International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI Assisted Grape Leaf Disease Detection

Keerthi M S, Meghana N K, Nesara N R, Pallavi N R

Department of Information Science and Engineering
Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India

Project Guide: **Mr. Sharath Kumar S R,** B.E., M.Tech., (Ph.D.)
Assistant Professor, Dept. of IS&E, JNNCE, Shivamogga

Abstract

Fungal diseases of grapevine leaves, black rot and esca (black measles) in particular, reduce yield and are often noticed late, because early lesions look alike to anyone without training. We built an image classifier for four leaf conditions (black measles, black rot, Isariopsis leaf spot and healthy) from 13,234 labelled photographs divided roughly 70/15/15 into training, validation and test sets. Three ImageNet-pretrained backbones were adapted to the task: EfficientNetB3 with a frozen base, ResNet50 trained in two stages with partial fine-tuning, and a VGG16 model with a small dense head. On the held-out test set EfficientNetB3 reached 98.69% accuracy (26 errors in 1,990 images) and ResNet50 reached 99.55% (9 errors in 1,989 images, macro F1 of 0.9955). Nearly all remaining mistakes involve black rot and black measles, and six of the nine ResNet50 errors were made with more than 99.8% confidence, which matters for anyone planning to deploy such a model. We also describe a web prototype that takes a leaf photograph and returns the predicted class, class probabilities and short management advice. Finally, we document a preprocessing mistake that held our first EfficientNetB3 run at 41.86% test accuracy, since it is an easy trap when these backbones are reused.

Index Terms—grape leaf disease, transfer learning, EfficientNet, ResNet50, VGG16, plant disease classification, web application

I. INTRODUCTION

A grapevine depends on its leaves for the sugar that ends up in the fruit. Leaves that are spotted, scorched or partly dead photosynthesise less, and several of the same pathogens go on to damage the fruit as well. Black rot, esca and Isariopsis leaf spot are among the more common leaf problems, and each of them has a rather similar first appearance: small brown or reddish lesions, sometimes with a yellow halo. Telling them apart in the field normally means a trained eye, and many small growers do not have one nearby.

Convolutional neural networks have been used for plant disease recognition for several years now. Models trained on the PlantVillage images reported accuracy above 99% on laboratory-style photographs [1], [2], and the idea has since been tried on most major crops. For grapes the picture is less settled. Published numbers are high, but they come from different splits, different image sizes and different amounts of tuning, so it is hard to say which backbone is a sensible starting point for a student project or a small startup that needs something working on a modest GPU.

This paper reports the classifier side of our final-year project. We trained three standard backbones on one fixed split of a four-class grape leaf dataset and then wrapped the trained model in a small web application. We do not claim a new architecture. What we offer is a careful, reproducible comparison, a candid look at where the models fail, and the practical details that are usually left out. The contributions are:

1) a like-for-like evaluation of EfficientNetB3 and ResNet50 on the same train, validation and test folders, with per-class metrics, confusion matrices and Wilson confidence intervals, plus a description of the VGG16 variant;
2) an error analysis showing that the remaining mistakes are concentrated in two classes and are frequently made with very high confidence;
3) a record of a preprocessing error (double input scaling) that dropped EfficientNetB3 to 41.86% test accuracy before we found it; and
4) a working web prototype with upload, prediction, class probabilities, short disease notes and a prediction log.

The rest of the paper is organised as follows. Section II reviews related work. Section III describes the dataset. Section IV explains the models and training procedure. Section V describes the prototype. Section VI presents the results, Section VII discusses limitations, and Section VIII concludes.

II. RELATED WORK

Early grape disease work relied on hand-crafted features. Typical pipelines segmented the leaf, picked out the diseased region by thresholding, computed colour and texture descriptors and passed them to an SVM or a random forest. These systems can work well on clean images but depend on segmentation quality, which breaks easily once the background is cluttered.

Deep learning removed most of that manual step. Mohanty et al. fine-tuned AlexNet and GoogLeNet on 54,306 PlantVillage images and reported 99.35% accuracy on heldout laboratory images, while also noting a steep fall when the same models saw images taken under different conditions [1]. Ferentinos compared several architectures on a much larger collection and reached 99.53% [3]. Too et al. found that finetuning a pretrained DenseNet gave better results than training from scratch or using other networks on the same task, with the added benefit of faster convergence [4]. For grapes specifically, Tockova et al. built a detection and classification service on Amazon SageMaker for black rot, black measles and leaf blight [5]. More recently, Grape Guard used YOLOv8 on a four-class public dataset, reported a mean average precision of 99.5%, and packaged the model as an Android application through TensorFlow Lite [6].

Three backbones keep appearing in this literature. VGG16 [7] stacks small 3×3 convolutions and is simple to adapt but heavy. ResNet [8] uses identity shortcuts so that much deeper networks can be trained. EfficientNet [9] scales depth, width and resolution together by a fixed rule and gets good accuracy from relatively few parameters. We picked one of each family because they are well supported in Keras, they all have ImageNet [10] weights, and they represent three quite different design choices. Where our work differs from the papers above is mostly in emphasis: we spend as much space on the failures and on deployment issues as on the headline accuracy.

III. DATASET

A. Composition and split

The data are a four-class collection of grape leaf photographs: black measles (esca), black rot, Isariopsis leaf spot (leaf blight) and healthy leaves. The class set matches the grape categories of PlantVillage [2]. The images came already divided into `train`, `val` and `test` folders. The EfficientNetB3 notebook counted 9,261, 1,983 and 1,990 images in the three folders, which is 13,234 in total and a split of 69.98%, 14.98% and 15.04%. The ResNet50 notebook found the same training and validation counts but 1,989 test images, one fewer in the black measles folder (568 instead of 569). We did not trace the difference, so each result below is reported with the test count of the notebook that produced it. Table I gives the per-class counts and Fig. 1 plots them.

TABLE I IMAGE COUNTS PER CLASS AND SPLIT

Class	Train	Val	Test	Total
Black measles	2,650	567	569*	3,786
Black rot	2,524	541	542	3,607
Healthy	1,826	391	393	2,610
Isariopsis leaf spot	2,261	484	486	3,231
Total	**9,261**	**1,983**	**1,990**	**13,234**

*568 in the ResNet50 notebook.

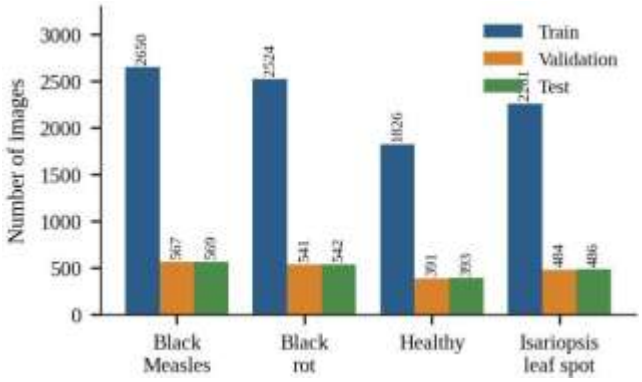


Fig. 1. Number of images per class in the training, validation and test folders.

The classes are moderately unbalanced. Healthy leaves are the smallest group (19.7% of the training set) and black measles the largest (28.6%), a ratio of about 1.45 to 1. We did not apply class weights or resampling, and the per-class results in Section VI suggest this was acceptable for a ratio this small.

B. Image characteristics

We opened a sample of 100 training images from each class to check sizes. Of those 400 files, 386 were 256×256 pixels; the rest were original photographs of various dimensions, the largest 3264×2448. Every image was resized to the input size of the model that consumed it. Most photographs show one leaf lying on a plain, grey-brown surface (Fig. 2), which makes the problem easier than field imaging, where soil, fruit, shadows and other leaves compete for attention. This matters for how far the reported accuracy can be trusted outside the dataset, and we return to it in Section VII.

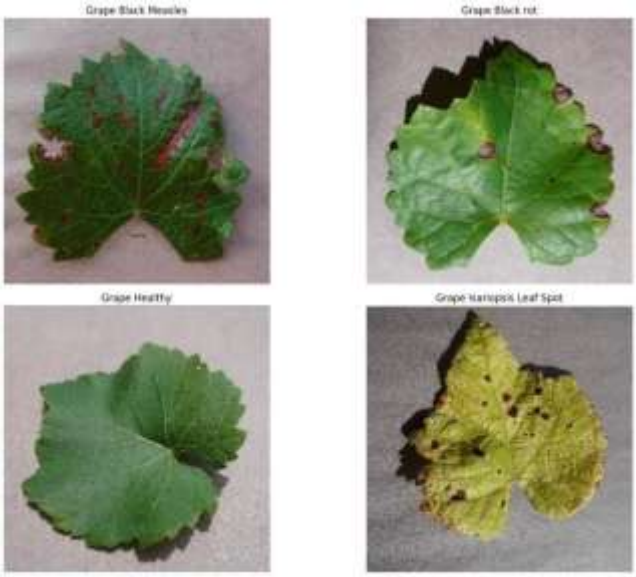


Fig. 2. One training image from each class: black measles, black rot, healthy and Isariopsis leaf spot.

IV. METHODOLOGY

A. Overall pipeline

Fig. 3 shows the workflow. All three models follow the same transfer-learning recipe [11]: start from ImageNet weights, remove the original 1,000-way classifier, attach a small new head for four classes, and train. They differ in input size, in how much of the backbone is allowed to change, and in the head. Training ran on a Colab GPU with TensorFlow 2.20 and Keras. Table II lists the main settings.

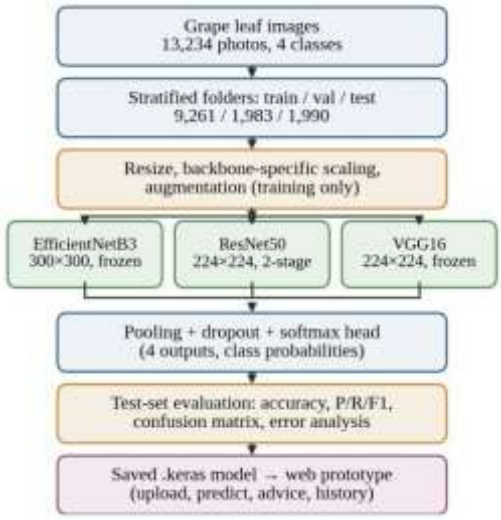


Fig. 3. End-to-end workflow, from the labelled folders to the saved model and the web prototype.

TABLE II TRAINING CONFIGURATION OF THE THREE MODELS

Setting	EffNetB3	ResNet50	VGG16
Input size	300×300	224×224	224×224
Base weights	ImageNet	ImageNet	ImageNet
Base trainable	No	Stage 2: last 35 layers†	No (13 conv)
Head	GAP, Drop 0.3, FC 4	GAP, BN, Drop 0.4, FC 256, Drop 0.3, FC 4	Flatten, FC 256, Drop 0.5, FC 4
Optimiser	Adam, 1e-3	AdamW, 1e-4 then 1e-5	Adam, 1e-4
Loss	Categorical CE	Sparse categorical CE	Categorical CE
Batch size	32	32	–
Max epochs	15	20 + 10	–
Trainable params	6,148	0.53 M (stage 1)	6.42 M
Total params	10.79 M	24.12 M	21.14 M

†Batch-norm layers stayed frozen, leaving 24 trainable layers. –: not recoverable from the saved file.

B. Preprocessing and augmentation

Validation and test images were never augmented. For EfficientNetB3 the generator resized images to 300×300 and passed raw pixel values in the 0 to 255 range, because the Keras implementation contains its own normalisation layers. Training images received random rotation up to 20°, horizontal flips, shifts and zoom of 10%, and shear of 10%. For ResNet50 we resized to 224×224 and applied the network's own `preprocess_input` (channel-wise mean subtraction); training-time augmentation was a horizontal flip, rotation by up to 8% of a full turn, 10% zoom and 10% contrast change, built from Keras layers inside the `tf.data` pipeline. The VGG16 model takes 224×224 input.

C. EfficientNetB3

EfficientNetB3 [9] has 10,783,535 parameters in its convolutional base. We froze all of them and trained only the head: global average pooling, dropout at 0.3, and a four-way softmax layer, which adds 6,148 trainable parameters. We used Adam [12] with a learning rate of 1e-3 and categorical cross-entropy for up to 15 epochs, with checkpointing on validation accuracy and early stopping with a patience of four epochs. Freezing the whole base is the cheapest option and it makes each epoch fast, but it also means the features cannot adapt to leaf textures. We kept that limitation in mind when comparing it with ResNet50.

D. ResNet50

ResNet50 [8] was trained in two stages. In stage one the 23.6 M parameter base was frozen and a head of global average pooling, batch normalisation, dropout (0.4), a 256unit ReLU layer with L2 penalty 1e-4, dropout (0.3) and a softmax layer was trained with AdamW [13] at 1e-4 and weight decay 1e-4. Training stopped by itself after nine epochs, because validation loss stopped improving. The learning rate was cut by a factor of five whenever validation loss stalled for two epochs, and the best weights were restored.

In stage two we unfroze the last part of the backbone: the first 140 of 175 layers stayed frozen, and batch-norm layers were kept frozen everywhere, which left 24 trainable layers inside the base. The learning rate dropped to 1e-5 with weight decay 1e-5, and we ran ten more epochs with the same callbacks. Batch normalisation [14] keeps running statistics that small batches can easily disturb, so freezing it is the usual precaution when fine-tuning with a small learning rate.

E. VGG16

The VGG16 variant [7] keeps all thirteen convolutional layers frozen (14.71 M parameters). On top of them sit a flatten layer, a 256-unit ReLU dense layer, dropout at 0.5 and a four-way softmax. The dense part alone has 6.42 M trainable parameters because flattening the 7×7×512 feature map yields a 25,088-dimensional vector. The saved model was compiled with Adam at a learning rate of 1e-4 and categorical crossentropy. The file we hold is a checkpoint of a continued training run; the epoch-level log of that run was not part of the material we analysed, so we describe its configuration but do not quote test figures for it in this paper.

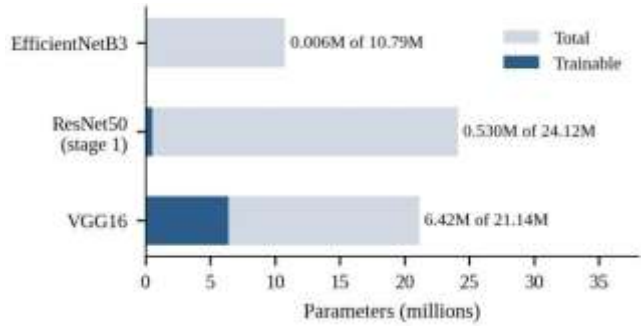


Fig. 4. Trainable versus total parameters for the three models as configured. Only a small slice of each network is trained.

F. Evaluation metrics

Test accuracy is the share of images classified correctly. For each class we report precision, recall and F1, with TP, FP and FN counted one-versus-rest:

$$P = TP/(TP+FP),\ R = TP/(TP+FN),\ F1 = 2PR/(P+R) \quad (1)$$

Macro averages weight the four classes equally. Because accuracy on a finite test set carries sampling noise, we also give 95% Wilson score intervals for the accuracy of each model.

V. WEB PROTOTYPE

To check that the model is usable by someone who has never opened a notebook, we put it behind a browser interface. The version built inside the ResNet50 notebook is a small Flask application. A user picks a JPG or PNG file; the server checks the extension, resizes the image, applies the same preprocessing as in training, runs one forward pass and returns the predicted class, its confidence and the probability for every class. A lookup table adds a short description of the predicted disease and a generic recommendation, for example to remove badly affected plant material and keep the vineyard clean. Each prediction is appended to a CSV file with the time, file name, class and confidence, and a separate page shows the history. A dashboard page presents the model's evaluation figures. The same upload-and-predict workflow is hosted publicly on Streamlit Community Cloud at `grape-leafdiseasedetection.streamlit.app`. Fig. 5 shows the request path.

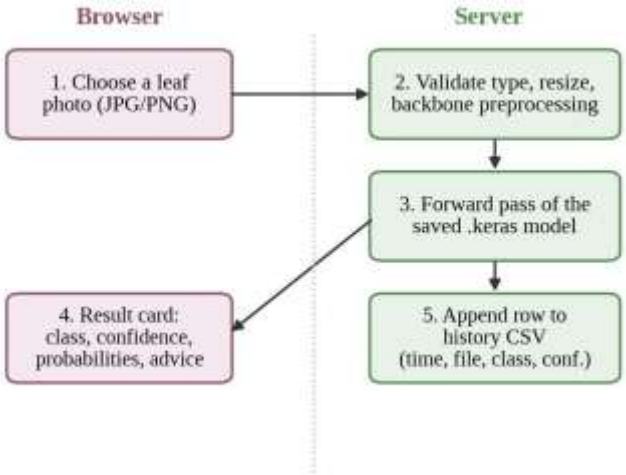


Fig. 5. Request path of the prototype: the browser sends an image, the server returns the prediction and stores a history row.

Two design points are worth recording. First, the classname list is saved next to the model as a small JSON file, so that the label order used in training cannot drift from the order used in the application. Second, the advice text is deliberately generic. It is meant to prompt a grower to look more closely or ask an advisor, not to replace a diagnosis; a spray programme depends on the cultivar, the weather and local regulations, none of which the model can see.

VI. RESULTS

A. Training behaviour

Fig. 6 shows the EfficientNetB3 run. Validation accuracy was already 96.47% after the first epoch and ended at 98.89% after the fifteenth, with validation loss falling from 0.186 to 0.049. Training accuracy stayed a little below validation accuracy for most of the run, which is normal when dropout and augmentation are active during training only. The curves had not completely flattened when the epoch limit was reached, so a longer run, or unfreezing part of the base, would probably gain a little more.

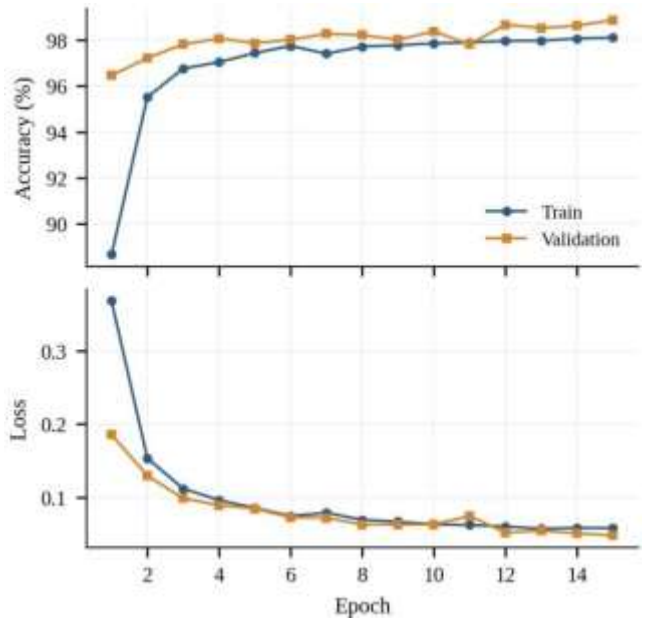


Fig. 6. EfficientNetB3 training and validation accuracy (top) and loss (bottom), 15 epochs.

ResNet50 (Fig. 7) started higher: 98.23% validation accuracy after one epoch of stage one and peaked at 99.09% before early stopping ended the stage. Fine-tuning in stage two lifted validation accuracy to 99.75% at its best epoch, although the validation loss moved up and down by a few thousandths between epochs, which we read as ordinary noise from a validation set of 1,983 images rather than a sign of overfitting. Epochs took about 140 s for ResNet50 and about 250 s for EfficientNetB3 on Colab GPUs, partly because of the larger input.

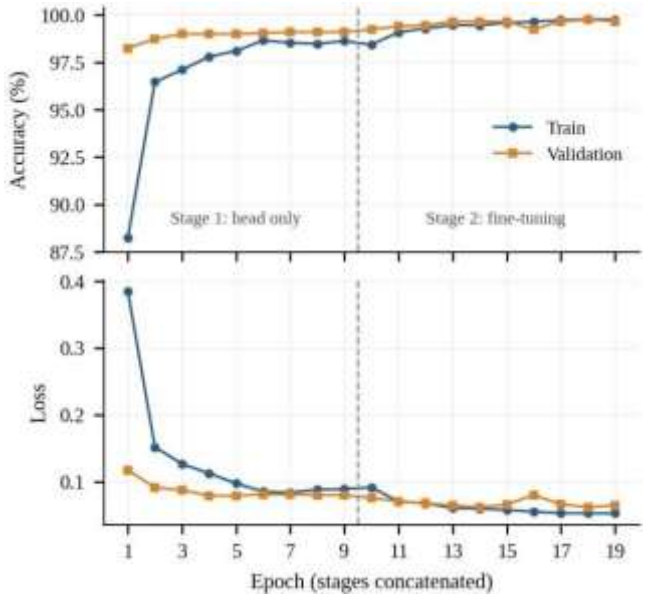


Fig. 7. ResNet50 training and validation curves; the dashed line marks the switch from head-only training to fine-tuning.

B. Test-set performance

Table III gives the headline numbers on the test folders, and Table IV the per-class precision, recall and F1. EfficientNetB3 scored 98.69% (1,964 of 1,990 correct; 95% interval 98.09% to 99.11%). ResNet50 scored 99.55% (1,980 of 1,989; 99.14% to 99.76%). The intervals only just fail to overlap, and the test sets differ by one image, so we describe ResNet50 as better in this experiment without claiming a large margin.

TABLE III TEST-SET RESULTS

Model	Test imgs	Errors	Accuracy	Macro F1	Test loss
EfficientNetB3	1,990	26	98.69%	0.9881	0.0485
ResNet50	1,989	9	99.55%	0.9955	0.0829
VGG16‡	—	—	—	—	—

‡Checkpoint described in Section IV-E; test metrics not included in the logs we analysed.

The test loss is lower for EfficientNetB3 even though its accuracy is lower. The explanation is probably the regularisation: ResNet50's loss includes an L2 term and the two networks use different heads, so the two loss values are not on the same footing and we do not compare them directly.

TABLE IV PER-CLASS PRECISION, RECALL AND F1 ON THE TEST SET

Class	Model	P	R	F1
Black measles	EffNetB3	0.9789	0.9807	0.9798
	ResNet50	0.9965	0.9965	0.9965
Black rot	EffNetB3	0.9778	0.9742	0.9760
	ResNet50	0.9944	0.9889	0.9917
Healthy	EffNetB3	0.9975	1.0000	0.9987
	ResNet50	0.9899	1.0000	0.9949
Isariopsis spot	EffNetB3	0.9979	0.9979	0.9979
	ResNet50	1.0000	0.9979	0.9990

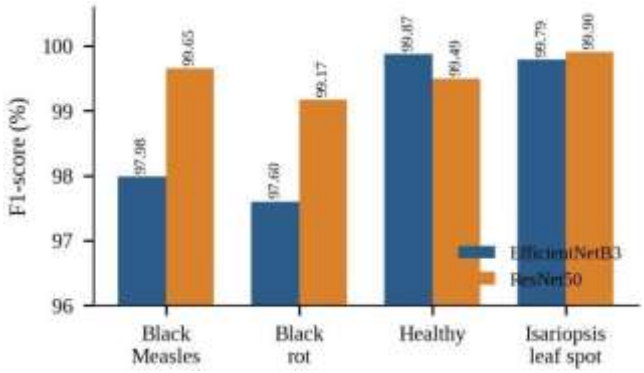


Fig. 8. Per-class F1-score of the two evaluated models.

C. Confusion matrices

Fig. 9 shows where the errors fall. Healthy leaves were never mistaken for a disease by either network: all 393 were

recognised, so recall for that class is 1.000 in both cases. Almost all of EfficientNetB3's 26 errors are exchanges between black measles and black rot (11 and 12 images), and the other three involve one image each of other pairs. ResNet50 makes fewer such exchanges (2 and 2) but has a different weakness: four black rot images were called healthy, which is the most costly error type for a grower because a diseased leaf passes as clean. That pattern also explains why ResNet50's healthy-class precision (0.9899) is lower than EfficientNetB3's (0.9975).

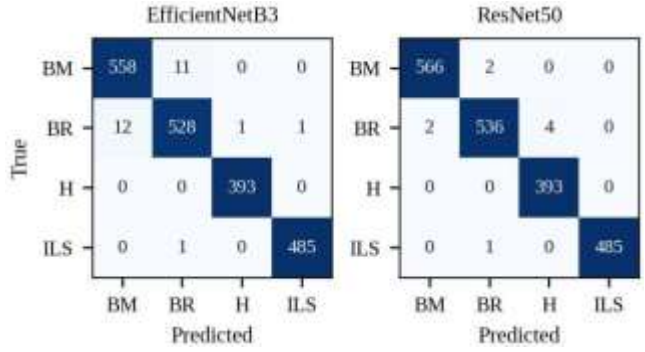


Fig. 9. Confusion matrices on the test set. BM: black measles, BR: black rot, H: healthy, ILS: Isariopsis leaf spot.

D. How confident are the mistakes?

We listed every ResNet50 test error together with the probability the network gave to its wrong answer (Fig. 10). Six of the nine had confidence of 99.89% or higher, and three of those were exactly 100.00% to two decimals. Only three errors came with a confidence under 70%. A simple rule such as "ask for a second photograph if the top probability is below 90%" would therefore catch about a third of the mistakes at best, and the rest would pass unnoticed. Softmax scores from a network trained with cross-entropy are known to be overconfident, and our results are consistent with that.

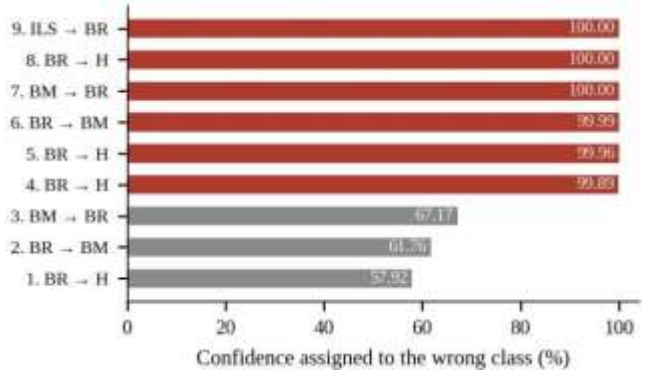


Fig. 10. Confidence given to the wrong class for each of the nine ResNet50 test errors (true class → predicted class). Red bars exceed 99%.

E. Qualitative examples

Fig. 11 shows five black measles leaves from the training folder that ResNet50 classified correctly, each with confidence reported as 100.00%. The lesions are dark red to brown patches with irregular edges, quite different from the small round spots of black rot. Fig. 12 shows two EfficientNetB3 outputs: a black rot leaf identified at 98.97% confidence, and an Isariopsis leaf spot image predicted as black rot at 99.50%. The second image has a black background, unlike most of the dataset, and the lesions on it are brown and round, which resembles black rot.



Fig. 11. ResNet50 predictions on five black measles leaves from the training folder.



Fig. 12. EfficientNetB3: a correct black rot prediction (left) and a misclassified Isariopsis leaf spot image (right).

F. Photographs from outside the dataset

We also tried a few pictures that were not part of the dataset, and these were much less comforting. A web image of a diseased leaf, passed through EfficientNetB3, produced black rot with 49.37%, black measles with 40.27% and Isariopsis leaf spot with 10.19%, which is about as unsure as a classifier can be. ResNet50 behaved worse on a field photograph with grape bunches and other leaves in the background (Fig. 13). It answered black rot with 99.9998% probability. A feature-similarity check using cosine similarity on the pooled features agreed with that choice only weakly (0.64 for black rot against 0.50 to 0.53 for the other classes), so the probability was much more certain than the underlying evidence. We read this as a domain-shift problem and not as proof that either network is poor, but it is the most important caveat in this paper.

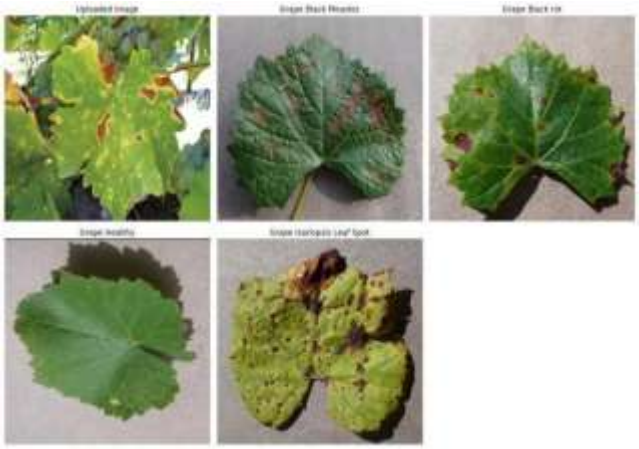


Fig. 13. A cluttered field photograph (top left) alongside one training example per class. ResNet50 labelled the field image black rot with neartotal confidence.

G. A preprocessing mistake

Our first EfficientNetB3 run scored 41.86% on the test set, only a little above the 28.6% that always guessing the largest class would give. Fig. 14 shows that the network never predicted the healthy class. In that run the data generator divided every pixel by 255 before the images reached the model, but the Keras EfficientNet implementation already rescales its input internally, so the values were scaled twice and the pretrained filters saw an almost flat image. Removing the extra division and keeping the rest of the pipeline gave the 98.69% reported above. We have not run a controlled ablation to prove the cause, but the repair worked at once and the symptom, a model that can hardly tell classes apart, is typical of inputs outside the range the weights expect. The lesson is to check the preprocessing contract of each backbone separately; ResNet50 and EfficientNet need different input ranges even though both are called through the same Keras API.

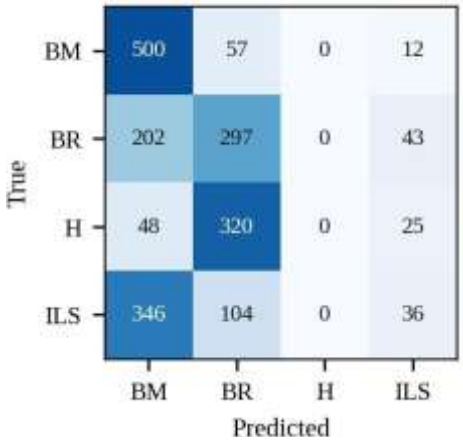


Fig. 14. Confusion matrix of the failed first EfficientNetB3 run (41.86% test accuracy). The healthy column is empty.

VII. DISCUSSION AND LIMITATIONS

The comparison is not a pure architecture contest. ResNet50 was fine-tuned in a second stage and EfficientNetB3 was not, the input sizes differ, and the heads differ. Part of ResNet50's lead may come from the extra finetuning and not from the backbone. We report what we ran; a fair architecture comparison would give each model the same two-stage recipe and several random seeds.

We see four other limitations. First, each result comes from a single training run on a single split, so we cannot say how much of the gap between the two models is seed noise. Second, the number of images is larger than we would expect from the original PlantVillage grape subset, which has roughly four thousand pictures. If the collection contains offline augmented copies of the same photographs, near-duplicates may sit in both the training and test folders, and that would inflate every score in this paper. We did not run a duplicate check, and anyone reusing these numbers should do so. Third, almost all images are single leaves on plain backgrounds, and the small experiments in Section VI-F show that performance outside this setting is unreliable. Fourth, we cover four classes only; a real vineyard also has downy and powdery mildew, nutrient deficiency, sunburn and insect damage, none of which the model can say “I do not know” about.

For deployment, the confidence analysis suggests three measures: reject low-confidence predictions, calibrate the probabilities (temperature scaling is cheap and effective [15]), and collect field photographs for retraining. Until those are in place the prototype should be treated as a screening aid.

VIII. CONCLUSION AND FUTURE WORK

We compared transfer-learning classifiers for grape leaf disease on a fixed four-class dataset and built a web prototype around the trained model. ResNet50 with two-stage finetuning reached 99.55% test accuracy and EfficientNetB3 with a frozen base reached 98.69%. In both cases the hard part is separating black rot from black measles, and ResNet50 occasionally calls black rot leaves healthy. Confident mistakes and poor behaviour on cluttered photographs show that dataset accuracy alone is a weak indicator of field readiness.

Next we plan to evaluate the VGG16 checkpoint on the same test folders and add it to Table III, fine-tune EfficientNetB3 with the same two-stage recipe, remove duplicates from the split, and collect field images from local vineyards. On the application side we want Grad-CAM maps [16] so users can see which part of the leaf drove a decision, a reject option for uncertain cases, and a TensorFlow Lite build for use on a phone without a network connection.

REFERENCES

[1] "Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG," *Artificial Intelligence in Agriculture*, vol. 6, pp. 23–33, 2022, doi: 10.1016/j.aiia.2021.12.002.

[2] I. Kunduracioglu and I. Pacal, "Advancements in deep learning for accurate classification of grape leaves and diagnosis of grape diseases," *Journal of Plant Diseases and Protection*, vol. 131, pp. 1061–1080, 2024, doi: 10.1007/s41348-024-00896-z.

[3] "GAPNet: Single and multiplant leaf disease classification method based on simplified SqueezeNet for grape, apple and potato plants," *PeerJ Computer Science*, 2025, doi: 10.7717/peerj-cs.2941.

[4] "Comparative Analysis of Transfer Learning Models for the Detection and Classification of Grape Leaf Diseases," in *Proc. 8th International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2024, doi: 10.1109/PDGC64653.2024.10984031.

[5] "Real-Time Automatic Detection of Grape Leaf Diseases with MobileNetV2 Deep Learning Network Based on Artificial Intelligence," in *Proc. Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, 2025, doi: 10.1109/ACCTHPA65749.2025.11168579.

[6] X. Xie, Y. Ma, B. Liu, J. He, S. Li, and H. Wang, "A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks," *Frontiers in Plant Science*, vol. 11, Art. no. 751751, 2020.

[7] R. Dwivedi, S. Dey, C. Chakraborty, and S. Tiwari, "Grape Disease Detection Network Based on Multi-Task Learning and Attention Features," *IEEE Sensors Journal*, vol. 21, no. 16, pp. 17573–17580, 2021.

[8] S. Mousavi and G. Farahani, "A Novel Enhanced VGG16 Model to Tackle Grapevine Leaves Diseases With Automatic Method," *IEEE Access*, vol. 10, pp. 113245–113260, 2022.

[9] R. Karthik, R. Menaka, S. Ompirakash, P. Bala Murugan, M. Meenakashi, S. Lingaswamy, and D. Won, "GrapeLeafNet: A Dual-Track Feature Fusion Network With Inception-ResNet and Shuffle-Transformer for Accurate Grape Leaf Disease Identification," *IEEE Access*, vol. 12, pp. 14567–14580, 2024.

[10] L. Falaschetti, L. Manoni, R. C. F. Rivera, D. Pau, G. Romanazzi, O. Silvestroni, V. Tomaselli, and C. Turchetti, "A Low-Cost, Low-Power and Image Detector for Grape Leaf Esca Disease Based on a CNN," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 3, pp. 468–480, 2021.

[11] K. M. Hosny, W. M. El-Hady, F. M. Samy, E. Vrochidou, and G. A. Papakostas, "Multi-Class Classification of Plant Leaf Diseases Using Feature Fusion of Deep Convolutional Neural Network and Local Binary Pattern," *IEEE Access*, vol. 11, pp. 65432–65445, 2023.

[12] M. Fraiwan, E. Faouri, and N. Khasawneh, "Multiclass Classification of Grape Diseases Using Deep Artificial Intelligence," *Agriculture*, vol. 12, no. 10, Art. no. 1542, 2022.

[13] B. Liu, C. Tan, S. Li, J. He, and H. Wang, "A Data Augmentation Method Based on Generative Adversarial Networks for Grape Leaf Disease Identification," *IEEE Access*, vol. 8, pp. 102188–102198, 2020.

[14] A. Adeel, M. A. Khan, M. Sharif, F. Azam, J. H. Shah, T. Umer, and S. Wan, "Diagnosis and Recognition of Grape Leaf Diseases: An Automated System Based on a Novel Saliency Approach and Canonical Correlation Analysis Based Multiple Features Fusion," *Sustainable Computing: Informatics and Systems*, vol. 24, Art. no. 100349, 2019.

[15] P. P. Kerkar, T. Veerakumar, A. D. Rahulkar, M. K. Panda, and B. N. Subudhi, "A Dual-Net With Graph Attention Network for Plant Diseases Identification," *IEEE Sensors Journal*, vol. 25, no. 19, pp. 37075–37090, 2025.

[16] A. Adeel, M. A. Khan, M. Sharif, F. Azam, J. H. Shah, T. Umer, and S. Wan, "Diagnosis and Recognition of Grape Leaf Diseases: An Automated System Based on a Novel Saliency Approach and Canonical Correlation Analysis Based Multiple Features Fusion," *Sustainable Computing: Informatics and Systems*, vol. 24, Art. no. 100349, 2019.

[17] V. Balafas, E. Karantoumanis, M. Louta, and N. Ploskas, "Machine Learning and Deep Learning for Plant Disease Classification and Detection," *IEEE Access*, vol. 11, pp. 114352–114377, 2023.

[18] D. J. Richter, M. I. Bappi, S. S. Kolekar, and K. Kim, "A Systematic Review of the Current State of Transfer Learning Accelerated CNN-Based Plant Leaf Disease Classification," *IEEE Access*, vol. 13, pp. 116262–116303, 2025.

[19] A. S. Al-Shamayleh and R. W. Ibrahim, "Grapevine Disease Detection Using (q, α)-Nabla Calculus Quantum Deformation With Deep Learning Features," *MethodsX*, vol. 15, Art. no. 103619, 2025.

[20] C. Zhou, Z. Zhang, S. Zhou, J. Xing, Q. Wu, and J. Song, "Grape Leaf Spot Identification Under Limited Samples by Fine-Grained GAN," *IEEE Access*, vol. 9, pp. 102783–102794, 2021.