

Mobile Application for Plant Leaf Disease Classification Using Convolutional Neural Networks

Luqman Nur Fajar¹, Witta Listiya Ningrum^{1*}, Ana Kurniawati¹

¹Information Systems, Universitas Gunadarma, Indonesia.

Article Info

Keywords:

Android;
Convolutional Neural Network;
Deep Learning;
Leaf Diseases Classification.

Article History:

Submitted: April 20, 2026
Accepted: October 15, 2026
Published: October 15, 2026

Abstract: Indonesia, as an agrarian country, faces serious challenges in agriculture, particularly plant leaf diseases that may lead to crop failure. Early detection is crucial to prevent their spread. This study aims to develop a leaf disease classification model using a Convolutional Neural Network (CNN) and implement it in an Android application called Leaf Lives. The model was developed using the EfficientNetB0 architecture and optimized with Stochastic Gradient Descent (SGD), trained for 30 epochs and fine-tuned for an additional 10 epochs. The dataset consisted of 44,030 leaf images covering 11 plant species and 60 disease or leaf-condition classes, obtained from public datasets available through platforms such as Kaggle and Mendeley Data. Four data split scenarios were evaluated (75:15:10, 70:20:10, 65:25:10, and 60:30:10). The best performance was achieved with the 70:20:10 split, yielding 92.03% accuracy, 91.12% precision, 89.89% recall, and 89.96% F1-score. The resulting model was then integrated into an Android application that classifies leaf images captured using the camera or selected from the gallery and stores classification history offline. The developed application is expected to assist farmers in identifying potential leaf diseases quickly and independently; however, further validation using real-field images and user-based evaluation is required to assess its practical effectiveness.

Corresponding Author:

Witta Listiya Ningrum
Email: wita_listiya@staff.gunadarma.ac.id

INTRODUCTION

Indonesia is an agrarian country that relies heavily on the agricultural sector as a major pillar of its economy. However, agricultural productivity is often constrained by plant diseases, most of which initially manifest through leaf symptoms such as changes in colour, shape, or texture (Rokhmana & Setiawan, 2022; Felix et al., 2020). Early detection of leaf diseases is essential to prevent disease spread and reduce the risk of crop failure. Currently, disease identification is still largely performed manually by farmers or agricultural experts, a process that requires specific expertise and may be affected by subjective judgement, potentially resulting in inaccurate identification. Therefore, an efficient, accurate, and accessible approach is needed to support plant leaf disease and condition classification.

The rapid development of Artificial Intelligence (AI), particularly deep learning, has created significant opportunities in the agricultural domain. Convolutional Neural Networks (CNNs) have been widely applied to plant disease detection because of their ability to automatically learn discriminative visual features from leaf images (Tugrul et al., 2022). Falaschetti et al. (2022), for example, developed a CNN-based image detector for resource-constrained hardware and achieved accuracies of 98.10% on the ESCA dataset and 95.24% on the augmented PlantVillage dataset. Their model was deployed on an OpenMV Cam H7 Plus for real-time classification, demonstrating the feasibility of deploying CNN-based plant disease detection on low-power devices. In the mobile application domain,

Rimon et al. (2022) developed PlantBuddy, an Android-based application using deep CNN models to classify plant diseases from leaf images. Their study used 54,305 images covering 14 plant species and 26 disease classes and evaluated MobileNetV2 and InceptionV3, with MobileNetV2 selected for the final application because of its accuracy and smaller model size.

Several studies have also investigated CNN-based plant disease detection for specific crops. Rachman et al. (2024) developed a CNN-based information system for rice disease detection, covering Brown Spot, Hispa, and Leaf Blast, using 3,355 images and reporting an accuracy of 85%. Similarly, Saputra et al. (2024) developed an Android-based application for Arabica coffee disease detection using 2,829 leaf images from three classes and reported a testing accuracy of 94.33% after hyperparameter optimization. These studies demonstrate the potential of CNN-based approaches and mobile applications for plant disease recognition. However, the reviewed studies differ in their classification scope, model architecture, evaluation strategy, and deployment platform. Rachman et al. (2024) and Saputra et al. (2024) focus on a single crop with a limited number of classes, while Rimon et al. (2022) addresses multiple plant species but uses MobileNetV2 and InceptionV3. Falaschetti et al. (2022), although demonstrating real-time deployment on constrained hardware, uses a dedicated embedded vision platform rather than an Android mobile application. These differences highlight an opportunity to develop a broader multi-plant and multi-class classification system that combines EfficientNetB0-based transfer learning and fine-tuning, detailed class-level performance analysis, and offline Android deployment within a single application.

In this study, the classification task covers **60 disease or leaf-condition classes across 11 plant species**, including disease classes as well as healthy leaves, nutritional deficiency, and pest-related conditions. The dataset comprises 44,030 leaf images collected from publicly available datasets on platforms such as Kaggle and Mendeley Data. This broader classification scope enables the model to distinguish not only among different plant diseases but also between diseased and non-diseased leaf conditions. The proposed approach uses the EfficientNetB0 architecture with ImageNet-pretrained weights and a fine-tuning process to adapt the learned visual representations to the plant leaf dataset. Model performance is evaluated using overall classification metrics as well as class-level precision, recall, F1-score, and confusion matrix analysis. The selected model is subsequently converted to TensorFlow Lite and integrated into an Android application named *Leaf Lives* to support offline image classification.

Based on this research gap, this study aims to develop a CNN-based plant leaf disease and condition classification model using EfficientNetB0 and implement the selected model in the *Leaf Lives* Android application. Four data split scenarios are evaluated to examine the effect of different training, validation, and testing proportions on model performance. The data split comparison is treated as part of the experimental evaluation rather than as the primary novelty of the study. The main contribution of this research lies in the integration of a broad multi-plant, multi-class dataset, EfficientNetB0 transfer learning and fine-tuning, detailed class-level evaluation, and offline mobile deployment within a single prototype system. The application is intended as a prototype for image-based leaf disease and condition classification, while its effectiveness in real agricultural environments and among farmers requires further validation through field testing, expert assessment, and user evaluation.

METHOD

This research employed the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, which consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment. The methodology was adopted to provide a systematic workflow for dataset preparation, model development, performance evaluation, and implementation of the plant leaf disease classification model in an Android application.

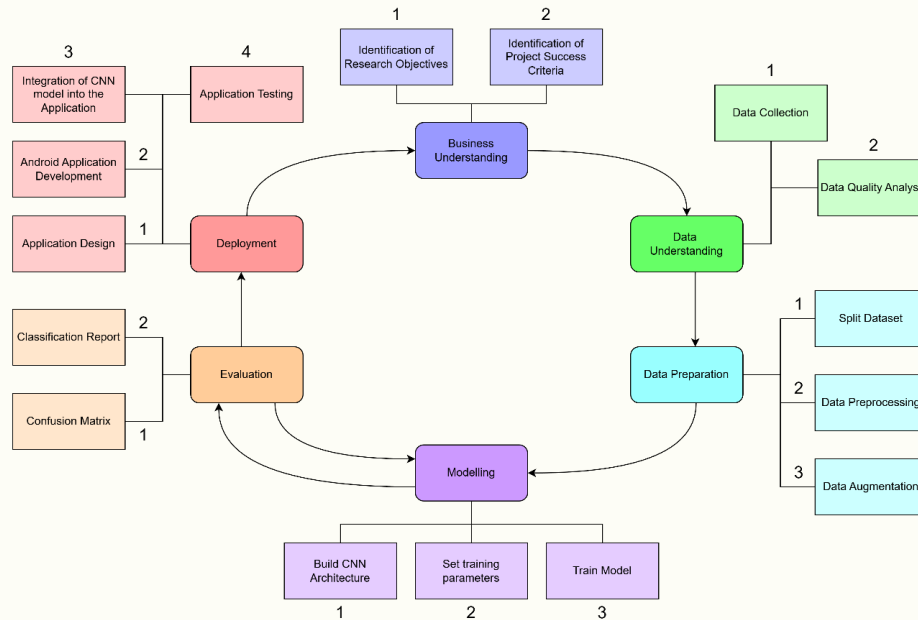


Figure 1. CRISP-DM Research Framework

Business Understanding

The Business Understanding phase focused on developing a plant leaf disease classification model based on a Convolutional Neural Network (CNN) and implementing the selected model in an Android mobile application named *Leaf Lives*. Four data split scenarios were investigated to examine the effect of different training, validation, and testing proportions on model performance. The success criteria were defined as achieving satisfactory classification performance across the evaluated data split scenarios and successfully integrating the selected model into an Android application capable of performing offline image classification.

Data Understanding

The dataset consisted of 44,030 colour leaf images representing 11 plant species and 60 disease or leaf-condition classes. The plant species included grape, chili, corn, soybean, potato, pumpkin, rice, banana, eggplant, cucumber, and tomato. The images were obtained from publicly available datasets hosted on Mendeley Data and Kaggle. The detailed distribution of images across plant species and disease or leaf-condition classes, together with the corresponding dataset sources, is presented in Tables 1, 2, and 3. Images were included when the leaf features were visually identifiable and accompanied by a valid class label corresponding to one of the predefined disease or leaf-condition classes. Images that were blurred, irrelevant to the target classes, or did not meet the predefined data criteria were excluded.

Table 1. Dataset Distribution by Plant Species

Plant Species	Number of Classes	Number of Images
Banana	4	1,669
Chili	4	1,488
Corn	6	5,436
Cucumber	4	4,387
Eggplant	7	2,211
Grape	5	2,824
Potato	4	2,751
Pumpkin	5	2,284
Rice	9	16,857
Soybean	7	1,784
Tomato	5	2,339
Total	60	44.030

Table 2. Dataset Distribution by Plant Species

Plant Species	Disease/Condition Class	Number of Images	Source
Banana	Cordana	324	D3
Banana	Healthy	433	D3
Banana	Pestalotiopsis	259	D3
Banana	Sigatoka	653	D3
Chili	Bacterial Spot	154	D5
Chili	Curl Virus	436	D5
Chili	Healthy	454	D5
Chili	Nutrition Deficiency	444	D5
Corn	Bacterial Leaf Streak	383	D2
Corn	Cercospora Leaf Spot	615	D1
Corn	Common Rust	374	D1, D2
Corn	Gray Leaf Spot	1484	D2
Corn	Healthy	1542	D1, D2
Corn	Northern Leaf Blight	1038	D1
Cucumber	Anthracnose	800	D10
Cucumber	Downy Mildew	1311	D10
Cucumber	Healthy	1084	D10, D12
Cucumber	Powdery Mildew	1192	D10, D12
Eggplant	Healthy	374	D7
Eggplant	Insect Pest	454	D7
Eggplant	Leaf Spot	359	D7
Eggplant	Mosaic Virus	253	D7
Eggplant	Small Leaf	242	D7
Eggplant	White Mold	204	D7
Eggplant	Wilt	325	D7
Grape	Black Rot	1088	D1, D9
Grape	Esca Black Measles	707	D1, D9
Grape	Fungal Diseases	113	D9
Grape	Healthy	443	D1, D9
Grape	Isariopsis Leaf Spot	473	D1
Potato	Early Blight	1152	D1
Potato	Healthy	362	D1
Potato	Late Blight	624	D1
Potato	Pest	613	D1
Pumpkin	Bacterial Leaf Spot	412	D1, D6
Pumpkin	Downy Mildew	505	D1, D6
Pumpkin	Healthy	554	D1, D6
Pumpkin	Mosaic Disease	413	D1, D6
Pumpkin	Powdery Mildew	400	D1, D6
Rice	Bacterial Leaf Blight	1451	D11
Rice	Bacterial Leaf Streak	684	D11
Rice	Brown Spot	2003	D11
Rice	Downy Mildew	621	D11
Rice	Healthy	2805	D11
Rice	Leaf Blast	3023	D11
Rice	Rice Hispa	2613	D11
Rice	Sheath Blight	1354	D11
Rice	Tungro	2303	D11
Soybean	Bacterial Pustule	184	D8

Soybean	Frogeye Leaf Spot	145	D8
Soybean	Healthy	455	D8
Soybean	Rust Disease	144	D8
Soybean	Sudden Death Syndrome	138	D8
Soybean	Target Leaf Spot	254	D8
Soybean	Yellow Mosaic	464	D8
Tomato	Bacterial Spot	346	D1, D4
Tomato	Early Blight	562	D1, D4
Tomato	Healthy	453	D1, D4
Tomato	Late Blight	324	D1, D4
Tomato	Leaf Mold	654	D1, D4

Table 3. Dataset Source

Code	Source
D1	Ali, <i>PlantVillage Dataset</i> , Kaggle
D2	Ahmad (2025), <i>Seasonal Corn Leaf Disease Dataset</i> , Mendeley Data
D3	Das et al. (2024), <i>Banana and Banana Leaf Dataset</i> , Mendeley Data
D4	Imtiaz et al. (2025), <i>Tomato Leaf Dataset</i> , Mendeley Data
D5	Nirob et al. (2025), <i>Chili Plant Leaf Disease and Growth Stage Dataset From Bangladesh</i> , Mendeley Data
D6	Rashid et al. (2024), <i>Pumpkin Leaf Diseases Dataset From Bangladesh</i> , Mendeley Data
D7	Rafe et al. (2025), <i>Eggplant_Leaf_Disease_Dataset</i> , Mendeley Data
D8	Rajput et al. (2023), <i>SoyNet</i> , Mendeley Data
D9	Shikalgar et al. (2024), <i>Grape Disease</i> , Mendeley Data
D10	Sultana et al. (2022), <i>Cucumber Disease Recognition Dataset</i> , Mendeley Data
D11	Trumanrase, <i>Rice Leaf Diseases</i> , Kaggle
D12	Zitu et al. (2024), <i>A Benchmark Dataset for Detecting Disease in Plant Leaves</i> , Mendeley Data

Data Preparation

Before data partitioning, images obtained from the different public repositories were combined into a single dataset and inspected for duplicate and low-quality images. Duplicate images were removed to reduce the possibility of identical images being included across the training, validation, and testing subsets, while images with poor visual quality or unclear leaf characteristics were excluded. Near-duplicate images were not systematically assessed using image-similarity or perceptual-hashing methods.

The cleaned dataset was manually partitioned into training, validation, and testing subsets according to four predefined proportions, namely 75:15:10, 70:20:10, 65:25:10, and 60:30:10, respectively. Stratified sampling was not applied during the partitioning process; therefore, the class proportions were not explicitly preserved across the three subsets. The selection of multiple data split scenarios was motivated by previous discussions regarding dataset partitioning in machine learning. Muraina (2022) highlighted that the suitability of a dataset splitting ratio depends on the characteristics of the dataset and modelling task. Therefore, multiple split configurations were evaluated in this study rather than assuming a single universally optimal ratio.

All images were resized to 224×224 pixels to match the EfficientNetB0 input dimensions. During data loading, pixel values were rescaled by $1/255$ using the ImageDataGenerator, resulting in input values in the $[0,1]$ range before being passed to the EfficientNetB0 model. Data augmentation was applied only to the training subset using rotation with a maximum range of 10 degrees, width and height shifting of up to 10%, shearing of up to 10%, zooming of up to 10%, and horizontal flipping. The nearest filling mode was used for newly generated pixels. The validation and testing subsets were not augmented and were only rescaled. This separation was applied to prevent augmentation leakage and to ensure that model performance was evaluated using non-augmented data.

Modelling

The classification model was developed using the EfficientNetB0 architecture with ImageNet-pretrained weights and implemented using TensorFlow 2.19.0. The input size was set to $224 \times 224 \times 3$. The model received the externally rescaled inputs generated by the ImageDataGenerator during the training, validation, and testing procedures. During the initial training stage, the EfficientNetB0 backbone was frozen, and a Global Average Pooling layer followed by a fully connected Softmax output layer was added for classification across 60 disease or leaf-condition classes. The model was compiled using the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001 and momentum of 0.9. Categorical cross-entropy was used as the loss function, while accuracy was used as the training metric. The model was initially trained for 30 epochs using a batch size of 32. The checkpoint with the highest validation accuracy was retained for subsequent fine-tuning. During fine-tuning, the previously trained model was loaded and all model layers were unfrozen by setting `model.trainable = True`. The model was then recompiled using the same SGD configuration and training was continued from epoch 30 to epoch 40, resulting in 10 additional fine-tuning epochs. No additional dropout layer was applied to the classification head. No explicit random seed was specified in the retained model-training implementation; therefore, NumPy and TensorFlow random seeds were not explicitly fixed.

Table 4. Model Training Configuration

Parameter	Configuration
Architecture	EfficientNetB0
Pretrained Weights	ImageNet
Input Size	$224 \times 224 \times 3$
Pooling	Global Average Pooling
Output Activation	Softmax
Number of Classes	60
Optimizer	SGD
Learning Rate	0.001
Momentum	0.9
Loss Function	Categorical cross-entropy
Batch Size	32
Initial Training	30 epochs
Fine Tuning	10 additional epochs
Initial Backbone State	Frozen
Fine Tuning Backbone State	Entire backbone unfrozen
Dropout	Not used
Framework	TensorFlow 2.19.0

Table 5. Computational Environment

Parameter	Configuration
Processor	AMD Ryzen 5 3600
GPU	NVIDIA RTX 2070 Super 8 GB
RAM	32 GB DDR4 3200 MHz
Storage	512 GB SSD
Operating System	Windows 11 Pro
Deep Learning Framework	TensorFlow 2.19.0

Evaluation

Model performance was evaluated using accuracy, macro-averaged precision, macro-averaged recall, macro-averaged F1-score, and confusion matrix analysis. Accuracy was calculated based on the proportion of correctly classified samples, while precision, recall, and F1-score were calculated using the macro-averaging strategy across the 60 disease or leaf-condition classes. The macro-average was

selected to give equal importance to each class despite differences in class size. The `zero_division=0` setting was applied when calculating the classification metrics to handle cases in which a metric could not be defined. The results obtained from the four data split scenarios were compared to determine which configuration produced the highest classification performance within the evaluated scenarios. The model obtained from the selected split was subsequently used for mobile application deployment.

Deployment

The selected model was converted into TensorFlow Lite format and integrated into an Android mobile application named *Leaf Lives* using the Flutter framework. The application allows users to select leaf images from the device gallery or capture images using the smartphone camera. The classification result is displayed within the application, and previous classification results can be stored as local history. The final TensorFlow Lite model had a file size of 15,935 KB (approximately 15.56 MiB). The application was tested on a POCO X6 Pro running Android 14, equipped with a MediaTek Dimensity 8300-Ultra processor, Mali-G615 MC6 GPU, 12 GB RAM, and 512 GB internal storage. Functional testing was performed to verify the integration of the TensorFlow Lite model, image acquisition from the camera and gallery, offline classification, and local storage of classification history. Quantitative measurements of inference time and memory consumption were not recorded during the original implementation. Therefore, numerical device-level performance benchmarks are not reported in this study. In addition, formal usability evaluation involving farmers and field validation using real agricultural images were outside the scope of the study.

RESULTS AND DISCUSSIONS

This section presents and discusses the results obtained in this study, which are divided into two main parts. The first part focuses on the evaluation of the proposed CNN model, including its classification performance, comparison across different data split scenarios, and analysis of model behaviour after the fine-tuning process. The second part discusses the results of the mobile application development, highlighting the integration of the trained model into an Android-based application and its practical functionality for offline plant leaf disease classification.

CNN Model Results

The performance of the proposed CNN model was evaluated using standard classification metrics, including accuracy, precision, recall, and F1-score. The model was developed using the EfficientNetB0 architecture and trained for 30 epochs, followed by a fine-tuning process of an additional 10 epochs to improve feature representation and classification stability. Model evaluation was conducted using the corresponding test subset for each training configuration. The same test subset was maintained across all four data split scenarios, enabling a direct comparison of model performance under different training and validation proportions.

Table 6. Model Performance Results Before Fine Tuning

	75:15:10	70:20:10	65:25:10	60:30:10
Accuracy %	75.08%	75.10%	74.81%	74.15%
Precision %	76.71%	76.82%	76.14%	75.83%
Recall %	74.26%	74.32%	73.76%	73.02%
F1-Score %	74.69%	74.74%	74.14%	73.50%

Table 7. Model Performance Results After Fine Tuning

	75:15:10	70:20:10	65:25:10	60:30:10
Accuracy %	91.78%	92.03%	91.21%	90.55%
Precision %	90.56%	91.12%	89.78%	89.40%
Recall %	89.69%	89.89%	89.33%	88.84%
F1-Score %	89.54%	89.96%	89.07%	88.57%

The performance comparison of the four data split scenarios before and after fine-tuning is presented in Tables 6 and 7, respectively. Fine-tuning substantially improved the classification performance across all evaluated scenarios. The average accuracy increased from 74.79% before fine-tuning to 91.39% after fine-tuning, representing an improvement of 16.61 percentage points. These results indicate that fine-tuning substantially improved the model's classification performance on the evaluated test datasets. Among the evaluated configurations, the 70:20:10 split achieved the highest performance after fine-tuning, with an accuracy of 92.03%, precision of 91.12%, recall of 89.89%, and F1-score of 89.96%. Compared with the other evaluated split configurations, this scenario produced the highest values for all four reported metrics. Each split scenario was trained only once; therefore, the observed differences should be interpreted as the results of the individual experimental runs rather than as evidence of statistically significant or consistently superior performance.

The four data split scenarios used the same test subset, while the remaining data were allocated differently between the training and validation subsets according to each predefined proportion. Therefore, the comparison of model performance was conducted on the same test samples across all scenarios, reducing the effect of differences in test-set composition. Nevertheless, because each scenario was trained only once, variation associated with model optimization, data augmentation, and data shuffling may still have contributed to the observed differences in performance. Thus, the 70:20:10 configuration is considered the best-performing split among the four evaluated configurations in this experiment, rather than a universally optimal data proportion.

Table 8. Per-Class Classification Performance of the Selected Model

No.	Plant Species	Disease	Test Samples	Precision (%)	Recall (%)	F1-score (%)
1	Banana	Cordana	32	97	100	98
2	Banana	Healthy	43	98	100	99
3	Banana	Pestalotiopsis	25	100	88	94
4	Banana	Sigatoka	65	98	97	98
5	Chili	Bacterial Spot	15	100	93	97
6	Chili	Curl Virus	44	98	100	99
7	Chili	Healthy	45	98	100	99
8	Chili	Nutrition Deficiency	44	100	100	100
9	Corn	Bacterial Leaf Streak	38	97	100	98
10	Corn	Cercospora Leaf Spot	61	81	92	86
11	Corn	Common Rust	37	94	89	92
12	Corn	Gray Leaf Spot	148	99	96	98
13	Corn	Healthy	154	98	99	98
14	Corn	Northern Leaf Blight	104	92	89	91
15	Cucumber	Anthracnose	80	96	100	98
16	Cucumber	Downy Mildew	131	100	96	98
17	Cucumber	Healthy	108	96	99	98
18	Cucumber	Powdery Mildew	119	99	98	99
19	Eggplant	Healthy	37	84	97	90
20	Eggplant	Insect Pest	45	94	98	96
21	Eggplant	Leaf Spot	36	91	86	89
22	Eggplant	Mosaic Virus	25	87	84	86
23	Eggplant	Small Leaf	24	92	92	92
24	Eggplant	White Mold	20	94	80	86
25	Eggplant	Wilt	32	100	97	98
26	Grape	Black Rot	109	97	99	98
27	Grape	Esca Black Measles	71	99	97	98
28	Grape	Fungal Diseases	11	92	100	96

29	Grape	Healthy	44	100	100	100
30	Grape	Isariopsis Leaf Spot	47	100	96	98
31	Potato	Early Blight	115	87	88	87
32	Potato	Healthy	36	82	92	87
33	Potato	Late Blight	62	84	95	89
34	Potato	Pest	61	92	61	73
35	Pumpkin	Bacterial Leaf Spot	41	71	54	61
36	Pumpkin	Downy Mildew	50	80	100	89
37	Pumpkin	Healthy	55	74	100	89
38	Pumpkin	Mosaic Disease	41	93	100	93
39	Pumpkin	Powdery Mildew	40	100	57	73
40	Rice	Bacterial Leaf Blight	145	96	91	93
41	Rice	Bacterial Leaf Streak	68	87	78	82
42	Rice	Brown Spot	200	89	94	91
43	Rice	Downy Mildew	62	77	82	80
44	Rice	Healthy	280	97	91	94
45	Rice	Leaf Blast	302	90	93	92
46	Rice	Rice Hispa	261	93	93	93
47	Rice	Sheath Blight	135	91	98	94
48	Rice	Tungro	230	93	93	93
49	Soybean	Bacterial Pustule	18	73	61	67
50	Soybean	Frogeye Leaf Spot	14	73	79	76
51	Soybean	Healthy	45	91	96	93
52	Soybean	Rust Disease	30	83	63	72
53	Soybean	Sudden Death Syndrome	14	100	64	78
54	Soybean	Target Leaf Spot	25	57	100	72
55	Soybean	Yellow Mosaic	46	100	96	98
56	Tomato	Bacterial Spot	35	97	91	94
57	Tomato	Early Blight	56	88	66	76
58	Tomato	Healthy	45	92	100	96
59	Tomato	Late Blight	32	92	75	83
60	Tomato	Leaf Mold	65	70	88	83

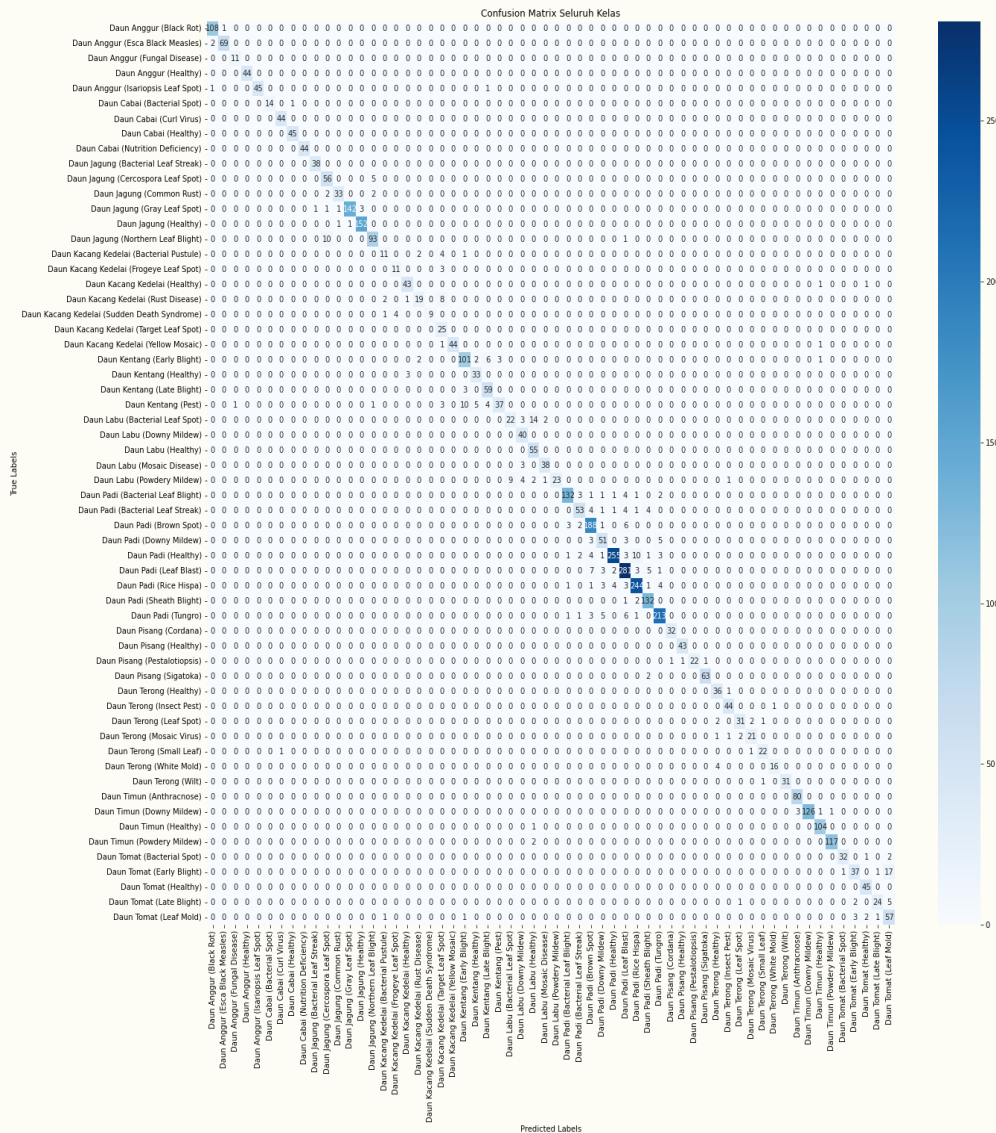


Figure 2. Confusion Matrix of the Selected Model

Figure 3 presents the 60-class confusion matrix of the selected model using the 70:20:10 data split after fine-tuning. The diagonal elements represent correctly classified samples, whereas the off-diagonal elements indicate misclassifications among the disease or leaf-condition classes. Most of the observations are concentrated along the diagonal, indicating that the model correctly classified the majority of the test samples. However, several classes exhibit more frequent off-diagonal predictions, which is consistent with the variation in class-level precision, recall, and F1-score reported in Table 8.

The class-level analysis reveals substantial variation in model performance that is not fully reflected by the overall accuracy of 92.03%. Pumpkin Bacterial Leaf Spot recorded the lowest F1-score at 61%, followed by Soybean Bacterial Leaf Pustule at 67%. Potato Pest and Pumpkin Powdery Mildew both achieved F1-scores of 73%. These lower values indicate that the model did not perform uniformly across all 60 classes. The differences may be associated with variations in visual characteristics, image quality, lighting, background conditions, disease severity, and the number of available samples per class. However, the individual contribution of these factors was not separately evaluated in this study.

Application Development Results

The trained CNN model was integrated into an Android-based mobile application named *Leaf Lives* using the Flutter framework. The application was designed to perform plant leaf disease classification directly on the device using a TensorFlow Lite model, allowing inference to be performed without an internet connection.

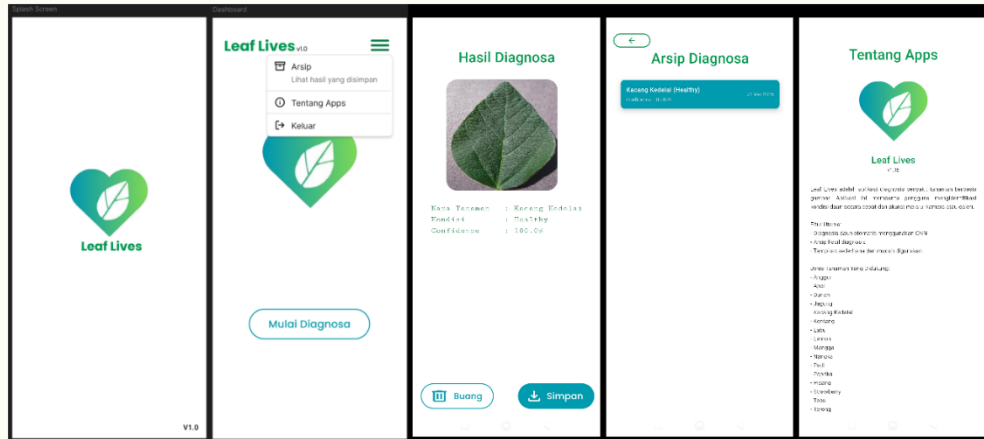


Figure 3. Leaf Lives User Interface

The application enables users to provide leaf images either by capturing them through the smartphone camera or by selecting images from the device gallery. The input image is processed locally by the embedded CNN model, and the predicted disease class is displayed as the classification result. The application also provides a local history feature for storing and reviewing previous classification results. The final TensorFlow Lite model had a file size of 15,935 KB (approximately 15.56 MiB). Functional testing was conducted using a POCO X6 Pro running Android 14, equipped with a MediaTek Dimensity 8300-Ultra processor, Mali-G615 MC6 GPU, and 12 GB RAM. The testing confirmed that the application could perform image classification from camera and gallery inputs and store classification history while operating offline. Quantitative measurements of inference latency and memory consumption were not collected during the original implementation. Therefore, the deployment results are limited to functional verification rather than quantitative evaluation of computational efficiency or usability.

The experimental results demonstrate a substantial improvement in classification performance after fine-tuning across all four data split scenarios. Recent EfficientNet-based work has similarly explored architectural refinement and interpretability for leaf disease classification (Assaduzzaman et al., 2025), while ensemble-based EfficientNet research further illustrates the potential benefits of complementing single-model decisions (Jannat et al., 2025). The average accuracy increased from 74.79% before fine-tuning to 91.39% after fine-tuning, indicating that further adapting the pretrained EfficientNetB0 representation to the plant leaf dataset substantially improved the classification performance. Among the four evaluated configurations, the 70:20:10 split achieved the highest accuracy, precision, recall, and F1-score, with values of 92.03%, 91.12%, 89.89%, and 89.96%, respectively. The same test subset was used across all four scenarios, while the remaining samples were allocated differently between the training and validation subsets. Therefore, the comparison was performed on the same test samples, reducing the influence of differences in test-set composition. Nevertheless, because each scenario was trained only once, the observed differences should be interpreted as outcomes of the individual experimental runs rather than as evidence of statistically significant superiority or a universally optimal data split.

The class-level analysis provides additional insight that is not fully represented by the overall accuracy. Performance varied considerably across the 60 disease or leaf-condition classes. Several classes achieved F1-scores close to or equal to 1.00, whereas others showed substantially lower performance. Pumpkin Bacterial Leaf Spot recorded the lowest F1-score at 61%, with a recall of 54%. This means that approximately 46% of the actual test samples belonging to this class were not correctly identified by the model. The importance of evaluating minority disease classes separately is also emphasized by a recent rice-leaf classification framework that addresses class imbalance using segmentation and resampling (Ongko et al., 2026). Other classes with relatively low recall included Pumpkin Powdery Mildew (57%), Soybean Bacterial Pustule (61%), Potato Pest (61%), Soybean Rust Disease (63%), Soybean Sudden Death Syndrome (64%), and Tomato Early Blight (66%). These results

are important because low recall indicates that a considerable proportion of actual cases may be missed by the classifier. In a potential agricultural decision-support context, such false-negative predictions could limit the usefulness of the application as an early screening tool, particularly when the misclassified class represents a condition requiring further inspection or treatment.

The confusion matrix in Figure 3 provides further evidence of the non-uniform classification performance across classes. Most observations are concentrated along the diagonal, indicating that the majority of samples were correctly classified. However, the presence of off-diagonal predictions demonstrates that certain classes were more difficult to distinguish from other classes. This pattern is consistent with the lower class-level precision, recall, and F1-score values reported in Table 8. In particular, the relatively low precision of Soybean Target Leaf Spot (57%) despite its recall of 100% indicates a different error pattern: the model identified all actual Target Leaf Spot samples in the test set, but a substantial number of samples from other classes were also predicted as Target Leaf Spot. Therefore, low performance should not be interpreted solely in terms of missed cases, as some classes may instead be affected primarily by false-positive predictions. Moreover, a closed-set classifier is generally forced to select one of its known classes, whereas an open-set potato-leaf study demonstrates the relevance of explicitly handling unfamiliar disease patterns (Verawati et al., 2025).

Several factors may contribute to the observed class-level variation. Classes with visually similar symptoms may be difficult to distinguish using leaf images alone, particularly when disease symptoms share similar patterns, colours, or lesion characteristics. Variations in image quality, lighting, background, viewing angle, and disease severity may also affect the visual features available to the model. In addition, the dataset contains substantial differences in the number of images available for each class. For example, some classes contain only a small number of images, whereas other classes contain several hundred or more. This imbalance may contribute to differences in class-level performance because the model is exposed to different amounts of visual variation for each class. Recent approaches have addressed these challenges with class-balancing and lesion-oriented segmentation (Ongko et al., 2026), while lightweight attention-based networks have been explored to enhance recognition of discriminative leaf features (Zhang et al., 2024). However, these factors were not independently controlled or experimentally isolated in this study; therefore, their individual contributions to the observed errors cannot be quantified.

The confusion matrix and class-level metrics should also be interpreted in the context of the composition of the dataset. The dataset was compiled from multiple public repositories, which introduces variation in image acquisition conditions, camera characteristics, backgrounds, and other dataset-specific properties. Although some source datasets contain field-captured images, the combined dataset does not necessarily represent the full range of conditions encountered in Indonesian agricultural environments. Real field images may include uncontrolled illumination, complex backgrounds, partial occlusion, varying leaf orientation, different camera qualities, and different stages of disease development. These factors may create a domain shift between the public images used for model development and images captured by users in actual agricultural settings. Concerns about the gap between curated image repositories and realistic agricultural scenes have been documented in recent dataset-focused research (Xu et al., 2024) and a review of field-based tomato disease studies (Jelali, 2024). Cross-domain experiments further indicate that adapting laboratory-trained disease recognition to field imagery remains a distinct methodological challenge (Yang et al., 2024).

Therefore, the reported 92.03% accuracy should be interpreted as classification performance on the evaluated test dataset rather than as evidence of diagnostic validity in the field. Leaf appearance alone may not always be sufficient to distinguish diseases with similar visual symptoms from nutritional deficiencies, pest-related damage, or other leaf conditions. Confirmation of the actual cause of a plant condition may require additional contextual information and expert assessment. The current study did not include validation by plant pathology experts, independent Indonesian field-image testing, or usability testing with farmers. Consequently, *Leaf Lives* should be regarded as a prototype for image-based classification rather than a validated agricultural diagnostic system. In addition, the open-set recognition literature shows why an unfamiliar symptom should not automatically be treated as one

of the predefined disease categories, supporting future consideration of unknown-class rejection and expert referral (Verawati et al., 2025).

From a deployment perspective, the selected model was successfully integrated into the *Leaf Lives* Android application and functionally operated offline on a POCO X6 Pro running Android 14. The TensorFlow Lite model had a file size of approximately 15.56 MiB, and the application was able to process images obtained from the camera or gallery and store classification history locally. These results demonstrate the technical feasibility of offline mobile integration. Research on lightweight attention-based architectures points to possible model-design directions for constrained deployment (Zhang et al., 2024); nevertheless, this study did not directly compare architectures or measure runtime efficiency. However, because inference latency and memory consumption were not quantitatively measured and testing was limited to a single device, the study does not establish computational efficiency across different smartphone configurations or broader device compatibility. Further validation is therefore required before practical use can be assessed.

Overall, the findings demonstrate that EfficientNetB0 with fine-tuning can provide promising multi-plant and multi-class classification performance on the public datasets evaluated in this study, while also showing that aggregate metrics can conceal substantial differences between individual classes. The results extend the application context of Indonesian CNN-based crop-disease systems already reported for rice (Rachman et al., 2024) and Android-based coffee-leaf identification (Saputra et al., 2024), but direct numerical comparisons would require harmonized datasets, labels, and evaluation protocols. Future research should extend the dataset with primary images collected under diverse Indonesian field conditions, evaluate the model with farmers and agricultural experts, verify predictions through plant pathology assessment, and investigate lightweight model optimization and device-level benchmarking to improve the reliability and efficiency of mobile deployment.

CONCLUSIONS

This study developed a Convolutional Neural Network (CNN)-based plant leaf disease and leaf-condition classification model using the EfficientNetB0 architecture and integrated the selected model into an Android mobile application named *Leaf Lives*. The dataset consisted of 44,030 leaf images covering 11 plant species and 60 disease or leaf-condition classes, including disease, healthy, nutritional deficiency, and pest-related conditions. The model achieved promising classification performance after fine-tuning. Among the four evaluated data split scenarios, the 70:20:10 configuration produced the highest performance, with an accuracy of 92.03%, macro-averaged precision of 91.12%, macro-averaged recall of 89.89%, and macro-averaged F1-score of 89.96%. Because each configuration was evaluated through a single training run, these results should be interpreted as the performance observed among the evaluated configurations rather than as evidence of a universally optimal data split.

The selected model was successfully converted into TensorFlow Lite format and integrated into the *Leaf Lives* Android application. The application supports image classification using images captured from the smartphone camera or selected from the gallery and provides offline inference and local classification history. The final TensorFlow Lite model had a size of approximately 15.56 MiB and was functionally tested on a POCO X6 Pro running Android 14. These findings demonstrate the technical feasibility of integrating the trained CNN model into an offline Android prototype. However, the current results do not establish field diagnostic validity, formal usability, broad device compatibility, or quantitative mobile computational efficiency. The study has several limitations. The dataset was compiled from multiple public repositories and, although duplicate images and low-quality images were removed before data partitioning, near-duplicate images were not systematically assessed using automated image-similarity methods. In addition, the classification experiments used non-stratified data partitioning and a single training run for each configuration. The public datasets may also not fully represent the visual conditions encountered in Indonesian agricultural fields. Furthermore, mobile deployment was functionally tested on a single device without quantitative measurements of inference latency or memory consumption, and the model was not independently validated after TensorFlow Lite conversion.

Future research should therefore evaluate the model using stratified and repeated experiments to obtain more robust estimates of performance, incorporate automated near-duplicate detection across data sources, and validate the model using independent field-image datasets collected under diverse agricultural conditions in Indonesia. Further work should also include post-TensorFlow Lite validation to verify consistency between the original and deployed models, device-level benchmarking of inference latency and memory consumption, and testing across multiple Android devices with different hardware configurations. In addition, evaluation involving farmers and validation by plant pathology experts should be conducted to assess usability and the diagnostic relevance of the model in practical agricultural settings. Lightweight model optimization may also be explored to improve mobile deployment efficiency.

REFERENCES

- Ahmad, M. H. (2025). *Seasonal Corn Leaf Disease Dataset: A Multi-Year Collection For Matang Analysis* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/vy629dngm8.1>
- Ali, A. (n.d.). *PlantVillage Dataset* [Data set]. Kaggle.
- Assaduzzaman, M., Bishshash, P., Nirob, M. A. S., Marouf, A. A., Rokne, J. G., & Alhaji, R. (2025). XSE-TomatoNet: An Explainable AI Based Tomato Leaf Disease Classification Method Using Efficientnetb0 With Squeeze-And-Excitation Blocks And Multi-Scale Feature Fusion. *MethodsX*, 14, 103159. <https://doi.org/10.1016/j.mex.2025.103159>
- Das, U., Shovo, S. S., Anan, I. J., & Mojumdar, M. U. (2024). *Banana And Banana Leaf Dataset For Classification And Disease Detection* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/5nfjzntwd8.1>
- Falaschetti, L., Manoni, L., Di Leo, D., Pau, D., Tomaselli, V., & Turchetti, C. (2022). A CNN-Based Image Detector For Plant Leaf Diseases Classification. *HardwareX*, 12, e00363. <https://doi.org/10.1016/j.ohx.2022.e00363>
- Felix, F., Wijaya, J., Sutra, S. P., Kosasih, P. W., & Sirait, P. (2020). Implementasi Convolutional Neural Network Untuk Identifikasi Jenis Tanaman Melalui Daun. *Jurnal SIFO Mikroskil*, 21(1), 1–10. <https://doi.org/10.55601/jsm.v21i1.672>
- Imtiaz, A., Swapnil, F. B. I., Masud, S. R., & Karmaker, D. (2025). *Tomato Leaf Dataset: A Dataset For Multiclass Disease Detection And Classification* (Version 2) [Data set]. Mendeley Data. <https://doi.org/10.17632/bpfd9cns5g.2>
- Jannat, N., Hasan, S. M. M., & Zibran, M. F. (2025). A Novel Ensemble Approach For Crop Disease Detection By Leveraging Customized Efficientnets And Interpretability. *Pattern Recognition Letters*, 197, 370–377. <https://doi.org/10.1016/j.patrec.2025.07.008>
- Jelali, M. (2024). Deep Learning Networks-Based Tomato Disease And Pest Detection: A First Review Of Research Studies Using Real Field Datasets. *Frontiers in Plant Science*, 15, 1493322. <https://doi.org/10.3389/fpls.2024.1493322>
- Muraina, I. O. (2022). Ideal Dataset Splitting Ratios In Machine Learning Algorithms: General Concerns For Data Scientists And Data Analysts. In *7th International Mardin Artuklu Scientific Researches Conference Proceedings*, 496–504.
- Nirob, M. A. S., Siam, A. K. M. F. K., Bishshash, P., & Assaduzzaman, M. (2025). *Chili plant leaf disease and growth stage dataset from Bangladesh* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/w9mr3vf56s.1>
- Ongko, E., Indrawati, A., & Sukiman, S. (2026). A Hybrid Framework Combining U-Net, Ant Colony Optimization, And CNN For Rice Leaf Disease Classification Under Class Imbalance. *Jurnal*

RESTI (Rekayasa Sistem dan Teknologi Informasi), 9(6), 1474–1484.
<https://doi.org/10.29207/resti.v9i6.6910>

- Rachman, Y., Susanti, P., Putra, A. B. R. P. A., & Rahmawati, N. I. (2024). Sistem Informasi Deteksi Penyakit Pada Tanaman Padi (Brown Spot, Hispa, Leaf Blast) Menggunakan Metode Convolutional Neural Network (CNN). *Decode: Jurnal Pendidikan Teknologi Informasi*, 4(3), 1193–1204. <https://doi.org/10.51454/decode.v4i3.846>
- Rafe, M. I. R., Nayem, F. M., Sarker, S. B., & Shiam, A. A. (2025). *Eggplant Leaf Disease Dataset* (Version 2) [Data set]. Mendeley Data. <https://doi.org/10.17632/mn8vfr9bw2.2>
- Rajput, A. S., Shukla, S., & Thakur, S. S. (2023). *Soynet: Indian Soybean Image Dataset With Quality Images Captured From The Agriculture Field (Healthy And Disease Images)* (Version 2) [Data set]. Mendeley Data. <https://doi.org/10.17632/w2r855hpx8.2>
- Rashid, M. R. A., Biswas, J., & Hossain, M. M. (2024). *Pumpkin Leaf Diseases Dataset From Bangladesh* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/wtxcw8wpbx.1>
- Rimon, S. I., Islam, M. R., Dey, A., & Das, A. (2022). PlantBuddy: An Android-Based Mobile Application For Plant Disease Detection Using Deep Convolutional Neural Networks. *Artificial intelligence and technologies*, 806, 275–285. https://doi.org/10.1007/978-981-16-6448-9_28
- Rokhmana, N., & Setiawan, D. (2022). Sistem Pakar Diagnosa Penyakit Tanaman Melon Berbasis Web Menggunakan Metode Forward Chaining Pada Kecamatan Gringsing. *Jurnal Publikasi Ilmu Komputer dan Multimedia (JUPIKOM)*, 1(2), 40–50. <https://doi.org/10.55606/jupikom.v2i1.887>
- Saputra, A. I., Weni, I., & Khaira, U. (2024). Implementasi Metode Convolutional Neural Network Untuk Deteksi Penyakit Pada Tanaman Kopi Arabika Melalui Citra Daun Berbasis Android. *Decode: Jurnal Pendidikan Teknologi Informasi*, 4(1), 41–51. <https://doi.org/10.51454/decode.v4i1.231>
- Shikalgar, A., Savalkar, A., Bhasme, A., Chavan, S., & Nikam, V. (2024). *Grape Disease* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/wkymf8bhcg.1>
- Sultana, N., Shorif, S. B., Akter, M., & Uddin, M. S. (2022). *Cucumber Disease Recognition Dataset* (Version 1) [Data set]. Mendeley Data. <https://doi.org/10.17632/y6d3z6f8z9.1>
- Trumanrase. (n.d.). *Rice Leaf Diseases* [Data set]. Kaggle.
- Tugrul, B., El Fatimi, E. H., & Eryigit, R. (2022). Convolutional Neural Networks In Detection Of Plant Leaf Diseases: A Review. *Agriculture*, 12, 1192. <https://doi.org/10.3390/agriculture12081192>
- Verawati, I., Hisam, M., & Pristyanto, Y. (2025). Open-Set Recognition For Potato Leaf Disease Identification Using OpenMax. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 9(4), 840–849. <https://doi.org/10.29207/resti.v9i4.6525>
- Xu, M., Park, J.-E., Lee, J., Yang, J., & Yoon, S. (2024). Plant Disease Recognition Datasets In The Age Of Deep Learning: Challenges And Opportunities. *Frontiers in Plant Science*, 15, 1452551. <https://doi.org/10.3389/fpls.2024.1452551>
- Yang, S., Feng, Q., Zhang, J., Yang, W., Zhou, W., & Yan, W. (2024). From Laboratory To Field: Cross-Domain Few-Shot Learning For Crop Disease Identification In The Field. *Frontiers in Plant Science*, 15, 1434222. <https://doi.org/10.3389/fpls.2024.1434222>
- Zhang, E., Zhang, N., Li, F., & Lv, C. (2024). A Lightweight Dual-Attention Network For Tomato Leaf Disease Identification. *Frontiers In Plant Science*, 15, 1420584. <https://doi.org/10.3389/fpls.2024.1420584>
- Zitu, M. Z. R., Shifat, S. R., & Mojumdar, M. U. (2024). *A Benchmark Dataset For Detecting Disease In Plant Leaves: An Essential Resource For Deep Learning Models* (Version 2) [Data set]. Mendeley Data. <https://doi.org/10.17632/v46jkbzbv3.2>