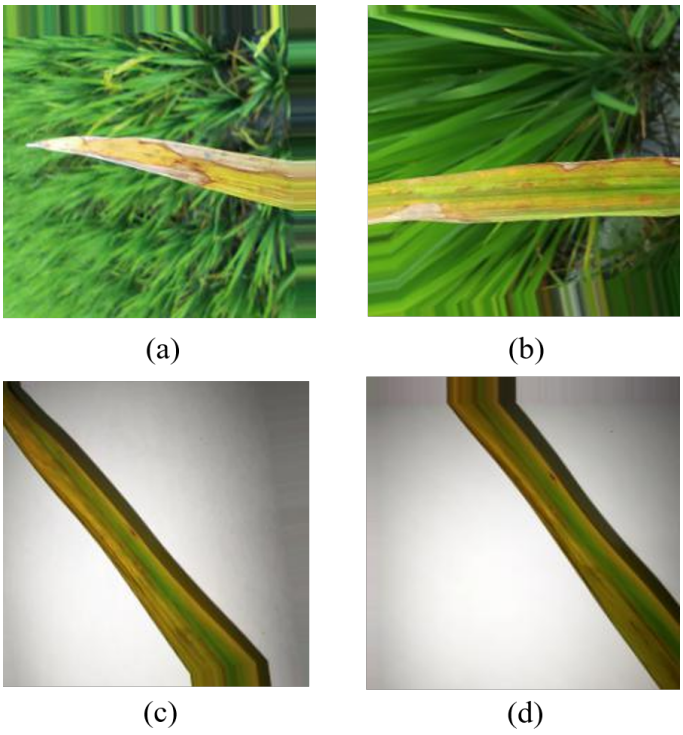

Figure 9. Misclassification Comparison – Part II. (a) Correctly Classified Image of Leaf Scald; (b) Leaf Blast Image Misclassified as Leaf Scald; (c) Correctly Classified Image of Leaf Scald; (d) Narrow Brown Spot Image Misclassified as Leaf Scald.

of Leaf Scald, while Figure 9(d) depicts a leaf affected by Narrow Brown Spot, which typically appears as short, brown lesions aligned parallel to the leaf veins. However, the misclassified image in Figure 9(d) seems to have undergone noticeable elongation during preprocessing, potentially distorting key visual cues and leading the model to mistakenly classify it as Leaf Scald. This error highlights the model's sensitivity to fine-grained features and its vulnerability to preprocessing artifacts.

These results show that fine-tuning the architecture and attention mechanisms of lightweight deep learning models can substantially boost their performance, particularly in resource-constrained environments, pointing to strong promise for real-world deployment.

6 Conclusion

With the rapid advancement of artificial intelligence and deep learning technologies, the classification of plant leaf diseases has greatly improved diagnostic efficiency while reducing the risk of human error. In this study, we proposed a rice leaf disease classification model that achieves both high accuracy and low computational cost. Building on the RegNetY architecture, we optimized key parameters—such as initial width, group width, and depth—to effectively reduce computational complexity, compressing the total GFLOPs to 0.1, thereby meeting the demands of edge computing environments.

To preserve the feature-weighting capability of channel attention mechanisms without the parameter overhead caused by the two fully connected layers in the SE module, we designed a modified ECA module. We also integrated a batch normalization layer to stabilize the learning process and further enhance model performance. Additionally, weight decay and a dynamic learning rate adjustment strategy were employed to improve convergence efficiency and generalization during training. Our model outperformed other lightweight architectures in accuracy, recall, and F1-score, while also achieving reductions in model size and inference time.

For future work, this method could be extended by incorporating a wider variety of rice leaf disease images captured under real-world field conditions, enhancing its applicability in practical scenarios. Furthermore, data augmentation techniques could be introduced to improve the model's robustness in non-ideal environments. To facilitate deployment on edge devices, the model can be packaged into a mobile application and integrated with smartphone cameras or drones for real-time disease diagnosis, further advancing the automation and operational efficiency of smart agriculture.

Data Availability Statement

The original data presented in the study are openly available in Kaggle at https://www.kaggle.com/datasets/loki4514/rice-leaf-diseases-detection.

Funding

This work was partially supported by the National Science and Technology Council, Taiwan, R.O.C., under Grant NSTC 114-2221-E-390-007.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture, 147*, 70-90. [CrossRef]

[2] Jackulin, C., & Murugavalli, S. J. M. S. (2022). A comprehensive review on detection of plant disease using machine learning and deep learning approaches. *Measurement: Sensors, 24*, 100441. [CrossRef]

[3] Dhanya, V. G., Subeesh, A., Kushwaha, N. L., Vishwakarma, D. K., Kumar, T. N., Ritika, G., & Singh, A. N. (2022). Deep learning based computer vision approaches for smart agricultural applications. *Artificial Intelligence in Agriculture, 6*, 211-229. [CrossRef]

[4] Pandian, J. A., Kumar, V. D., Geman, O., Hnatiuc, M., Arif, M., & Kanchanadevi, K. (2022). Plant disease detection using deep convolutional neural network. *Applied Sciences, 12*(14), 6982. [CrossRef]

[5] Sajitha, P., Andrushia, A. D., Anand, N., & Naser, M. Z. (2024). A review on machine learning and deep learning image-based plant disease classification for industrial farming systems. *Journal of Industrial Information Integration, 38*, 100572. [CrossRef]

[6] Prasad, K. V., Vaidya, H., Rajashekhar, C., Karekal, K. S., Sali, R., & Nisar, K. S. (2024). Multiclass classification of diseased grape leaf identification using deep convolutional neural network (DCNN) classifier. *Scientific reports, 14*(1), 9002. [CrossRef]

[7] Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., & Dollár, P. (2020). Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10428-10436).

[8] Bhujel, A., Kim, N. E., Arulmozhi, E., Basak, J. K., & Kim, H. T. (2022). A lightweight attention-based convolutional neural networks for tomato leaf disease classification. *Agriculture, 12*(2), 228. [CrossRef]

[9] Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018, September). CBAM: Convolutional Block Attention Module. In *European Conference on Computer Vision* (pp. 3-19). Cham: Springer International Publishing. [CrossRef]

[10] Zhou, Y., Fu, C., Zhai, Y., Li, J., Jin, Z., & Xu, Y. (2023). Identification of rice leaf disease using improved ShuffleNet V2. *Computers, Materials & Continua, 75*(2), 4501–4517. [CrossRef]

[11] Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018, September). ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *European Conference on Computer Vision* (pp. 122-138). Cham: Springer International Publishing. [CrossRef]

[12] Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020, June). GhostNet: More Features From Cheap Operations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1577-1586). IEEE. [CrossRef]

[13] Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., & Hu, Q. (2020, June). ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11531-11539). IEEE. [CrossRef]

[14] Brawijaya, H., Rahmawati, E., & Haryanto, T. (2024). Optimization of Potato Leaf Disease Identification With Transfer Learning Approach Using Mobilenetv1 Architecture. *Jurnal Pilar Nusa Mandiri, 20*(1), 33-40. [CrossRef]

[15] Bijoy, M. H., Hasan, N., Biswas, M., Mazumdar, S., Jimenez, A., Ahmed, F., ... & Momen, S. (2024). Towards sustainable agriculture: a novel approach for rice leaf disease detection using dCNN and enhanced dataset. *IEEE Access, 12*, 34174-34191. [CrossRef]

[16] Sethy, P. K., Barpanda, N. K., Rath, A. K., & Behera, S. K. (2020). Deep feature based rice leaf disease identification using support vector machine. *Computers and Electronics in Agriculture, 175*, 105527. [CrossRef]

[17] Bari, B. S., Islam, M. N., Rashid, M., Hasan, M. J., Razman, M. A. M., Musa, R. M., ... & Majeed, A. P. A. (2021). A real-time approach of diagnosing rice leaf disease using deep learning-based faster R-CNN framework. *PeerJ Computer Science, 7*, e432. [CrossRef]

[18] Haque, M. E., Rahman, A., Junaeid, I., Hoque, S. U., & Paul, M. (2022). Rice leaf disease classification and detection using yolov5. *arXiv preprint arXiv:2209.01579*. [arXiv]

[19] Ahad, M. T., Li, Y., Song, B., & Bhuiyan, T. (2023). Comparison of CNN-based deep learning architectures for rice diseases classification. *Artificial Intelligence in Agriculture, 9*, 22-35. [CrossRef]

[20] Ritharson, P. I., Raimond, K., Mary, X. A., & Robert, J. E. (2024). DeepRice: A deep learning and deep feature based classification of Rice leaf disease subtypes. *Artificial Intelligence in Agriculture, 11*, 34-49. [CrossRef]

[21] Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2019). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(8), 2011-2023. [CrossRef]

[22] Basha, S. S., Farazuddin, M., Pulabaigari, V., Dubey, S. R., & Mukherjee, S. (2024). Deep model compression based on the training history. *Neurocomputing, 573*, 127257. [CrossRef]

[23] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr.

[24] Ying, X. (2019, February). An overview of overfitting

and its solutions. In *Journal of physics: Conference series* (Vol. 1168, No. 2, p. 022022). IOP Publishing. [CrossRef]

[25] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. [arXiv]

[26] Li, L., Zhang, S., & Wang, B. (2021). Plant disease detection and classification by deep learning—a review. *IEEE Access*, *9*, 56683-56698. [CrossRef]

[27] Kaggle. (n.d.). Rice Leaf Diseases Detection [Data set]. Retrieved May 20, 2025, from https://www.kaggle.com/datasets/loki4514/rice-leaf-diseases-detection

[28] Park, J., Woo, S., Lee, J. Y., & Kweon, I. S. (2018). Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*. [arXiv]

[29] Yang, L., Zhang, R. Y., Li, L., & Xie, X. (2021, July). Simam: A simple, parameter-free attention module for convolutional neural networks. In *International conference on machine learning* (pp. 11863-11874). PMLR.

[30] Liu, Y., Li, Z., Chen, X., Gong, G., & Lu, H. (2020, May). Improving the accuracy of SqueezeNet with negligible extra computational cost. In *2020 International Conference on High Performance Big Data and Intelligent Systems* (HPBD&IS) (pp. 1-6). IEEE. [CrossRef]

[31] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019, June). MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*) (pp. 2815-2823). IEEE. [CrossRef]

[32] Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L. C., Tan, M., ... & Le, Q. (2019, October). Searching for MobileNetV3. In *2019 IEEE/CVF International Conference on Computer Vision* (*ICCV*) (pp. 1314-1324). IEEE. [CrossRef]

Chao-Yun Chang received the B.S. degree in green energy and information technology from National Taitung University, Taitung, Taiwan, in 2023 and the M.S. degree in Electrical Engineering from the National University of Kaohsiung, Kaohsiung, Taiwan, in 2025. He is currently an Engineer with the Bumping CIM Department, ASE Group, Kaohsiung, Taiwan. His research interests include artificial intelligence and image processing.(Email: kevin900709@gmail.com)

Chih-Chin Lai received the B.S. degree in computer science and information engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1993 and the Ph.D. degree in computer science and information engineering from National Central University, Chungli, Taiwan, in 1999. He is currently a Professor with the Department of Electrical Engineering, National University of Kaohsiung, Kaohsiung, Taiwan. His research interests include computer vision, pattern recognition, and lightweight deep learning models.(Email: cclai@nuk.edu.tw)