Journal of Computer and Communications, 2025, 13(8), 121-137
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Identification of Cherry Powdery Mildew Using Deep Convolutional Neural Network

Qiang Jiang[1,2], Xu Ming[1*], Duyi He[1], Shaojie Guo[1], Tao Zuo[3]

[1]School of Electronic Information and Artificial Intelligence, Leshan Normal University, Leshan, China
[2]Sichuan Prefabricated Cabin Power Equipment Engineering Technology Research Center, Leshan, China
[3]Leshan Yi Lade Power Grid Automation Co., Ltd., Leshan, China
Email: *mingxu@lsnu.edu.cn

How to cite this paper: Jiang, Q., Ming, X., He, D.Y., Guo, S.J. and Zuo, T. (2025) Identification of Cherry Powdery Mildew Using Deep Convolutional Neural Network. *Journal of Computer and Communications*, **13**, 121-137.
https://doi.org/10.4236/jcc.2025.138006

Received: June 30, 2025
Accepted: August 9, 2025
Published: August 12, 2025

Copyright © 2025 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Agriculture is essential for humanity's survival, and productivity is crucial in agriculture. Owing to its multiple nutrients, cherry has become an important fruit for daily consumption; however, crop productivity is mainly reduced by powdery mildew. Hence, recognizing this disease is vital to farmers. We developed a deep convolutional neural network (DCNN)-based method to assess cherry tree health using leaf images. We constructed the DCNN model on Keras with TensorFlow, preprocessing the original images for convenience. We used a visualization method to analyze intermediate model layers, comparing features of healthy and diseased leaves. The model achieved 99.2% accuracy after 10 training epochs. Misclassification occurred when leaf shadow edges were close to actual leaf edges, leading the model to mistake actual edges as features indicating disease, offering new insights for distinguishing the disease. Simulation results demonstrate that identifying plant disease via a DCNN-based protocol is suitable and adaptable to other plants, providing high practical value.

Keywords

Disease Identification, Deep Learning, Convolutional Neural Network, Recognition

1. Introduction

Although most countries consider yearly economic growth, as indicated by measures such as the gross domestic product, food is unarguably the most important resource for the survival of humans and animals, especially in countries with large populations such as China and India [1] [2]. As a nation develops, more categories

of food, such as grains, rice, vegetables, fruits, and meat, are required daily. Thus, food productivity and quality are crucial. However, diseases are a key cause of low productivity and reduced quality. Food security is low in resource-poor countries [3], which directly affects human health and survival. Cherry is a fruit that is common in daily consumption. It contains many nutrients such as vitamins, minerals, and energy. It also contains important nutrients that can supply dietary elements lacking in rice and grains [4]. Hence, the diagnosis of diseases in cherry trees is vital. Unfortunately, few studies on this topic have been published.

Rapidly recognizing the health of a cherry tree and classifying disease with high accuracy is a goal of many researchers. Some plant disease symptoms first appear on the leaves; thus, plant leaves can be studied to determine diseases [5]. Molecular techniques for disease identification are the foundation of this work. Over the decades, many plant researchers have constructed plant disease datasets. However, this method is only suitable for experts and small quantities of crops [6]. Fortunately, image-based techniques have been developed, and they can be combined with machine learning to effectively identify powdery mildew on plants. Machine learning, which consists of over 10 algorithms, is actively used in crop disease identification [7]. Using machine learning to identify plant pests and diseases is an efficient and precise method. By collecting images of plant leaves, stems, fruits, and other parts and combining environmental data such as temperature, humidity, and soil conditions to label the images, the types and severity of pests and diseases can be identified, assisting farmers and agricultural experts in quickly diagnosing the health status of plants and taking corresponding prevention and control measures [8] [9].

For example, Liu *et al.* proposed an Internet-of-Things-based method to directly perceive environmental conditions in crop fields [10]. This helps machine learning methods accurately predict the occurrence of plant diseases, with an accuracy as high as 91%. Kumar *et al.* used soil sensors and satellites to collect real-time data for in-depth analysis and a machine learning model to detect, evaluate, and predict plant diseases, with an average prediction accuracy above 98% [11]. To reduce the damage of white scale disease to date palm trees, an algorithm based on feature extraction and machine learning was proposed using support vector machine (SVM), *K*-nearest neighbors, and ensemble learning to analyze date palm tree leaf images, reaching an identification accuracy of 98.29% [12].

In practical applications, traditional machine learning is suitable for scenarios with limited resources or small data volumes, while convolutional neural networks (CNNs) are more suitable for large-scale high-precision plant pest and disease identification tasks and perform well in processing high-dimensional data, achieving superior recognition accuracy [13] [14]. Khattak *et al.* used a CNN to automatically detect citrus leaf diseases, with an accuracy of 94.55% [15]. Shruthi *et al.* proposed a CNN called TomSevNet for detecting tomato diseases and their severity, with a test accuracy of 96.91% [16]. In the past few years, deep learning

has developed at an astonishing speed and become a popular research topic. Moreover, the sudden outbreak of the coronavirus disease, COVID-19, has made food security even more important. Therefore, many studies have utilized deep learning techniques to identify plant diseases to maintain the quality of agricultural products and increase yields. Studies have proposed plant disease detection models based on optimized CNNs [17]-[20]. A hybrid deep learning model combining CNN and optimized recurrent neural network was developed for precise segmentation and disease detection of rice spikelet sterility, achieving an accuracy of 96.34% [17]. The EfficientNetV2 model was demonstrated to be capable of detecting anthracnose and leaf spot diseases in cardamom plants, as well as black rot, esca, and isariopsis leaf spot in grapes, with a detection accuracy up to 98.26% [18], outperforming other CNN and EfficientNet models. In the identification and detection of tea plant diseases and pests, transfer learning and the freezing core strategy were employed, resulting in a detection accuracy of 98.231% [19]. For detection and classification of cherry maturity and disease status, a MobileNet_v2 with SVM was adopted, achieving an identification accuracy of 98.3% [20]. The PlantDet model was proposed to enhance the robustness of disease detection in betel nut and rice leaves [21]. This model integrated multiple powerful models, achieving a detection accuracy of 98.53% and demonstrating the effectiveness of deep learning for plant disease prediction. The main drawback of deep learning is its high computational demand. Additionally, accuracy must be optimized; thus, the performance and computational load must be balanced [22].

In this study, we trained a deep CNN (DCNN) according to a dataset, optimized the hyperparameters, and classified the pixels of leaf images using the DCNN. This model performed well, and its computation was smaller than that of other methods, requiring only 72 s per epoch and 10 epochs to train. We analyzed the reasons for misclassifications of some leaves and present our conclusions.

In brief, our major contributions are as follows:

1) We built an improved DCNN model for cherry powdery mildew leaf diagnosis and classification and obtained a set of high-performing hyperparameters for our task.

2) By conducting standardized preprocessing on the dataset (including image size normalization and key information retention algorithms), we achieved an optimal balance between the model training effect and computational efficiency, and proposed a solution that strikes a balance between recognition performance and computational consumption.

3) We discovered that when the leaf shadow edge is close to the actual leaf edge, the DCNN model mistook the actual edge for a feature of disease during image processing.

The remainder of this paper is organized as follows. Section 2 presents the CNN architectures. Section 3 describes the strategy for cherry disease identification, including the model design and recognition algorithm. In Section 4, we present our experiments, including dataset preparation and processing as well as the simula-

tion results. The discussion is presented in Section 5, and conclusions are drawn in Section 6.

2. CNN Architectures

2.1. CNNs

Artificial neural networks were inspired by the seemingly inconceivable learning capability of the human brain. The human brain consists of billions of neurons interconnected by synapses, and their relationship changes as learning proceeds. Therefore, similar to a human brain, artificial neural networks can learn any task in principle. To improve learning capabilities, in 1998, LeCun *et al.* proposed the CNN architecture focusing on recognition, which is especially useful in image identification [23].

CNNs use image input directly and use the image pixels to automatically obtain features by convolving the pixels with filters. Multiple layers of convolutions can be calculated, which are followed by several fully connected layers. A set of weights and biases can be obtained through learning, and after the model has been trained, it can perform classification. In general, a CNN used in supervised learning contains an input layer, convolutional layer, pooling layer, flattening layer, and softmax layer, as illustrated in **Figure 1**. CNNs are applied in many areas such as handwriting recognition, traffic flow, automatic driving, face identification, and plant disease diagnosis.

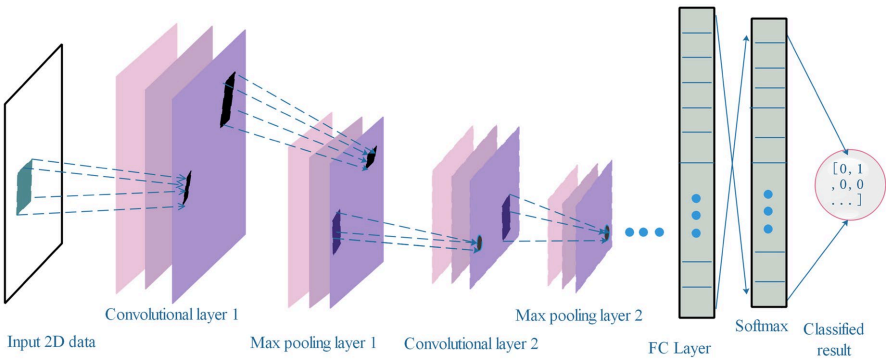

Figure 1. Architecture of DCNN.

2.2. Design of CNN Model

In this study, we designed the architecture of our DCNN based on our goal. The model consists of three dense layers, one max-pooling layer, two dropout layers, and one flattening layer. Furthermore, the dense layers contain two convolutional layers and one softmax output layer. The architecture of our DCNN is specified in **Table 1**. The total number of parameters is 420,738, but if we use an image 256 × 256 pixels, the total number of parameters is 30,938,690.

The convolutional layer computes the output of each tensor. Each computation is a dot product between the weights and a small region input covered by a kernel window of 5 × 5 pixels. Moreover, each computation extracts a feature map from

the input image. When the window finishes sliding over the image, a map of the main features is obtained, and the tensor output dimensionality is reduced. We design the pooling layer to subsample the features. The pooling kernel size is (4 × 4), which also reduces the dimensionality of the tensor output and can decrease the probability of overfitting. We introduce two dropout layers to avoid overfitting and improve the training speed. The final CNN model is the result of many experimental trials.

Table 1. Architecture of proposed DCNN.

Layer (type)	Tensor output (shape)	No. params.
Dense_1 (conv2D)	(None, 60, 60, 32)	832
Maxpooling2D_1 (max pooling)	(None, 57, 57, 32)	0
Dropout_1 (dropout)	(None, 57, 57, 32)	0
Dense_2 (conv2D)	(None, 52, 52, 64)	73,792
Dropout_2 (dropout)	(None, 52, 52, 64)	0
Flatten_1 (flattening)	(None, 173,056)	0
Dense_3 (softmax)	(None, 2)	346,114
Total no. parameters	420,738	

3. Cherry Disease Recognition Strategy

3.1. Design of Identification Architecture

We used Python 3 as our design environment platform and the Keras deep learning application programming interface. Therefore, we must convert the raw images into the data type required by the dataset. Hence, we must first preprocess the data and then design the recognition architecture, as shown in **Figure 2**. This architecture contains two main parts, data preprocessing and the DCNN model, in which we embed the identification algorithm. Data preprocessing includes reducing the number of pixels, creating the label set, and reshaping and creating the training and test sets.

3.2. Recognition Algorithm

Artificial neural networks can learn by adjusting the weights and bias to minimize the error between the true and predicted labels. In general, there are two common methods for describing errors, either as a loss function or as a cross-entropy function. Moreover, there are several methods (called optimizers) for finding the best weights, including stochastic gradient descent (SGD), RMSprop, and Adam. Finally, using an index to evaluate our model performance, we design an accuracy function.

1) *Loss function*

The loss function is an important part of machine learning. Treating it as the

standard of model learning, we use the loss function to measure the classification performance. It measures the error between the actual and predicted labels of an object. Many statistical measures can be used as a loss function, such as the mean squared error, mean absolute error, mean absolute percentage error, and mean squared logarithmic error. Cross-entropy, used in binary classification, allows to measure two probability distributions. Hence, we use the cross-entropy loss to increase the predicted probability as follows:

$$E(w) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \bar{y}_n + (1-y_n)\log(1-\bar{y}_n)\right] \tag{1}$$

where $\bar{y} = \mathrm{ReLU}(w \cdot x + b)$, w is the weight vector of each x, b is the vector of the bias, y_n is the actual label, and $\bar{y}_n$ denotes the predicted value.

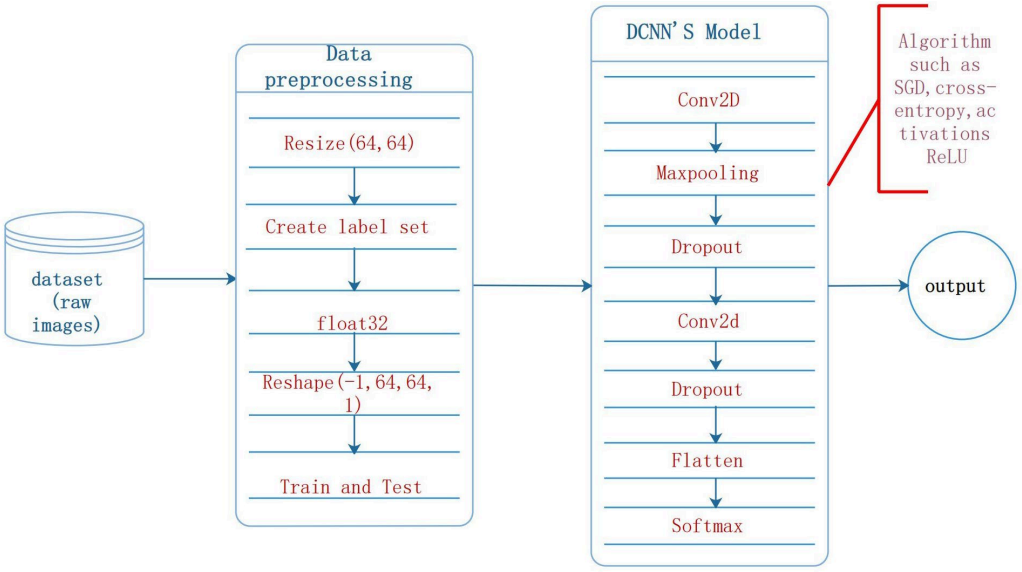

Figure 2. Architecture of recognition method for diseased cherry leaves.

2) *SGD*

SGD is a fundamental optimization method that stochastically approximates gradient descent through iterative parameter updates, capable of achieving strong performance in specific tasks despite its simplicity. The DCNN uses SGD to change w to minimize the following objective function:

$$w_i = w_{i-1} - \eta \frac{\partial E(w)}{\partial w} \tag{2}$$

where η is the learning ratio, which is an important parameter. A smaller η easily improves the performance but increases the computation burden. If η is large, the performance fluctuates. Hence, the best value per case must be determine experimentally. In addition, w_i is the current weight, and w_{i-1} is the previous weight, which is the weight updated using the SGD algorithm and the one that minimizes $E(w)$.

3) *Softmax*

In the DCNN, the output of the last layer is the classification result, and hence in this layer, dense output numbers are based on the number of classes. We determine them according to the softmax function, which takes the maximum value as the classification result as follows:

$$\text{soft max}\left(y = c_i\right) = \frac{e^{x \cdot w + b_i}}{\sum_{j=1}^{N} e^{x \cdot w + b_j}} \tag{3}$$

where N is the number of classes, y is the learning output, c_i is the i^{th} class, w is the set of weights, and b_i is the i^{th} bias.

To evaluate our model, we tested the performance using the following accuracy calculation:

$$\text{Score}\left(c_i\right) = \frac{P_T}{P_T + P_R} \tag{4}$$

where $\text{Score}\left(c_i\right)$ denotes the accuracy of the i^{th} class, and P_T and P_R are the numbers of correctly classified and misclassified images, respectively.

4. Experiments

4.1. Dataset and Preprocessing

We used the Python 3 platform, and the model was implemented using Keras, with TensorFlow being the backend. Our raw cherry leaf dataset contains 1906 leaf images, which includes 854 healthy leaf images and 1052 diseased leaf images. Each image is 256 × 256 pixels. We show four 256 × 256-pixel images from the healthy and diseased leaf datasets in **Figure 3**.

Preprocessing raw data is necessary and involves many tasks, as shown in **Figure 2**. Before feeding an image into a model, we must substantially reduce its pixel size such that it drastically decreases the computation cost. In addition, we must create the image label set. Next, we convert its type into float32 and reshape the images into size (−1, *x*, *y*, 1). Therefore, we must convert each image of the training and test sets into $x \times y \times 1$, where x and y are the image width and height, respectively. Here, we convert images into 64 × 64-pixel images. The number of pixels in each image reduces from 65,535 to 4,096, a decrease be a factor of 16. Example 64 × 64-pixels images including their features are shown in **Figure 4**. Then, we create the label set of the dataset for supervised learning. First, we reshape the image size to (−1, 64, 64, 1) and divide the dataset into training and test sets in an 80% - 20% split, where all the sample images are randomly selected. Moreover, no cross-samples exist between the training and test sets. Therefore, the dimensions of the training and test sets are (1, 524, 64, 64, 1) and (382, 64, 64, 1), respectively.

Comparing the raw images in **Figure 3** and **Figure 4**, although the number of pixels is reduced, **Figure 4(a)** and **Figure 4(c)** remain clear after conversion to float32 and reshaping into size (−1, 64, 64, 1).

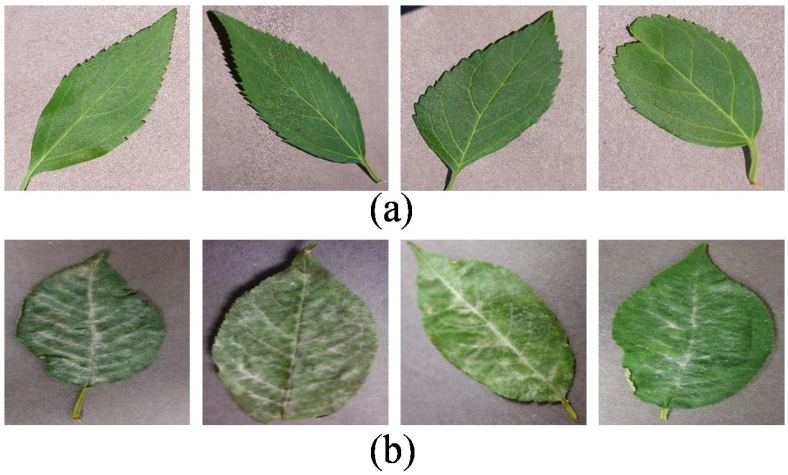

Figure 3. Raw 256 × 256-pixel images of cherry leaves in our dataset. (a) Healthy leaves; (b) leaves with powdery mildew.

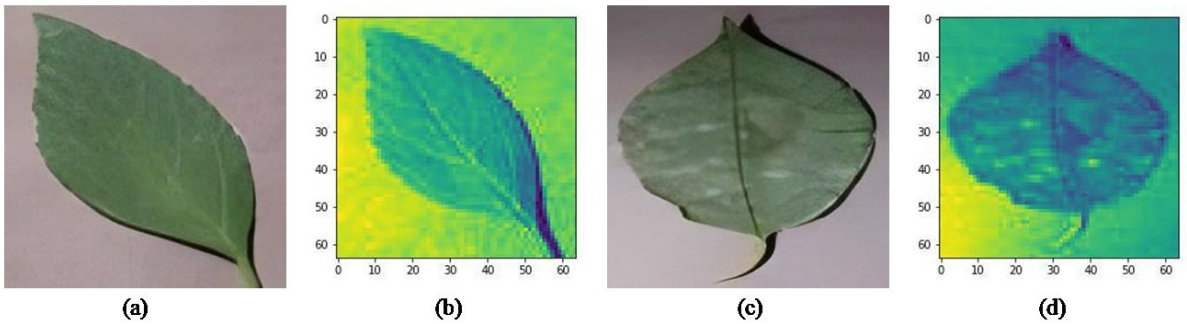

Figure 4. Converted 64 × 64-pixel images of cherry leaves in our dataset. (a), (b) Healthy leaves. (c) (d) Leaves with powdery mildew disease. (a) and (c) are reduced to 64 × 64 pixels. (b) and (d) are the final input images, which are reshaped and transformed into float32 images with dimension (−1, 64, 64, 1).

4.2. DCNN Model Hyperparameters

In deep learning, hyperparameters play a key role, and the model performance substantially differs under diverse hyperparameters for the same model, causing problems ranging from overfitting to nonconvergence. In our experiments, we used the hyperparameters listed in Table 2.

Table 2. Hyperparameters of DCNN layers.

Layer	Parameter	Values/specifications
Dense 1 (conv2D)	Filter Activation	Kernel size = (5, 5), rides = (1,1) ReLU
Max_pooling2d 1 (max pooling)	Pool	pool size = (4,4), strides = (1,1)
Dropout 1 (dropout)	Dropout ratio	0.5
Dense 2 (conv2D)	Filter Activation	Kernel size = (6,6), strides = (1,1) ReLU

Continued

Dropout_2 (dropout)	Dropout ratio	0.5
Flatten_1 (flattening)	None	None
Dense_3 (softmax)	None	None
Batch size	16	
No. epochs	10	
Learning ratio	0.005	

4.3. Results

The results from our experiments are shown in Figure 5 as loss and accuracy curves for both training and testing. The visual output from the intermediate layers 1 - 5 are shown in Figures 6-9.

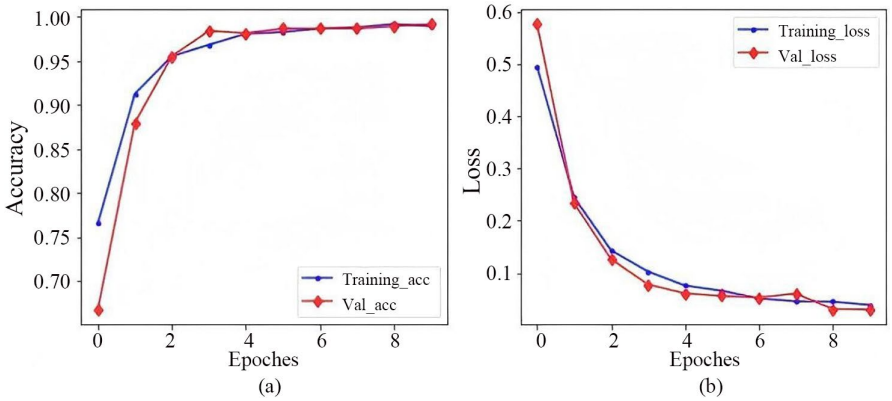

Figure 5. Results for training and testing regarding; (a) accuracy and (b) loss.

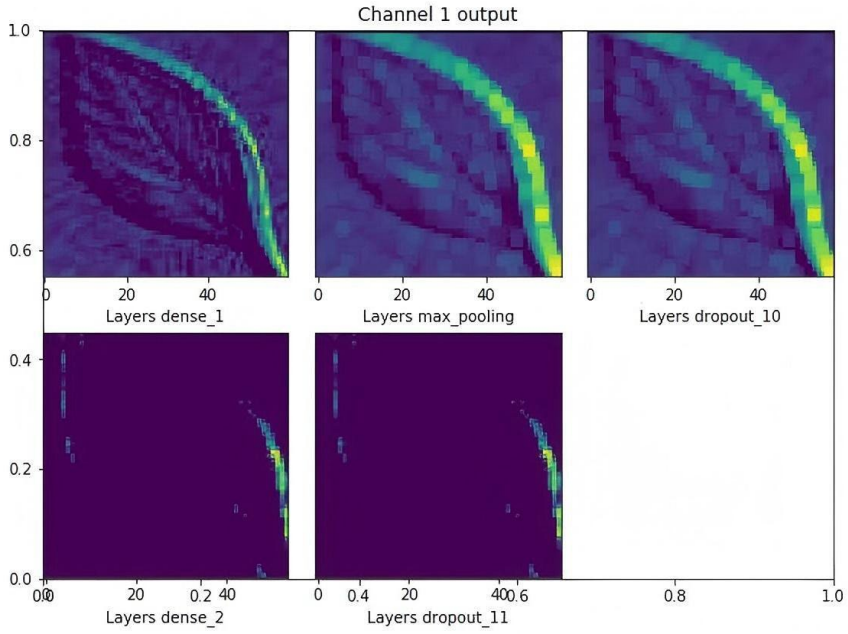

Figure 6. Visual output of channel 1 for healthy leaf.

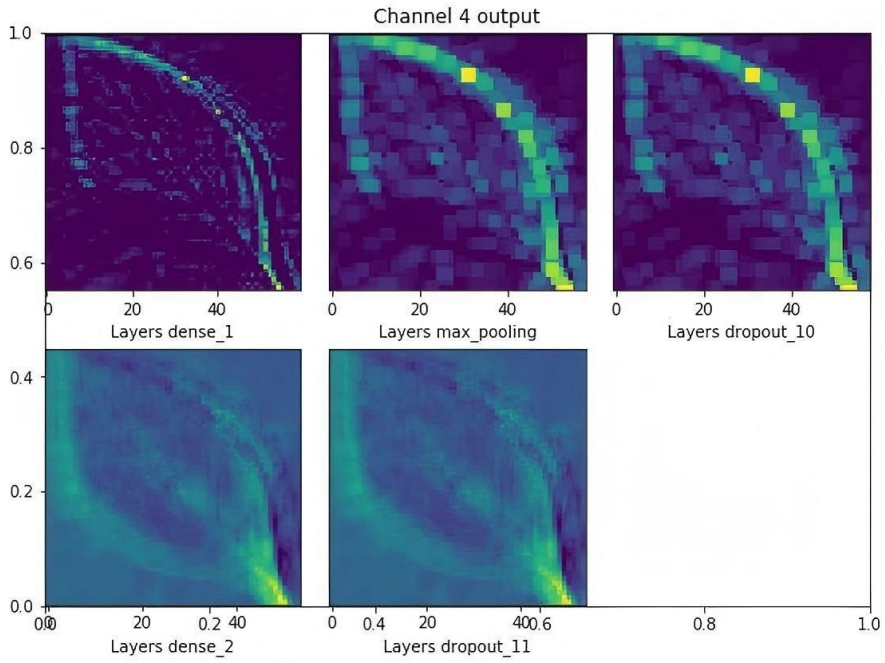

Figure 7. Visual output of channel 4 for healthy leaf.

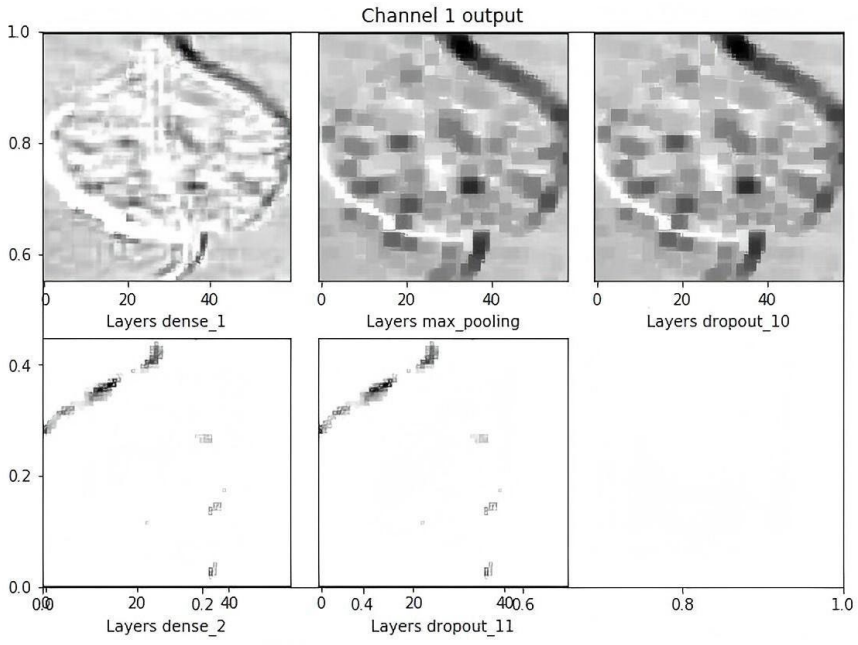

Figure 8. Visual output of channel 1 for diseased leaf.

Figure 5(a) shows the change in accuracy. During both training (blue curve) and testing (red curve), the accuracy continues to increase, except for a fluctuation at epoch 3 on the red curve. Figure 5(b) shows the change in cross-entropy, which indicates that over the whole course of both training (blue curve) and testing (red curve), the curve decreases stably, except for a fluctuation at epoch 7 on the red curve. Table 3 lists the classification results, indicating an accuracy of 99.2%, pre-

cision of both classes of 99%, and recall of class 1 (diseased) of 100%, indicating that all samples of class 1 diseased leaves are recognized correctly. Therefore, the training and testing results show that the model performs well.

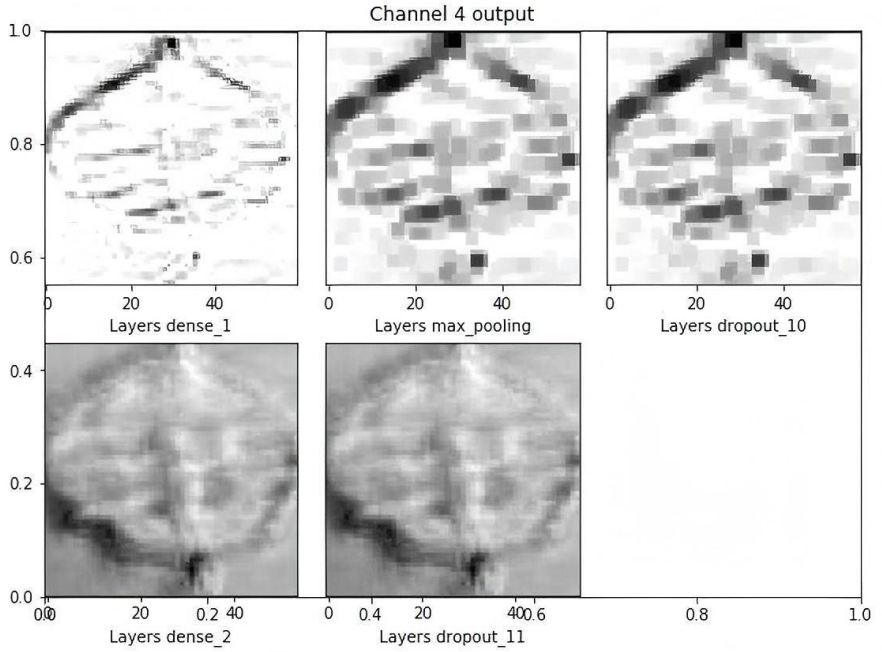

Figure 9. Visual output of channel 4 for diseased leaf.

Table 3. Classification result for diseased cherry leaves.

Class	Precision	Recall	F1-score	Support
Class 0	0.99	0.99	0.99	152
Class 1	0.99	1	0.99	230
Avg./Total	0.99	0.99	0.99	382
Test loss:			0.03	
Test accuracy:			0.992	

Figures 6-9 show output images from the intermediate layers. Figure 6 and Figure 7 show the outputs of healthy leaves for each layer from channels 1 and 4, respectively. Figure 8 and Figure 9 show the output for each layer of channels 1 and 4 for diseased leaves, respectively. Figure 6 shows that the extracted features are different between layers. The dense 1 and pooling layers extract many large-scale features. By contrast, the dense 2 layer extracts more detailed characteristics. These images also help understanding the dropout method, which not only prevents overfitting but also retains feature information by comparing the outputs of the dense 2 and dropout layers.

Comparing the outputs of channels 1 and 4, we can easily determine that channel 1 suitably represents the edge information of an object, and channel 4 includes

a large amount of feature information contained in the edge and inner regions. Hence, in deep learning, the channels of the tensor have different roles in feature detection.

Further comparison between **Figure 6** and **Figure 7** and **Figure 8** and **Figure 9** reveals that the characteristics of healthy leaves are mainly on the edge. However, the dominant information of diseased leaves is in the inner region of the object. **Figure 8** clearly shows that the pooling layer can extract crucial features from the object. In addition, **Figure 8** also shows that the DCNN model is efficient for our recognition goal. Thus, it is important to design the model architecture using several methods, which is a key advantage of DCNNs.

Table 4 lists the performance of our model, which includes accuracy, test loss, number of epochs, and training time per epoch. The accuracy is high, and training consumes only 72 s/epoch over 10 epochs. Regarding the other methods, Bhange *et al.* reported an accuracy of 82% using SVM classification, and Mousavi *et al.* proposed a classification method based on Gabor wavelet features and SVM, achieving an accuracy of 90.4%. Yang *et al.* [14] proposed an approach to identify rice disease using deep learning and achieved an accuracy of 95.48%. Ferentinos [23] used a deep learning model in 2018 to achieve 99.53% accuracy after training for 10,000 epochs over 5.5 days and testing over 67 epochs at 7,034 s each using a VGG. Obviously, our model provides a high performance and balance between accuracy and speed.

Table 4. Performance comparison of proposed and existing models. (Not all metrics were reported for all methods).

Model	Accuracy	Test loss	No. epochs	Training time (s/epoch)
Proposed DCNN model	99.21%	0.03	10	72
SVM classification	82.00%	–	–	–
GWF and SVM	90.40%	–	–	–
CNN model in [14]	95.48%	–	–	–
Model in [23]	99.53%	0.02	67	7,034

5. Discussion

DCNNs have shown excellent performance in many research domains, especially for processing images, achieving a classification efficiency of over 99% in a few published papers. In this section, we discuss objects misclassified by our model and analyze their characteristics to identify reasons that explain the misclassifications. This discussion will be valuable for further CNN development.

The three images shown in Figure 10 were classified incorrectly. **Figure 10(b)** and **Figure 10(c)** have the same shadow endpoint and real leaf endpoint. Hence, we analyze the images shown in **Figure 10(a)** and **Figure 10(c)** because they have a large shadow as a similar feature. However, we obtain different results. The first

image label is diseased, but our model prediction is healthy, and the third image class is healthy. However, our classifier provides prediction diseased. To explain the prediction of our model, we highlight some features in the leaf images shown in Figure 11. The marked areas are used to analyze their effect on the DCNN ability to recognize powdery mildew.

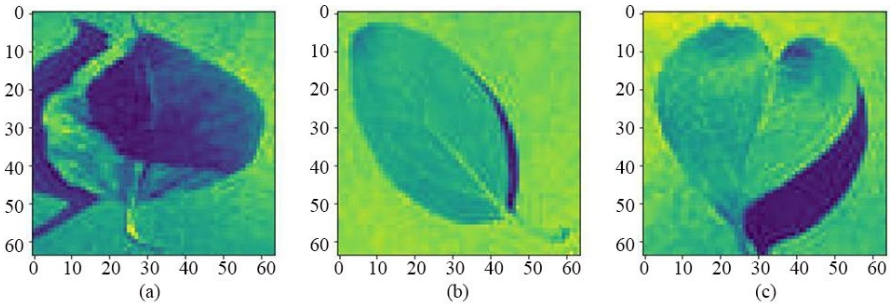

Figure 10. Misclassified leaves. (a) Diseased leaf classified as healthy and (b), (c) healthy leaves classified as diseased.

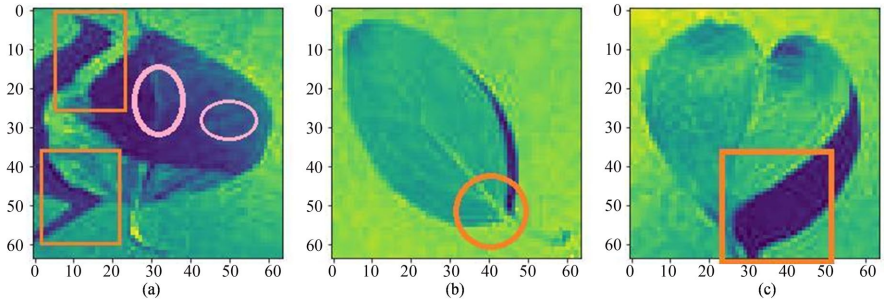

Figure 11. Annotated misclassified leaves. The features in (a), (b), and (c) are indicated by the red shapes, and the diseased regions of the image in (a) are indicated by pink ellipses.

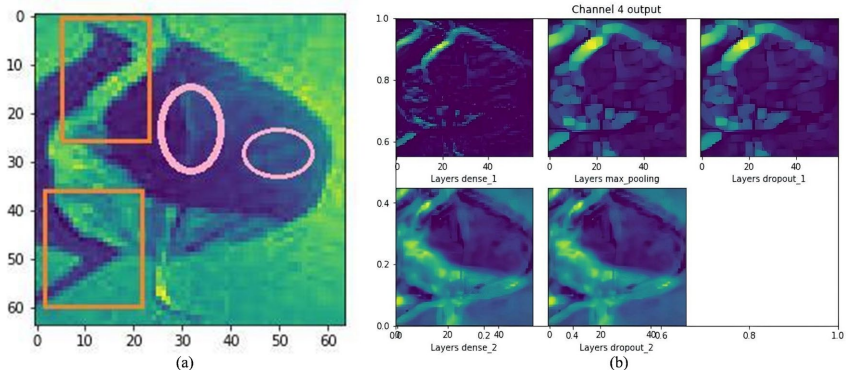

Figure 12. Channel 4 output image of misclassified leaf in Figure 10(a). (a) Raw 64 × 64-pixel leaf image and (b) outputs of intermediate layers.

We show details from the intermediate layer outputs in Figure 12. In Figure 12(a), a shadow is located along the leaf edge. In Figure 12(b), both edges are recognized. Diseased regions indicated by the pink ellipses are hidden in the shadow. Thus, the disease features are ignored, and the other shadows are incor-

porated into another leaf. Therefore, our model misclassifies the disease. Making a correct inference by human visualization is difficult because the diseased leaf regions are occluded.

Because the leaf shown in Figure 10(b) has the same features as that shown in Figure 10(c), we input that leaf into our model and show the outputs of the layers in Figure 13. The shadow of Figure 13(a) is recognized as a leaf edge, and the actual leaf edge is misclassified as a diseased region, causing misclassification. Comparing the raw images with those shown in Figure 12, we observe that the shadow terminal vertex and edges in Figure 12 are far from the true leaf edge. However, in Figure 13, the endpoint of the shadow edge crosses the actual leaf edge. Therefore, in Figure 13, the shadow edge is mistaken for a leaf edge, and the real edge is mistaken as a diseased region. To address this issue, we will consider two methods in future work. The first method is training the model further to use the shadow characteristics of the object. The second method is creating an image dataset that avoids sampling under strong light and maximizes leaf flattening before capturing the images.

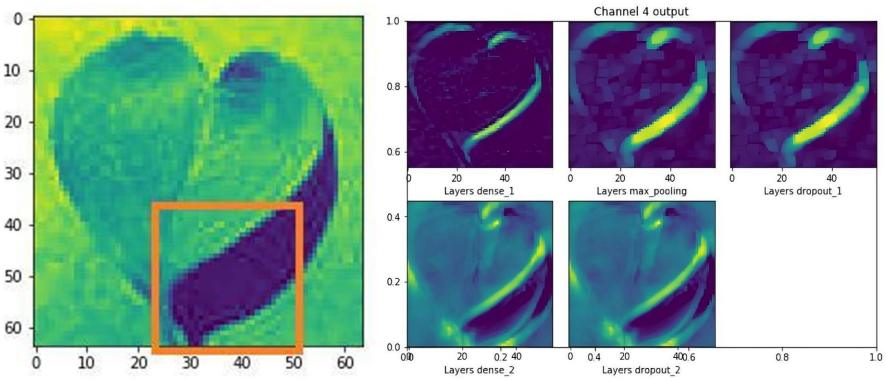

Figure 13. Channel 4 output image of misclassified leaf in Figure 10(b). (a) Raw 64 × 64-pixel leaf image and (b) outputs of intermediate layers.

6. Conclusions

We propose an approach based on a DCNN to recognize disease in cherry leaf images. In experiments, we obtained an accuracy of 99.2% after just 10 epochs. A comparison with other methods revealed that the proposed method has a high accuracy, but it also reduces the training time. A main drawback of deep learning is high computational cost. Hence, we devised a balance for the tradeoff between learning performance and computational consumption, which decreased the computation cost rapidly by reducing the size of the raw image to 64 × 64 pixels. By determining suitable hyperparameters, our method achieved high performance. Subsequently, we discussed the misclassification of our model for some instances. We interpret this case by considering that when the shadow edge is close to the actual leaf edge, the DCNN mistakes the edge for the diseased region. This hypothesis needs further evaluation in other cases, but provides a new research di-

rection. The methods used in this study have also been retrained and can be applied to other plants.

Acknowledgements

This work is supported in part by the Education Department of Sichuan Province Foundation of China (18ZB0273), in part by the key Laboratory of Sichuan Province (ZL2019004), and in part by the Central Guidance for Local Science and Technology Development Fund Projects (2024ZYD0268, 2024ZYD0311).

Code and Data Availability

The experiment dataset used in this research is publicly available on GitHub: https://github.com/spMohanty/PlantVillage-Dataset; we would like to express our gratitude for this resource.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Coletta, A., Bartolini, N., Maselli, G., Kehs, A., McCloskey, P. and Hughes, D.P. (2022) Optimal Deployment in Crowdsensing for Plant Disease Diagnosis in Developing Countries. *IEEE Internet of Things Journal*, **9**, 6359-6373. https://doi.org/10.1109/jiot.2020.3002332

[2] Nagpal, J., Goel, L. and Shekhawat, P.S. (2024) A Hybrid Deep Learning Neural Network for Early Plant Disease Diagnosis Using a Real-World Wheat-Barley Vision Dataset: Challenges and Solutions. *International Journal of Data Science and Analytics*, **18**, 1-15. https://doi.org/10.1007/s41060-024-00578-x

[3] Pierre Nyakuri, J., Nkundineza, C., Gatera, O. and Nkurikiyeyezu, K. (2024) State-of-the-Art Deep Learning Algorithms for Internet of Things-Based Detection of Crop Pests and Diseases: A Comprehensive Review. *IEEE Access*, **12**, 169824-169849. https://doi.org/10.1109/access.2024.3455244

[4] Mikiciuk, G., Rogowska, M. and Mikiciuk, M. (2015) The Effects of Foliar Nutrition with InCa Fertilizer on the Chemical Composition of Leaves and Fruits of Sweet Cherry. *Journal of Ecological Engineering*, **16**, 116-119.

[5] Moupojou, E., Tagne, A., Retraint, F., Tadonkemwa, A., Wilfried, D., Tapamo, H., *et al.* (2023) Fieldplant: A Dataset of Field Plant Images for Plant Disease Detection and Classification with Deep Learning. *IEEE Access*, **11**, 35398-35410. https://doi.org/10.1109/access.2023.3263042

[6] Hassan, S.M. and Maji, A.K. (2022) Plant Disease Identification Using a Novel Convolutional Neural Network. *IEEE Access*, **10**, 5390-5401. https://doi.org/10.1109/access.2022.3141371

[7] Bhargava, A., Shukla, A., Goswami, O.P., Alsharif, M.H., Uthansakul, P. and Uthansakul, M. (2024) Plant Leaf Disease Detection, Classification, and Diagnosis Using Computer Vision and Artificial Intelligence: A Review. *IEEE Access*, **12**, 37443-37469. https://doi.org/10.1109/access.2024.3373001

[8] Chouhan, S.S., Singh, U.P., Sharma, U. and Jain, S. (2024) Classification of Different Plant Species Using Deep Learning and Machine Learning Algorithms. *Wireless Per-*

sonal Communications, **136**, 2275-2298.
https://doi.org/10.1007/s11277-024-11374-y

[9] Shinde, N. and Ambhaikar, A. (2024) An Efficient Plant Disease Prediction Model Based on Machine Learning and Deep Learning Classifiers. *Evolutionary Intelligence*, **18**, 14-28. https://doi.org/10.1007/s12065-024-01000-y

[10] Liu, Z., Bashir, R.N., Iqbal, S., Shahid, M.M.A., Tausif, M. and Umer, Q. (2022) Internet of Things (IoT) and Machine Learning Model of Plant Disease Prediction-Blister Blight for Tea Plant. *IEEE Access*, **10**, 44934-44944.
https://doi.org/10.1109/access.2022.3169147

[11] Kumar, M., Kumar, A. and Palaparthy, V.S. (2021) Soil Sensors-Based Prediction System for Plant Diseases Using Exploratory Data Analysis and Machine Learning. *IEEE Sensors Journal*, **21**, 17455-17468. https://doi.org/10.1109/jsen.2020.3046295

[12] Hessane, A., Mesbahi, M. and El Fkihi, S. (2023) A Machine Learning Based Framework for a Stage-Wise Classification of Date Palm White Scale Disease. *Big Data Mining and Analytics*, **6**, 263-272.

[13] Joseph, D.S., Pawar, P.M. and Pramanik, R. (2022) Intelligent Plant Disease Diagnosis Using Convolutional Neural Network: A Review. *Multimedia Tools and Applications*, **82**, 21415-21481. https://doi.org/10.1007/s11042-022-14004-6

[14] Yang, B., Li, M., Li, F., Wang, Y., Liang, Q., Zhao, R., *et al.* (2024) A Novel Plant Type, Leaf Disease and Severity Identification Framework Using CNN and Transformer with Multi-Label Method. *Scientific Reports*, **14**, Article No. 11664.
https://doi.org/10.1038/s41598-024-62452-x

[15] Khattak, A., Asghar, M.U., Batool, U., Asghar, M.Z., Ullah, H., Al-Rakhami, M., *et al.* (2021) Automatic Detection of Citrus Fruit and Leaves Diseases Using Deep Neural Network Model. *IEEE Access*, **9**, 112942-112954.
https://doi.org/10.1109/access.2021.3096895

[16] Shruthi, U. and Nagaveni, V. (2023) TomSevNet: A Hybrid CNN Model for Accurate Tomato Disease Identification with Severity Level Assessment. *Neural Computing and Applications*, **36**, 5165-5181. https://doi.org/10.1007/s00521-023-09351-w

[17] Patel, B. (2024) Rice Spikelet's Disease Detection Using Hybrid Optimization Model and Optimized CNN. *Soft Computing*, **28**, 12787-12806.
https://doi.org/10.1007/s00500-024-10367-0

[18] Sunil, C.K., Jaidhar, C.D. and Patil, N. (2022) Cardamom Plant Disease Detection Approach Using EfficientNetV2. *IEEE Access*, **10**, 789-804.
https://doi.org/10.1109/access.2021.3138920

[19] Li, Z., Sun, J., Shen, Y., Yang, Y., Wang, X., Wang, X., *et al.* (2024) Deep Migration Learning-Based Recognition of Diseases and Insect Pests in Yunnan Tea under Complex Environments. *Plant Methods*, **20**, Article No. 101.
https://doi.org/10.1186/s13007-024-01219-x

[20] Yang, J. and Wang, G. (2023) Identifying Cherry Maturity and Disease Using Different Fusions of Deep Features and Classifiers. *Journal of Food Measurement and Characterization*, **17**, 5794-5805. https://doi.org/10.1007/s11694-023-02091-4

[21] Shovon, M.S.H., Mozumder, S.J., Pal, O.K., Mridha, M.F., Asai, N. and Shin, J. (2023) Plantdet: A Robust Multi-Model Ensemble Method Based on Deep Learning for Plant Disease Detection. *IEEE Access*, **11**, 34846-34859.
https://doi.org/10.1109/access.2023.3264835

[22] Dang, M., Wang, H., Li, Y., Nguyen, T., Tightiz, L., Xuan-Mung, N., *et al.* (2024) Computer Vision for Plant Disease Recognition: A Comprehensive Review. *The Bo-*

tanical Review, **90**, 251-311. https://doi.org/10.1007/s12229-024-09299-z

[23] Ferentinos, K.P. (2018) Deep Learning Models for Plant Disease Detection and Diagnosis. *Computers and Electronics in Agriculture*, **145**, 311-318. https://doi.org/10.1016/j.compag.2018.01.009