Journal
Vol. 8, No. 2 (2024) pp. 504-515
https://jurnal.politeknik-kebumen.ac.id/index.php/E-KOMTEK
p-ISSN: 2580-3719 e-ISSN : 2622-3066

Optimization of Convolutional Neural Network (CNN) Using Transfer Learning for Disease Identification in Rice Leaf Image

Abdul Azis[1,2], Abdul Fadlil[3], Tole Sutikno[4]

[1]Informatics Study Program, Universitas Ahmad Dahlan, Indonesia, 55191
[2]Information System Study Program, Universitas Amikom Purwokerto, Indonesia 53127
[3,4]Department of Electrical Engineering, Ahmad Dahlan University, Indonesia, 53127

abdazis9@amikompurwokerto.ac.id

https://doi.org/10.37339/e-komtek.v8i2.2132

Published by Politeknik Piksi Ganesha Indonesia

Article Info
Submitted:
24-11-2024
Revised:
26-12-2024
Accepted:
29-12-2024
Online first:
29-12-2024

Abstract

Rice productivity as one of the main commodities in Southeast Asia is often hampered by various plant diseases such as Rice Blast, Bacterial Leaf Blight, and Brown Spot, which can cause significant economic losses for farmers. This research aims to develop a deep learning-based rice leaf disease detection system using Convolutional Neural Networks (CNN) architecture technology with a transfer learning approach. The dataset captured by Kaggle is a dataset of 10,407 rice leaf images categorized into 10 classes, including various diseases and healthy leaves. The dataset was divided into three parts, 80% (8,323 images) for training, 15% (1,557 images) for validation, and 5% (527 images) for testing. The EfficientNetB0 pretrained model was used for feature extraction and classification. Data evaluation used accuracy matrix, precision matrix, recall matrix, and F1-score based on confusion matrix. The results showed that the model achieved global accuracy of 98.48%, micro precision of 100%, micro recall of 99.42%, and micro F1-score of 99.70%. These findings confirm the effectiveness of the proposed approach in automatically detecting rice leaf diseases, contributing significantly to technology-based agricultural solutions.

Keywords *Convolutional Neural Network, Transfer Learning, EfficientNet*, Disease Detection, Rice Leaf Image

Abstract

Rice productivity, as one of the key commodities in Southeast Asia, is often hindered by various plant diseases such as Rice Blast, Bacterial Leaf Blight, and Brown Spot, which can cause significant economic losses for farmers. This study aims to develop an automated rice leaf disease detection system using deep learning, specifically leveraging the Convolutional Neural Networks (CNN) architecture with a transfer learning approach. The dataset used comprises 10,407 images of rice leaves categorized into 10 classes, including various diseases and healthy leaves. The dataset is divided into three parts: 80% (8,323 images) for training, 15% (1,557 images) for validation, and 5% (527 images) for testing. The trained EfficientNetB0 model was utilized for feature extraction and classification. The evaluation used metrics such as accuracy, precision, recall, and F1-score based on a confusion matrix. The results revealed that the model achieved a global accuracy of 98.86%, a micro precision of 100%, a micro recall of 99.42%, and a micro F1-score of 99.70%. These findings underscore the effectiveness of the proposed approach in automating rice leaf disease detection, providing a significant contribution to technology-based agricultural solutions.

Keywords: *Convolutional Neural Network, Transfer Learning, EfficientNet, Disease Detection, Rice Leaf Images*

1. Introduction

Rice is one of the main commodities in many countries, especially in Southeast Asia, which economically and socially depend on optimal yields [1] . However, rice productivity is often threatened by various leaf diseases , such as Rice Blast, Bacterial Leaf Blight, and Brown Spot that cause significant economic losses to farmers [2] . Early identification of these diseases can help farmers take timely mitigation measures. However, traditional methods such as visual inspection are often inaccurate and require specialized skills [3] . Developing fast and accurate disease detection methods is essential to support sustainable agricultural practices. Artificial intelligence technologies, particularly deep learning have provided potential solutions in the automated detection of plant diseases [4].

In recent years, deep learning-based approaches, namely Convolutional Neural Networks (CNNs), have shown many promising results on plant disease identification in analyzing images [5]. Convolutional Neural Networks (CNNs) have become one of the dominant deep learning architectures in image recognition tasks due to their ability to extract features from visual data hierarchically [6]. However, training CNN models from scratch requires large datasets and high computational resources. Transfer Learning, which utilizes the weights of trained models that have been trained on large datasets such as ImageNet, is becoming a widely used approach to improve the efficiency and performance of CNN models on specific tasks [7] . In this research, the use of transfer learning to detect rice leaf diseases has been explored by utilizing EfficientNet trained models.

2. Method

The method used for the research can be seen in Figure 1 below. Figure 1 below is an overview of the stages of research carried out by researchers:

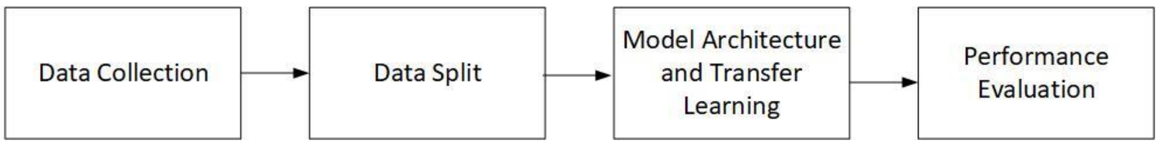


Figure 1. Stages of the research process

Figure 1 can be described regarding the explanation per stage. For the stages of the research process can be seen below:

2.1 Data Collection

The dataset in the study consists of images of rice leaves based on rice leaf diseases into 10 categories. The dataset was obtained from open sources and has been annotated by experts to ensure the accuracy of the disease classification. The data was collected from several open sources, including databases provided by research institutions and plant image datasets published online

2.2 Dataset Split

The research dataset is broken down into three main parts to ensure that the training and evaluation processes are carried out effectively. The first part is the training data, which covers 80% of the entire dataset. This data is used to train the model so that it can recognize patterns and features relevant to the classification task. Next, the validation data, which covers 15% of the dataset, evaluates the model's performance during the training process. Through the validation data, it can be identified whether the model is overfitting or underfitting, allowing for adjustments to the training parameters. Finally, the testing data, which covers 5% of the dataset, is used for the final evaluation. This data measures the model's generalization ability to new data that has not been seen during training.

The dataset is divided randomly but maintains a proportion of the data between classes. This aims to prevent class bias so that each category in the dataset has a balanced representation in all stages, namely training, validation, and testing. This approach ensures that the evaluation results are fair and reflect the model's overall performance.

2.3 Model Architecture and Transfer Learning

2.3.1 CNN Architecture Model

This research uses Google Collab as a platform for model development with TensorFlow and Keras libraries to build and train CNNs. CNN is a deep learning architecture built specifically for processing data in network networks, such as 2D images. CNN models excel in handling visual pattern recognition due to their ability to extract features hierarchically, from simple features (edges, corners) to complex features (objects)[6][8] . The architecture model in CNN involves the design of the network structure to capture data features efficiently. Here are the main components:

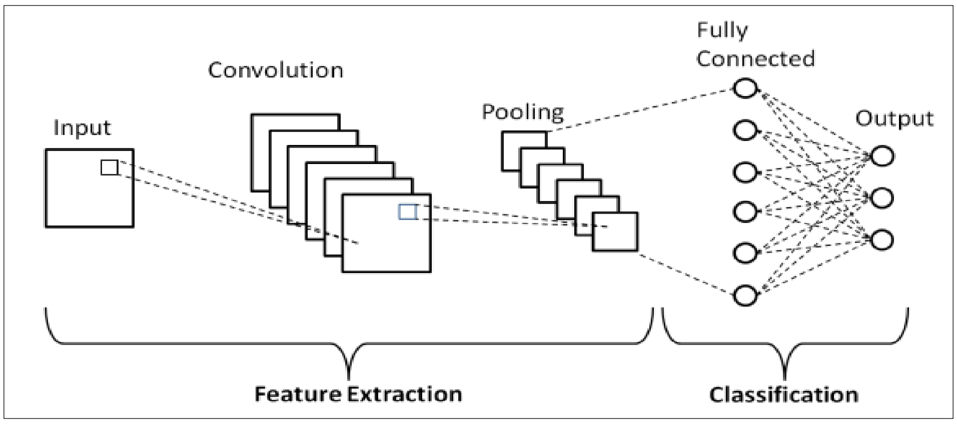
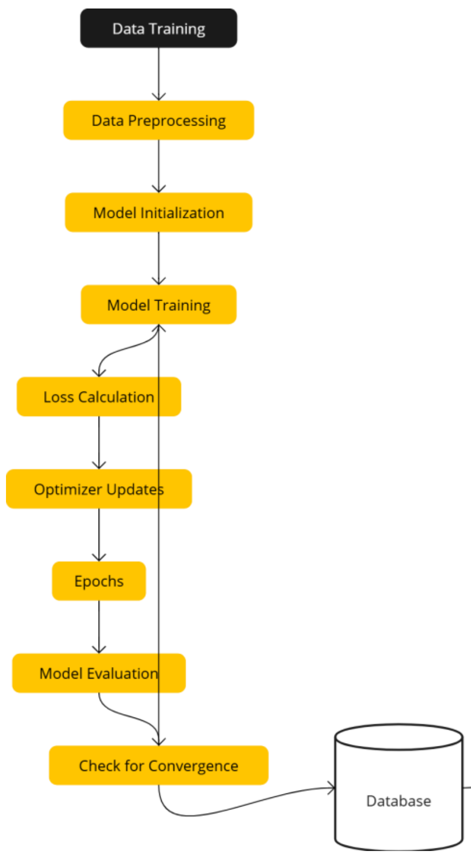


Figure 2. CNN Architecture Model

Figure 2 explains the process of the CNN architecture model. The first process is the Input Layer used to receive input data such as images from the original image dimensions with a size of 480x480 in (RGB) and training images with a size of 256x256 in (RGB). The second process is Convolutional Layer: This layer performs the convolution operation, where a filter (kernel) is used to extract important features from the input image. This filter moves across the image to generate a feature map. This operation enables detecting features such as edges, textures, or certain patterns.[9] . The Third Process is the Pooling Layer: A pooling function (usually max pooling) is used to reduce features' dimensionality while retaining important information. This reduces the number of parameters and the risk of overfitting. The Fourth Process is the Fully Connected Layer (Dense Layer): Once the feature extraction is complete, this layer connects all the neurons to process the feature map into a prediction output i.e. Activation Function: Activation functions such as ReLU (Rectified Linear Unit) are used to introduce non-linearity, thus allowing the network to learn complex patterns[10][11] , Batch Normalization: Used to normalize the data before it goes to the next layer, which speeds up training and improves stability, and Dropout: This technique is used to prevent overfitting by randomly disabling neurons during training .[12]

2.3.2. Transfer Learning

Transfer learning takes models pre-trained in large datasets (such as ImageNet) to solve new tasks with smaller datasets. This reduces training time and computational resources. The pretrained model used in this research is EfficientNet80, because of its efficiency in parameter usage and high performance[13] . The stages of applying transfer learning are as follows: first Output Layer Modification: The last fully connected layer is replaced with a Dense layer of size 10 (number of classes) with softmax activation function. Second Layer Freezing: The initial layer of the pretrained model is frozen to retain the basic features. Third Fine-tuning: The last few layers are unfrozen for adaptation to the new dataset.



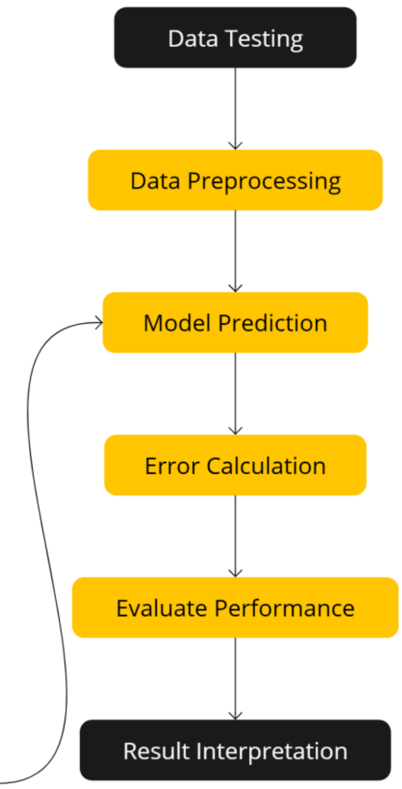


Figure 3. Training Flowchart **Figure 4.** Testing Flowchart

In Figure 3, the Training process can be explained from Data Collection: Collecting relevant data (images, text, etc.) for the task. Data Processing: Clean the data by handling missing values, normalizing it, and dividing it into training, validation, and testing sets. Model Initialization: Determine the model structure (e.g., number of layers, neurons, activation function). Model Training: The model is trained by inputting data, and the weights are adjusted at each iteration. Loss Calculation: Calculate how far the model's predictions are from the actual values using a loss function (e.g., mean square error). Optimizer Update: The optimizer (e.g., Adam) uses the calculated loss to adjust the model weights. Epochs: Training is repeated for several epochs (iterations) until the model is optimized. Model Evaluation: Evaluates model performance on training data, using metrics such as accuracy or loss. Check Convergence: The process stops when the model achieves optimal performance or the set number of epochs is reached

The testing process can be explained from Data Collection: Use a test dataset that the model has not seen during training. Data Processing: Apply the test data preprocessing steps as done during training. Model Prediction: Use the trained model to predict the outcome of the test data. Error Calculation: Calculate the error or loss for model prediction using appropriate evaluation metrics. Performance Evaluation: Analyze performance metrics e.g. accuracy matrix, precision matrix, recall matrix, etc. Interpretation of Results: Interpret the results to understand the model behavior and decide if the model is ready for use.

2.4 Performance Evaluation

Once the model is trained, its performance is evaluated using several metrics to assess the accuracy and generalization ability of the new data[14] . Since the dataset in this study consists of 10 disease classes, the formula used for each measurement parameter is as follows:

1. Global Accuracy: Predicts and Measures the correct percentage of total predictions, providing a generalized view of the model's performance. Accuracy is calculated using the global accuracy formula in equation (1).[15]

$$Accuracy = \frac{\sum TP}{\sum SAMPLE} \qquad (1)$$

In model evaluation using confusion matrix, the Total True Positive (TP) refers to the accumulation of all predictions for each class, which are the diagonal elements of the matrix. In other words, TP indicates the number of times the model accurately identifies the sample in the correct category. It directly measures the model's ability to correctly recognize the data corresponding to each class. Meanwhile, Total Sample refers to the overall amount of data used in the evaluation, calculated by summing up all the elements in the confusion matrix. Total Sample provides context for understanding the distribution of model predictions and calculating performance metrics such as accuracy, precision, and recall.

The function of calculating Total TP is to assess the extent to which the model can make correct predictions, which is an important component in calculating precision and recall metrics. Total Sample is the basis for calculating the global accuracy metric, which is the ratio of correct predictions to the overall data. Combining these two values helps provide an overall picture of the model's effectiveness in data classification.

2. Micro Precision: Calculates the proportion of correct positive predictions in each class. Precision is very important in detecting certain diseases to avoid false positive results, where healthy leaves are misclassified as diseases. Precision is calculated using the micro precision formula in equation (2).[15]

$$Precision_{micro} = \frac{\sum TP}{\sum(TP + FP)} \tag{2}$$

3. Recall: Calculates the proportion of actual cases from each class that are correctly detected. Recall is useful to ensure that the model identifies all disease cases. Recall is calculated using the micro recall formula in equation (3).[15]

$$Recall_{micro} = \frac{\sum TP}{\sum(TP + FN)} \tag{3}$$

4. F1-score: F1-score combines harmonized precision and recall to provide an overall picture of the model's performance in each class, especially when there is an imbalance in the number of samples. F1-score is calculated using the micro F1-score formula in equation (4).[15]

$$F1_{micro} = \frac{2 \times Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}} \tag{4}$$

To ensure that the model evaluation results are not biased towards a particular subset of the dataset, we also applied k-fold cross-validation with 5-fold, where the dataset is divided into five subsets and the model is trained five times, with each subset alternating as validation data. This technique allows for more reliable estimation of model performance.

3. Results and Discussion

This section explains and presents each processed data result from each stage described in the previous section.

3.1 Data Collection

The dataset used in the research includes 10,407 rice leaf images categorized into 10 classes, including various diseases and healthy leaves. Each class represents a specific condition of rice leaf images, namely rice leaf diseases (Normal, Bacterial Leaf Streak, Dead Hearth, Hispa Bacterial Leaf Blight, Blast, Brown Spot, Downy Mildew, Bacterial Panicle Blight, and Tungro). The example image used in this research is as shown in the picture below:

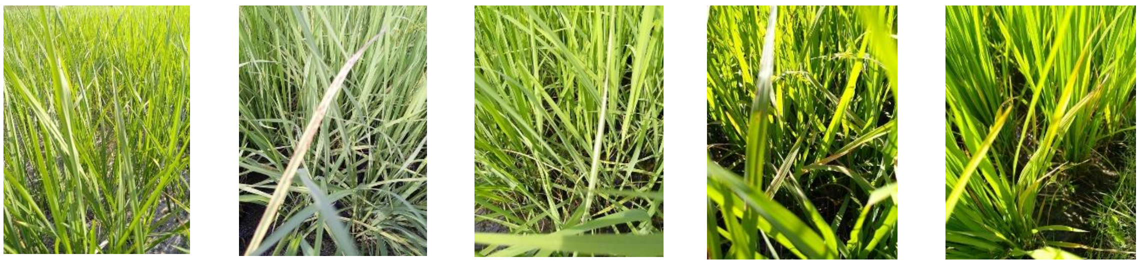


Image a) Normal | Image b) Brown Spot | Image c) Bacterial leaf blight | Image d) Blast | Image e) Tungro

Figure 5. Example of Rice Leaf Disease Image

Each class is provided with representative images to train and evaluate the performance of the disease detection model. For example, the Normal class includes images of healthy rice leaves with no signs of disease, while the Bacterial Leaf Blight class contains images of leaves with typical brown spot symptoms. The images in the dataset show variations in pattern, color, and intensity of symptoms, ensuring that the dataset covers the diversity of leaf conditions found in the field.

The following visual illustration of some of the images used for each class. These images provide a real picture of the difference in characteristics between the disease and normal classes, which is the basis for the model to learn to recognize relevant patterns. Table 1 shows example images for each of the 10 labels.

Table 1. Dataset from Kaggle

No.	Name of disease	Number of Original Datasets	Number of Testing Datasets	Number of Training Datasets	Number of Validation Datasets
1	Normal	1764	89	1411	264
2	Bacterial leaf blight	479	25	383	71
3	Bacterial leaf streak	380	19	304	57
4	Bacterial panicle blight	337	18	269	50
5	Blast	1738	88	1390	260
6	Brown spot	965	49	772	144
7	Dead hearth	1442	73	1153	216
8	Downy mildew	620	31	496	93
9	Hispa	1594	80	1275	239
10	Tungro	1088	55	870	163
Total	Number of Diseases = 10	Original Dataset = 10.407	Dataset Testing = 527	Traning Dataset = 8.323	Dataset Validation = 15.557

3.2 Dataset Split

Furthermore, the dataset is split into three main datasets: TrainingValidation, and Testing. The training data covers 80% of the total 10,407 dataset, is 8,323 images used to train the model. Meanwhile, the validation data, which amounted to 1,557 images constituting 15% of the total dataset, was used to monitor the model's performance during training. The testing data consisted of 527 images which constituted 5% of the dataset and was used to measure the model's accuracy after the training was completed. This division aims to ensure that the model can learn well without overfitting.

3.3 Model Architecture and Transfer Learning

The CNN model is applied using a customized EfficientNet architecture to classify rice leaf diseases. The training process is performed by setting the initial learning rate at 0.1, with learning rate adjustments every epoch to improve training accuracy and stability. The optimizer uses the Stochastic Gradient Descent (SGD) method with a momentum of 0.9, which helps accelerate convergence and reduce fluctuations during the training process. The number of iterations is 100 epochs to provide an opportunity for the model to update parameters (weights and biases) and improve accuracy and reduce errors. The experimental results show that applying an appropriate learning rate can significantly improve the performance of the model.

At this stage the layers in the Convolutional Neural Network (CNN) are integrated with EfficientNet which consists of:

1. Input Layer: The image is input into the network.
2. EfficientNet Block: This section applies compound scaling settings (depth, width, and resolution adjustments).
3. Convolutional Layers: Feature extraction using filters.
4. Pooling Layers: Lower the resolution of features to reduce dimensions.
5. Fully Connected Layer: Combines features to produce the final output.
6. Output Layer: The result of the network's classification or prediction.

3.4 Performance Evaluation

In the evaluation stage, model performance is measured using the model confusion matrix (CM) which is divided into four main parts: CM True Positive (TP), CM True Negative (TN), CM False Positive (FP), and CM False Negative (FN). The explanations of TP, TN, FP and FN in this study are as follows:

1. True Positive (TP): The number of images that are actually detected as a disease corresponding to the correct class. For example, an image of a rice leaf that is indeed infected with a disease and the model also identifies it as that disease.
2. True Negative (TN): The number of images that are truly not infected with the disease, and the model also classifies them as healthy.

3. False Positive (FP): The number of images that are actually healthy, but the model classifies them as disease-infected (type I error).
4. False Negative (FN): The number of images that are actually infected with the disease, but the model misclassifies them as healthy (type II error).

The following is the confusion matrix generated by the model which in this study consists of 10 classes:

Table 2. Testing Dataset

Class Name	bacterial leaf blight	Bacterial leaf streak	Bacterial panicle blight	blast	Brown spot	Dead heart	Downy mildew	hispa	normal	tungro	TRUE	FALSE
bacterial leaf blight	25	0	0	0	0	0	0	0	0	0	25	0
Bacterial leaf streak	0	18	0	1	0	0	0	0	0	0	18	1
Bacterial panicle blight	0	0	18	0	0	0	0	0	0	0	18	0
blast	0	0	0	88	0	0	0	0	0	0	88	0
Brown spot	0	0	0	0	49	0	0	0	0	0	49	0
Dead heart	0	0	0	0	0	72	1	0	0	0	72	1
Downy mildew	0	0	0	3	0	0	28	0	0	0	28	3
hispa	0	0	0	1	0	0	0	78	1	0	78	2
normal	0	0	0	0	0	0	0	0	89	0	89	0
tungro	0	0	0	0	0	0	1	0	0	54	54	1
										SUBTOTAL	519	8
										TOTAL	527	

Based on the confusion matrix, it can be seen that the TP value for each class is bacterial leaf blight 25, bacterial leaf streak 18, bacterial panicle blight 18, blast 88, brown spot 49, dead hearth 72, downy mildew 28, hispa 78, normal 89 and tungro 54 as shown in the diagonal confusion matrix above. This shows that from all the testing data used to test the model, most of the data is predicted correctly, resulting in a high TP value from the total testing data. By using the values generated in the confusion matrix, the percentage level for each measurement parameter is then obtained as follows.

$$Akurasi = \frac{521}{527} = 0{,}9886\ atau\ 98{,}86\%$$

$$Presisi = \frac{521}{521 + 0} = 1\ atau\ 100\%$$

$$Recall = \frac{521}{521 + 3} = 0{,}9942\ atau\ 99{,}42\%$$

$$F1 - Score = \frac{2 \times 1 \times 0{,}9942}{1 + 0{,}9942} = 0{,}9970\ atau\ 99{,}70\%$$

Based on the calculation for each measurement parameter used, it can be seen that the accuracy rate obtained is 98.86%, precision is 100%, recall is 99.42% and F1-Score is 99.70%. This shows that the performance of the CNN algorithm using Transfer Learning

EfficientNet is able to provide excellent performance on disease identification in rice plant images.

4. Conclusion

In this research, the Convolutional Neural Network (CNN) method using Transfer Learning is applied in identifying various types of rice leaf diseases using a dataset of rice leaf images. The CNN model implemented, after going through an optimization process with EfficientNet, showed excellent performance results in processing and classifying images of rice leaf diseases.

The model evaluation results show a global accuracy of 98.86%, with Micro Precision, Micro Recall, and Micro F1-Score results of rice leaf diseases reaching 100%, 99.42%, and 99.70%. This shows that the CNN model applied with augmentation and transfer learning techniques is able to recognize diseases in rice leaves with a very low error rate, even on varied datasets.

Overall, this research shows that the use of transfer learning on CNN architecture, with effective data augmentation, can produce highly accurate models for plant disease image classification, which can be applied in an automated rice disease detection system.

References

[1] IRRI, "Enriching Rice-Based Economies: SOUTHEAST ASIA, SOUTH ASIA, & AFRICA." 2018.

[2] S. Shekhar, D. Sinha, and A. Kumari, "An Overview of Bacterial Leaf Blight Disease of Rice and Different Strategies for its Management," vol. 9, no. 4, pp. 2250-2265, 2020.

[3] J. Li *et al.*, "An Interpretable High-Accuracy Method for Rice Disease Based on Multisource Data and Transfer Learning," *Plants*, vol. 12, no. 18, pp. 1-22, 2023.

[4] A. Jafar, N. Bibi, and R. A. Naqvi, "Revolutionizing agriculture with artificial intelligence: plant disease detection methods, applications, and their limitations," *Front. Plant Sci.*, vol. 15, no. March, pp. 1-20, 2024, doi: 10.3389/fpls.2024.1356260.

[5] B. Tugrul, E. Elfatimi, and R. Eryigit, "Convolutional Neural Networks in Detection of Plant Leaf Diseases: A Review," *agriculture*, vol. 12, no. 8, 2022.

[6] M. M. Taye, "Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions," *Computation*, vol. 11, no. 3, pp. 1-23, 2023.

[7] A. W. Salehi, S. Khan, G. Gupta, B. I. Alabduallah, and A. Almjally, "A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope," *sustainability*, vol. 15, no. 7, pp. 1-28, 2023.

[8] Z. Kelta, "An Introduction to Convolutional Neural Networks (CNNs)," *datacamp*, 2023. https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns.

[9] T. Wiatowski and H. Bolcskei, "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, 2018.

[10] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, and M. A. Ayidzoe, "RMAF: ReLU-Memristor-like Activation Function for Deep Learning," *IEEE Access*, vol. 1, no. 1, 2020,

doi: 10.1109/ACCESS.2020.2987829.

[11] D. Hartmann, D. Franzen, and S. Brodehl, "Studying the Evolution of Neural Activation Patterns During Training of Feed-Forward ReLU Networks," *Front. Artif. Intell.*, vol. 4, no. 1, pp. 1-13, 2021, doi: 10.3389/frai.2021.642374.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929-1958, 2014.

[13] Y. Fu, "Image classification via fine-tuning with EfficientNet," *Keras*, 2023. https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/#image-classification-via-finetuning-with-efficientnet.

[14] S. K. Agrawal, "Metrics to Evaluate your Classification Model to take the right decisions," *Analytics Vidhya*, 2024. https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/.

[15] V. V. Kumar, "Evaluating machine learning models-metrics and techniques," *AI Accelerator Institute*, 2024. https://translate.google.com/?sl=id&tl=en&text=evaluasi performance&op=translate.