J-INTECH (Journal of Information and Technology)
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 10/C/C3/DT.05.00/2025
E-ISSN: 2580-720X || P-ISSN: 2303-1425

Implementation of a Plant Disease Identification System using the CNN Algorithm and the Web-Based Django Framework

Syifa Ilafiah[1], Hendri Ardiansyah[2*]

[1,2] Pamulang University, Faculty of Computer Science, Computer Science Department, Jl. Raya Puspitek, Buaran, Kec. Pamulang, South Tangerang, Banten 15310*

Keywords
Agriculture 4.0; Convolutional Neural Network (CNN); Django; Image classification; Machine Learning; Plant disease detection; Web-based system

Corresponding Author:
dosen00832@unpam.ac.id

Abstract
This study addresses the need for efficient and accessible plant disease identification systems in the era of Agriculture 4.0, where advances in artificial intelligence (AI) and machine learning (ML) support data-driven agricultural practices. The increasing popularity of home gardening highlights challenges faced by users in identifying plant diseases due to limited knowledge and diagnostic tools. Therefore, this research aims to develop a web plant disease detection system using the Django framework and convolutional neural networks (CNNs). The model was trained on a controlled dataset consisting of 57,320 leaf images collected from the PlantVillage and Turmeric Plant Disease datasets. Image preprocessing was applied, including resizing, normalization, and data augmentation such as image rotation, zooming, image inversion and brightness adjustmen. Class imbalance during training was handled using class weighting. The dataset is divided into a training set and a validation set for model development and evaluation. The CNN model achieved an accuracy of 92% on the labeled validation dataset, with a mean F1 score of 0.79 and a weighted mean F1 score of 0.92. For generalization testing, an uncontrolled (wild) dataset consisting of 223 images collected from online sources was used, resulting in an accuracy of 11%, indicating limited real-world generalization due to domain differences. Despite this limitation, the proposed system demonstrates the feasibility of CNN-based plant disease classification in a web application.

1. Introduction

Recent breakthroughs in information and communication technologies in recent years has fundamentally changed the way we do business in many sectors, and agriculture in particular stands out as a sector that has experienced significant changes. Progress in artificial intelligence (AI) and machine learning (ML) has enabled the implementation of data driven systems that can improve the efficiency and accuracy of decision making [1] . The concept of Agriculture 4.0 is the foundation of this transformation, integrating digital technologies such as the Internet of Things (IoT), AI, cloud computing, big data, and machine learning to support more productive and sustainable agriculture [2] . Technology that was previously only applied to large-scale agriculture is now being adopted by individual users, including home gardeners and urban communities.

Changes in people's lifestyles have contributed to an increase in interest in independent gardening, especially since the COVID-19 pandemic due to the implementation of lockdown policies. These restrictions on activities have had an impact on people's health and well-being, creating a need for activities that can support mental health while at home [3] . Gardening has been proven to provide psychological and physical benefits, especially during times of crisis, including for vulnerable groups such as the elderly [4] . In addition, integrated home garden interventions also contribute positively to improving household nutrition outcomes [5] . The emergence of healthy living and food self-sufficiency trends among urban youth has encouraged an increase in home gardening practices, including conventional and hydroponic vegetable cultivation. However, beginner gardeners often find it difficult to identify plant diseases when symptoms such as discoloration, spots, or wilting appear due to limited knowledge and a lack of diagnostic tools. This phenomenon is also reflected in online communities, where users often ask for help in identifying diseases by uploading pictures of their plants. On a broader scale, farmers still rely on conventional methods such as manual inspection, which is slow, requires special expertise, and is inefficient. This situation highlights the need for technology-based solutions that can support fast, accurate, and easily accessible plant disease identification for various groups.

Machine learning currently has become instrumental in the rapid, accurate, and quantifiable analysis of leaf images, significantly improving the efficiency of plant disease detection systems. Various machine learning algorithms have been widely adopted in previous studies, although their performance has demonstrated considerable variability, The Artificial Neural Network (ANN) achieved 97% accuracy in thyroid disease detection [6] and 90.48% accuracy in wheat grain classification. Support Vector Machine (SVM) has also been widely employed in various applications, achieving 96% accuracy in footwear type classification [7] and 76% accuracy in detecting apple plant diseases through leaf image analysis [8] Other algorithms, such as Self-Organizing Maps (SOM), achieve up to 83.3% accuracy in Painting Image Classification [9] though the classification results require additional interpretation and tend to be less robust when applied to complex image data. Convolutional Neural Networks (CNN) have proven to be superior because they are able to automatically extract features from raw images without manual feature engineering [10]. Furthermore, CNNs have been successfully applied across various image processing domains, including face recognition, object detection, and image classification [11], [12], [13], [14], [15], [16] . Based on these advantages, CNN was chosen in this study as the primary method for detecting plant diseases through analysis of leaf images uploaded by users. The model was trained using open source datasets such as PlantVillage, and Turmeric Plant Disease to automatically recognize disease patterns. Next, the CNN model was integrated into a web-based system using the Django framework.

Django is a Python-based framework that supports rapid, scalable web application development and is easily integrated with machine learning models, enabling users to obtain practical plant disease diagnoses by uploading leaf images [17]. The use of Django in this research is supported by scientific studies that highlight its capabilities in seamless integration between the backend and machine learning models, database management through ORM, and built-in security features such as protection against CSRF and SQL injection. In the context of ML-based applications, Django is widely used for developing web applications that present models in real-time or semi-real-time, as it facilitates efficient integration of models and web interfaces [18]. Another study comparing Django with other Python frameworks, such as Flask and FastAPI, shows that although Django has greater overhead, its built-in features, security, and ease of scalability make it more suitable for complex applications with end-to-end deployment needs, including web-based ML applications [19] . In its development process, this research used the Waterfall method because the system requirements were clearly defined from the outset, including leaf image upload functions, image processing using CNNs, disease prediction, and presentation of results via a web interface. The linear and systematic characteristics of Waterfall allow for structured documentation and evaluation at each stage of development, from requirements analysis to maintenance. The waterfall method is suitable for projects with stable requirements and a clearly defined scope [20], [21].

Based on this background, this study aims to to develop a web plant disease detection system using the Django framework and convolutional neural networks (CNNs). With this system, it is hoped that users can easily identify plant diseases through leaf image analysis, thereby increasing efficiency in agricultural management and preventing losses due to plant diseases that are not detected early.

2. Research Method

This research uses an experimental approach by develop a web plant disease detection system using the Django framework and convolutional neural networks (CNNs). The Waterfall approach was adopted because system requirements were clearly defined at the project's outset. This clarity enabled a structured development process, supported by comprehensive documentation across five sequential phases, from analysis to evaluation [22]. Figure 1 illustrates the stages of the Waterfall methodology in a graphical format.

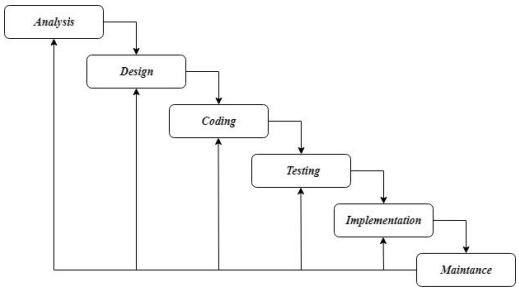

Figure 1. the Waterfall methodology
(Source: riset.guru)

The dataset used in this study consisted of plant leaf images from two open source datasets, namely PlantVillage [23] , and Turmeric Plant Disease Dataset [24] . These datasets were chosen because they provided plant leaf labeled images based on disease type and healthy condition, and were widely used in plant disease detection research, ensuring data reliability and relevance.

In the preprocessing stage, all images are resized to a resolution of 128 × 128 pixels using a padding-based resizing function to preserve aspect ratio. Pixel values are normalized to a range between 0 and 1 during the preprocessing stage. Augmentation was applied to the data using TensorFlow's ImageDataGenerator with a validation split of 0.3. The advanced settings include rotation up to 30°, horizontal and vertical shift by 0.2, zoom change by 0.3, shear strength change by 0.2, brightness adjustment from 0.7 to 1.3 and random horizontal mirroring, while in fill mode, interpolation of the nearest neighboring pixels is used. These transformations help to increase the diversity of the dataset and improve the model's ability to generalize results to different data distributions.. The study used a dataset of 57,320 images under controlled conditions, comprising 54,528 images from the PlantVillage dataset and 2,792 images from the Turmeric Disease dataset after data curation and removal of underrepresented classes. The controlled dataset is split into 40,124 training images (70%) and 17,222 validation images (30%). In addition, an uncontrolled (wild) dataset consisting of 223 images collected from online sources is used exclusively for external testing and is not included in the training or validation process.

A web plant disease detection system using the Django framework and convolutional neural networks (CNNs) implemented with TensorFlow and Keras. Input images with dimensions of 128 × 128 × 3 pixels are processed through a network structure with three sequential convolutional blocks. Each block uses a 3 × 3 kernel with ReLU activation, featuring filters of 32, 64, and 128 respectively, with 2 × 2 MaxPooling layers for spatial dimension reduction. Following feature extraction, the resulting output is flattened before being passed into a dense layer containing 256 units, activated by ReLU and incorporating L2 regularization at 0.001. To reduce overfitting, a dropout layer with a dropout factor of 0.4 was used. The output layer consists of fully connected layers activated by a softmax function, and the number of neurons corresponds to the total number of disease classes. The training setup consisted of the Adam optimizer, a cross-category cross-

entropy loss function, and accuracy as the primary evaluation metric. Figure 2 illustrates the proposed CNN architecture.

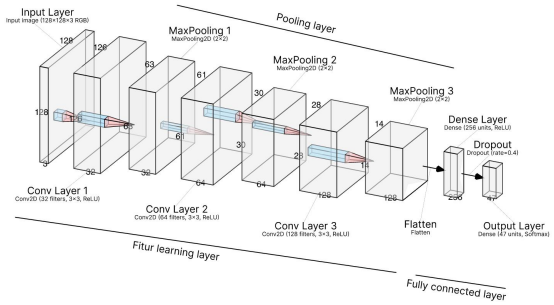

Figure 2. CNN-based framework for automated plant health classification

To test how well the model can distinguish healthy leaves from diseased leaves, its predictive performance was quantitatively assessed performance based on sensitivity, accuracy, precision, and F1 score. As detailed in Equation (1), the accuracy metric represents the proportion of correctly classified instances relative to the entire set of predictions. [25].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

Precision Focuses only on positive prediction results, shown in Equation (2). This metric is computed as the ratio of true positive (TP) classifications relative to the aggregate of all positive outcomes generated by the model (TP + FP). High precision means that the model's prediction is highly likely to be correct if it is "positive." This metric is especially important in scenarios where minimizing false positives is crucial.

$$Precision = \frac{TP}{TP+FP} \qquad (2)$$

As illustrated in Equation (3), recall is determined as the ratio of true positive (TP) classifications over the entirety of actual positive instances present within the dataset (TP + FN). This metric measures the network's efficiency in capturing every positive instance within the dataset. Consequently, elevated recall scores indicate a minimal rate of false negatives, showing that the architecture rarely misses true cases. The calculation emphasizes the importance of minimizing false negatives, especially in critical contexts such as plant disease diagnosis.

$$Recall = \frac{TP}{TP+FN} \qquad (3)$$

According to the formulation given in Equation (4), the F1 score serves as a composite metric that calculates the harmonic mean to balance precision and hit rate. This metric is essentially derived from the harmonic mean of the precision and hit rate values. In other words, both metrics are processed through a specialized formula yielding a balanced score that favors neither precision nor recall. When accuracy and precision are equally good, F1 scores are high. However, weaknesses in any of these components lead to a lower F1 score.

$$F1 = 2 \times = \frac{Precision \times Recall}{Precision+Recall} \qquad (4)$$

Additionally, a confusion matrix served to examine how the model predictions are distributed across the classes. To verify model stability and convergence throughout training, graphs depicting training and validation loss alongside accuracy metrics are examined. The confusion matrix itself does not directly produce a single number, but rather a table containing four main values: TP, TN, FP, and FN. From this table, all of the above metrics can be calculated. To use it, the first step is to record the number of predictions in each category (positive/negative correct or incorrect). These values are entered into the accuracy, precision,

recall, and F1-score calculations according to their respective definitions. Thus, the confusion matrix becomes the basis for calculations that combine all model evaluation indicatorsThis evaluation serves as the foundation for determining whether the model is suitable for integration into the developed plant disease detection system. Typically, the confusion matrix formula is presented in a tabular format as shown below [26]:

Table 1. Evaluation matrix

	Predticted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Following training and evaluation, the CNN model was implemented in a web application built using the Django framework. This application offers an interface that allows users to upload images of plant leaves. The uploaded images are analyzed by a CNN model that generates suggestions for disease detection and treatment. This system is based on user-friendly design principles and ensures accessibility on a variety of devices. After implementation, the web application goes through a testing process before release. System testing uses black box testing techniques that all functions operate in compliance with system requirements. System evaluation is based on the accuracy of diagnosis results, ease of use of the application, and the effectiveness of the system in helping users independently identify plant diseases.

3. Result and Discussions

Implementation is the stage where all system designs created in the previous chapter are realized in the form of an application that can be run and used by users. This process encompasses implementing a Convolutional Neural Network (CNN)-based model for plant disease detection and creating a web-based user interface through the Django framework. The implementation process follows the Convolutional Neural Network (CNNs) model architecture and the data preprocessing stages described earlier. The model has been trained using a combined dataset that has undergone augmentation and normalization processes to ensure optimal model performance in classifying plant disease types based on visual leaf characteristics. All system components, from the image processing backend to the user interface frontend, are integrated and tested to ensure that the system runs according to the designed objectives.

3.1 *Training Configuration*

The model was developed using Python 3.10 with TensorFlow and Keras and trained on 128×128 pixel leaf images. Training was performed for 50 epochs with a batch size of 32, using the Adam optimizer and categorical cross-entropy loss function. The controlled dataset was divided into 70% training data and 30% validation data. Following data curation and the removal of underrepresented classes, the final controlled dataset comprised 57,320 images, including 54,528 images from the PlantVillage dataset and 2,792 images from the Turmeric Disease dataset. To evaluate the model's generalization capability, an additional uncontrolled (wild) dataset consisting of 223 images collected through online scraping was used exclusively for external testing.

3.2 *Model Training Process*

The Convolutional Neural Network (CNN) model was trained for 50 epochs with an architecture consisting of three convolutional layers, each followed by a pooling layer, as well as one dense layer and dropout, which functions as a regularization mechanism. The training process produced accuracy and loss graphs as shown in Figure 3.

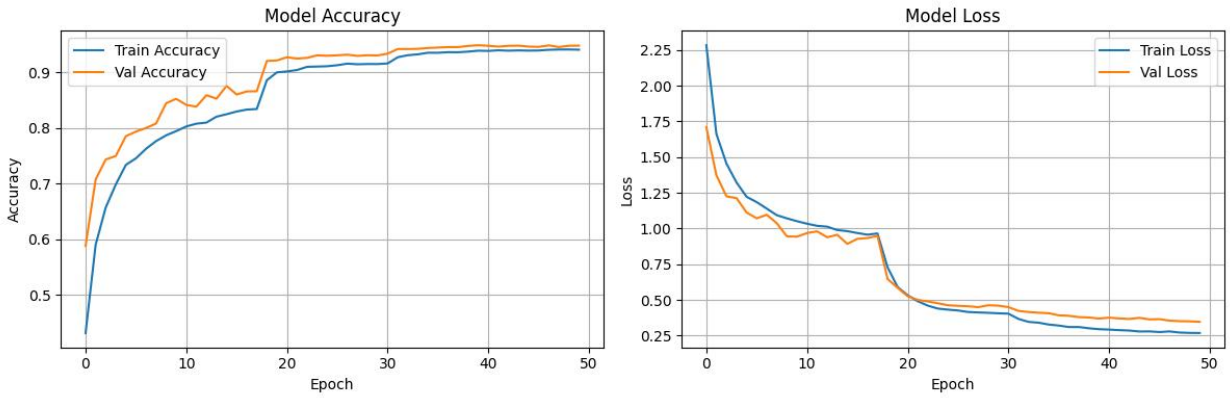

Figure 3. Accuracy and loss graphs during the training process

Based on the accuracy curves presented in Figure 3, the CNN model demonstrated a substantial improvement in classification performance throughout the training process. During the initial training phase, the training accuracy increased markedly from approximately 43% at the first epoch to nearly 80% by the 10 epoch. This rapid increase suggests that the model was able to effectively capture and learn discriminative features from the leaf image dataset, including visual characteristics associated with plant health conditions and disease symptoms.

The validation accuracy exhibited a similar upward trajectory and, in several early epochs, slightly exceeded the training accuracy. This phenomenon may be attributed to the implementation of regularization techniques, particularly dropout, which is applied during training but disabled during evaluation. Consequently, the model may exhibit slightly higher performance when assessed on validation data. As training progressed, both training and validation accuracy curves gradually converged and stabilized within the range of 92-95% after approximately the 20 epoch. The close alignment of these curves indicates that the model achieved a satisfactory balance between learning and generalization, with no evident signs of overfitting.

The loss curves further support these observations. At the beginning of training, the model recorded a relatively high training loss of approximately 2.30, reflecting the substantial prediction errors typically observed during the early learning stages. However, the loss decreased rapidly during the first several epochs and continued to decline progressively throughout the training process. By the final epoch, the training loss had reached approximately 0.26, while the validation loss stabilized around 0.34. The consistent reduction in both metrics indicates that the optimization algorithm effectively minimized the classification error and facilitated stable convergence. A notable decline in both training and validation loss is observed around epochs 18–20, suggesting that the model entered a more effective optimization phase and acquired increasingly representative feature embeddings. Following this stage, both loss curves continued to decrease smoothly with minimal fluctuations, indicating a stable learning process. Furthermore, the absence of a substantial gap between training and validation loss suggests that the model maintained strong generalization performance and did not exhibit significant overfitting throughout the training period.

Overall, the results demonstrate that the proposed CNN architecture successfully learned meaningful representations from the leaf image dataset and achieved robust classification performance over 50 training epochs. The high validation accuracy, coupled with the relatively low validation loss, indicates that the model generalized effectively to unseen data. Moreover, the close correspondence between the training and validation metrics throughout the training process confirms the effectiveness of the adopted training configuration and regularization strategy in producing a reliable and well-generalized classification model.

3.3 *Model Evaluation*

The model evaluation aims to measure the CNN's ability to classify plant diseases under two conditions: (1) controlled test dataset and (2) wild data that reflects real conditions. The evaluation was conducted using a confusion matrix, accuracy, precision, recall, and F1-score. The confusion matrix for the test data is shown in Figure 4.

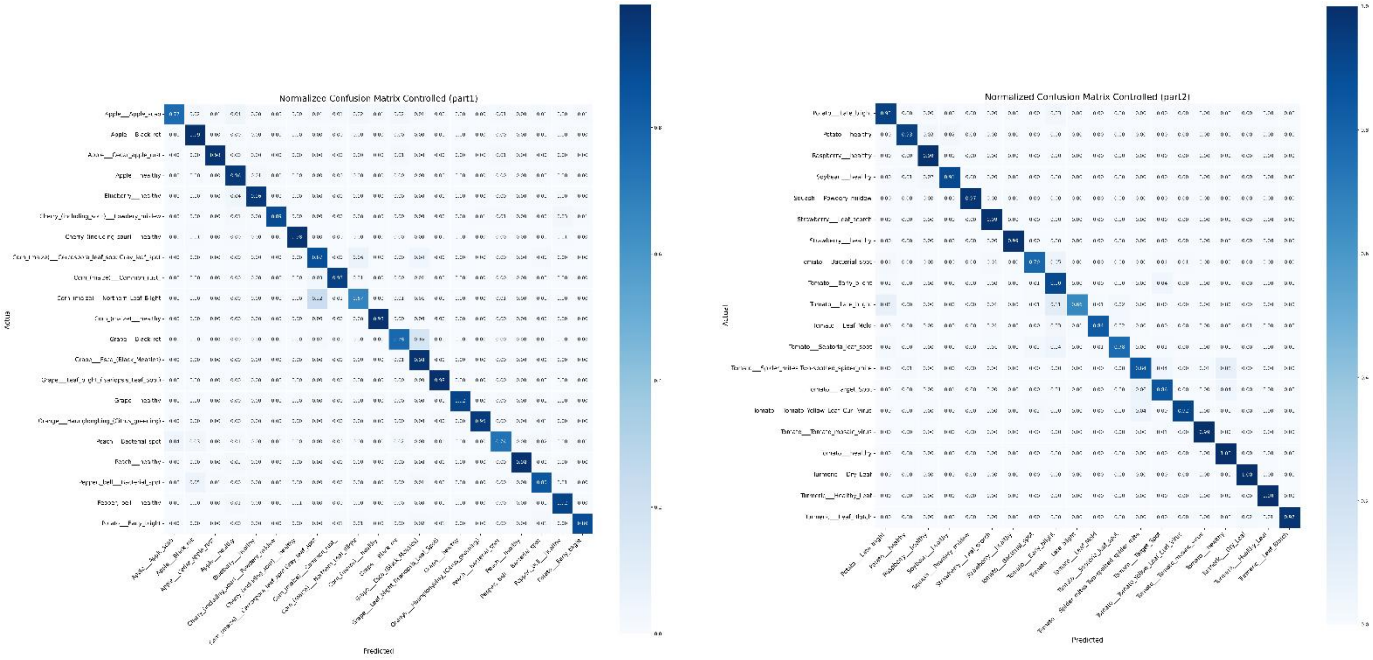

Figure 4. Confusion matrix CNN model disease detection with controlled test dataset

The confusion matrix presented in Figure 4 illustrates the classification performance of the proposed CNN model on the validation subset of the controlled dataset divided into 2 images . The matrix is characterized by a strong concentration of values along the main diagonal, represented by darker color intensities, indicating that the majority of validation samples were correctly classified. In contrast, the off-diagonal regions appear relatively sparse and lighter in color, suggesting a low frequency of misclassification across classes. This pattern demonstrates that the model was able to effectively learn and distinguish the visual characteristics associated with different plant species and disease categories within the controlled dataset. The limited occurrence of classification errors further indicates that the model achieved high predictive accuracy and consistent performance under standardized data conditions.

Based on the quantitative evaluation results presented in Table 2, the proposed model achieved an overall accuracy of 0.92. The macro F1-score of 0.79 suggests that the model achieved relatively consistent performance across classes, although some variation remains among underrepresented categories. Meanwhile, the weighted F1-score of 0.92 indicates strong overall predictive performance when accounting for class distribution. The difference between these metrics highlights the impact of class imbalance, where majority classes contribute more substantially to the aggregated performance scores.

Table 2. Classification report controlled test dataset

No	Plant Disease Class	Precision	Recall	F1-Score	Number of Test Samples
1	Apple_Apple_scab class	0.91	0.77	0.83	196
2	Apple_Black_rot class	0.72	0.99	0.84	186
3	Apple_Cedar_apple_rust class	0.97	0.91	0.94	82
4	Apple_healthy class	0.84	0.99	0.91	493
5	Blueberry_healthy class	0.99	0.97	0.98	450
6	Cherry_(including_sour)_Powdery_mildew class	1.00	0.90	0.95	315
7	Cherry_(including_sour)_healthy class	0.96	0.98	0.97	256
8	Corn_maize_Cercospora_leaf_spot_Gray_leaf_spot	0.63	0.87	0.73	159
9	Corn_(maize)_Common_rust class	0.96	0.93	0.94	365
10	Corn_(maize)_Northern_Leaf_Blight class	0.89	0.75	0.81	306
11	Corn_(maize)_healthy class	0.99	0.98	0.98	348
12	Grape_Black_rot class	0.89	0.87	0.88	363
13	Grape_Esca_(Black_Measles) class	0.85	0.99	0.92	414
14	Grape_Leaf_blight_(Isariopsis_Leaf_Spot) class	0.91	1.00	0.95	322
15	Grape_healthy class	0.90	0.94	0.92	126
16	Orange_Haunglongbing_(Citrus_greening) class	1.00	0.99	1.00	1652
17	Peach_Bacterial_spot class	0.98	0.93	0.95	689
18	Peach_healthy class	1.00	0.94	0.97	108
19	Pepper_bell_Bacterial_spot class	0.88	0.82	0.85	299
20	Pepper_bell_healthy class	0.96	0.91	0.94	443
21	Potato_Early_blight class	0.67	0.90	0.77	309
22	Potato_Late_blight class	0.91	0.85	0.88	300
23	Potato_healthy class	0.71	0.27	0.39	45
24	Raspberry_healthy class	0.83	0.98	0.90	111
25	Soybean_healthy class	0.94	0.99	0.96	1527
26	Squash_Powdery_mildew class	0.97	1.00	0.98	550
27	Strawberry_Leaf_scorch class	0.92	0.98	0.95	332
28	Strawberry_healthy class	0.94	0.95	0.95	136
29	Tomato_Bacterial_spot class	0.94	0.95	0.94	638
30	Tomato_Early_blight class	0.80	0.81	0.81	300
31	Tomato_Late_blight class	0.97	0.78	0.86	582
32	Tomato_Leaf_Mold class	0.98	0.84	0.91	292
33	Tomato_Septoria_leaf_spot class	0.95	0.82	0.88	531
34	Tomato_Spider_mites Two-spotted_spider_mite class	0.94	0.61	0.74	502
35	Tomato_Target_Spot class	0.87	0.76	0.81	421
36	Tomato_Tomato_Yellow_Leaf_Curl_Virus class	1.00	0.97	0.98	1607
37	Tomato_Tomato_mosaic_virus class	1.00	0.80	0.89	111
38	Tomato_healthy class	0.68	1.00	0.81	477

No	Plant Disease Class	Precision	Recall	F1-Score	Number of Test Samples
39	Turmeric_Dry_Leaf class	0.95	1.00	0.97	243
40	Turmeric_Healthy_Leaf class	0.98	1.00	0.99	295
41	Turmeric_Leaf_Blotch class	0.96	0.94	0.95	298
	accuracy	0.92	0.92	0.92	0.92
	macro avg	0.80	0.79	0.79	17222
	weighted avg	0.93	0.92	0.92	17222

The achieved accuracy of 92% on the controlled validation dataset demonstrates robust performance and is comparable to several recent studies in plant disease detection. For comparison, Tejaswini et al. [27] used a custom CNN architecture on the PlantVillage dataset and achieved an accuracy of approximately 86,21%, which closely aligns with our results using a similar lightweight approach. In contrast, more complex architectures employing transfer learning, such as the ResNet-50 model used by Walewangko S et al [28] , reported substantially higher accuracies exceeding 96,32% on the same dataset.

To evaluate the model's generalization ability in real-world conditions, additional testing was conducted using wild data, namely leaf images obtained from external sources such as the internet. Unlike the training dataset, which is standardized with uniform lighting, homogeneous backgrounds, and consistent image quality, wild data has a much higher level of variation. This variation includes differences in lighting, shooting angles, complex backgrounds, and inconsistent image quality, resulting in a distribution that differs significantly from the training data.

The evaluation results show a significant decline in model performance when tested using wild data. The confusion matrix in (Figure 5) shows that the model predictions are not centered on the main diagonal, with a very limited number of correct predictions and an error distribution spread across various classes. This pattern indicates that the model has difficulty recognizing plant diseases in real-world images, mainly due to the high variation in lighting, background, and leaf texture that is not adequately represented in the training data.

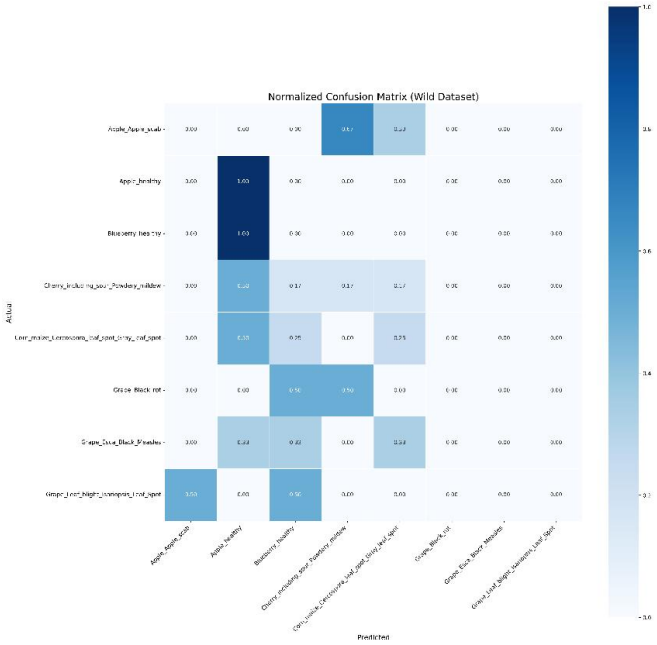

Figure 5. Confusion matrix CNN model disease detection with wild dataset

Overall accuracy on the wild data only reached 0.11, with most classes having precision, recall, and F1-score values of 0, as shown in Table 3. The low scores on almost all of these metrics indicate that the model failed to provide meaningful predictions in most classes. This condition illustrates that the model is unable to generalize to new domains, despite showing excellent performance on controlled test data.

Table 3. Classification report wild dataset

No	Plant Disease Class	Precision	Recall	F1-Score	Number of Test Samples
1	Apple_Apple_scab class	0	0	0	24
2	Apple_Black_rot class	0	0	0	0
3	Apple_Cedar_apple_rust class	0	0	0	0
4	Apple_healthy class	0	0	0	0
5	Blueberry_healthy class	0	0	0	0
6	Cherry_including_sour_Powdery_mildew class	0	0	0	0
7	Cherry_including_sour_healthy class	0	0	0	0
8	Corn_maize_Cercospora_leaf_spot_Gray_leaf_spot class	0.29	0.6	0.39	20
9	Corn_maize_Common_rust class	0.06	0.04	0.05	26
10	Corn_maize_Northern_Leaf_Blight class	0.38	0.23	0.29	35
11	Corn_maize_healthy class	0	0	0	0
12	Grape_Black_rot class	0.38	0.1	0.16	30
13	Grape_Esca_Black_Measles class	0	0	0	0
14	Grape_Leaf_blight_Isariopsis_Leaf_Spot class	0	0	0	0
15	Grape_healthy class	0	0	0	0
16	Orange_Haunglongbing_Citrus_greening class	0	0	0	0
17	Peach_Bacterial_spot class	0	0	0	0
18	Peach_healthy class	0	0	0	0
19	Pepper_bell_Bacterial_spot class	0	0	0	0
20	Potato_Early_blight class	0	0	0	31
21	Tomato_Late_blight class	0.5	0.03	0.06	34
22	Tomato_Leaf_Mold class	0	0	0	23
	accuracy	0.11	0.11	0.11	223
	macro avg	0.03	0.02	0.02	223
	weighted avg	0.22	0.11	0.11	223

The difference between the 41 classes in the original data and the 22 experimental classes in the real-world data is in consequence of by the difficulty of obtaining verifiable real-world images for all plant disease classes using web scraping techniques. Classes with a support value of 0 indicate a lack of representative real-world data, further highlighting the uneven distribution of agricultural data in reality.

The extremely significant decline in accuracy to 11% underscores the presence of a serious domain shift challenge, a phenomenon frequently observed in Agriculture 4.0 implementations. Models trained on controlled or "studio-like" datasets, such as PlantVillage, often exhibit limited generalization when applied to real-world field images because of high-dimensional noise, including variations in illumination and complex backgrounds [29]. Similar generalization gaps have also been reported during the transition from controlled experimental environments to real-world agricultural settings, which remains one of the primary obstacles to

the large-scale deployment of deep learning models [30] . Furthermore, high-performance deep learning models frequently require extensive domain adaptation to maintain robust performance under diverse environmental conditions, particularly in localized agricultural contexts such as Indonesia [31] . Therefore, the 11% accuracy obtained in this study is consistent with the generalization gap reported in previous research, indicating that model robustness is substantially degraded when the testing data distribution differs from that of the training environment.

The contrasting performance between controlled test data and wild data indicates a domain shift problem, where the model becomes overly dependent on the visual characteristics present in the training dataset and struggles to generalize to unseen data distributions. This finding suggests that the internal accuracy of 92% does not fully reflect the model's performance under real-world operational conditions. Domain shift remains a critical challenge in plant disease recognition systems, as variations between training and testing data distributions can substantially reduce model performance. Models trained using controlled datasets frequently exhibit decreased accuracy when deployed in field environments due to differences in illumination, background complexity, image quality, and environmental conditions. Therefore, improving model generalization requires strategies such as increasing data diversity, applying advanced data augmentation techniques, balancing class distribution, and adopting domain adaptation methods[32].

3.4 System Testing

System testing was conducted to evaluate whether all functionalities of the developed system operated correctly and fulfilled the specified requirements. The testing process aimed to verify that each feature could receive user inputs, process data appropriately, and generate outputs consistent with the expected results. In addition, system testing was performed to identify potential functional errors before the application was deployed for actual use.

The evaluation employed the Black Box Testing approach [33] , which focuses on assessing system functionality without examining the internal implementation or source code. This method evaluates the system from the end-user perspective by validating whether the outputs produced correspond to the given inputs. Therefore, the testing process emphasizes functional correctness rather than program logi .

To ensure comprehensive validation, several testing scenarios were designed for each major module of the application, including the Home page, Detection page, Result page, and About page. Each scenario examined specific functionalities such as page accessibility, navigation, image upload processing, prediction execution, result visualization, recommendation generation, and input validation. The detailed testing scenarios and results are presented in Tables 4–8.

Table 4. Black Box Testing Results for the Home Page

No	Feature Tested	Input	Expected Output	Status
1	Home Page Loading	Access the main URL	The system displays the website description correctly	Passed
2	"Try Now" Button	Click the button	The system redirects the user to the Detection (Use) page	Passed

Table 5. Black Box Testing Results for the Detection (Use) Page

No	Feature Tested	Input	Expected Output	Status
1	Image Upload	Upload a .jpg or .png file	The uploaded file name is displayed in the preview area	Passed
2	"Detect"	Click the button after	The system displays the	Passed

No	Feature Tested	Input	Expected Output	Status
	Button	uploading an image	prediction result page	
3	Input Validation	Click the "Detect" button without uploading an image	The system prevents the detection process from being initiated	Passed

Table 6. Black Box Testing Results for the Result Page

No	Feature Tested	Input	Expected Output	Status
1	Disease Prediction Result Display	Access the page after a successful detection process	The predicted plant disease name is displayed automatically	Passed
2	Treatment Recommendation Display	Predicted disease generated by the model	A list of treatment recommendations is displayed according to the prediction result	Passed
3	"Re-Detect" Button	Click the "Re-Detect" button	The system redirects the user to the Detection (Use) page for a new detection process	Passed

Table 7. Black Box Testing Results for the About Page

No	Feature Tested	Input	Expected Output	Status
1	About Page Loading	Access the About page	The system displays information about the application correctly	Passed

Based on the testing results, all evaluated functionalities successfully produced outputs that matched the expected behavior. The system was able to perform image uploads, execute disease classification, display prediction results, provide treatment recommendations, and support navigation between pages without functional issues. These findings indicate that the developed system satisfies its functional requirements and is ready for practical implementation.

4. Conclusions and Future Works

This study successfully designed and implemented a web-based plant disease identification system using a Convolutional Neural Network (CNN) algorithm and the Django framework. The developed system enables users to identify plant diseases by uploading leaf images through a user-friendly web interface. The CNN model was trained using a controlled dataset consisting of 57,320 images from the PlantVillage and Turmeric Disease datasets, supported by preprocessing techniques such as image resizing, normalization, and data augmentation. Evaluation results on the controlled validation dataset showed that the model achieved a final classification accuracy of 92%, with a macro-average F1-score of 0.79 and a weighted-average F1-score of 0.92, indicating good classification performance under controlled conditions. In addition, black-box testing confirmed that all major system functionalities operated according to the specified requirements, demonstrating the successful implementation of the web application.

Despite these promising results, evaluation on an uncontrolled (wild) dataset consisting of 223 images revealed a significant decrease in performance, achieving only 11% accuracy with a macro-average F1-score of 0.02 and a weighted-average F1-score of 0.11. This finding indicates that the model's generalization capability remains limited when exposed to real-world conditions. Variations in image quality, lighting conditions, backgrounds, camera angles, and disease symptom representation likely contributed to the decline in performance. Therefore, while the proposed system demonstrates the feasibility of integrating CNN-based plant disease classification into a web platform, further improvements are necessary before it can be reliably applied in practical agricultural environments.

Future work should focus on improving the model's robustness and generalization performance by expanding and diversifying the training dataset with more real-world images collected under varying environmental conditions. Addressing class imbalance and exploring more advanced deep learning architectures, such as MobileNet, EfficientNet, ResNet, or ensemble learning approaches, may further enhance classification performance. Additional optimization of training parameters and regularization techniques should also be considered. From the system development perspective, future enhancements may include features such as detection history, geolocation support, cloud deployment, and integration with IoT-based agricultural technologies. Furthermore, conducting usability evaluations involving farmers and agricultural practitioners, as well as improving responsiveness across mobile devices, would provide valuable insights into the system's practical applicability and user acceptance in real-world agricultural settings.

5. References

[1] Hilman Ferdinandus Pardede, *Penerapan Pembelajaran Mesin (Machine Learning) dan Pembelajaran Dalam (Deep Learning) Berkinerja Tinggi untuk Mendukung Sektor Pertanian di Indonesia*. Penerbit BRIN, 2023. doi: 10.55981/brin.872.

[2] S. O. Araújo, R. S. Peres, J. Barata, F. Lidon, and J. C. Ramalho, "Characterising the agriculture 4.0 landscape—emerging trends, challenges and opportunities," Apr. 01, 2021, *MDPI AG*. doi: 10.3390/agronomy11040667.

[3] A. Theodorou, A. Panno, G. Carrus, G. A. Carbone, C. Massullo, and C. Imperatori, "Stay home, stay safe, stay green: The role of gardening activities on mental health during the Covid-19 home confinement," *Urban For. Urban Green.*, vol. 61, p. 127091, Jun. 2021, doi: 10.1016/J.UFUG.2021.127091.

[4] J. Corley *et al.*, "Home garden use during COVID-19: Associations with physical and mental wellbeing in older adults," *J. Environ. Psychol.*, vol. 73, Feb. 2021, doi: 10.1016/j.jenvp.2020.101545.

[5] L. Depenbusch, P. Schreinemachers, S. Brown, and R. Roothaert, "Impact and distributional effects of a home garden and nutrition intervention in Cambodia," *Food Secur.*, vol. 14, no. 4, pp. 865–881, Aug. 2022, doi: 10.1007/s12571-021-01235-y.

[6] M. D. Rengganis, T. I. Hermanto, and M. A. Sunandar, "Implementasi Metode ANN untuk Klasifikasi Diagnosis Tiroid Berbasis Aplikasi Mobile," *Jurnal Riset dan Aplikasi Mahasiswa Informatika (JRAMI)*, vol. 6, no. 4, pp. 828–837, 2025, doi: https://doi.org/10.30998/jrami.v6i04.14502.

[7] Gina Annisa and Ninuk Wiliani, "Perbandingan Model CNN dan SVM untuk Klasifikasi Jenis Footwear pada Dataset Alas Kaki Berbasis Citra," *Teknomatika: Jurnal Informatika dan Komputer*, vol. 18, no. 1, pp. 28–36, Jun. 2025, doi: 10.30989/teknomatika.v18i1.1564.

[8] E. Nahak, R. P. Putra, and F. Marisa, "Klasifikasi Penyakit Pada Tanaman Apel Melalui Citra Daun Menggunakan Metode Multiclass Support Vector Machine," vol. 11, no. 3, pp. 401–408, 2024, doi: https://doi.org/10.35957/jatisi.v11i3.9153.

[9] R. Eka Nanda, Y. Denny Prabowo, F. Industri Kreatif, and I. Teknologi dan Bisnis Kalbis Jalan Pulomas Selatan Kav, "Pengembangan Model Pembelajaran Mesin untuk Klasifikasi Citra Lukisan Menggunakan Self-Organizing Map dengan Library Minisom," 2022.

[10] D. Sembiring, I. Taufik, H. Syahputa, Z. Indra, and K. Saputa, "Classification of Stinging Nettle Plants Based on Leaf Images Using the CNN Method (Case Study: Biru-Biru Village)," *Journal of Information and Technology) Accredited Sinta*, vol. 13, no. 02, Dec. 2025, doi: https://doi.org/10.32664/j-intech.v13i02.2131.

[11] M. L. Prasetyo *et al.*, "Face Recognition Using the Convolutional Neural Network for Barrier Gate System," *International Journal of Interactive Mobile Technologies*, vol. 15, no. 10, pp. 138–153, 2021, doi: 10.3991/ijim.v15i10.20175.

[12] A. TiaraSari and E. Haryatmi, "Penerapan Convolutional Neural Network Deep Learning dalam Pendeteksian Citra Biji Jagung Kering," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 2, pp. 265–271, Apr. 2021, doi: 10.29207/resti.v5i2.3040.

[13] E. Predianto and B. Sutomo, "Klasifikasi Jenis Bunga dengan Algoritma Convolutional Neural Network (CNN) Menggunakan Metode Region-Based Convolutional Neural Network (R-CNN)," *Jurnal Pendidikan Teknologi informasi*, vol. 8, no. 2, pp. 1–15, 2024.

[14] H. Hairani and T. Widiyaningtyas, "Augmented Rice Plant Disease Detection with Convolutional Neural Networks," *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi*, vol. 8, no. 1, pp. 27–39, Feb. 2024, doi: 10.29407/intensif.v8i1.21168.

[15] A. Indarto and K. Kusrini, "Application of Convolutional Neural Network Based on ResNet18 for Alzheimer Disease Classification," *International Journal of Artificial Intelligence Research*, vol. 9, no. 2, Dec. 2025, doi: 10.29099/ijair.v9i2.1504.

[16] A. Latifa, H. Kurniawan, K. Rohmat Hidayat, and N. Larasati, "Implementasi Deep Learning Algoritma Convolutional Neural Network untuk Klasifikasi Kesegaran Buah dan Sayur," vol. 13, no. 2, pp. 319–328, 2026, doi: https://doi.org/10.25126/jtiik.2026131.

[17] Sadriddinovich Jalolov Stand up and O. F. Altiboyevna, "Innovative Developments and Research in Education International scientific-online conference," in *International scientific-online conference*, Canada, May 2023, pp. 538–541. Accessed: Jun. 10, 2026. [Online]. Available: https://www.researchgate.net/publication/388277035_INNOVATIVE_DEVELOPMENTS_AND_RESEARCH_IN_EDUCATION_International_scientific-online_conference

[18] K. Mani and A. K. B. Shenoy, "Machine learning models in web applications: A comprehensive review," *ICT Express*, no. April, 2025, doi: 10.1016/j.icte.2025.09.001.

[19] Abhinaya V, Akhila S, Deepika K, and Hamsa S, "Comparative Analysis of Full-Stack Python Frameworks: Django, Flask, and FastAPI," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 09, no. 05, pp. 1–9, 2025, [Online]. Available: www.ijsrem.com

[20] A. Widyantoro, F. Faradisa Al Bina, T. Prayoga, R. Safei, and M. Akmal Arrasid, "Systematic Literature Review: Membandingkan Pendekatan Metode Agile dan Waterfall Dalam Pengembangan Perangkat Lunak," *Journal of Comprehensive Science (JCS)*, vol. 4, no. 1, pp. 183–193, 2025, doi: 10.59188/jcs.v4i1.2969.

[21] T. Thesing, C. Feldmann, and M. Burchardt, "Agile versus Waterfall Project Management: Decision model for selecting the appropriate approach to a project," *Procedia Comput. Sci.*, vol. 181, pp. 746–756, 2021, doi: 10.1016/j.procs.2021.01.227.

[22] A. M. Dawis *et al.*, *Rekayasa Perangkat Lunak Panduan Praktis untuk Pengembangan Aplikasi Berkualitas*, Cetakan Pe. Bandung: Widina Media Utama, 2023.

[23] Samir Bhattarai, "New Plant Diseases Dataset," San Francisco, CA, USA, 2018. Accessed: Jun. 16, 2025. [Online]. Available: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

[24] SIAM, A K M FAZLUL KOBIR, Nirob, Sharker Asraful, Bishshash, and Prayma, "Turmeric Plant Disease Dataset: Advancing AI for Agricultural Sustainability," 2025. doi: 10.17632/g46dvrcvwn.2.

[25] M. Ulfa, T. F. Abidin, and K. Muchtar, "Ripeness Detection and Shelf Life Prediction of Avocados Using YOLOv8 and Hybrid Machine Learning," *Journal of Information Systems Engineering and Business Intelligence*, vol. 12, no. 1, pp. 31–42, Feb. 2026, doi: 10.20473/jisebi.12.1.31-42.

[26] A. A. Anggara and A. Ridho, "Analisa Penyakit Pada Tanaman Cabai Merah (Capsicum Annuum L) dengan Membandingkan Tingkat Akurasi Menggunakan Metode Convolutional Neural Network (CNN) dan K-Nearest Neighbor (KNN)," vol. 4, no. 1, p. 52, 2025, doi: 10.35308/jti.v4i1.11816.

[27] Tejaswini, Rastogi Priyanka, Dua Swayam, Manikanta, and Dagar Vikas, "Early Disease Detection in Plants using CNN," 2024. doi: https://doi.org/10.1016/j.procs.2024.04.327.

[28] S. Walewangko *et al.*, "Deep Learning-Based Plant Leaf Disease Classification Using Convolutional Neural Network," *Kohesi: Jurnal Multidisiplin Saintek*, vol. 10, no. 6, 2025, doi: 10.8734/Kohesi.v1i2.365.

[29] M. Xu *et al.*, "Embracing limited and imperfect training datasets: opportunities and challenges in plant disease recognition using deep learning," *Front. Plant Sci.*, vol. 14, 2023, doi: 10.3389/fpls.2023.1225409.

[30] S. O. Araújo, R. S. Peres, J. Barata, F. Lidon, and J. C. Ramalho, "Characterising the agriculture 4.0 landscape—emerging trends, challenges and opportunities," Apr. 01, 2021, *MDPI AG*. doi: 10.3390/agronomy11040667.

[31] Hilman Ferdinandus Pardede, *Penerapan Pembelajaran Mesin (Machine Learning) dan Pembelajaran Dalam (Deep Learning) Berkinerja Tinggi untuk Mendukung Sektor Pertanian di Indonesia*. Penerbit BRIN, 2023. doi: 10.55981/brin.872.

[32] M. Xu *et al.*, "Embracing limited and imperfect training datasets: opportunities and challenges in plant disease recognition using deep learning," *Front. Plant Sci.*, vol. 14, 2023, doi: 10.3389/fpls.2023.1225409.

[33] M. Aqil *et al.*, "Penerapan Black Box Testing untuk Evaluasi Fungsionalitas Website Maggoplast," *Jurnal Mahasiswa Teknik Informatika (JATI)*, vol. 9, no. 1, pp. 294–306, Dec. 2025, doi: https://doi.org/10.36040/jati.v9i1.12177.