International Journal of Environmental Science and Green Technology

https://journals.ayu.edu.kz

Comparative Evaluation of ResNet18, ResNet34, and ResNet50 Models in Classifying Plant Leaf Diseases

Arypzhan Aben[1*], Dinmukhammed Gabdullauly[2]

[1]*Department of Computer Engineering, Faculty of Engineering. Khoja Akhmet Yassawi International Kazakh-Turkish University, Turkistan, Kazakhstan*
[2]*Master's student, Department of Computer Engineering, Faculty of Engineering. Khoja Akhmet Yassawi International Kazakh-Turkish University, Turkistan, Kazakhstan*

*Corresponding Author e-mail: arypzhan.aben@gmail.com

Keywords

Plant disease classification

Leaf images

Deep learning

Transfer learning

ResNet

Abstract

This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines. Abstract: This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines. Abstract: This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines. Abstract: This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines. Abstract: This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines. Abstract: This is the section where the purpose, the method(s) used, the findings obtained, the results and evaluations are briefly stated. It should not exceed 350 words and 2600 characters including spaces. It should be written in 9pt with single space between the lines.

Cite

Aben, A., & Gabdullauly, D. (2026). Comparative Evaluation of ResNet18, ResNet34, and ResNet50 Models in Classifying Plant Leaf Diseases. *International Journal of Environmental Science and Green Technology, 1*(1), 30-45. doi: 10.5281/zenodo.23485351

Article Process

Submission Date: 17.01.2026; Revision Date: 10.02.2026; Accepted Date: 02.03.2026; Published Date: 30.03.2026.

1. INTRODUCTION

It is necessary to diagnose plant diseases at early stages so that the state of the crop can be evaluated and preventive measures considered. One of the available tools that can resolve this issue in a quick fashion is automated analysis of leaf images. Diagnosis by type of lesion may not be easy - similar manifestations of lesion types, different degrees of lesion visibility, variation of image quality all play a role in making diagnosis more challenging. However, Ferentinos (2016) exemplified how convolutional neural networks can discriminate between healthy and diseased leaf images, which underlies the complexities of visual based plant disease diagnosis, with research in such methods serving as a methodological background of autonomous tools which can evaluate the state of plants through image.

In traditional leaf image classification approaches, features like colour, texture, shape are computed first and then fed into a classifier. The result varies according to how descriptive the features selected are of the diseases under survey. Deep learning techniques can directly learn features from the images; Mohanty et al (2016) showed how effective this approach could be by classifying 38 categories in the Plant Village dataset, on the internal test, their best performing model showed 99.35% accuracy on the Plants Village data and dropped to 31% on external images captured with varying conditions. This suggests an explanation why having a high internal test score does not necessarily ensure same results on new conditions under which pictures are taken.

When we do not have adequate labels in the new data but still want to achieve impressive results, the method of adapting models that have already been trained on one field to a different task called transfer learning can be used. When training models, some parameters, such as the initial learning rate, have to be set while training. When using the transfer-learning method, we can initialize and fine-tune such parameters from large image dataset using our own ones. Research has looked at comparing pre-trained networks when detecting plant diseases. Tu et al. had conducted a comparison between fine-tuned deep learning models for classification of plant disease. Research of others can help in motivating selection of models for the problem. ResNet family of networks is one of the types of architecture used in image recognition that uses residual connections. He et al. (2016) had proposed residual learning to solve the degradation problem by making the deep networks easy to improve. In a residual block, the input is combined with transformed features and further passed through the network. ResNet18, ResNet34, ResNet50 work using the same principle but vary in terms of depth, architecture of the blocks, and so forth. Comparing them would allow us to evaluate how well different models of one architecture family perform for our classification task. Additionally, to differentiate the differences in ResNet models solely because of the number of layers, the other structural properties such as the depth/type of convolutional kernel used in each have been compared by the researchers.

The problem being solved here is how accurately ResNet can distinguish between healthy, powdery mildew infected, and rust infected leaves. In the literature, studies are shown using large data sets but do not draw any final conclusion on which model is better at a 3-class classification problem. This is why similar modelling conditions and data partition methods must be used along with data-preprocessing techniques as well as training duration. By doing so, results obtained can be studied which vary only because of different models and not due to other training conditions.

In this study, our primary goal is to take a look at the ResNet18, ResNet34 and ResNet50 models (pre-trained on the data from ImageNet) that are adapted to detect the categories "Healthy," "Powdery," and "Rust" in the Plant Disease Recognition Dataset and compare their results using one consistent evaluation protocol. We train them separately over separate training, validation and test sets. We consider the random augmentation of images in our training set only and the best version (selected on the basis of validation accuracy) of each of the three models is used for our calculations on the test set. For our comparison we will refer to the overall accuracy as well as the average of the different class results for precision, recall and F1-score which make sure that all the classes contribute equally to the result when estimating the performance of the classification.

The main contribution of this work is an experimental comparison of the three ResNets described before under the same experimental conditions for healthy leaves and leaves of two types of diseases. Which one models provides best classification quality under limited training budget is discussed along with question whether advantage in terms of accuracy remains when it comes to macroaveraged measures. Results are interpreted in terms of data and evaluation protocol used and it is underlined, that successful real world deployment of studied models needs to be validated using images collected under different conditions.

2. LITERATURE REVIEW

Plant diseases detection from pictures was previously dealt with using classical machine learning methods based on manually designed features. The researchers would manually define some features, for instance leaf surface colour or texture or shape of spots and then use classifiers like support vector machines or ensembles of trees (random forest) to classify them. While in some cases this approach might have worked depending on the chosen features and also on consistency in image-taking conditions (the accuracy has been shown to drastically drop when background and lighting change), with the development of convolutional neural networks (CNNs) manual feature engineering was no longer needed, the features could be learnt straight from the data (Ahmed & Yadav, 2023; Demilie, 2024). Various reviews confirm the robustness and greater generalization ability as compared to the classical approach (Shafik et al., 2023). While this is sometimes true, comparative reviews show that the CNN performance depends on the training set available, on the number of classes and on the protocol of evaluation (Demilie, 2024; Qadri et al., 2024), which is why the claim that "deep learning is always better" is not correct. One of the first experimental studies to train a deep CNN for 38 classes (14 crops and 26 diseases) from PlantVillage dataset, 54,306 images taken in controlled conditions and classify them with a 99.35% accuracy on a controlled test set, was performed by Mohanty et al. (2016) The authors showed that the accuracy drops drastically if the model is applied to pictures taken under various conditions that have been fetched from the internet, which implied that the model could not be generalized for conditions in the field. A similar research has been carried out by Ferentinos but using a much bigger dataset of 87,848 images, 25 crops and 58 classes and comparing different architectures (AlexNet, VGG, GoogLeNet, Overfeat) with 99.53% success rate (2018). In both cases high accuracies were shown using datasets that employed controlled acquisition system, but the robustness of the model when deployed in the field was not addressed. Then Pandian et al. (2022) combined existing data from open-source resources and made a new dataset with 58 classes and 16 plants of 147,500 images ; they came up with a special 14-layer CNN (14-DCNN) and to deal with the problems of unbalanced classes they performed image augmentation using basic transformations, Generative Adversarial Networks (DCGAN) and Neural Style Transfer (NST). These three studies differ in their specific aims, for example number of classes, but also crop types and dataset sizes, however they all highlight that deep learning models achieve high accuracies when run on controlled datasets. Most of the research on plants diseases uses transfer learning, an approach that allows the features pre-trained on large image datasets like

ImageNet to be transferred and reused for the task of detecting crop diseases on smaller datasets. Too et al. (2019) used VGG16, Inception-v4, ResNet with different depths (50, 101, 152 layers) and DenseNet121 and fine-tuned them for the 38 classes of the PlantVillage dataset, showing that DenseNet models continued improving as number of epochs increased without any clear signs of overfitting. The ResNet family was used too and performance of ResNet models was not only dependent on depth of the network, but also on training strategy that was used to develop the architecture. The foundations of ResNet architecture came from work of He et al. (2016) who showed that thanks to residual connections deep networks could be trained without the degradation problem of vanishing gradient. There is a difference between ResNet18, ResNet34 and ResNet50 in number of layers, but also in the block structure; ResNet18 and ResNet34 implementations use the standard residual block, but to increase efficiency ResNet50 and following architectures use the "bottleneck" residual block, this changes the number of parameters and computation complexity therefore it does not scale linearly with the depth of the network.

Table 1. *Structural differences between the ResNet18, ResNet34, and ResNet50 architectures described in the work of He et al. (2016).*

Model	Residual Block Type	Number of Blocks per Stage	Total Number of Parameters (Approx.)
ResNet18	Basic Block	[2, 2, 2, 2]	~11.7 million
ResNet34	Basic Block	[3, 4, 6, 3]	~21.8 million
ResNet50	Bottleneck Block	[3, 4, 6, 3]	~25.6 million

From this table, the reader can see that although ResNet34 and ResNet50 consist of the same number of "blocks," the change in block type led to a variation in the number of parameters and, consequently, the network's characteristics; therefore, we cannot draw the definitive, broad conclusion that a "deeper model is a better model."

Factors related to the dataset also significantly impact model performance. With datasets like PlantVillage-featuring controlled backgrounds and a single leaf per image-models can learn distinct disease features; however, such conditions do not exist in the field. To illustrate this issue, Moupojou (2023) et al. introduced the FieldPlant dataset, which comprises field images featuring multiple leaves, complex backgrounds, and varying lighting conditions; their model demonstrated a significant drop in accuracy compared to performance on PlantVillage. This finding quantifies the impact of factors such as background complexity and leaf occlusion resulting from various capture angles-both of which are frequently highlighted in review articles (Jafar et al., 2024; Shafay et al., 2025). It must be acknowledged that the PlantVillage, FieldPlant, and Plant Disease Recognition (Healthy, Powdery Mildew, Rust) datasets used in this experiment are distinct and independent datasets, characterized by different collection conditions and category structures; they cannot be merged.

Common shortcomings and recurring issues have been highlighted in recent review articles. Shoaib et al. (2023) noted that the "small, class-imbalanced datasets" used to evaluate deep learning models lead to overfitting errors. Demli (2024), in a critical review of various methods across existing studies, pointed out that the lack of a standardized testing procedure makes it difficult to compare different approaches. In their review, Shafiq et al. (2023) highlighted the presence of "overlap (data leakage) between training and testing sets" in many studies-an issue often overlooked, leading to inaccurate evaluations. Quadri et al. Regarding the issues identified, studies often "do not employ an external test set to evaluate their models; instead, they rely on internal datasets. Consequently, it is difficult to compare them with other research utilizing similar classification categories (e.g.,

tomato leaves, early blight)". Jafar et al. (2024) highlighted the problem of experimental reproducibility, noting that in many studies, "the source code and hyperparameters of the proposed models are not fully disclosed, making it virtually impossible to replicate the reported accuracy results" (2024). Each of these aspects-namely, the risk of overfitting on small datasets, the difficulty of comparing methods due to the lack of a standardized evaluation protocol, the need to prevent data leakage between training and testing sets, the requirement for external test set validation, and the feasibility of reproducing experiments has been addressed individually rather than as a unified set of recommendations, as their relative importance may vary depending on the specific study. While there is ample evidence of high accuracy for individual models in this field, it is difficult to find a study that compares ResNet architectures of varying depths against one another-specifically for the same three-class task (Healthy, Powdery Mildew, Rust) under identical conditions, including the same training/validation/test splits, preprocessing, and training protocols. Although studies such as have included ResNet models in their architectural comparisons, the proposed work differs by focusing specifically on comparing ResNet18, ResNet34, and ResNet50 on a single problem while strictly controlling other training factors; in contrast, the earlier study utilized a different class structure and dataset (2019). We view the study presented here as an applied comparative evaluation project: it does not propose a new architecture, nor does it fully resolve the challenge of generalizing to external environments using a single dataset; instead, it aims to empirically demonstrate how network depth-beyond merely the number of parameters influences performance in a specific, constrained problem setting.

3. MATERIALS AND METHODS

Dataset and problem formulation

The study utilizes the "Plant Disease Recognition Dataset," which comprises three classes of plant leaves: Healthy, Powdery (powdery mildew), and Rust. The dataset description indicates a total of 1,530 images. In this work, the dataset is sourced from the Kaggle platform with a pre-existing directory structure, featuring pre-split Train, Validation, and Test folders; while literature often cites a 1200/170/160 ratio (approximately 80%/11%/10%) for this dataset, the actual file count in this specific implementation must be verified by directly counting the contents of each folder: [clarification required: exact number of images per class in the Train, Validation, and Test folders].

This data split is not performed randomly or via stratification by the research code itselfit is provided as a pre-organized directory structure within the dataset; consequently, the process is not described as "random splitting" or "stratified splitting" in this work. Similarly, no specific verification was conducted to ensure the absence of duplicate images across the train, validation, and test folders, and no such verification is claimed.

The task is formally defined as a three-class classification problem:

$$D = \{(x_i, y_i)\}_{i=1}^{N}, \quad x_i \in \mathbb{R}^{3x224x224}, \quad y_i \in \{0, 1, 2\} \tag{1}$$

Here, x_i is an RGB input image resized to 224×224, and y_i is its class label. The class codes are designated as follows: 0 – Healthy, 1 – Powdery, 2 – Rust. The training set is used for gradient-based updates of model parameters; the validation set is used after each epoch to comparatively evaluate model versions and select the best checkpoint; and the test set is used solely to independently assess the final quality of the selected model after the training process has fully concluded.

Image Preprocessing

All images are first resized to 224×224 and converted into tensors using `torchvision.transforms.ToTensor()`, which transforms 8-bit values in the [0, 255] range to the [0, 1] interval. Subsequently, channel-wise normalization is applied based on ImageNet statistics:

$$\tilde{x}_{i,h,v} = (x_{i,h,v} / 255 - \mu) / \sigma \quad (2)$$

Here, $x_{i,h,v}$ represents the pixel value in the original 0–255 range, while $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$ are the mean and standard deviation corresponding to the RGB channels. In practice, the division by 255 is performed within the `ToTensor()` transformation rather than as a separate step; consequently, the image is not re-scaled during the normalization stageinstead, only the final two operations from formula (2) are executed: subtracting the mean and dividing by the standard deviation.

For the training set, the `RandomHorizontalFlip()` transformation is applied with a probability of 0.5, as the class identity of leaf images remains unchanged by horizontal flipping; this helps mitigate model overfitting on the limited training data. No random augmentation is applied to the validation and test setsonly deterministic resizing, tensor conversion, and normalization are performed for these sets, ensuring that evaluation results do not depend on random transformations.

Model Architectures

The study employs three deep convolutional architecturesResNet18, ResNet34, and ResNet50based on the residual learning principle introduced by He et al. (2016). In that work, the authors demonstrated that increasing network depth could lead to higher training error (the degradation problem) and proposed a residual block to address it. The general expression for the block is:

$$h_{l+1} = \mathrm{ReLU}(F(h_l W_l) + S_l(h_l)) \quad (3)$$

where h_l is the input to the l-th block, $F(h_l W_l)$ is the residual transformation learned via weights W_l, and S_l is the shortcut connection. When input and output dimensions match, $S_l(h_l) = h_l$ acts as an identity mapping; if the number of channels or spatial dimensions changes, S_l becomes a projection transformation implemented via 1×1 convolution and batch normalization, aligning the dimensions of the two branches prior to the addition operation.

Table 2. *Architectures used and the internal structure of the model's residual block*

Model	Block type	Number of blocks in the four stages	Number of features after global average pooling	New classifier
ResNet18	BasicBlock	[2, 2, 2, 2]	512	Linear(512, 3)
ResNet34	BasicBlock	[3, 4, 6, 3]	512	Linear(512, 3)
ResNet50	Bottleneck	[3, 4, 6, 3]	2048	Linear(2048, 3)

ResNet 18 and Resnet34 used BasicBlock 3×3which has 2 consecutive a1×1 convolutions with batchnorm and ReLU and a shortcut line of 2 layers over batchnorm. On the other hand ResNet 50 has Bottleneck (to reduce computation while adding more channels). It reduces channels via a3×3convolution, works spatially via 1×1 convolution and then back up 4 times the channel number

to LWMK4KMWL with another convolution, for each of convolution there exist batchnorm but reLU after summing only.

Though using the same number of blocks per stage for ResNet34 and ResNet50 (which was modified in the experiment), changing the block from simple block to bottleneck block makes huge change in parameter amount thus it is hard to estimate one from the other [3, 4, 6, 3]; the figure here refers to the parameter amount of the original 1000-class ImageNet models and is not the actual parameter amount of the three-class and a little bit more modified ones.

All models are loaded as pretrained models (using weights trained on ImageNet already) by passing the parameter "pretrained=True" to the torchvision loading function. I remove the last (fully connected) layer of the backbone of the pretrained model and add one fully connected layer that is suitable for three output classes (hence number of classes). Now, using full fine tuning meaning that I do not remove, nor freeze any parts inside the pretrained backbone so that when the parameters are updated the layers in the torchvision part too get adjusted. Workflow shown in figure 1. Architektur shown in Figure 2.

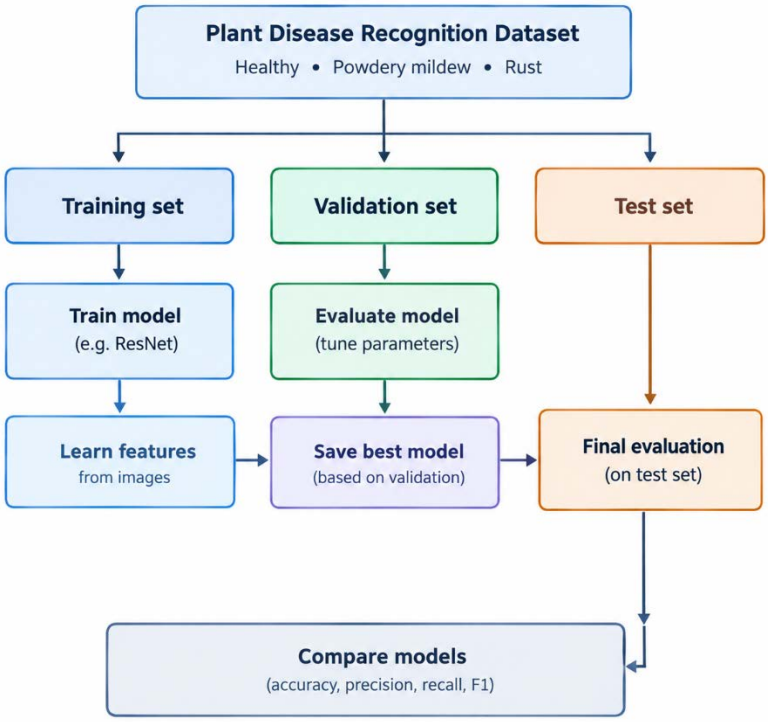


Fig. 1. *Classification of plant diseases - experimental process*

It depicts separate processing flows for the training, validation and test subsets, with the latter being used solely for test phase evaluation and the first using random horizontal flipping to update the model. However, note that three separate models are running independently the full cycle in the diagram above - ResNet18, ResNet34 and ResNet50, and these are not used as an ensemble. The validation subset is processed deterministically for evaluation.

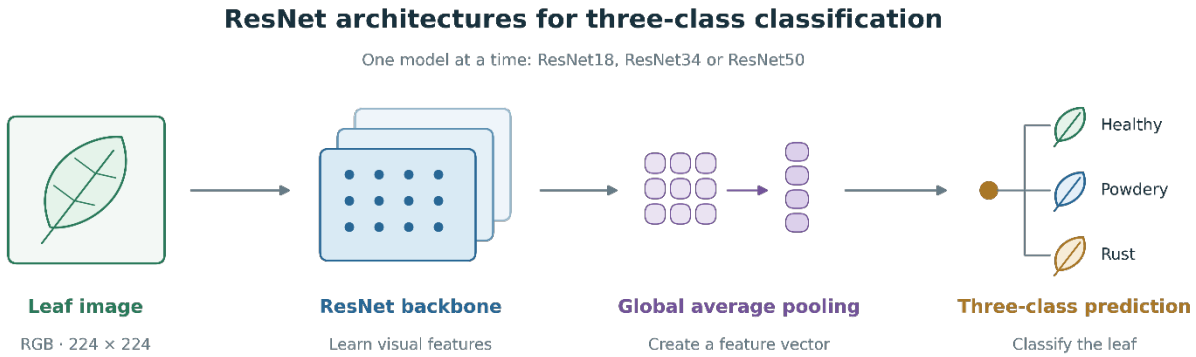


***Fig. 2**. Example of architectures used, ResNets configured for three-class classification*

Figure 2 comprises three sections: (A) presents the entire flow of a 224×224 RGB input passing through a 7×7 convolution (stride 2), batch normalization, ReLU, max pooling, four residual stages, global average pooling and then a new linear layer resulting in three logit values; (B) shows the BasicBlock internal structure from ResNet18/ResNet34, with two 3×3 convolutions and a shortcut connection; (C) details the 1×1→3×3→1×1 convolution sequence of a Bottleneck, characteristic of ResNet50, plus when a projection shortcut is used. The chosen stride placements in all sections are in accordance with the common ResNet implemented in torchvision.

Training and Model Selection

All three models are obtained by transfer learning, which consists in taking weights of a network pretrained on ImageNet and continue training all layers including a new three-class classifier on a target dataset by fine-tuning their weights. To perform classification we transition classifier output from logs to probabilities by using softmax function:

$$p_{ik} = \exp(z_{ik}) / \sum_{j=1}^{3} \exp(z_{ij}) \quad (4)$$

while multi-class cross-entropy is used as the loss function:

$$L_{CE} = -(1/B) \sum_{i=1}^{B} \sum_{kk1}^{3} (y_i = k-1) \log p_{ik} \quad (5)$$

where B is the batch size (16), and the indicator function corresponds to the indexing $y_i \in \{0,1,2\}$. In practice, at the code level, there is no separate softmax layer at the end of the model: `torch.nn.CrossEntropyLoss` accepts raw logits directly and internally combines log-softmax and negative log-likelihood calculations; formula (7) is provided solely to explain the mathematical meaning of the probability.

A total of 5 sweeps (epochs) of the system are applied for training. After each training epoch, the model is tested on its validating dataset and accuracy is calculated. The model parameters from the training epoch that achieves the highest accuracy is stored away as a checkpoint; if there training epochs achieve the same maximum score, then older checkpoint model would be given priority. At the end of five epochs the winning checkpoint is restored and final testing is done on the testing set.Early stopping methods, learning-rate scheduling, cross-validation, SMOTE, GAN-based augmentation, segmentation and testing on external (independent) dataset were not performed. The seed to the PyTorch and DataLoader is set with random state 42. In addition, when CUDA is available, I set the seed to the CUDA but did not enable cuDNN deterministic mode nor the fully deterministic operational mode since these were not specified in the code itself; therefore, no

deterministic claims are made, but that the primary random number generators are set. I ran the experiment on a single seed value and therefore no average, standard deviation or confidence interval is computed over repeated runs in the experiments.

Evaluation Metrics

Model quality is evaluated on the test set using Accuracy, as well as macro-averaged Precision, Recall, and F1-score metrics.

$$\text{Accuracy} = (1 / N_{test}) \sum_{i=1}^{Ntest} (y_i = \hat{y}_i) \quad (6)$$

where N_{test} is the number of samples in the test set, y_i is the true class of the i-th sample, $\hat{y}_i$ is the class predicted by the model, and $(\cdot)$ is an indicator function that takes the value 1 if the condition is true and 0 otherwise.

For each k-th class, Precision, Recall, and F1-score are calculated using the following formulas:

$$P_k = TP_k/(TP_k+FP_k), \quad R_k = TP_k/(TP_k+FN_k), \quad F1_k = 2P_kR_k/(P_k+R_k) \quad (7)$$

where TP_k is the number of positive samples correctly assigned to class k, FP_k is the number of samples from other classes incorrectly assigned to class k, and FN_k is the number of samples belonging to class k that are incorrectly classified into other classes. In the case of three-class classification, macro-averaged metrics are defined as the arithmetic mean of individual metrics across all classes:

$$P_{macro} = (1/3) \sum_{k=1}^{3} P_k, \quad R_macro = (1/3) \sum_{k=1}^{3} R_k, \quad F1_{macro} = (1/3) \sum_{k=1}^{3} F1_k \quad (8)$$

Here, P_{macro} is macro-averaged Precision; R_macro is macro-averaged Recall; F1_macro is macro-averaged F1-score; and k = 1, 2, 3 represent the three classes in the classification system.

If the number of classes in the classification system differs from three, macro-averaged metrics are calculated across K classes in the general case. In this instance, the coefficient 1/K is used instead of 1/3.

4. RESULTS

During the study, leaves classification results with Healthy, Powdery, Rust classes from using ResNet18, ResNet34 and ResNet50 models were compared. The training sets, validation sets and test sets used in the networks were the same and all models were trained for 5 epochs. The performance of the networks are evaluated by the validation and test accuracy and the macro average of precision, recall and F1-score. There were 1532 pictures; these belonged to the "Healthy" (528) (34.46%), "Powdery" (500) (32.64%) and "Rust" (504) (32.90%) classes. There were relatively equal amounts of images in the three classes, but in the training, the healthy class represented more images than the other two did. Figure 3 shows the data distribution and the ratio of experiments set that were shown to the models.

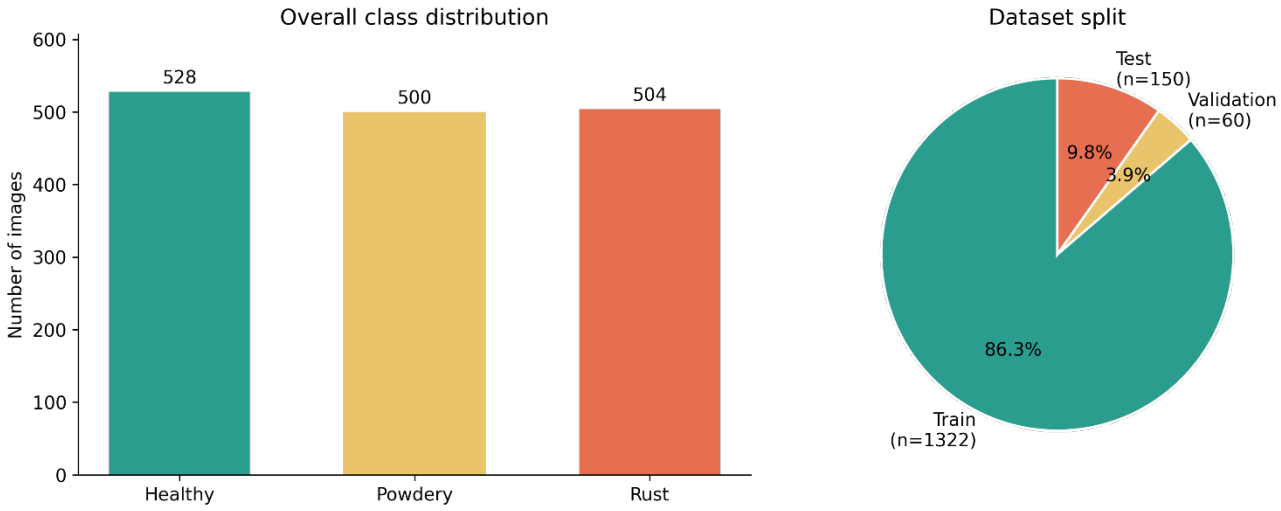


Fig. 3. *Highlights the division of dataset into training, validation and test, and how the overall classes are distributed*

The training set has a division of 86.29% of the images while the validation has 3.92% as the test images being 9.79%. Since the class representation was 20 and 50 respectively for the validation and testing images, hence both validation and testing images have been divided into all the classes equally. The number of classes has been shown in Figure 4.

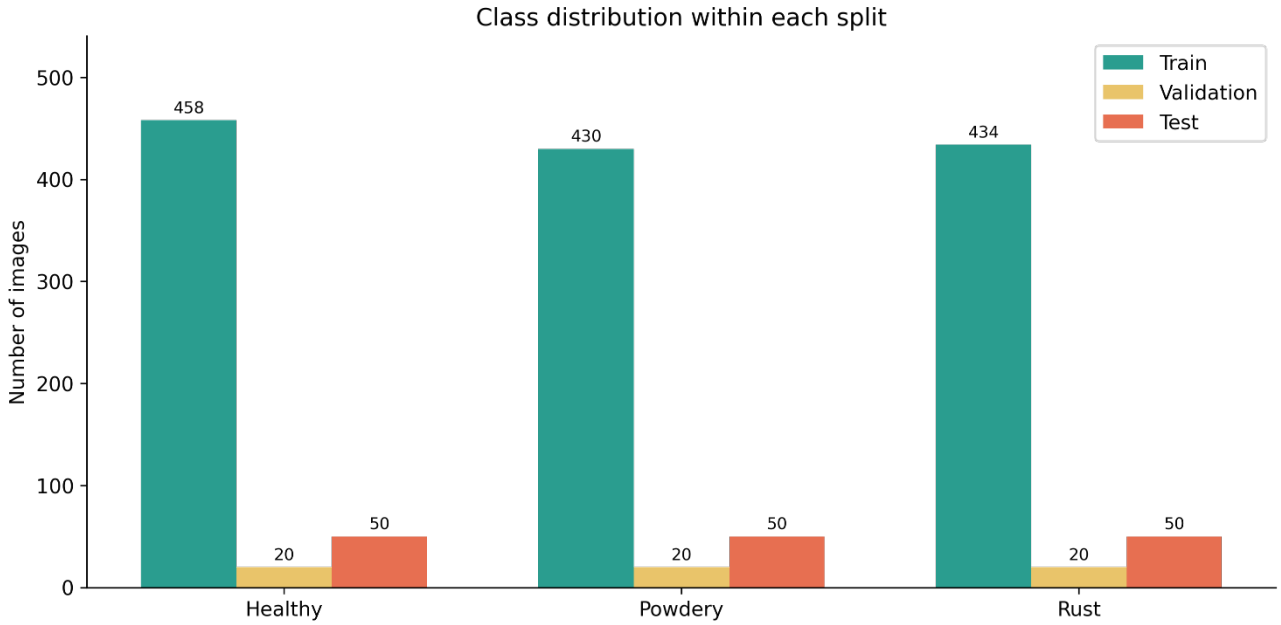


Fig. 4. *Number of images per class in the training, validation, and test sets*

Table 3. *Distribution of the dataset across classes and experimental subsets*

Class	Train	Validation	Test	All
Healthy	458	20	50	528
Powdery	430	20	50	500
Rust	434	20	50	504
All	1322	60	150	1532

Images from the provided sample show green leaves for the "Healthy" class, whitish coating for "Powdery" class and yellowish-brown spots for "Rust" class. Images appear to be taken under various surroundings, different lighting and at different angles of capture. Samples of images showing typical appearances for the three different classes are provided in Figure 5.

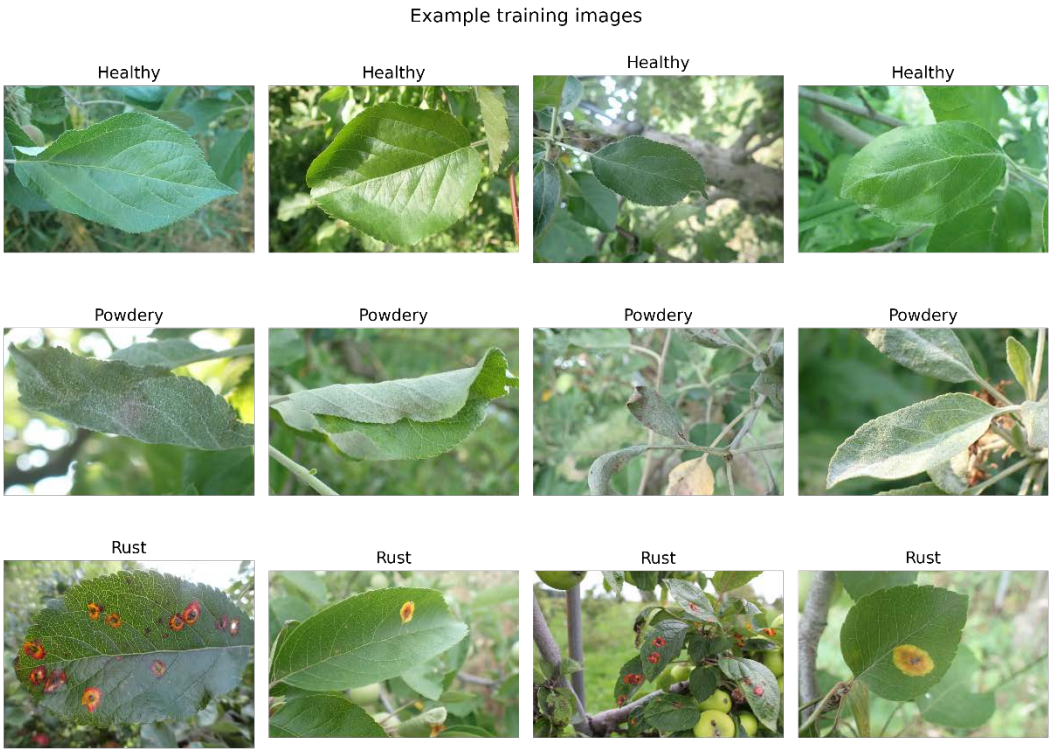


Fig. 5. *Examples of leaf images belonging to the Healthy, Powdery, and Rust classes*

File sizes in relation to images are spread between 700 - 1100 Kbytes according to the file size histogram (Figure 6 displays distributions for the 3 data partitions). Although this might affect the quality of classification by file size characteristics due to heterogeneity in the technical acquisition of source data, this was not measured.

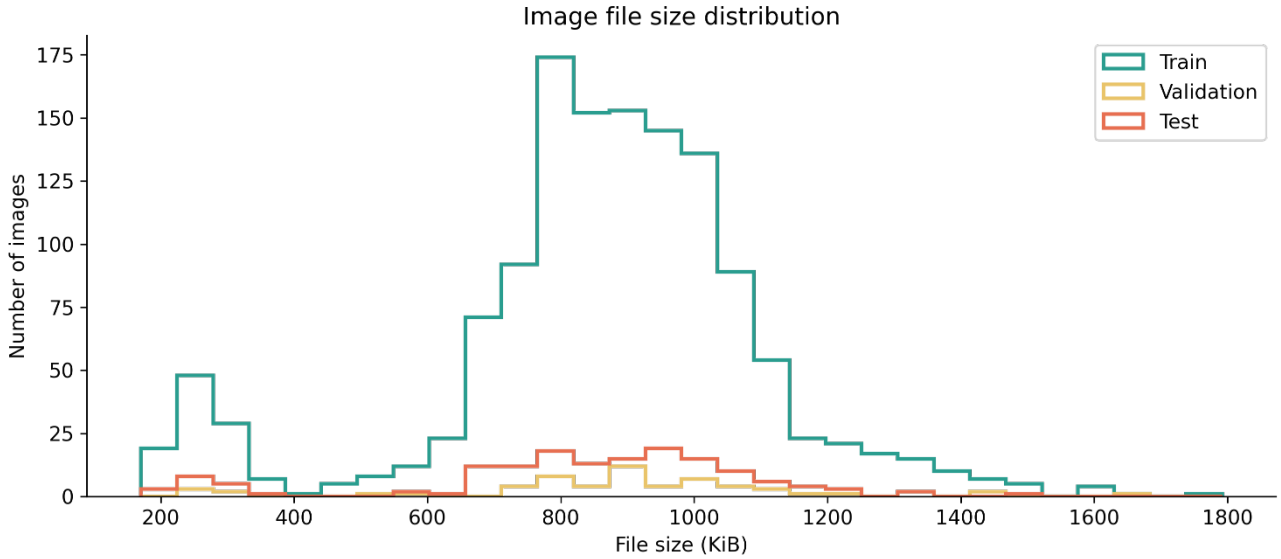


Fig. 6. *Distribution of image file sizes in the training, validation, and test sets.*

Over the course of five epochs, each of the models acquired to an astonishingly high validation accuracy of 1.0000 in the first phase and no change from this value was evident at each of assessment of accuracy in the remaining four epochs over which study of accuracy took place. It effectively follows that all 60 images within the given set were classified correctly every single time the accuracy was determined over the course of training. Since the algorithm retains the model only when validation accuracy is in excess of maximum accuracy achieved before, the algorithm retained the model at state after the first epoch. No significant improvement in validation came forth in the epochs after that.ResNet18, ResNet34 and ResNet50 had identical accuracies of 97.33% when evaluated on the test set. Given the size of the test set, this corresponded to each model accurately predicting 146 out of 150 images and misclassifying four. Just because four images were misclassified by each model did not mean it was the same set of four images. There was a difference of 2.67-percentage points between the validation and test accuracy of all three models. A comparison of these accuracies is provided below in Figure 7.

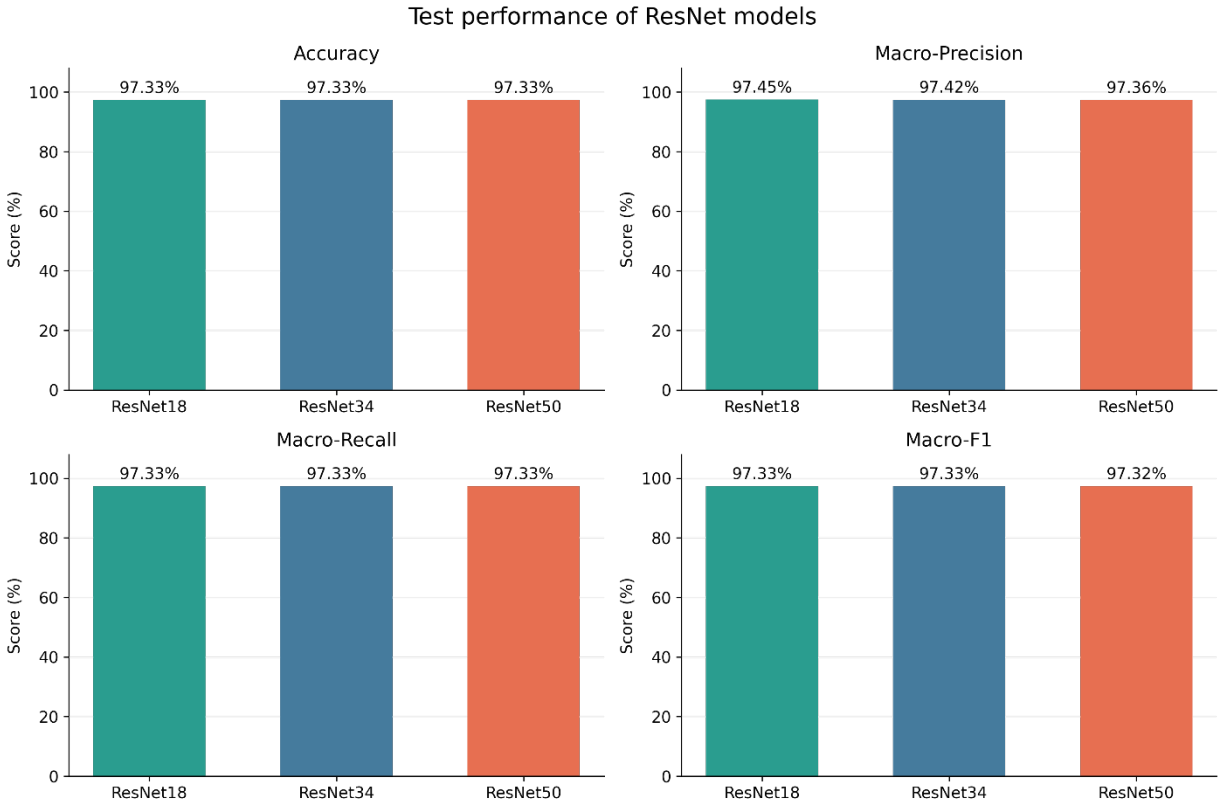


Fig. 7. *Comparison of validation and test accuracy for ResNet18, ResNet34, and ResNet50 models*

Table 4*. Classification results of ResNet models*

Model	**Maximum Validation Accuracy, %**	**Test Accuracy, %**	**Precision (macro), %**	**Recall (macro), %**	**F1 (macro), %**
ResNet18	100.00	97.33	97.45	97.33	97.33
ResNet34	100.00	97.33	97.42	97.33	97.33
ResNet50	100.00	97.33	97.36	97.33	97.32

Note: Percentage figures were calculated from experimental results rounded to four decimal places.

In all models the models differed when we looked at the macro-averaged precision metric, as the value for ResNet18 was 97.45%, while it was 97.42% for ResNet34 and value of RepsResNet50was 97.36%. The difference in this metric between ResNet18 and ResNet50 was 0.09 percentage points, while the difference between ResNet18 and ReResNet34 was just 0.03 percentage points. It means that the best numerical result had ResNet18 but the differences here are minimal. The macro-averaged recalls were all 97.33% and the same reason, which explained agreement between overall accuracy and itself, explained this one too. Again it does not mean that the recalls are the exact same for each class. About F1, again ResNet18 and ResNet34 generated 97.33% and ResNet50 generated 97.32%, thus a maximum difference of 0.01% between values. Figure 8 shows comparison between test accuracy, precision, recall and F1.

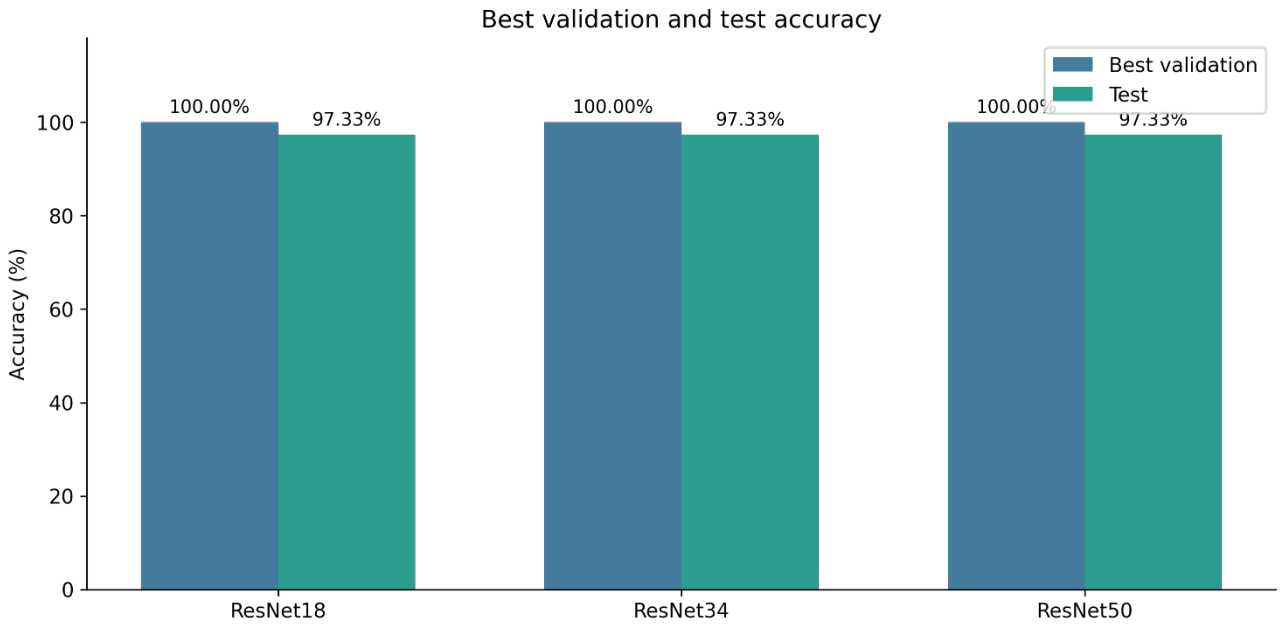


Fig. 8*. Comparison of test accuracy, macro-averaged precision, recall, and F1 scores for ResNet18, ResNet34, and ResNet50 models*

The conclusion is that with training for 5 epochs on this split, increasing the depth does not imply an increase in accuracy on the test set. All ResNets achieve about the same accuracy. Furthermore, their precision and f1 scores are similar. They are not statistically compared to draw concrete conclusions.Validation accuracy reached 100% after an epoch, but this does not imply that no overtraining took place or that models will work equally well on "real" problems. Results for classification performance on the given validation and test sets were displayed. Further analysis on class errors, robustness to new test data and computational efficiency would be possible by analyzing the errors for each class, by testing on another test data set and respectively on model execution times.

5. DISCUSSION

An advantage is comparing ResNet18, ResNet34 and ResNet50 models in similar settings (same data splitting, same feature extraction, same hyperparameters during training and a budget for 5 epochs). Ferentinos and Too et al.(2016) compared different architectures while this work evaluates three configurations of a single architecture in a specific 3-class disease prediction. This means that model selection can thus be justified/derived from experiment results on this specific task.In addition, macro-averaged precision, recall and F1-score could be applied in the evaluation, along with overall accuracy. With 50 images for each of the classes in the test data, it ensured that the classes are weighted equally. To finally evaluate the models on the test set, after training the models on training data and making selections based on data from evaluation, allowed a proper distinction between selection and evaluation of classification performance for the final models. These approaches strengthened the methodological basis of the comparison although this is not to imply that they had not been used in other studies.ConclusionWe modified models such as ResNet18, ResNet34 and ResNet50 (which are pre-trained on ImageNet data to classify images of plant leaves into "Healthy", "Powdery" and "Rust"). The models were compared on the same train/test splits, image pre-processing procedures and training parameters. Of 1532 pictures, 1,322 images were used to train, 60 for validation and 150 for testing purposes. The validation accuracy was used to choose the 'best versions of models' which were then evaluated using the test set.Test accuracy of all models was

identical at 97.33%, which classified 146 out of 150 images correctly. Macro-averaged F1-score was 97.33% in the ResNet18 and ResNet34 cases and 97.32% in the ResNet50 case. Macro-averaged precisions were 97.45%, 97.42%, and 97.36% respectively. Results above imply that in the 3-class classification case, a deeper model would yield better classification performance. The main contribution of the project is testing and comparing three kinds of ResNet models based on the same experimental protocol for evaluating classification capability in a certain kind of application using ResNet. Since ResNet18 showed the same accuracy as the deeper ones, it would still be considered for use in practical tests. However, as computation speed, memory, and energy use were not measured, no certain conclusion can be drawn yet as for its resource use efficiency.

CONFLICT OF INTEREST

The author declares that they have conflict of interest.

REFERENCES

Ahmed, I., & Yadav, P. K. (2023). Plant disease detection using machine learning approaches. *Expert Systems*, *40*(5), e13136.

Demilie, W. B. (2024). Plant disease detection and classification techniques: a comparative study of the performances. *Journal of Big Data*, *11*(1), 5.

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

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Jafar, A., Bibi, N., Naqvi, R. A., Sadeghi-Niaraki, A., & Jeong, D. (2024). Revolutionizing agriculture with artificial intelligence: plant disease detection methods, applications, and their limitations. *Frontiers in Plant Science*, *15*, 1356260.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science, 7*, Article 1419. https://doi.org/10.3389/fpls.2016.01419

Moupojou, E., Tagne, A., Retraint, F., Tadonkemwa, A., Wilfried, D., Tapamo, H., & Nkenlifack, M. (2023). FieldPlant: A dataset of field plant images for plant disease detection and classification with deep learning. *Ieee Access*, *11*, 35398-35410.

Pandian, J. A., Kumar, V. D., Geman, O., Hnatiuc, M., Arif, M., & Kanchanadevi, K. (2022). Plant disease detection using deep convolutional neural network. *Applied Sciences*, *12*(14), 6982.

Qadri, S. A. A., Huang, N. F., Wani, T. M., & Bhat, S. A. (2024). Advances and challenges in computer vision for image-based plant disease detection: a comprehensive survey of

machine and deep learning approaches. *IEEE Transactions on Automation Science and Engineering*, *22*, 2639-2670.

Shafay, M., Hassan, T., Owais, M., Hussain, I., Khawaja, S. G., Seneviratne, L., & Werghi, N. (2025). Recent advances in plant disease detection: challenges and opportunities. *Plant Methods*, *21*(1), 140.

Shafik, W., Tufail, A., Namoun, A., De Silva, L. C., & Apong, R. A. A. H. M. (2023). A systematic literature review on plant disease detection: motivations, classification techniques, datasets, challenges, and future trends. *Ieee Access*, *11*, 59174-59203.

Shoaib, M., Shah, B., Ei-Sappagh, S., Ali, A., Ullah, A., Alenezi, F., ... & Ali, F. (2023). An advanced deep learning models-based plant disease detection: A review of recent research. *Frontiers in plant science*, *14*, 1158933.

Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture, 161*, 272–279. https://doi.org/10.1016/j.compag.2018.03.032