

Plant disease identification from images using machine learning algorithms

Svitlana Antoshchuk¹, Tetiana Kunup^{1*}, Oleg Kravchenko¹

^{1,2,3} Odessa Polytechnic National University, Shevchenko Avenue 1, Odessa, 65001, Ukraine

Abstract

Plant-disease identification in real-world conditions remains challenging because models trained on laboratory images often lose accuracy when applied to images with complex backgrounds, variable illumination, and viewpoint changes. This study aims to develop and evaluate a mobile-oriented plant-disease identification system and determine the most effective CNN architecture for practical deployment. Four models—EfficientNet-B0, MobileNetV3-Small, DenseNet-121, and ConvNeXt-Tiny—were pretrained on PlantVillage, fine-tuned on PlantWild, and independently evaluated on PlantDoc. The models were trained using transfer learning, data augmentation, AdamW optimization, label smoothing, and early stopping. ConvNeXt-Tiny achieved the best PlantDoc performance, with 62.65% top-1 and 90.52% top-5 accuracy, compared with 48.38% and 82.47% for DenseNet-121, 46.28% and 83.23% for EfficientNet-B0, and 35.22% and 69.38% for MobileNetV3-Small. Thus, ConvNeXt-Tiny improved top-1 accuracy by 14.27 percentage points over the second-best model. The results confirm a substantial domain shift between controlled and real-world images and demonstrate the feasibility of integrating the selected model into a mobile application for on-device plant-disease identification. The proposed system can support accessible preliminary disease diagnosis in practical agricultural applications.

Keywords

image classification, plant disease identification, convolutional neural networks, deep learning, PlantVillage, PlantWild, PlantDoc, artificial intelligence

1. Introduction

Agricultural production is exposed to environmental, technological, and economic risks, while plant diseases remain an important factor affecting crop quality and yield [1, 2]. Automated image-based identification can support the rapid preliminary diagnosis of plant diseases. However, the practical performance of such systems depends on their ability to generalize from controlled training images to photographs captured under real-world conditions.

A major challenge is the difference between laboratory and in-the-wild datasets. PlantVillage contains images captured under relatively uniform conditions, whereas PlantWild includes complex backgrounds, variations in lighting and viewpoint, partial occlusion, and other real-world factors [8, 9]. Consequently, high accuracy on PlantVillage does not necessarily indicate comparable performance on field images. An external dataset such as PlantDoc is therefore required to evaluate model generalization under more realistic conditions [12].

The objective of this study is to develop and evaluate an image-based plant-disease identification system suitable for deployment on mobile devices. Four convolutional neural-network architectures with different accuracy–efficiency trade-offs—EfficientNet-B0, MobileNetV3-Small, DenseNet-121, and ConvNeXt-Tiny—were compared. The models were pretrained on PlantVillage, fine-tuned on

ADT-ISMDDC 2026: *Advanced Digital Technologies: Intelligent Systems, Modeling, and Data-Driven Control*, September 22–24, 2026, Odessa, Ukraine

^{1*} Corresponding author.

✉ asg@op.edu.ua. (S. Antoshchuk); kunup.t.v@op.edu.ua (T. Kunup); 10253024@stud.op.edu.ua (O. Kravchenko);

ORCID 0000-0002-9346-145X (S. Antoshchuk); 0000-0003-0246-0951 (T. Kunup); 0009-0006-6982-0965 (O. Kravchenko);



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

PlantWild, and evaluated on PlantDoc. The best-performing model was then converted to a mobile inference format and integrated into an Android application. The main contributions of this study are: (1) a two-stage training pipeline combining controlled and in-the-wild plant images; (2) a comparative evaluation of four CNN architectures using PlantDoc as an external evaluation dataset; and (3) the deployment of the best-performing model in a mobile application for on-device plant-disease identification.

2. Review of CNN architectures and plant-disease datasets

Today, many convolutional neural network (CNN) architectures are publicly available. This paper analyzes and trains several CNN architectures specifically optimized for operation on devices with limited computational power, enabling their use in mobile systems. One of the well-known groups of convolutional neural networks is EfficientNet, developed and presented by Google engineers in 2019. A distinctive feature of these models is the use of compound scaling, which involves scaling the depth (number of layers), width (number of channels in each layer), and input image resolution by applying specific scaling coefficients. The EfficientNet group consists of eight convolutional neural network models, from EfficientNet-B0 to EfficientNet-B7. These models differ in resource requirements and target applications. EfficientNet-B0 is the most compact and was used as the basis for the other models, from EfficientNet-B1 to EfficientNet-B7.

This model has been widely used in real-world image analysis tasks and can therefore also be applied to plant disease classification. Key features of this model include:

- use of modified inverted residual MBConv blocks, borrowed from the MobileNetV2 model;
- high classification accuracy with a small number of parameters, achieved using a compound scaling approach;
- efficient use of computational resources;
- support for deep networks without significantly increasing complexity;
- use of a squeeze-and-excitation mechanism to improve feature extraction.

EfficientNet-B0 was designed to provide a favorable balance between classification accuracy and computational efficiency by jointly scaling network depth, width, and input resolution [3].

The MobileNet family was designed for efficient inference on mobile and embedded devices. MobileNetV3 was optimized for mobile-phone CPUs using hardware-aware neural architecture search and the NetAdapt algorithm. Its architecture extends the inverted residual structure used in MobileNetV2 by incorporating squeeze-and-excitation modules and the h-swish activation function. MobileNetV3-Small was designed for low-resource use cases and was therefore selected for the experiments in this study [4]. ResNet was introduced by He et al. and is based on residual blocks with skip connections, which make substantially deeper networks easier to optimize [5].

DenseNet uses dense connectivity, in which each layer receives the feature maps produced by all preceding layers. This design encourages feature reuse, strengthens feature propagation, and improves gradient flow. DenseNet-121 is a 121-layer variant of this architecture [6]. ConvNeXt is a modernized pure convolutional architecture inspired by several design choices used in vision transformers. ConvNeXt-Tiny was included in the experiments because it provides a strong accuracy–efficiency trade-off [7]. These architectures provide different accuracy–efficiency trade-offs and were considered as candidate models for the experimental comparison. This demonstrates the growing importance and widespread application of deep learning in computer vision. Training

an image classifier for many plant-disease classes requires a sufficiently large and diverse dataset. Such datasets are also needed to evaluate model accuracy and generalization. In the field of computer vision, datasets containing images of leaves, fruits, and other plant organs are particularly important for agricultural applications because of their distinctive visual characteristics. One of the most widely used datasets for plant-disease classification is PlantVillage. Mohanty et al. trained convolutional neural networks on 54,306 images representing 14 crop species and 38 crop-disease classes. The images were collected under controlled conditions with relatively uniform backgrounds and lighting [8]. The main strengths of the PlantVillage dataset are its large number of images, clear class organization, and suitability for training convolutional neural networks (CNNs). Mohanty et al. reported 99.35% accuracy on a held-out PlantVillage split [8]; however, this result does not establish field performance because the images were collected under controlled conditions. In the dataset version used in this study, tomato-related classes accounted for approximately 43.4% of the images, indicating substantial class imbalance. The analysis of the PlantVillage dataset revealed several limitations that may reduce its effectiveness for model training. An example image from this dataset is shown in Fig. 1.



Figure 1: Example image from the PlantVillage dataset “Apple__Black_rot” [8]

The controlled backgrounds and class imbalance of PlantVillage limit direct generalization to field images. To improve generalization to real-world conditions, the models were subsequently fine-tuned on PlantWild using transfer learning. The PlantWild dataset, which includes 89 classes representing 33 different crops, meets these requirements. It contains 18,542 images, collected from in-the-wild internet sources and exhibiting diverse backgrounds, viewpoints, and lighting conditions (Fig. 2). The objects in the images vary in size, and leaf overlap and background noise are also present.



Figure 2: Image from the PlantWild dataset for the class "apple black rot"

These characteristics make PlantWild well suited for training models intended for real-world agricultural applications. A unique feature of the PlantWild dataset is the availability of a JSON file containing textual descriptions of the visual characteristics of each class (50 descriptions per class), enabling multimodal training with vision-language models [9].

As with the PlantVillage dataset, the PlantWild dataset does not include augmented images; data augmentation will be applied programmatically before model training. The dataset contains 89 classes for 33 different crops. Each crop has one class representing healthy plants (e.g., a healthy apple leaf) and at least one disease class. In total, the PlantWild dataset includes 56 disease classes.

Each class contains between 44 and 589 images [9]. The distribution of photographs by healthy and diseased plant classes can be seen in Fig. 3.

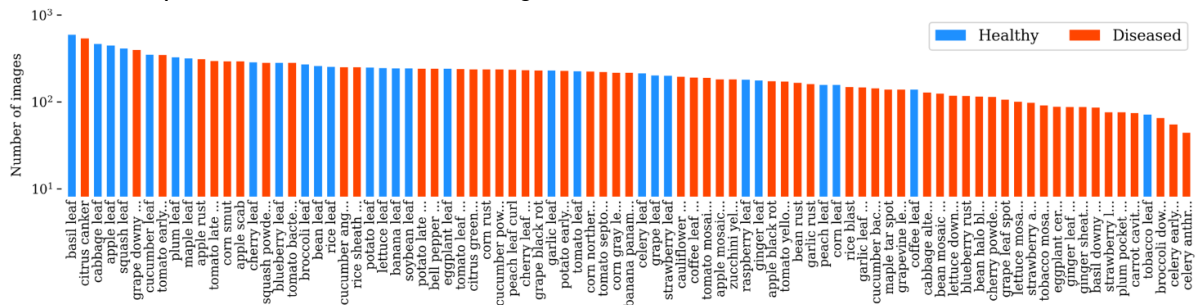


Figure 3: Distribution of images across the 56 diseased and 33 healthy classes in the PlantWild dataset [9]

PlantPAD is a large-scale plant-disease platform containing 421,314 images covering 63 crops and 310 diseases [10]. The distribution of images among the 63 plant species is shown in Fig. 4.

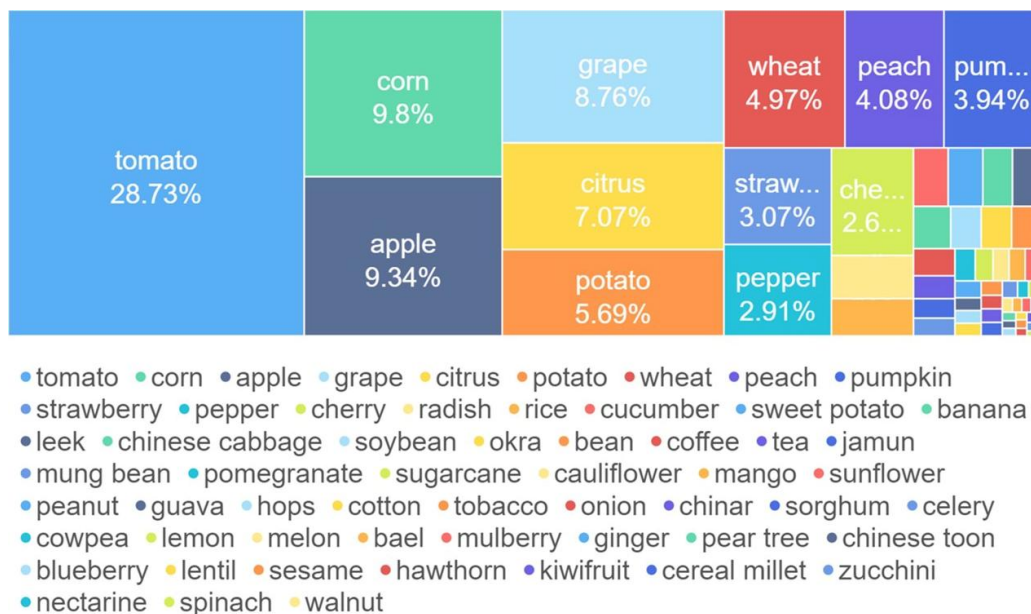


Figure 4: Distribution of images among the 63 plant species in the PlantPAD dataset [10]

Published in February 2026, PlantSeg contains 7,774 in-the-wild diseased images with corresponding pixel-level segmentation masks, covering 115 plant-disease categories across 34 host plants [11]. Unlike laboratory datasets, PlantSeg primarily contains images collected in complex real-world settings. Its pixel-level masks support the localization of diseased regions and the evaluation of plant-disease segmentation models [11]. Examples of PlantSeg images and segmentation masks are shown in Fig. 5.



Figure 5: Examples of PlantSeg images and segmentation masks [11]

After analyzing the features, advantages, and limitations of each dataset, it was decided to perform an initial training phase using the PlantVillage dataset, which consists of laboratory images, followed by a second training phase using the PlantWild dataset, which contains field images of plants. The PlantDoc dataset will be used as the test dataset to evaluate the performance of the trained models [12].

3. Experimental setup and model training

Training and evaluation were performed in three stages: pretraining on PlantVillage, fine-tuning on PlantWild, and evaluation on PlantDoc. Because the datasets use different class-label formats, the labels were normalized to a common lowercase, space-separated form. For example, “Apple__Apple_scab,” “Apple_scab,” and “Apple Scab” were all normalized to “apple scab.” A mapping was then created from each PlantDoc class to the corresponding PlantWild output class. PlantDoc classes for which no corresponding PlantWild class was found were marked as unmatched and excluded from the mapped-class accuracy calculation. The trained models were subsequently exported to ONNX and LiteRT (.tflite) formats. All four models were implemented in Python using PyTorch and torchvision. The models were initialized with the default ImageNet-pretrained weights, and their classification heads were replaced according to the number of target classes: 38 for PlantVillage and 89 for PlantWild. All input images were processed at a resolution of 224×224 pixels and normalized using the ImageNet channel means and standard deviations. PlantVillage and PlantWild were each divided into training and validation subsets using an 85:15 ratio and a fixed split seed of 42. PlantDoc was not used during model training or checkpoint selection. It was used for cross-dataset evaluation and for selecting the architecture deployed in the mobile application.

The models were optimized using AdamW and cross-entropy loss with label smoothing of 0.1. During PlantVillage pretraining, the initial learning rate was 1×10^{-3} , the weight decay was 1×10^{-4} , and training was scheduled for up to 30 epochs. A three-epoch linear warm-up was followed by cosine-annealing learning-rate decay. Fine-tuning on PlantWild was scheduled for up to 60 epochs and divided into two phases. During the first 15 epochs, only the classification head was trained with a learning rate of 5×10^{-3} . During the remaining 45 epochs, the backbone was progressively unfrozen, using learning rates of 1×10^{-3} for the classification head and 1×10^{-4} for the backbone. Early stopping monitored validation accuracy with a patience of eight epochs and a minimum improvement threshold of 0.001. For each stage, the checkpoint with the highest validation accuracy was retained.

Table 1: Main training parameters

Parameter	PlantVillage pretraining	PlantWild fine-tuning
-----------	--------------------------	-----------------------

Target classes	38	89
Input resolution	224×224	224×224
Initialization	ImageNet-pretrained weights	Best PlantVillage checkpoint
Train/validation split	85% / 15%, seed 42	85% / 15%, seed 42
Maximum epochs	30	60
Training phases	Full model	15 epochs head-only + 45 epochs progressive unfreezing
Optimizer	AdamW	AdamW
Learning rate	1×10^{-3}	Head: 5×10^{-3} in Phase A; head: 1×10^{-3} and backbone: 1×10^{-4} in Phase B
Weight decay	1×10^{-4}	1×10^{-4}
Loss function	Cross-entropy, label smoothing 0.1	Cross-entropy, label smoothing 0.1
LR schedule	3-epoch linear warm-up followed by cosine annealing	Cosine annealing
Early stopping	Patience 8, minimum improvement 0.001	Patience 8, minimum improvement 0.001
Model selection	Highest validation accuracy	Highest validation accuracy

Moderate data augmentation was applied during PlantVillage pretraining and during the head-only phase of PlantWild fine-tuning. It included random resized cropping, horizontal and vertical flipping, color jitter, and rotations of up to 20° . During full fine-tuning on PlantWild, stronger augmentation was used, including random resized cropping, flips, color jitter, rotations of up to 45° , perspective transformation, occasional grayscale conversion, Gaussian blur, and random erasing. MixUp with $\alpha = 0.2$ and class-balanced weighted sampling were applied during this phase to reduce overfitting and mitigate the class imbalance of PlantWild. Training additionally used automatic mixed precision and gradient clipping with a maximum norm of 1.0.

PlantDoc was used as an external evaluation dataset without random splitting or training-time augmentation. Each image was resized to 256 pixels along its shorter dimension, center-cropped to 224×224 pixels, and normalized using the same ImageNet statistics as the validation data. Five-view test-time augmentation was applied by averaging the predictions for the original image and four randomly augmented views. The augmentations included horizontal and vertical flipping, rotations of up to 15° , and moderate color jitter. Top-1 and top-5 accuracies were calculated for PlantDoc classes that could be mapped to corresponding PlantWild output classes.

During the study, the following convolutional neural network models were trained: EfficientNet-B0, MobileNetV3-Small, DenseNet-121, and ConvNeXt-Tiny. The EfficientNet-B0 model demonstrated the following accuracy results:

- Pretraining on PlantVillage: 99.97% on the training set, 99.89% on the validation set;

- Fine-tune on PlantWild: 38.90% on the training set, 60.59% on the validation set;
- Testing on PlantDoc: 46.28% of cases of providing the correct disease with the highest probability, 83.23% of cases of the correct disease being in the top 5 most likely. Graphs showing the successive accuracy progression of the EfficientNet-B0 model with each training epoch are shown in Fig. 6 and Fig. 7. The MobileNetV3-Small model has the worst results:
- the pretraining stage on PlantVillage (Fig. 18) – 99.97% on the training set, 99.78% on the validation set;
- the fine-tune stage on PlantWild (Fig. 9) – 32.58% on the training set, 51.45% on the validation set;
- testing on PlantDoc – 35.22% of cases of providing the correct disease with the highest probability, 69.38% of cases of the correct disease being in the top 5 most likely.

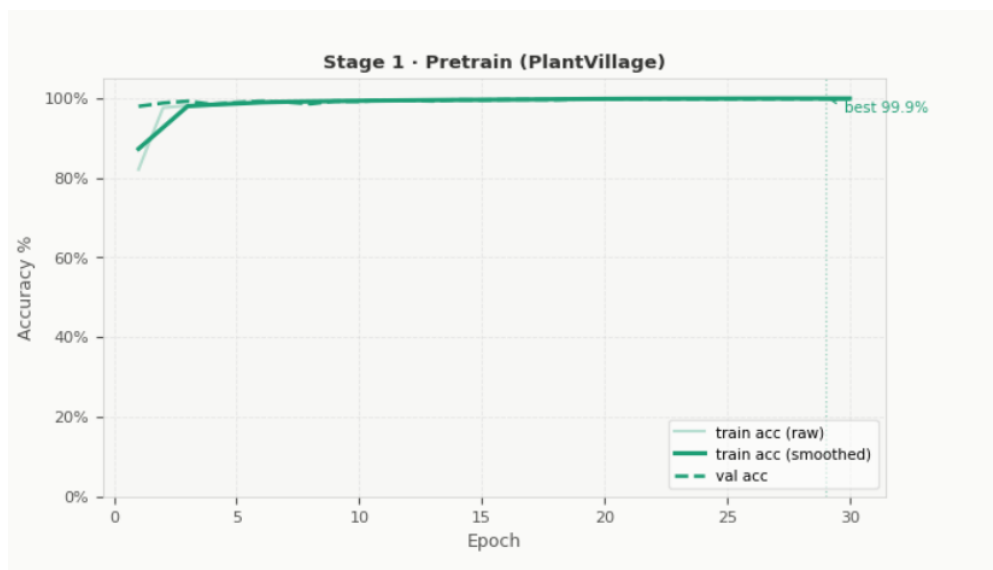


Figure 6: Training the EfficientNet-B0 model, pretrain stage

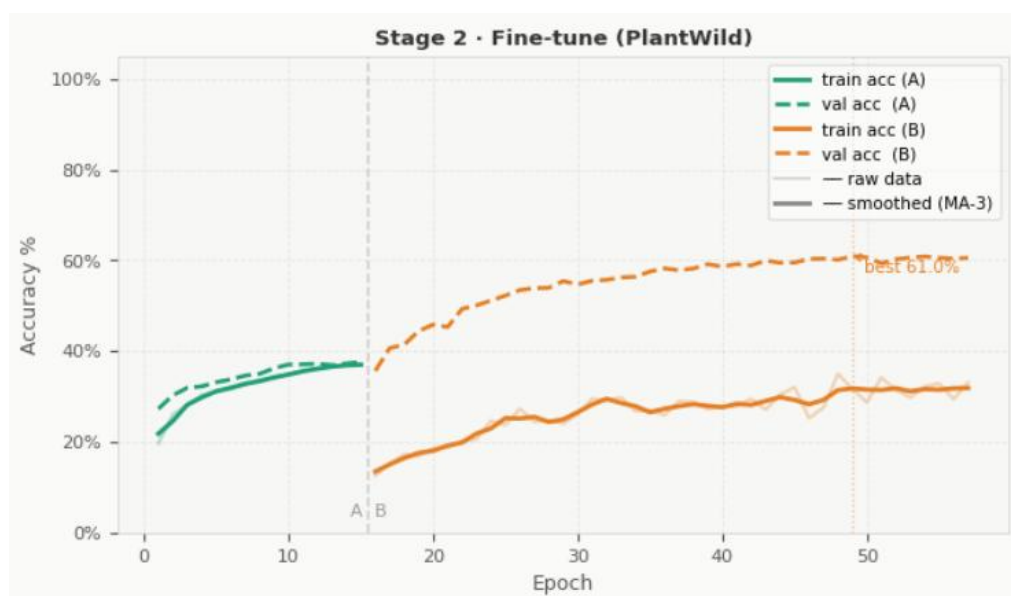


Figure 7: Training the EfficientNet-B0 model, fine-tune stage

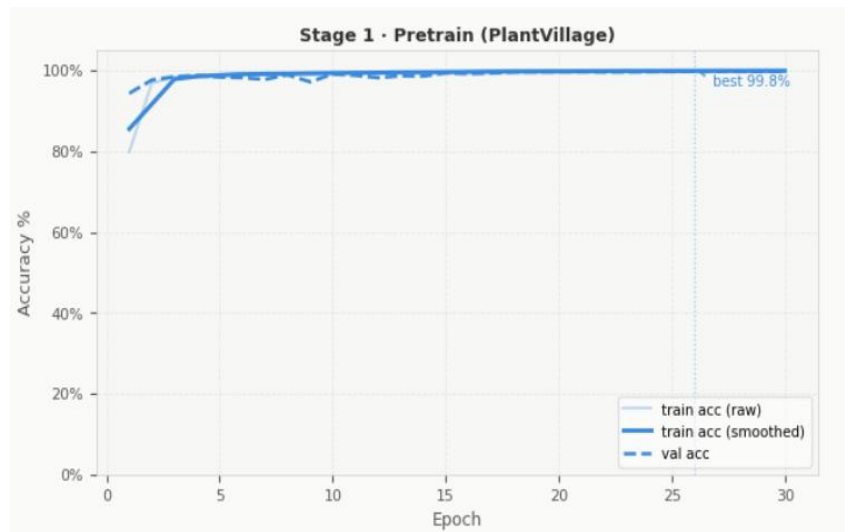


Figure 8: Training the MobileNetV3-Small model, pretrain stage

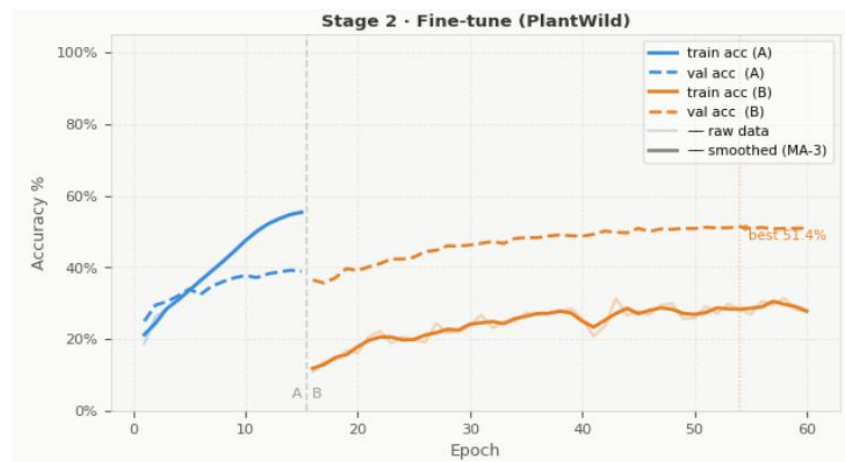


Figure 9: Training the MobileNetV3-Small model, fine-tune stage

The DenseNet-121 model demonstrated the following results:

- pretraining stage on PlantVillage – 99.98% on the training set, 99.88% on the validation set (Fig. 10);
- fine-tune stage on PlantWild – 35.97% on the training set, 58.62% on the validation set (Fig. 11);
- PlantDoc evaluation yielded a top-1 accuracy of 48.38% and a top-5 accuracy of 82.47%.

The model with the best results among all was ConvNeXt-Tiny. It demonstrated the following accuracy rates:

- pretraining on PlantVillage: 98.12% on the training set, 99.01% on the validation set (Figure 12);
- fine-tune on PlantWild: 47.05% on the training set, 72.17% on the validation set (Figure 13);
- testing on PlantDoc: 62.65% of cases provided the correct disease with the highest probability, 90.52% of cases of the correct disease being in the top 5 most likely.

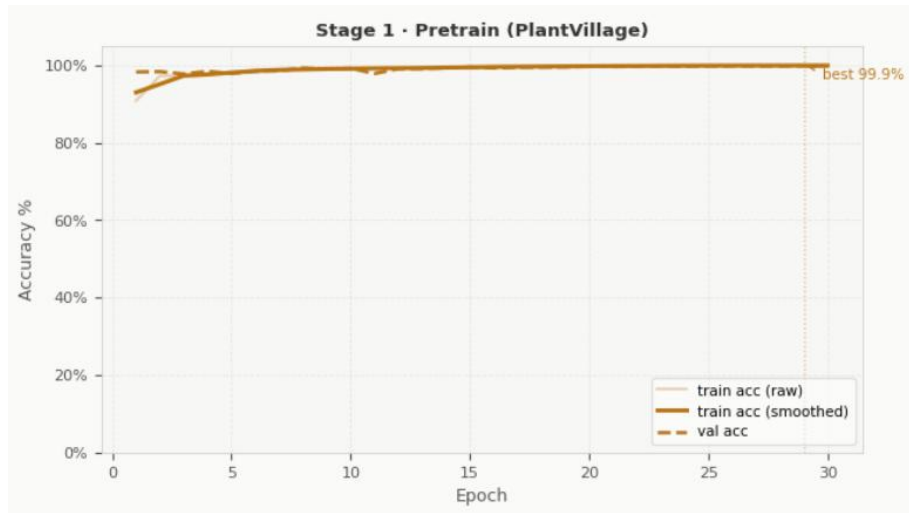


Figure 10: Training the DenseNet-121 model, pretrain stage

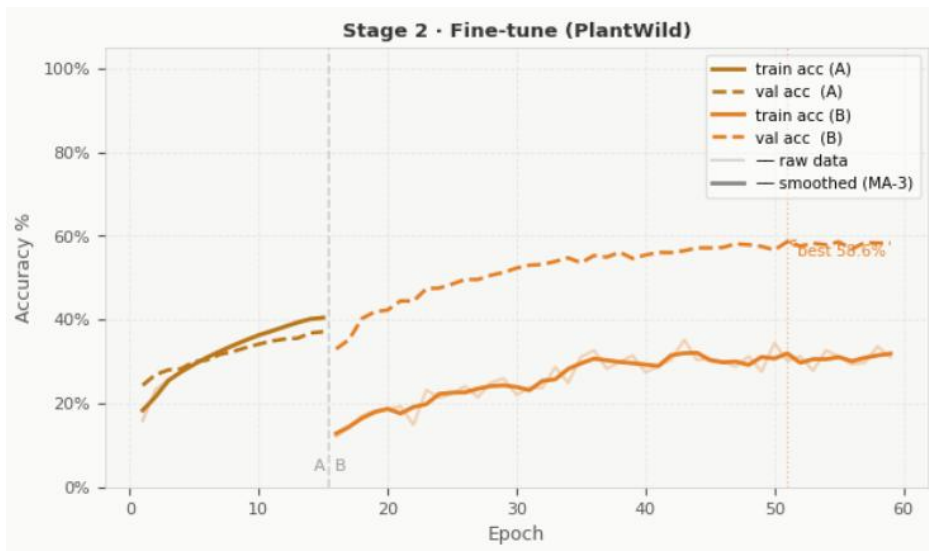


Figure 11: Training the DenseNet-121 model, fine-tune stage

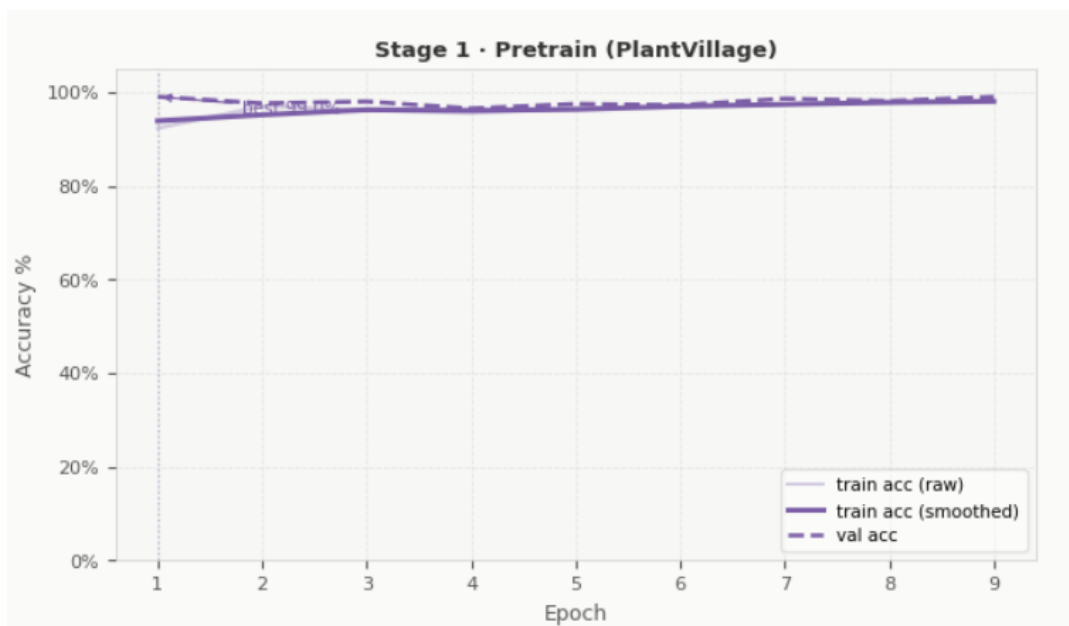


Figure 12: Training the ConvNeXt-Tiny model, pretrain stage

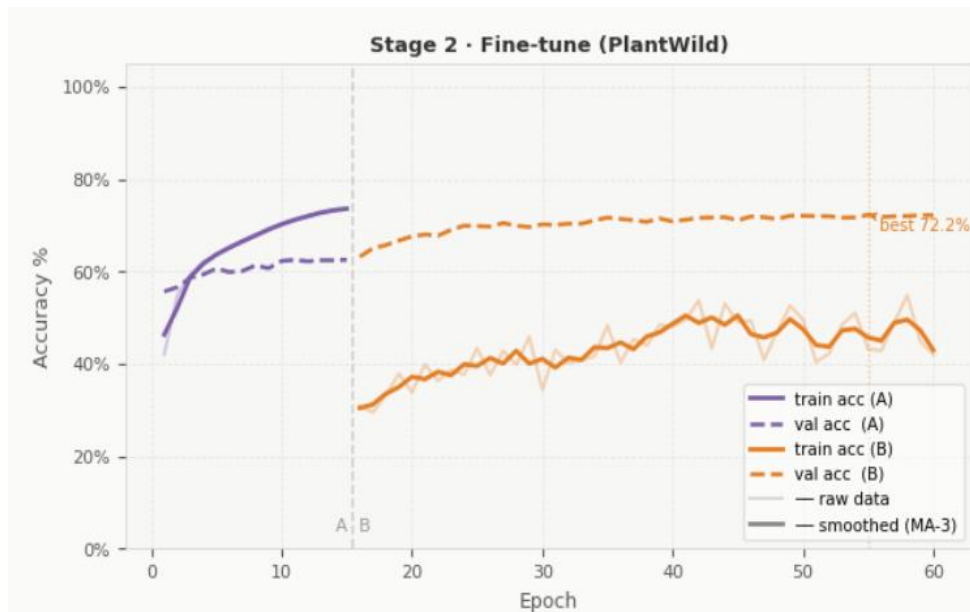


Figure 13: ConvNeXt-Tiny model training, fine-tune stage

ConvNeXt-Tiny pretraining stopped after 9 epochs because validation accuracy did not improve by more than 0.001 for eight consecutive epochs. Since this model achieved the highest PlantDoc accuracy, it was selected for deployment in the mobile application. ConvNeXt-Tiny has the highest accuracy rates, so it was chosen for use in the mobile app. Training was performed using a CUDA-enabled GPU in Google Colab. Each architecture required approximately 30 minutes to train. After analyzing the training results, we can conclude that all models demonstrated the ability to quickly learn to accurately distinguish plant diseases on the PlantVillage dataset, all photographs of which were taken under identical conditions with a simple background. Performance decreased substantially on PlantWild, reflecting the domain shift and the greater visual complexity of in-the-wild images. The use case diagram for the mobile application for identifying plant diseases is shown in Fig. 14. It sequentially displays all the actions the program provides the user. The application can be used in guest mode; registration is required only to save results and access recognition history. The mobile application for identifying agricultural diseases identifies the plant's condition after a visual inspection. To successfully perform this task, the system requires only one photo of the affected leaves or fruit, which the user can provide in two ways: using the mobile phone camera or its gallery.

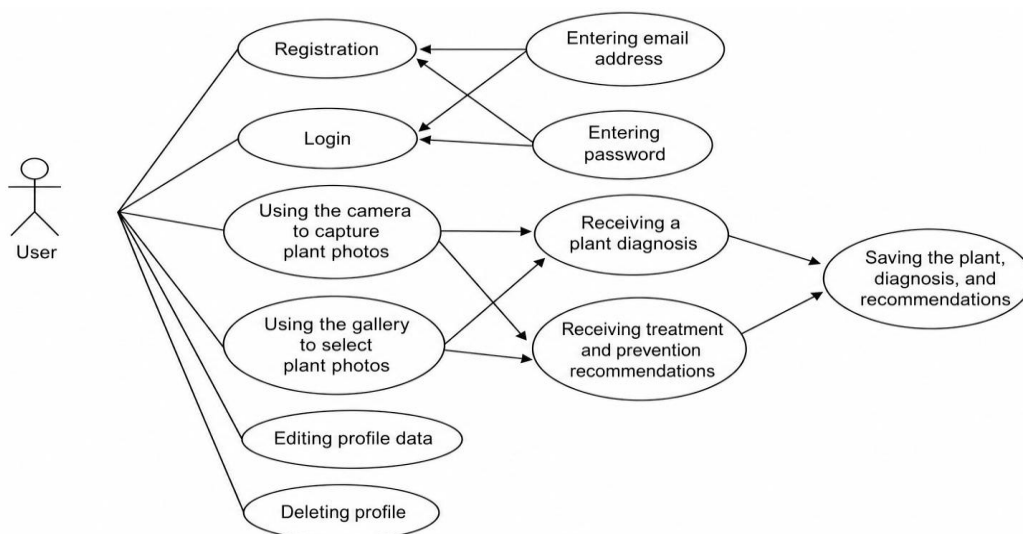


Figure 14: Use-case diagram

After receiving an image, the system analyzes it and provides a status report, including the plant species, diagnosis, and brief treatment recommendations. The user can save the results in a database. The program also includes the ability to edit profile data, including changing the email address, password, and username. The user can also delete their profile along with all data if they no longer wish to use the program. The results screen displays a list of the five most likely diagnoses and treatment methods for each.

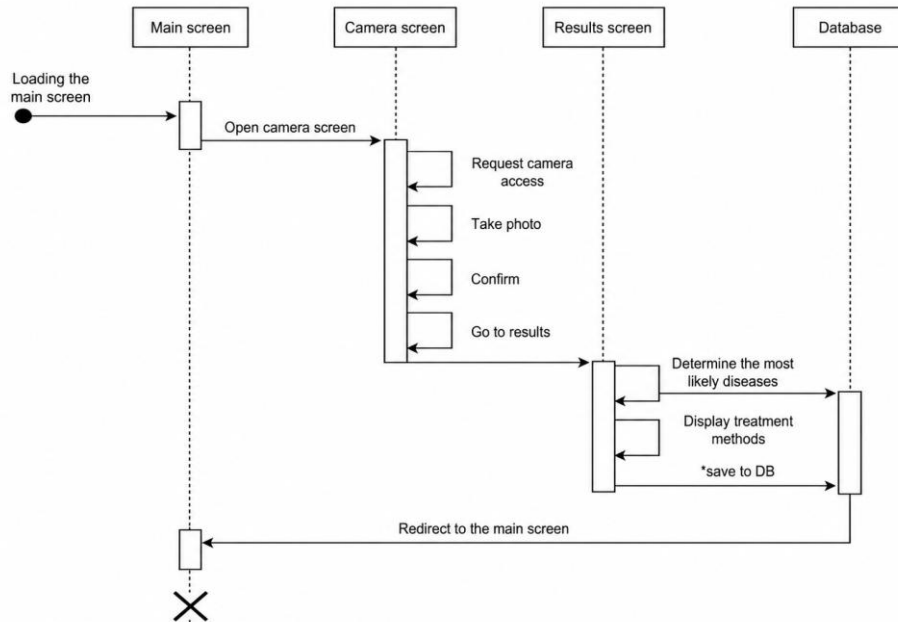


Figure 15: Image Diagnostic Sequence Diagram

This screen allows the user to save the diagnostic results in a database. Saving the results enables the recognition history feature. The sequence of interactions between the application screens and the local database during plant diagnosis is shown in Figure 15.

Working with the application begins with registration in the system, or authorization, which involves logging into an existing account, if one exists. This step is optional - you can skip it and use the program in guest mode. After successfully logging in (or choosing to remain in guest mode), the user is taken to the application's main page, which contains a brief description of the program and a "Diagnose" button, which redirects the user to the image selection page. The user can use the camera to take a photo of the plant they've just seen or select a photo from their smartphone's gallery. The appearance of the pages described above is shown in Figure 16.

The next step involves using a neural network model to analyze the image to identify the plant and its disease. The program integrates the `convnext_tiny_fp32_litert.tflite` model, which demonstrated the highest accuracy during training. Figure 18 shows how the program displays the most probable image class and its description to the user, followed by a list of the top five most probable classes to which the image may belong, along with the model confidence score of its membership in each class. In the case shown in Figure 18, the image was classified correctly - the model assigned the highest model confidence score to the correct class. If the correct class is ranked second to fifth among the top five most probable classes, the user can click on its name - the page will refresh and display a description and recommendations specifically for the selected class.

Optionally, the user can save the plant and its diagnosis by clicking the "Save to My Plants" button. Saving a plant is only available to registered users. The image is saved with the class assigned by the user (by default, this is the class with the highest probability).

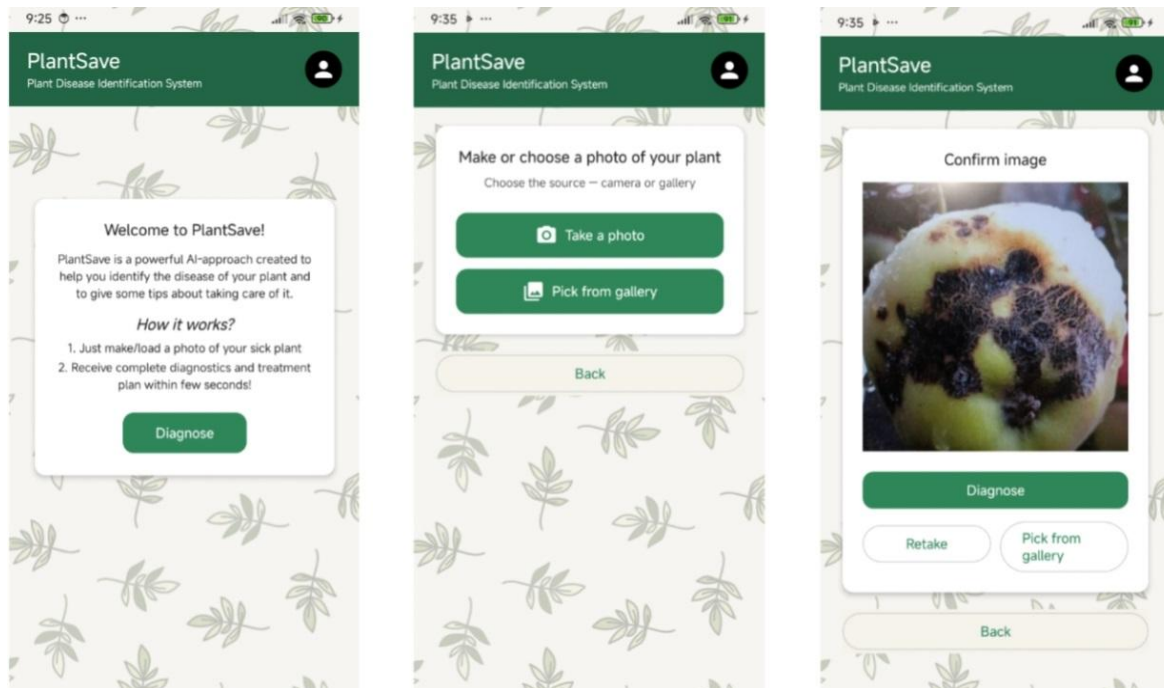


Figure 16: Application home page and image selection pages

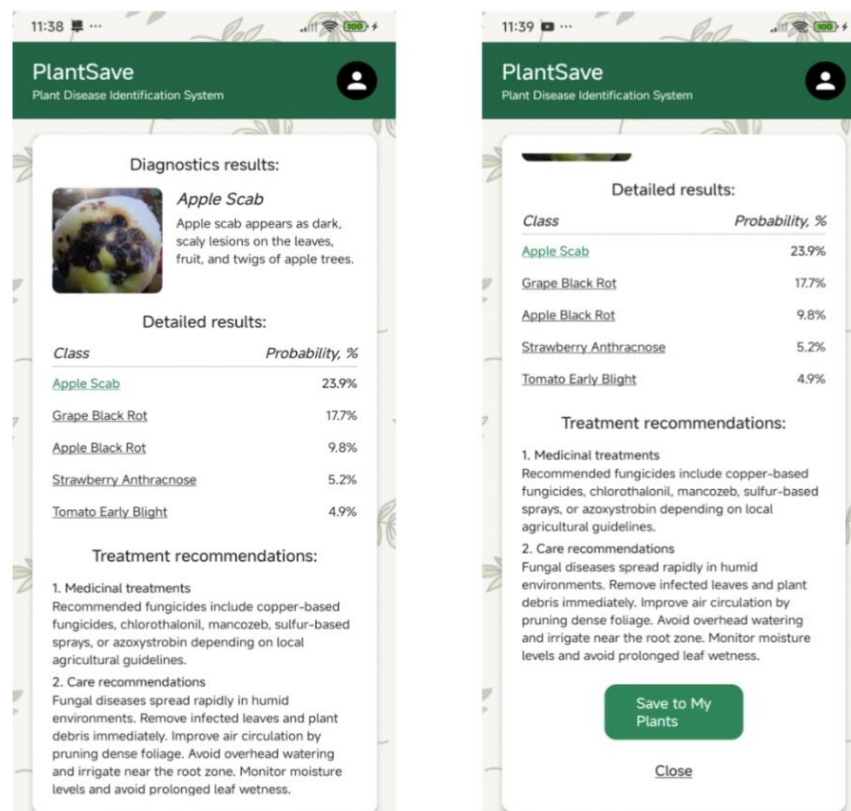


Figure 17: Diagnostic results page graphical interface

The My Plants page is shown in Figure 18. You can also see the user icon in the top right corner of the screen. Clicking it opens the user settings page, which allows you to change your username, email address, and password, access the My Plants page, log out, and delete your profile.

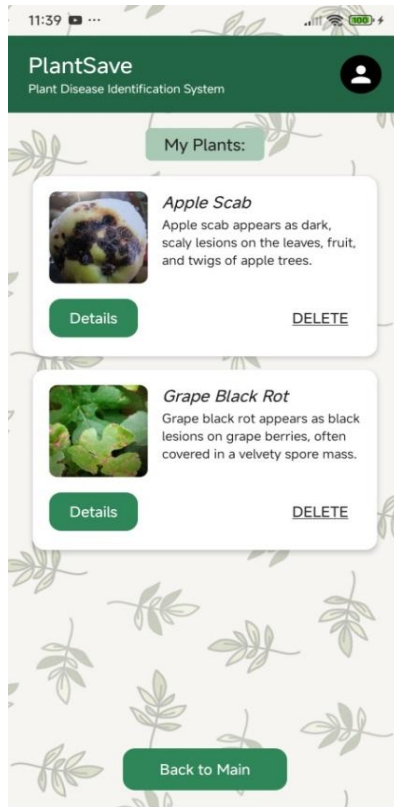


Figure 18: Graphical interface of the "My Plants" page

Analyzing the completed work, we can conclude that the application successfully fulfills its intended purpose, and its graphical interface is user-friendly and intuitive. To further analyze the mobile application's functionality, manual testing was conducted. This approach involved the tester manually executing interaction scenarios with the application, without the use of automation tools.

4. Conclusion

This study developed and evaluated an image-based plant-disease identification pipeline and integrated the best-performing model into a mobile application. The study reviewed several compact vision architectures. Four CNNs-EfficientNet-B0, MobileNetV3-Small, DenseNet-121, and ConvNeXt-Tiny-were trained and evaluated. ConvNeXt-Tiny achieved the highest PlantDoc top-1 accuracy of 62.65%. The project consisted of two phases: training neural network models and developing a mobile app. Models were trained on two different datasets: PlantVillage (laboratory photos) and PlantWild (field images). During the first training stage, all models achieved high accuracy on PlantVillage, whereas top-1 accuracy decreased substantially on PlantWild because of the domain shift and the greater visual complexity of in-the-wild images. The model that demonstrated the highest level of accuracy, ConvNeXt-Tiny, was integrated into the mobile app. The mobile application was developed in Kotlin using Android Studio, Jetpack Compose for the user interface, LiteRT (formerly TensorFlow Lite) for on-device inference, and Room for local data storage. These technologies helped develop an app characterized by low computational costs, ease of use, and high performance. In the future, the system can be improved by applying methods to increase model accuracy and by training the models on datasets containing more classes, allowing the application to identify a wider range of crops and diseases.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT to check citation consistency, verify bibliographic metadata, and improve grammar and wording. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

- [1] Chornobay L., Mushenyk I., Modern Risks of Agricultural Activity and Methods of Their Leveling, *Modern Economics* 42 (2023) 111–116. doi:10.31521/modecon.V42(2023)-16.
- [2] Agrios G.N., *Plant Pathology*, 5th ed., Academic Press, 2005. doi:10.1016/C2009-0-02037-6.
- [3] Tan M., Le Q.V., EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, *Proceedings of the 36th International Conference on Machine Learning* 97 (2019) 6105–6114. URL: <https://proceedings.mlr.press/v97/tan19a.html>.
- [4] Howard A., Sandler M., Chu G., Chen L.-C., Chen B., Tan M., Wang W., Zhu Y., Pang R., Vasudevan V., Le Q.V., Adam H., Searching for MobileNetV3, *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019) 1314–1324. URL: https://openaccess.thecvf.com/content_ICCV_2019/html/Howard_Searching_for_MobileNetV3_ICCV_2019_paper.html.
- [5] He K., Zhang X., Ren S., Sun J., Deep Residual Learning for Image Recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016) 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.
- [6] Huang G., Liu Z., van der Maaten L., Weinberger K.Q., Densely Connected Convolutional Networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017) 4700–4708. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.
- [7] Liu Z., Mao H., Wu C.-Y., Feichtenhofer C., Darrell T., Xie S., A ConvNet for the 2020s, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022). URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.pdf.
- [8] Mohanty S.P., Hughes D.P., Salathé M., Using Deep Learning for Image-Based Plant Disease Detection, *Frontiers in Plant Science* 7 (2016) 1419. doi:10.3389/fpls.2016.01419.
- [9] Wei T., Chen Z., Huang Z., Yu X., Benchmarking In-the-wild Multimodal Disease Recognition and A Versatile Baseline, *Proceedings of the 32nd ACM International Conference on Multimedia* (2024) 1593–1601. doi:10.1145/3664647.3680599.
- [10] Dong X., Zhao K., Wang Q., et al., PlantPAD: a platform for large-scale image phenomics analysis of disease in plant science, *Nucleic Acids Research* 52 (2024) D1556–D1568. doi:10.1093/nar/gkad917.
- [11] Wei T., Chen Z., Yu X., Chapman S., Melloy P., Huang Z., A Large-Scale In-the-wild Dataset for Plant Disease Segmentation, *Scientific Data* 13 (2026) 205. doi:10.1038/s41597-025-06513-4.

- [12] Singh D., Jain N., Jain P., Kayal P., Kumawat S., Batra N., PlantDoc: A Dataset for Visual Plant Disease Detection, Proceedings of the 7th ACM IKDD CoDS and 25th COMAD (2020) 249–253. doi:10.1145/3371158.3371196.
- [13] Fuentes A., Yoon S., Kim S., Park D., A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (2017) 2022. doi:10.3390/s17092022.
- [14] Kravchenko O.M., Kunup T.V., Analysis of deep research methods for identifying disease in plants based on photographs of leaves and fruits, Advanced Technologies in Scientific Research: material. IV Mizhnarod. scientific-practical Conf., 13–15 May 2026, Rotterdam, the Netherlands (2026) 121–126. URL: <https://isu-conference.com/en/archive/advanced-technologies-in-scientific-research-13-05-26>.
- [15] Kravchenko O.M., Kunup T.V., Development of a mobile device for detecting illness in agricultural crops, Problems and prospects for the implementation and promotion of interdisciplinary scientific achievements: material. XI Mizhnarod. Sci. Conf., 29 May 2026, Kamianets-Podilskyi, Ukraine (2026) 352–360. URL: <https://archives.mcnd.org.ua/index.php/conference-proceeding/issue/view/29.05.2026/96>.
- [16] Zhang Y., et al., ConvNeXt-based anchor-free object detection model for infrared image of power equipment, Energy Reports 9 (2023) 1121–1132. doi:10.1016/j.egy.2023.04.145..