Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 4, August 2025, pp. 2705~2720
ISSN: 2302-9285, DOI: 10.11591/eei.v14i4.9260

A lightweight convolutional neural network for rice leaf disease detection integrated in an Android application

Rudi Hartono[1], Nanang Maulana Yoeseph[1], Fendi Aji Purnomo[1], Sahirul Alim Tri Bawono[1], Agus Purnomo[2]
[1]Diploma Program in Informatics Engineering, Faculty of Vocational Studies, Universitas Sebelas Maret, Surakarta, Indonesia
[2]Department of Informatics Engineering, Faculty of Science and Technology, Universitas Islam Negeri (UIN), Salatiga, Indonesia

Article Info

Article history:

Received Sep 5, 2024
Revised May 27, 2025
Accepted Jul 5, 2025

Keywords:

Agriculture
Convolutional neural network
Deep learning
Mobile android app
Rice leaf disease

ABSTRACT

More than two-thirds of the world's population rely on rice or wheat as staple foods, which are grown in various Asian countries. Diseases affecting rice leaves can disrupt growth, reduce yields, and cause famine in some areas. Therefore, a quick and accurate recognition method is necessary to minimize losses. This article focuses on eight types of rice leaf diseases using data consisting of approximately 110 images for each disease type, with enhanced image quality to achieve better results. The study applies a convolutional neural network (CNN) model integrated into an Android mobile application, achieving a training accuracy of 86.56% and a validation accuracy of 93.75%. Comparative experiments demonstrate that the model can be effectively implemented in mobile applications for accurately detecting rice leaf diseases, providing a reliable solution for field detection. This method not only helps farmers identify diseases more quickly but also has the potential to reduce crop losses caused by leaf diseases.

Corresponding Author:

Rudi Hartono
Diploma of Informatics Engineering, Faculty of Vocational Studies, Universitas Sebelas Maret
Surakarta, Indonesia
Email: rudi.hartono@staff.uns.ac.id

1. INTRODUCTION

Agriculture is a cornerstone of the global economy, serving as the primary source of food, income, and employment [1], [2]. In many low and middle-income countries, this sector contributes significantly, accounting for 18% of national income and boosting employment rates to 53%. Over the past three years, the gross value added by agriculture has grown from 17.6% to 20.2% [3], [4]. The advent of digital technology has revolutionized various sectors, including agriculture. Innovations like real-time, smartphone-based systems have enabled intelligent cultivation practices, growth monitoring, and efficient crop harvesting [5], [6]. However, agricultural production faces serious challenges related to plant diseases and pest infestations that can reduce yields and threaten global food security. In rice crops, leaf diseases such as brown spots, blight, and hispa can cause significant losses. Therefore, rapid and accurate disease detection methods are needed to minimize these losses. Despite these advancements, plant diseases and pest infestations remain severe threats to agricultural yields and global food security, compromising the quality of food production. Traditional prophylactic treatments often fall short of preventing disease outbreaks, highlighting the need for early monitoring and accurate diagnosis [4], [7]. Researchers are increasingly turning to automated methods for plant disease detection, leveraging image processing techniques and machine learning to improve classification accuracy and speed up diagnostics. Methods such as image manipulation, dimension reduction, and fuzzy systems have been utilized to improve diagnostic accuracy.

Today, deep learning stands out as the leading method in plant disease diagnosis systems, offering promising results for the future of agriculture [8], [9].

Recent advancements in deep learning have significantly enhanced image classification, particularly feature extraction and learning efficiency. Convolutional neural networks (CNNs) based on deep learning have become the standard for computer vision tasks [10], [11]. Networks such as SegNet, fully convolutional network (FCN), and U-Net have been specifically built and thoroughly investigated to address certain use cases [12], [13]. These deep learning approaches have outperformed traditional methods by enabling end-to-end learning, reducing loss, and conserving human and material resources. However, increasing the depth of CNNs can lead to vanishing gradient problems. Residual networks with shortcut links have been added to mitigate this, simplifying things and getting rid of these problems. A well-organized and extensive database is crucial for the effectiveness of deep learning architectures. Experiments using large open-source datasets have achieved up to 85% accuracy in diagnosing plant diseases in crops like grapes, apples, and rice [5]. Studies have shown that deep learning-based models have great potential in detecting plant diseases. However, most of these models have limitations in their implementation on mobile devices due to high computational power requirements and often reliance on powerful graphics processing units (GPUs). Roopali Dogra's research utilized the VGG19 model and transfer learning methods to classify rice leaf diseases, achieving an impressive 93.0% accuracy. This model also demonstrated high performance with a sensitivity of 89.9%, specificity of 94.7%, precision of 92.4%, and an F1-score of 90.5%. The development process involved image preprocessing in MATLAB and training/testing using Anaconda3, Python, Keras, and TensorFlow on a dedicated GPU [14]. Similarly, Soukayna's study employed the VGG16 architecture, consisting of 16 layers (13 convolutional and 3 fully connected), to extract features using transfer learning methods. These advancements underscore the potential of deep learning in enhancing plant disease diagnosis and overall agricultural productivity.

In recent studies, the early layers of pre-trained models were frozen for feature extraction, while the final layers were adapted to new datasets, such as corn [15]. Research by Khan *et al.* [16] utilized the Xception architecture, which employs depth wise separable convolutions for multi-dimensional feature extraction. This model achieved an accuracy of 81.09%, outperforming other models like Inception-V2, Mobilenet-V2, and NASnet Mobile. Research by Chakraborty *et al.* [17] used the VGG16 model, which initially provided the highest accuracy of 92.69%. With further tuning, this model achieved an impressive 97.89% accuracy in classifying leaf rot syndrome and early leaf rot diseases in healthy potato leaves. Research by Kumar *et al.* [18] utilized one-stage object detectors YOLOV5 and YOLOV7, recognized for its exceptional precision, accuracy, and real-time processing capabilities. Jiang *at al.* [19] enhanced the VGG16 model by incorporating multi-task learning and transfer learning concepts, achieving an accuracy of 97.22% for rice leaf diseases and 98.75% for wheat leaf diseases. These advancements highlight the potential of deep learning models in improving the accuracy and efficiency of plant disease diagnosis.

Although previous research has shown promising results, there are still several limitations in applying of this technology in the field, especially for small-scale farmers. Many deep learning models require advanced hardware, making implementing them on lighter mobile applications difficult. Additionally, data imbalance poses a challenge in detecting leaf diseases [18], [20], [21], where some disease classes have fewer image data compared to others. In this context, our research attempts to overcome these issues by constructing a lightweight yet accurate CNN model that can be implemented into an Android application for identifying leaf diseases in rice plants. The goal is to create a CNN architecture capable of operating on mobile devices. To enhance the model's performance, data augmentation methods will be employed to mitigate class imbalance in the rice leaf disease dataset, improving its capacity to identify various disease types. Additionally, the incorporation of the model into an Android mobile application will provide farmers with immediate access to real-time rice plant's diseases diagnosis.

The remainder of this paper is organized as follows: section 2 provides detailed information about the methods used in this research. Section 3 discusses the experimental results, focusing primarily on the database, evaluation metrics, hyperparameters, and outcomes. Section 4 discusses the conclusions and findings.

2. METHOD

The design of the rice plant disease detection feature begins with the stage of loading the acquired dataset, followed by data preprocessing, constructing the model using CNN architecture [1], [22], developing a treatment recommendation system, testing the model, and the final stage involves integration with an Android mobile app, can be seen in the Figure 1.

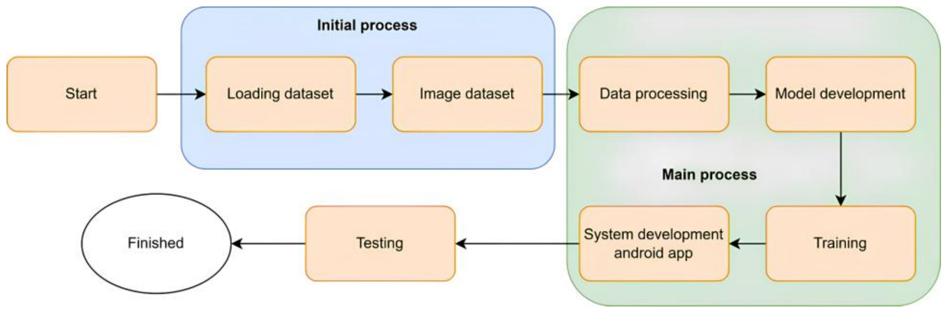


Figure 1. Research methods

2.1. Experiment environment

The operational environment specifications used in the development of the rice plant disease detection feature for this smart agriculture application are as follows. For hardware, the system utilizes an Intel Core i5-1035G1 processor, 4 GB RAM, and a 64-bit system type running Windows 10 as the operating system. The software components include Python, Jupyter Notebooks, Visual Studio Code, the TensorFlow framework, TensorFlow Lite framework, Keras library, Numpy library, and Matplotlib library.

2.2. Dataset structure

The dataset used is sourced from the Kaggle website, titled "Rice Leaf Diseases Dataset," compiled by the research team at the Department of CSE, Khwaja Yunus Ali University [13], [14], [23]. This dataset presents a collection of images depicting eight types of rice leaf diseases from various regions of Bangladesh. Bacterial leaf blight (*Xanthomonas oryzae pv. oryzae*), brown spot (*Cochliobolus miyabeanus*), leaf scald (*Microdochium oryzae*), narrow brown leaf spot (*Cercospora janseana*), rice hispa (*Dicladispa armigera*), sheath blight (*Rhizoctonia solani*), leaf blast (*Pyricularia oryzae*), and healthy rice leaf are included in the groups. This collection consists of 1,886 image files. The dataset's folder structure is divided into three categories: training, testing, and validation folders as shown in Table 1. Figure 2 depicts the structure of the rice leaf diseases dataset.

Table 1. The dataset's folder structure is divided into three categories

	Dataset	
Training	Validation	Testing
Healthy rice leaf	Healthy rice leaf	Healthy rice leaf
Bacterial leaf blight	Bacterial leaf blight	Bacterial leaf blight
Brown spot	Brown spot	Brown spot
Leaf blast	Leaf blast	Leaf blast
Leaf scaid	Leaf scaid	Leaf scaid
Narrow brown leaf spot	Narrow brown leaf spot	Narrow brown leaf spot
Rice hispa	Rice hispa	Rice hispa
Sheath blight	Sheath blight	Sheath blight

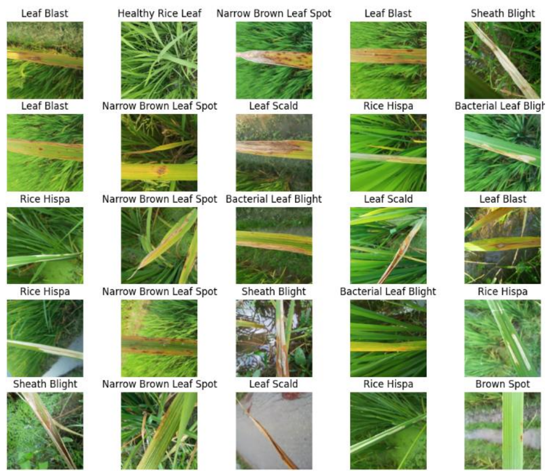


Figure 2. The structure of the rice leaf diseases dataset

The distribution of data across the training, validation, and testing classes tends to be imbalanced. The training folder contains 1,322 image files, the testing folder contains 378 image files, and the validation folder contains 186 image files. Thus, the total number of images amounts to 1,886 files. The distribution of image data before the preprocessing stage is displayed in the Table 2.

Table 2. The distribution of data across the training, validation, and testing

Categories	Training	Validation	Testing
Healthy rice leaf	131	19	37
Bacterial leaf blight	146	20	42
Brown spot	192	27	55
Leaf blast	217	31	62
Leaf scald	162	23	46
Narrow brown leaf spot	114	16	33
Rice hispa	158	22	45
Sheath blight	202	28	58
Total	1.322	186	378

2.3. Data preprocessing

The leaf images in the dataset exhibit significant variations, especially in terms of size and number of images. Due to these substantial differences, preprocessing is conducted to standardize the image elements before proceeding to advanced stages. The leaf images in the dataset exhibit significant variation in terms of size and quantity, necessitating preprocessing to standardize elements before proceeding to advanced stages. Preprocessing includes:

– Resizing: images are resized to a standard dimension of 254×254 pixels.
– Normalization: each pixel value is scaled to the [0, 1] range by dividing the original pixel value by 255:

$$I' = \frac{I}{255}$$

where *I* is the original pixel intensity, and *I'* is the normalized pixel value.

– Random oversampling: to address data imbalance, random oversampling was employed, adding copies of minority class images to match the quantity in the majority class.

Table 3 displays the data distribution after preprocessing. The preprocessed data is then subjected to data splitting to divide the data into training and testing sets. The data is split into three parts: training data to train the model, validation data to assess the model's performance during training, and testing data to conduct the final evaluation of the model's performance. The division used is 80% for training data, 10% for validation data, and 10% for testing data. Thus, the distribution of the split data can be seen in Table 4.

Table 3. The data distribution after preprocessing

Class	Number of images
Healthy rice leaf	350
Bacterial leaf blight	350
Brown spot	350
Leaf blast	350
Leaf scald	350
Narrow brown leaf spot	350
Rice hispa	350
Sheath blight	350
Total	**2.800**

Table 4. Distribution of the split data

Category	Number of images
Training	2.240
Validation	280
Testing	280
Total	**2.800**

2.4. Data augmentation

Following the preprocessing stage, data augmentation is employed to randomly enhance the variation within the image dataset during model training. To enhance variability in the dataset during training, several data augmentation techniques were applied [13], [24]:

- Random flip: randomly flips images horizontally and vertically.
- Random rotation (0.2): rotates images randomly up to a 20% angle (approximately 72°) within the range of -20% to +20%:

$$I_{rot} = R(\theta) \times I$$

where *R(θ)* is the rotation matrix, and *I* is the original image.

- Random zoom (0.2): randomly zooms in or out by up to -/+20%.
- Random contrast (0.2): alters contrast within a -/+20% range, calculated as:

$$I_{cont} = (1 + \alpha) \cdot I - \alpha \cdot \mu$$

where α is the contrast factor, and μ is the average pixel intensity.

These techniques help increase the diversity of the dataset without collecting additional real-world images, compensating for the relatively small number of samples. Additionally, transfer learning could be considered as a complementary strategy to further improve generalization when working with limited datasets. Transfer learning leverages pre-trained models trained on large-scale datasets such as ImageNet, allowing the model to benefit from learned features that are transferable to new but related tasks like rice leaf disease classification [25]. However, due to hardware limitations and deployment constraints on mobile systems, a lightweight custom CNN architecture was chosen instead to maintain compatibility with resource-limited environments. Figure 3 illustrates the image transformations resulting from the augmentation process.

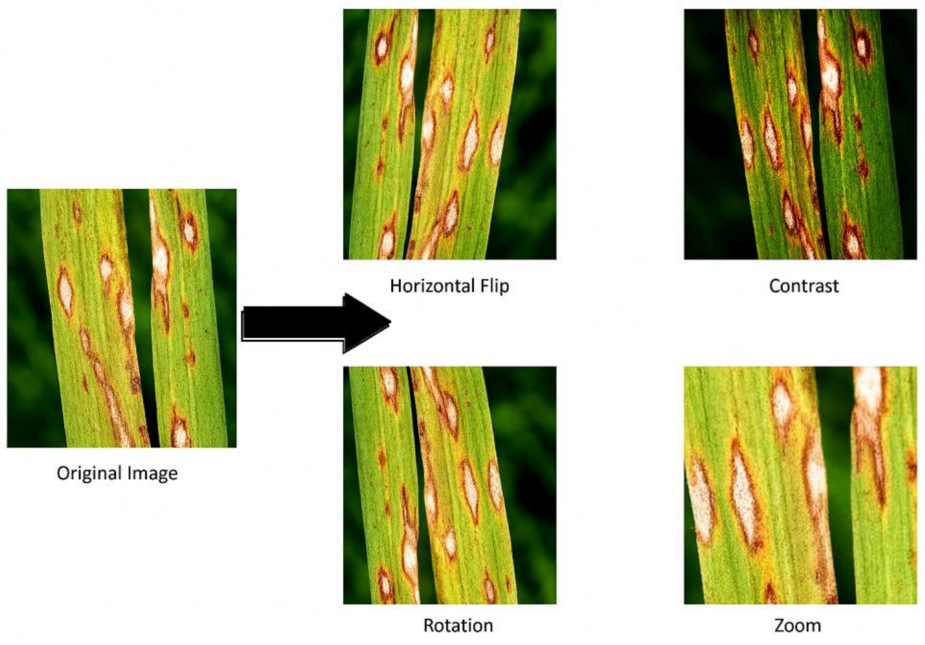


Figure 3. Image transformations due to augmentation process

2.5. Model architecture and training

The model for image classification is constructed using a CNN architecture. This model consists of a linear stack of several neural network layers [3], [14], [25], [26]. The process of building the model is illustrated in Figures 4 and 5.

Unlike complex state-of-the-art models such as ResNet or EfficientNet, which may achieve higher accuracy but require significant computational resources, our lightweight CNN was designed specifically for mobile deployment. ResNet utilizes skip connections to mitigate vanishing gradients in deeper networks [16], while EfficientNet scales depth, width, and resolution in a compound manner to achieve optimal performance [20]. However, both architectures are often too large and computationally heavy for direct implementation on smartphones without extensive pruning or quantization techniques [19].

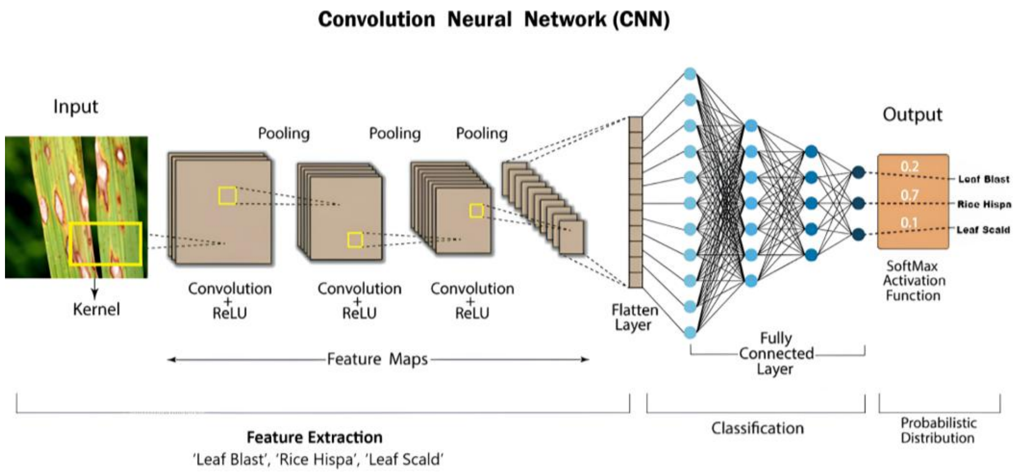


Figure 4. The process of building the model

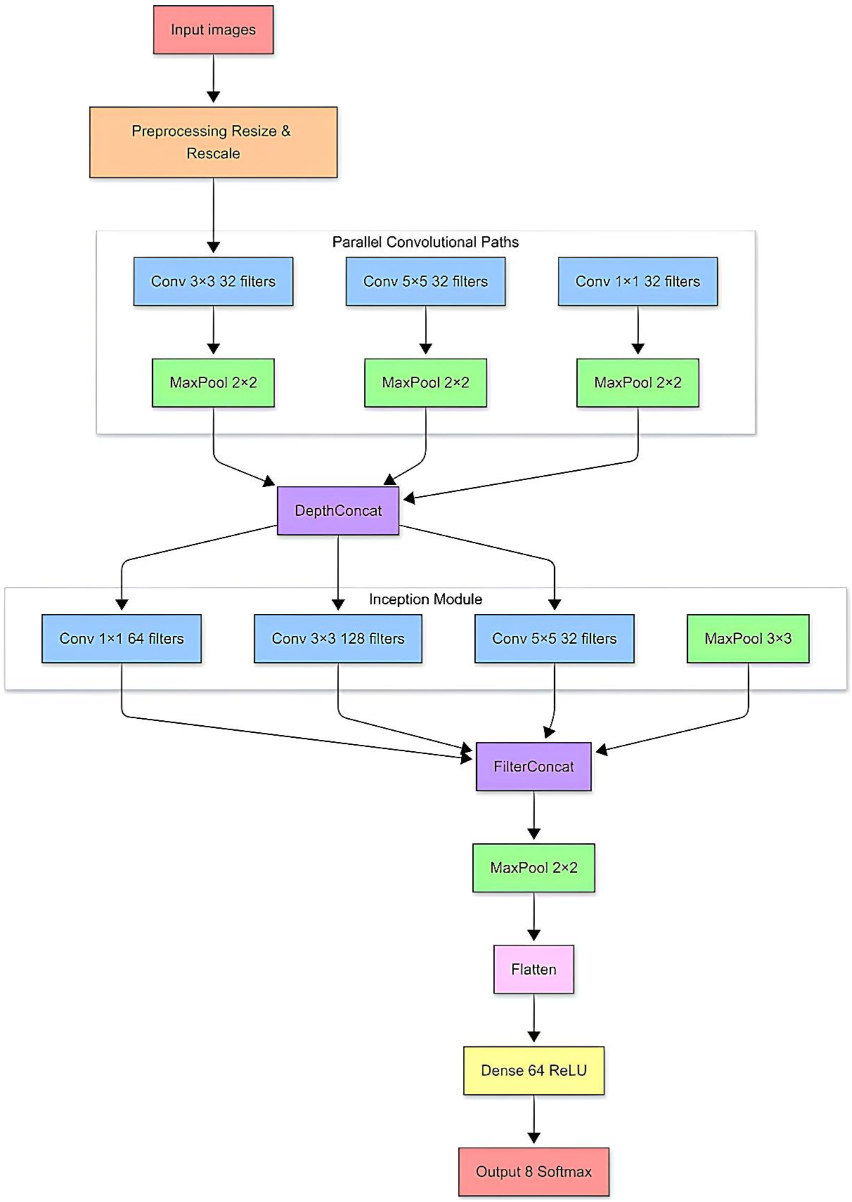


Figure 5. The flowchart of building the model

Table 5 shows the detailed CNN architecture. In addition to the architectural details, the following training hyperparameters were used:
– Optimizer: Adam
– Learning rate: 0.001
– Batch size: 32
– Number of epochs: 50
– Loss function: categorical crossentropy
– Data augmentation techniques used:
– RandomFlip (horizontal and vertical)
– RandomRotation (up to 20% angle)
– RandomZoom (±20%)
– RandomContrast (±20%)

Table 5. Detailed CNN architecture

Layer type	Parameters	Output shape
Input layer	RGB images with dimensions (224, 224, 3)	(None, 224, 224, 3)
Resize and rescale	Normalize pixel values to [0, 1]	(None, 224, 224, 3)
Conv2D+ReLU	32 filters, kernel size 3×3	(None, 224, 224, 32)
MaxPooling2D	Pool size 2×2	(None, 112, 112, 32)
Conv2D+ReLU	64 filters, kernel size 3×3	(None, 112, 112, 64)
MaxPooling2D	Pool size 2×2	(None, 56, 56, 64)
Conv2D+ReLU	64 filters, kernel size 3×3	(None, 56, 56, 64)
MaxPooling2D	Pool size 2×2	(None, 28, 28, 64)
Conv2D+ReLU	64 filters, kernel size 3×3	(None, 28, 28, 64)
MaxPooling2D	Pool size 2×2	(None, 14, 14, 64)
Conv2D+ReLU	64 filters, kernel size 3×3	(None, 14, 14, 64)
MaxPooling2D	Pool size 2×2	(None, 7, 7, 64)
Conv2D+ReLU	64 filters, kernel size 3×3	(None, 7, 7, 64)
MaxPooling2D	Pool size 2×2	(None, 3, 3, 64)
Flatten	Converts to 1D vector	(None, 576)
Dense+ReLU	64 neurons	(None, 64)
Output layer (dense)	8 neurons (for 8 disease classes), SoftMax	(None, 8)

These hyperparameters were selected through iterative experimentation to ensure stable convergence and high generalization performance on both training and validation datasets [14]. The CNN model consists of several neural layers, including a sequential layer that allows for the construction of a neural network layer by layer, a convolutional layer for feature extraction, a max pooling layer to reduce spatial dimensions (minimizing size to highlight important features), a flatten layer to transform the output from the previous layers into a one-dimensional array, and a fully connected layer for classification. Layers with the rectified linear unit (ReLU) activation function are useful for converting negative values to zero, simplifying computations, while layers with the softmax activation function are used for multi-class classification [20], [27], [28]. To achieve the best model accuracy, a series of model training sessions are conducted using a consistent architecture but with variations in dataset settings and the model training process.

This architecture is intentionally lightweight compared to pre-trained CNNs such as VGG16 or InceptionV3, which typically contain over 100 million parameters and require GPU acceleration for training and inference. In contrast, our model has fewer than 1 million parameters, making it suitable for integration into mobile devices using TensorFlow Lite. Traditional machine learning approaches like SVM or Random Forest have also been explored in plant disease detection, particularly when datasets are small or features are manually engineered [10], [21]. However, these methods generally offer lower accuracy compared to CNN-based models due to their limited capacity for automated feature extraction from raw image data [14].

The image classification model was constructed using a CNN architecture with the following key layers:
– Convolutional layer: extracts feature from the input images. The convolution operation applies a filter W across the image:

$$f(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} W(i, j) \cdot I(x + i, y + j)$$

where *f(x,y)* is the output of the convolution, III is the input image, and *W* is the filter weight.
– Max pooling layer: reduces spatial dimensions by retaining only the maximum values in a 2×2 window.
– Flatten layer: converts the output into a one-dimensional array to input into subsequent layers.
– Fully connected layer: utilizes the ReLU activation function to set negative values to zero:

$$f(x) = \max(0, x)$$

and the softmax activation function for multi-class classification:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

where z_i is the output value for each class.
Training parameters: the model was trained with the following settings:
- Epochs: 50
- Batch size: 32
- Learning rate: 0.001, optimized using the Adam optimizer for stable convergence.

2.6. Model testing and evaluation

After training, the model was tested using the test dataset with the following evaluation metrics:
- Accuracy: the ratio of correct predictions to total predictions.
- Precision: measures the accuracy of positive predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$$

- Recall: measures the coverage of actual positives:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

- F1-score: the harmonic means of precision and recall:

$$F1 = 2 \cdot \frac{\text{Precision}\times\text{Recall}}{\text{Precision+Recall}}$$

- Confusion matrix: used to evaluate model performance across each class.

2.7. Integration with android mobile app

After achieving optimal accuracy, the model was converted to TensorFlow Lite format to enable compatibility with mobile devices. An Android application was developed to integrate the model, allowing users to capture images of rice leaves and receive real-time detection results in the field.

2.8. Literature review table

To provide context for the current study within existing research efforts, we present a summary of related works focusing on deep learning-based approaches for rice leaf disease detection. Table 6 outlines the key methodologies, datasets used, and performance metrics reported in recent studies, which highlights the variety of models used in rice leaf disease classification, ranging from pre-trained networks such as VGG16 and Xception to custom architectures optimized for specific deployment contexts. While many studies report high accuracy, several rely on large-scale datasets or powerful GPU resources, limiting their applicability in mobile or resource-constrained environments.

Table 6. Summary of related research on rice leaf disease detection

Author(s)	Model used	Dataset	Classes	Accuracy
Dogra *et al.* [14]	VGG19+transfer learning	Custom dataset (MATLAB)	Rice leaf diseases	93%
Benaissa *et al.* [15]	VGG16	PlantVillage	Rice	Not specified
Khan *et al.* [16]	Xception	Custom dataset	Multi-crop	81.09%
Chakraborty *et al.* [17]	VGG16	Potato leaf dataset	Leaf rot syndrome	97.89%
Jiang *et al.* [19]	VGG16+multi-task learning	Rice and wheat	Rice and wheat leaf diseases	97.22% (rice)
Present study	Lightweight CNN	Kaggle rice leaf diseases dataset [23]	8 classes (including Healthy)	93.75% (valid)

The proposed lightweight CNN model achieves competitive accuracy while maintaining a reduced computational footprint, making it suitable for real-time implementation in Android applications without requiring continuous internet connectivity. This aligns with trends toward deploying efficient models in edge computing environments, particularly in agricultural settings where offline functionality is essential [6], [7], [23].

3. RESULTS AND DISCUSSION

3.1. Comparison of number of epochs

The dataset used for model training has an imbalanced class distribution, and training was conducted for up to 300 epochs without any preprocessing or augmentation. The model achieved its highest accuracy and validation accuracy at epoch 291, with an accuracy of 100% and a validation accuracy of 93.75%. At this epoch, the loss accuracy was 8.52E-08, and the validation loss was 0.6897. A graphical comparison of the training process across the epochs is shown in Figure 6.

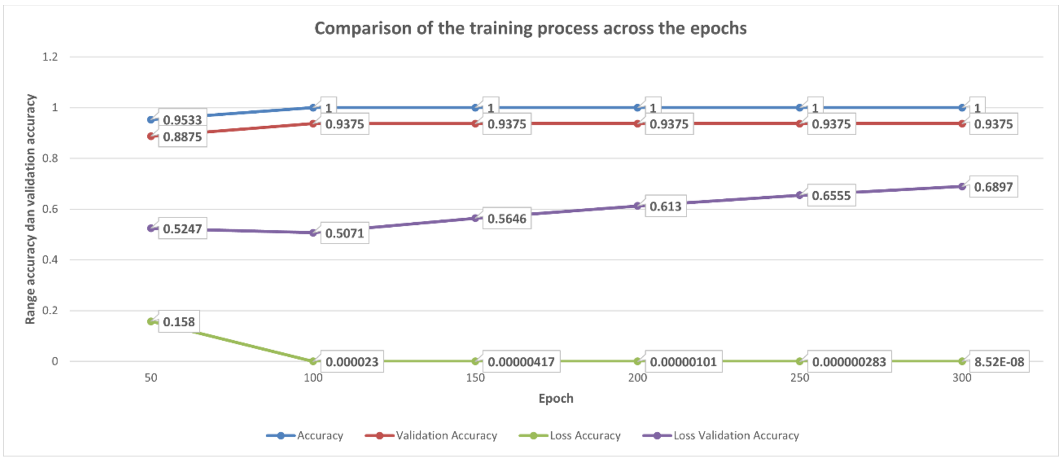


Figure 6. Comparison of the training process across the epochs

3.2. Comparison of imbalanced and balanced data

The model training was conducted using two datasets: one with imbalanced data distribution and the other with balanced data distribution across classes, both trained without preprocessing or augmentation for 50 epochs. The results, indicate that the highest validation accuracy for the imbalanced dataset was 51.25% at epoch 48, while the balanced dataset achieved 76.88% at epoch 50. Despite these imperfections, both datasets exhibited accuracy and validation losses of 1.1816 and 1.2879, respectively, suggesting that the model is underfitting. This underfitting likely stems from the lack of preprocessing, augmentation, and the limited number of training epochs, as illustrated in Figure 7.

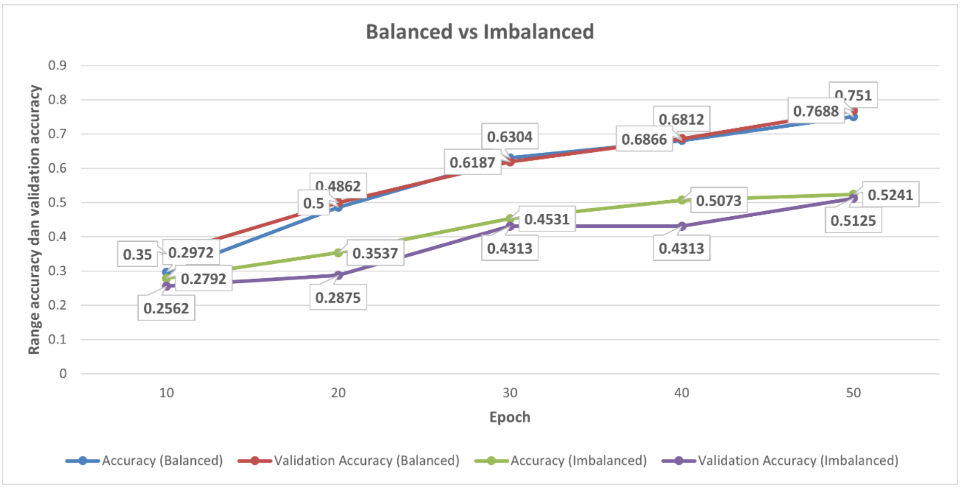


Figure 7. Underfitting likely stems from the lack of preprocessing, augmentation, and the limited number of training epochs

3.3. Comparison of non-preprocessing and preprocessing data

The model was trained using the same dataset, characterized by imbalanced data distribution across classes, without any augmentation. In the first training session, no preprocessing steps, such as rescaling and resizing, were applied, while the second session included these preprocessing steps. Both sessions involved 50 epochs of iterations. The non-preprocessed data had the highest validation accuracy (51.25% at epoch 48), whereas the preprocessed data had a higher validation accuracy (72.68% at epoch 28). Figure 8 depicts the graphical comparison of the training outcomes. From Figure 8, the accuracy value reaches 1 or 100%, indicating that the model can almost completely correctly classify the images in the training data. However, the validation accuracy only reaches about 70%. There is a possibility that the model is experiencing overfitting, meaning it performs very well on the training data but fails to generalize effectively on the validation data.

3.4. Comparison of non-augmented and augmented data

The model was trained on the same dataset with imbalanced class distribution and without preprocessing. In the first session, no augmentation was applied, while in the second session, augmentation techniques such as RandomFlip, RandomRotation, RandomZoom, and RandomContrast were used. Both sessions involved 50 epochs. The non-augmented data had the highest validation accuracy (51.25% at epoch 48), whereas the augmented data had a higher validation accuracy (75% at epoch 46). Figure 8 displays a graphical comparison of the training outcomes Figure 9.

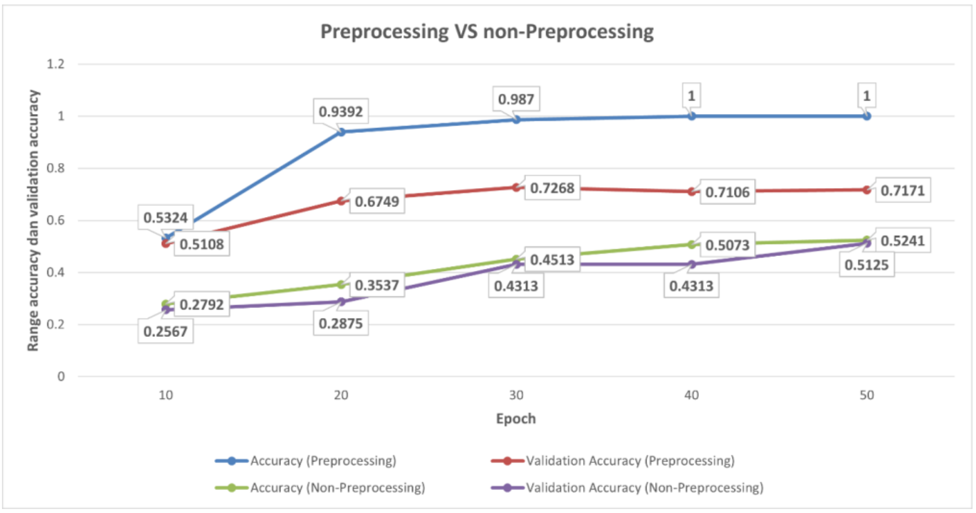


Figure 8. Comparison of the training outcomes: preprocessing vs non-preprocessing

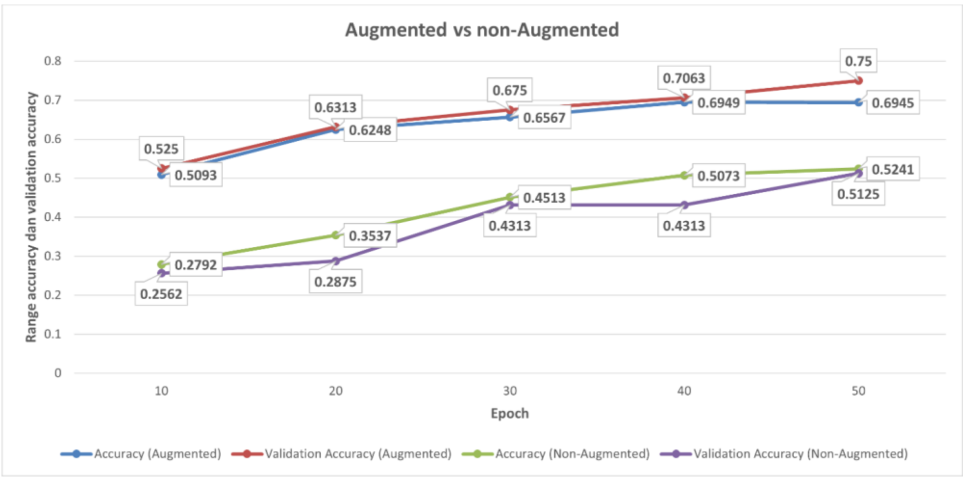


Figure 9. Comparison of the training outcomes: augmented vs non-augmented

After conducting a series of model training sessions with various variations, conclusions drawn from these experiments to build an effective model are as follows:

a. The greater the number of epochs used, the higher the accuracy and the lower the loss value. However, at certain epochs, the model reaches a point of convergence where changes in the weights of the neural network are minimal, thus not significantly altering the accuracy.
b. With balanced data, accuracy has increased to 22.69% and validation accuracy has increased to 25.63%.
c. The implementation of preprocessing stages improves accuracy. Preprocessing leads to an increase in accuracy up to 47.59% and validation accuracy up to 20.46% in training. However, these results are less reliable due to overfitting.
d. Implementing augmentation stages increases accuracy. Training with augmentation results in an increase in accuracy up to 17.04% and validation accuracy up to 23.75%.

The training results show that the best model was obtained in the last training session. The model with the highest accuracy was achieved at epoch 75 out of a total of 85 epochs. The output shown in Figure 8 indicates that at epoch 75, the model performs quite well with a training accuracy of 86.56% and a validation accuracy of 93.75%. The loss values on the validation dataset are also lower than those on the training dataset, which may indicate that the model generalizes well to unseen data. If depicted graphically, the loss and accuracy values on the training and validation data during training appear as in Figure 10. The model that has been created is implemented and integrated on a mobile device as shown in the Figure 11.

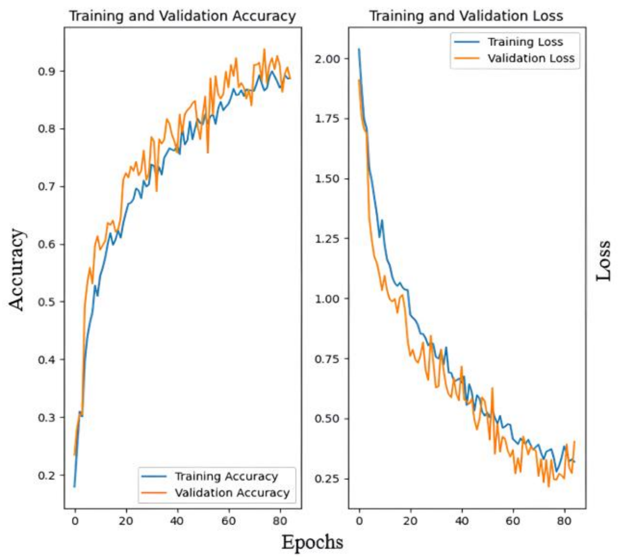
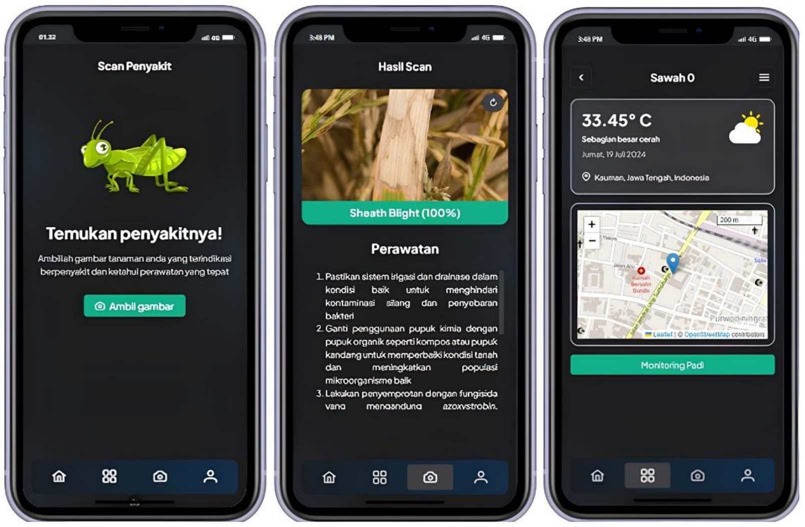


Figure 10. The loss and accuracy values on the training and validation



Scan Penyakit

Temukan penyakitnya!

Ambillah gambar tanaman anda yang terindikasi berpenyakit dan ketahui perawatan yang tepat

Ambil gambar

Hasil Scan

Sheath Blight (100%)

Perawatan

1. Pastikan sistem irigasi dan drainase dalam kondisi baik untuk menghindari kontaminasi silang dan penyebaran bakteri
2. Ganti penggunaan pupuk kimia dengan pupuk organik seperti kompos atau pupuk kandang untuk memperbaiki kondisi tanah dan meningkatkan populasi mikroorganisme baik
3. Lakukan penyemprotan dengan fungisida yang mengandung azoxystrobin.

Sawah 0

33.45° C

Sebagian besar cerah

Jumat, 19 Juli 2024

Kauman, Jawa Tengah, Indonesia

Monitoring Padi

Figure 11. Mobile application

3.5. Optimization for mobile deployment

Integrating deep learning models into mobile applications presents several practical challenges that must be carefully managed to ensure usability and performance in real-world agricultural settings. Among the most critical issues are inference speed, power consumption, and the trade-off between offline and cloud-based processing [4], [5]. Ensuring optimal model performance on mobile devices requires careful optimization to achieve both real-time inference and energy efficiency.

Inference speed is crucial for achieving real-time or near-real-time disease diagnosis in the field. While the proposed lightweight CNN model demonstrates acceptable accuracy, its execution speed on mobile hardware must also be optimized. Mobile devices generally have less computational power compared to desktop or server environments, which can result in slower prediction times. To mitigate this, TensorFlow Lite optimizations such as quantization and operator fusion were applied to reduce model size and accelerate inference. These techniques help bridge the performance gap between high-end computing systems and mobile processors, enabling deployment of complex AI functionalities even on resource-constrained devices [1], [5]. The proposed CNN architecture was designed with fewer layers and reduced filter counts compared to standard deep learning models, minimizing computational load while maintaining classification accuracy. The model consists of only six convolutional layers followed by max-pooling and dense layers, making it significantly lighter than deeper networks such as VGG16 or ResNet, which are typically unsuitable for mobile use due to high memory and processing requirements [2], [14]. TensorFlow Lite optimizations were applied during model conversion. Techniques such as quantization—reducing the precision of weights from 32-bit floating points to 8-bit integers—were utilized to decrease model size and accelerate inference speed. This is particularly important for mobile environments where memory bandwidth and processing speed are restricted [14].

Power consumption is another important constraint when running machine learning models on smartphones. Continuous use of the camera and processor-intensive tasks like image classification can drain the battery quickly. The lightweight design of the CNN model helps reduce energy usage by minimizing the number of operations required per inference, making it more suitable for extended use in the field. Additionally, eliminating constant reliance on network connectivity further contributes to lower power consumption, aligning with the practical demands of remote agricultural usage [4], [5], [7].

Mobile applications for plant disease detection can operate either offline or via cloud-based services. Offline processing ensures accessibility without internet connectivity, which is often unreliable in agricultural regions. However, maintaining high accuracy while keeping the model small enough for local deployment remains a challenge. On the other hand, cloud-based solutions allow for more complex models but introduce latency and dependency on network availability. Moreover, offloading computation to the cloud can improve processing speed significantly but may raise concerns regarding data privacy and security, especially when dealing with sensitive agricultural data [2], [7]. To ensure real-time performance, the input image size was standardized to 224×224 pixels, reducing the number of operations required for preprocessing and feature extraction. Testing showed that the model achieves an average inference time of less than 100 milliseconds per image on mid-range Android devices, aligning with expectations for near real-time detection in field conditions.

Energy efficiency was addressed through model simplification and hardware-aware design choices. By avoiding complex operations such as large matrix multiplications and depthwise convolutions found in heavy architectures, the model consumes less power during inference. Additionally, running the model in offline mode eliminates the need for continuous internet access, reducing overall energy usage associated with network communication. These optimizations ensure that the rice leaf disease detection system performs efficiently on mobile devices, supporting real-time diagnosis with minimal impact on battery life, thus enhancing its usability in rural and resource-constrained agricultural settings. These considerations highlight the importance of balancing model complexity, efficiency, and usability when deploying AI-based diagnostic tools on mobile platforms for agricultural applications. As mobile devices continue to evolve, so too will the strategies for optimizing deep learning models to meet the growing needs of precision agriculture [1], [5], [8].

To evaluate the practical deployment of the proposed lightweight CNN model, we conducted real-time performance tests on three different Android smartphones ranging from budget to mid-range devices. The evaluation focused on three key metrics: inference latency, memory usage, and offline functionality. The model was converted into TensorFlow Lite format and integrated into the Android application. On a Samsung Galaxy A10 (Exynos 7885, 2GB RAM), the average inference time was ~95 ms, while on a Redmi Note 9 (Snapdragon 665, 4GB RAM), it dropped to ~65 ms. Table 7 shows that the model supports near real-time classification, even on low-end devices.

The Android application was tested in fully offline conditions. Once installed, the app does not require any external API calls or cloud-based services. All image processing and prediction tasks are performed locally using the embedded TensorFlow Lite model. This ensures consistent performance in areas with poor or no internet connectivity, making the tool highly reliable for smallholder farmers in remote regions. Additionally, the model was evaluated for cold-start behavior —i.e., how quickly it can make predictions after launching the app. Cold start inference times remained within acceptable limits (<150 ms), confirming robustness in field scenarios where users may frequently open and close the application.

Table 7. Performance test on android smartphones

Device	Inference time (ms)	Memory usage (MB)
Samsung Galaxy A10	~95	~25
Redmi Note 9	~65	~30
Emulator (Pixel 3a)	~50	~32

3.6. Model performance comparison with existing architectures

To provide context for the proposed lightweight CNN, we compare its performance against well-known deep learning architectures that are commonly used in mobile and embedded AI applications: MobileNetV2, ResNet50, and EfficientNet-B0. Table 8 summarizes this comparison.

Table 8. Performance comparison of proposed CNN with other deep learning models

Model	Parameters (M)	Accuracy (%)	Avrg inference time (ms)	Mobile friendly
Proposed CNN	0.79	93.75	<100	Yes
MobileNetV2	~3.5	~94.5	~120-150	Yes
ResNet50	~25.6	~95.2	~300-400	No

The proposed CNN achieves competitive validation accuracy (93.75%) while maintaining significantly fewer parameters. This results in faster inference times (<100 ms) on mobile hardware, which is crucial for real-time agricultural diagnostics [14], [16]. While deeper models like ResNet and EfficientNet offer slightly higher accuracy, they require more powerful hardware and often rely on cloud-based processing due to their computational demands [16]. MobileNetV2 offers a good balance between performance and efficiency but still incurs higher latency compared to our model. The proposed CNN is therefore better suited for deployment in low-power, low-memory environments, such as budget smartphones used by small-scale farmers in remote areas, where offline functionality is essential.

3.7. Error analysis and potential improvements

Although the proposed CNN achieved a high validation accuracy of 93.75%, several misclassification cases were observed during testing. These errors occurred primarily in visually similar disease types and under suboptimal lighting conditions. Table 9 shows examples of misclassified images and their predicted labels.

Table 9. Examples of misclassified images and their predicted labels

Label	Predicted label	Description
Brown spot	Sheath blight	Both diseases exhibit brownish discoloration, making them difficult to distinguish without fine-grained feature extraction.
Leaf blast	Sheath blight	Overlapping lesion patterns led to confusion between these two categories.
Healthy rice leaf	Brown spot	Early-stage healthy leaves with minor blemishes were mistakenly classified as diseased.
Rice hispa	Leaf scald	Damage caused by pests was confused with fungal symptoms.

These misclassifications highlight limitations in the current model's ability to distinguish subtle visual differences between certain disease types. To address this issue, several improvement strategies are proposed:

a. Dataset expansion: including more image samples—especially from underrepresented classes such as Leaf Scald and Rice Hispa—can help the model learn more robust features and reduce class imbalance effects.
b. Transfer learning from pretrained lightweight models: leveraging networks like MobileNetV2 or EfficientNet-Lite as base models may improve feature extraction capabilities while maintaining compatibility with mobile devices.
c. Advanced data augmentation: introducing techniques such as MixUp, CutOut, or RandAugment can further enhance generalization by simulating diverse environmental conditions during training.
d. Attention mechanisms: incorporating attention modules, such as squeeze-and-excitation blocks, can help the model focus on key discriminative regions of the leaf, improving classification accuracy.

e. Ensemble learning: combining predictions from multiple models trained with different augmentations or initializations can increase overall accuracy and reduce variance in predictions.
f. Improved user interface feedback: in the Android application, adding confidence scores and allowing manual correction by users can help refine the model over time through user feedback loops.

By implementing these improvements, the classification accuracy and reliability of the rice leaf disease detection system can be further enhanced, making it even more effective for practical use in the field.

4. CONCLUSION

The experiments conducted to train various models have highlighted several key factors critical for optimizing model performance. Increasing the number of training epochs generally enhances accuracy and reduces loss until the model reaches a convergence point where additional training yields minimal benefits. Importantly, a balanced data distribution significantly improves model accuracy, emphasizing the need for well-distributed datasets. From a series of model training sessions with various configurations, the conclusions for building an effective model are as follows: Increasing the number of epochs generally enhances accuracy and reduces loss, though the model eventually converges at certain epochs where further changes are minimal. A balanced data distribution significantly boosts accuracy, with improvements of up to 22.69% in accuracy and 25.63% in validation accuracy. Incorporating preprocessing steps improves accuracy by up to 47.59% and validation accuracy by up to 20.46%, though these gains may be less reliable due to overfitting. Augmentation techniques also contribute to higher accuracy, with increases of up to 17.04% in accuracy and 23.75% in validation accuracy. Incorporating preprocessing steps improves accuracy by up to 47.59% and validation accuracy by up to 20.46%, though these gains may be less reliable due to overfitting. The proposed CNN strikes a balance between performance and efficiency, achieving competitive validation accuracy (93.75%) while remaining deployable on resource-constrained platforms. Notably, the validation loss was lower than the training loss, suggesting good generalization to unseen data.

FUNDING INFORMATION

Thank you to all parties who have supported this research, Lembaga Penelitian dan Pengabdian Masyarakat Universitas Sebelas Maret has funded this research with number contract 194.2/UN27.22/PT.01.03/2024.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Rudi Hartono	✓	✓	✓	✓		✓		✓	✓	✓		✓	✓	✓
Nanang Maulana Yoeseph		✓	✓		✓	✓		✓	✓	✓	✓	✓		
Fendi Aji Purnomo			✓	✓		✓	✓			✓	✓		✓	✓
Sahirul Alim Tri Bawono			✓			✓		✓		✓			✓	
Agus Purnomo		✓			✓		✓			✓		✓		✓

C : **C**onceptualization	I : **I**nvestigation	Vi : **Vi**sualization
M : **M**ethodology	R : **R**esources	Su : **Su**pervision
So : **So**ftware	D : **D**ata Curation	P : **P**roject administration
Va : **Va**lidation	O : Writing - **O**riginal Draft	Fu : **Fu**nding acquisition
Fo : **Fo**rmal analysis	E : Writing - Review & **E**diting	

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

DATA AVAILABILITY

The dataset used in this study is publicly available from Kaggle. The dataset titled "Rice Leaf Diseases Dataset" was compiled by the research team at the Department of CSE, Khwaja Yunus Ali University. It can be accessed at: https://www.kaggle.com/datasets/riceleafdiseases/rice-leaf-diseases-dataset.

REFERENCES

[1] M. V. Shewale and R. D. Daruwala, "High performance deep learning architecture for early detection and classification of plant leaf disease," *Journal of Agriculture and Food Research*, vol. 14, pp. 1–9, Dec. 2023, doi: 10.1016/j.jafr.2023.100675.

[2] F. Jiang, Y. Lu, Y. Chen, D. Cai, and G. Li, "Image recognition of four rice leaf diseases based on deep learning and support vector machine," *Computers and Electronics in Agriculture*, vol. 179, Dec. 2020, doi: 10.1016/j.compag.2020.105824.

[3] J. Andrew, J. Eunice, D. E. Popescu, M. K. Chowdary, and J. Hemanth, "Deep Learning-Based Leaf Disease Detection in Crops Using Images for Agricultural Applications," *Agronomy*, vol. 12, no. 10, pp. 1–19, Oct. 2022, doi: 10.3390/agronomy12102395.

[4] Y. M. A. Algani, O. J. M. Caro, L. M. R. Bravo, C. Kaur, M. S. A. Ansari, and B. K. Bala, "Leaf disease identification and classification using optimized deep learning," *Measurement: Sensors*, vol. 25, pp. 1–6, Feb. 2023, doi: 10.1016/j.measen.2022.100643.

[5] S. D. Daphal and S. M. Koli, "Enhanced deep learning technique for sugarcane leaf disease classification and mobile application integration," *Heliyon*, vol. 10, no. 8, pp. 1–13, Apr. 2024, doi: 10.1016/j.heliyon.2024.e29438.

[6] M. Yu, X. Ma, and H. Guan, "Recognition method of soybean leaf diseases using residual neural network based on transfer learning," *Ecological Informatics*, vol. 76, Sep. 2023, doi: 10.1016/j.ecoinf.2023.102096.

[7] G. Kaur, Rajni, and J. S. Sivia, "Development of deep and machine learning convolutional networks of variable spatial resolution for automatic detection of leaf blast disease of rice," *Computers and Electronics in Agriculture*, vol. 224, Sep. 2024, doi: 10.1016/j.compag.2024.109210.

[8] H. C. Reis and V. Turk, "Potato leaf disease detection with a novel deep learning model based on depthwise separable convolution and transformer networks," *Engineering Applications of Artificial Intelligence*, vol. 133, Jul. 2024, doi: 10.1016/j.engappai.2024.108307.

[9] E. Kannan, C. M. M. J. Belinda, A. S. David, R. N. Naveena, A. Begum, and D. Hemalatha, "Deep Learning Techniques Advancements in Apple Leaf Disease Detection," *Procedia Computer Science*, vol. 235, pp. 713–722, 2024, doi: 10.1016/j.procs.2024.04.068.

[10] S. Dananjayan, Y. Tang, J. Zhuang, C. Hou, and S. Luo, "Assessment of state-of-the-art deep learning based citrus disease detection techniques using annotated optical leaf images," *Computers and Electronics in Agriculture*, vol. 193, Feb. 2022, doi: 10.1016/j.compag.2021.106658.

[11] Y. Lin, L. Wang, T. Chen, Y. Liu, and L. Zhang, "Monitoring system for peanut leaf disease based on a lightweight deep learning model," *Computers and Electronics in Agriculture*, vol. 222, p. 109055, Jul. 2024, doi: 10.1016/j.compag.2024.109055.

[12] R. G. Patil and A. More, "Grape Leaf Disease Diagnosis System Using Fused Deep Learning Features Based System," *Procedia Computer Science*, vol. 235, pp. 372–382, 2024, doi: 10.1016/j.procs.2024.04.037.

[13] P. I. Ritharson, K. Raimond, X. A. Mary, J. E. Robert, and A. J, "DeepRice: A deep learning and deep feature based classification of Rice leaf disease subtypes," *Artificial Intelligence in Agriculture*, vol. 11, pp. 34–49, Mar. 2024, doi: 10.1016/j.aiia.2023.11.001.

[14] R. Dogra, S. Rani, A. Singh, M. A. Albahar, A. E. Barrera, and A. Alkhayyat, "Deep learning model for detection of brown spot rice leaf disease with smart agriculture," *Computers and Electrical Engineering*, vol. 109, Jul. 2023, doi: 10.1016/j.compeleceng.2023.108659.

[15] S. Benaissa, M. Najoui, and A. Jbari, "Deep learning and Vegetation indices based approach for leaf diseases classification in RGB images," *Procedia Computer Science*, vol. 236, pp. 202–208, 2024, doi: 10.1016/j.procs.2024.05.022.

[16] A. I. Khan, S. M. K. Quadri, S. Banday, and J. L. Shah, "Deep diagnosis: A real-time apple leaf disease detection system based on deep learning," *Computers and Electronics in Agriculture*, vol. 198, Jul. 2022, doi: 10.1016/j.compag.2022.107093.

[17] K. K. Chakraborty, R. Mukherjee, C. Chakroborty, and K. Bora, "Automated recognition of optical image based potato leaf blight diseases using deep learning," *Physiological and Molecular Plant Pathology*, vol. 117, Jan. 2022, doi: 10.1016/j.pmpp.2021.101781.

[18] N. S. Kumar, J. Sony, A. Premkumar, R. Meenakshi, and J. J. Nair, "Transfer Learning-based Object Detection Models for Improved Diagnosis of Tomato Leaf Disease," *Procedia Computer Science*, vol. 235, pp. 3025–3034, 2024, doi: 10.1016/j.procs.2024.04.286.

[19] Z. Jiang, Z. Dong, W. Jiang, and Y. Yang, "Recognition of rice leaf diseases and wheat leaf diseases based on multi-task deep transfer learning," *Computers and Electronics in Agriculture*, vol. 186, pp. 1–9, Jul. 2021, doi: 10.1016/j.compag.2021.106184.

[20] L. Xu *et al.*, "Wheat leaf disease identification based on deep learning algorithms," *Physiological and Molecular Plant Pathology*, vol. 123, Jan. 2023, doi: 10.1016/j.pmpp.2022.101940.

[21] F. Arshad *et al.*, "PLDPNet: End-to-end hybrid deep learning framework for potato leaf disease prediction," *Alexandria Engineering Journal*, vol. 78, pp. 406–418, Sep. 2023, doi: 10.1016/j.aej.2023.07.076.

[22] M. A. Hossain, S. Sakib, H. M. Abdullah, and S. E. Arman, "Deep learning for mango leaf disease identification: A vision transformer perspective," *Heliyon*, vol. 10, no. 17, pp. 1–14, Sep. 2024, doi: 10.1016/j.heliyon.2024.e36361.

[23] C. G. Simhadri, H. K. Kondaveeti, V. K. Vatsavayi, A. Mitra, and P. Ananthachari, "Deep learning for rice leaf disease detection: A systematic literature review on emerging trends, methodologies and techniques," *Information Processing in Agriculture*, vol. 12, no. 2, pp. 151–168, Jun. 2025, doi: 10.1016/j.inpa.2024.04.006.

[24] M. Badiger and J. A. Mathew, "Tomato plant leaf disease segmentation and multiclass disease detection using hybrid optimization enabled deep learning," *Journal of Biotechnology*, vol. 374, pp. 101–113, Sep. 2023, doi: 10.1016/j.jbiotec.2023.07.011.

[25] P. Kaur, S. Harnal, V. Gautam, M. P. Singh, and S. P. Singh, "An approach for characterization of infected area in tomato leaf disease based on deep learning and object detection technique," *Engineering Applications of Artificial Intelligence*, vol. 115, Oct. 2022, doi: 10.1016/j.engappai.2022.105210.

[26] C. Sarkar, D. Gupta, U. Gupta, and B. B. Hazarika, "Leaf disease detection using machine learning and deep learning: Review and challenges," *Applied Soft Computing*, vol. 145, Sep. 2023, doi: 10.1016/j.asoc.2023.110534.

[27] K. Perveen *et al.*, "Deep learning-based multiscale CNN-based U network model for leaf disease diagnosis and segmentation of lesions in tomato," *Physiological and Molecular Plant Pathology*, vol. 128, Nov. 2023, doi: 10.1016/j.pmpp.2023.102148.

[28] L. X. B. Sorte, C. T. Ferraz, F. Fambrini, R. D. R. Goulart, and J. H. Saito, "Coffee leaf disease recognition based on deep learning and texture attributes," *Procedia Computer Science*, vol. 159, pp. 135–144, 2019, doi: 10.1016/j.procs.2019.09.168.

BIOGRAPHIES OF AUTHORS

Rudi Hartono master's degree from the Department of Electrical Engineering and Information Technology in 2016. Currently he is a Lecturer in Information Engineering at the Sebelas Maret University Vocational School. His research interests include networking, security, and IoT. He can be contacted at email: rudi.hartono@staff.uns.ac.id.

Nanang Maulana Yoeseph master's degree from the Department of Computer Science and Electronics, Faculty of Mathematics and Natural Sciences in 2015. Currently he is a Lecturer in Informatics Engineering at the Sebelas Maret University Vocational School. His research interests include IoT. He can be contacted at email: nanang.my@staff.uns.ac.id.

Fendi Aji Purnomo master's degree from the Department of Electrical Engineering and Information Technology in 2016. Currently he is a Lecturer in Information Engineering at the Sebelas Maret University Vocational School. His research interests include AR/VR and IoT. He can be contacted at email: fendi_aji@staff.uns.ac.id.

Sahirul Alim Tri Bawono master's degree from the Department of Electrical Engineering and Information Technology in 2015. Currently he is a Lecturer in Information Engineering at the Sebelas Maret University Vocational School. His research interests include information systems. He can be contacted at email: sahirul@staff.uns.ac.id.

Agus Purnomo master's degree from the Department of Electrical Engineering and Information Technology in 2016. Currently he is a Lecturer in Information Engineering at the Universitas Islam Negeri (UIN) of Salatiga, Indonesia. His research interests include networking, information systems and optimization. He can be contacted at email: agus.purnomo@uinsalatiga.ac.id.