Leaf Disease Classification Exploration of Contemporary Models

Rotem Adar 323874511, Rotemcomp@gmail.com and Nikita Nedostoup 336495593 ,
nikolos7771@gmail.com
Moussaffi John Meir 322923244, johnhmm2001@gmail.com

Bar-Ilan University, Department of Mathematics

1 Introduction

The goal of this project is multi-class classification on a leaf disease dataset, where each RGB image is hierarchically labeled by split (train, test, val), leaf type, and disease. https://github.com/JohnLemonAid/LeafDiseaseClassification

Figure 1: Examples of plants with their disease

Our goal here was to test the performance of a fine-tuned resnet18 model, which is a relatively simple model, with 11.7 million parameters, in comparison to resnet50 (25.6m), or DeiT-small (21.7m). We then compare ResNet18's performance to that of 4 other, more complex CNN models (ResNet-50, EfficientNet-B2, ConvNeXt-Tiny, DeiT-Small, along with Random Forest, just to see how the classical algorithm we've learned about performs in 'the big 2025'. The evaluation metrics we used were accuracy, F1, and ROC-AUC, along with runtime and parameter size.

2 Dataset & Data Analysis

2.1 Dataset and Distributions

Data source: The leaf disease dataset (PlantVillage-style), RGB leaf images organized by hierarchical folders seemed to have been an amalgamation of 3 merged datasets, giving rise to the problem of duplicates, which may

cause leakage. The following graphs show the 'plant vs disease' heatmap, along with distributions of 'Plants vs Diseases' in the data, scaled by $log(1+n)$, where n in the number of plants with the specific disease.

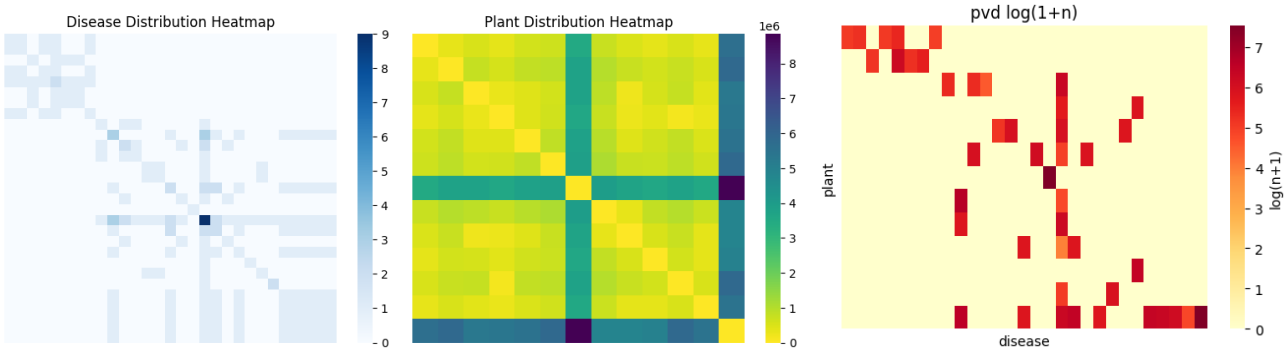


Figure 2: Full graphs in Addendum

We can see that the heatmap matrices are relatively sparse.

2.2 Preprocessing, Re-splitting and Normalization

The dataset included a total of 53,303 images, categorized by the plant (leaf type), and its state (type of disease or healthy). We removed the folders of the 'orange' and 'squash' plant, since they didn't have any 'healthy' pictures. We also scanned the dataset using ImageHash, and removed a total of 12,135 pictures (first copy was saved).

After the cleaning, a disproportionate amount was left in the train folder, so we reassembled the train, val, and test folders in the standard ratio of train 80%, test 10%, and val 10%. Also, we made Individual Standardization Normalization. This type of normalization normalized each photo by individual characteristics. It helps standardize each photo to the same spectral characteristics. That will help clearly see specific patterns on the photo and recognize them after.

3 Training

3.1 Models

ResNet-18 (baseline). We chose ResNet-18 as a baseline: small (11.7M params), fast on a T4, and reliably strong when fine-tuned from ImageNet. It also serves as a common baseline in public Kaggle notebooks on plant disease, so results are easy to compare.
Other models. ResNet-50 (deeper residual CNN), EfficientNet-B2 (accuracy/efficiency trade-off), ConvNeXt-Tiny (modern ConvNet), DeiT-Small (ViT), and a Random Forest on frozen CNN features.

3.2 Loss and metrics

We trained with multiclass cross-entropy, with class weights to reduce imbalance:

$$L_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(y_i \mid x_i), \quad p_\theta = \text{softmax}(f_\theta(x)).$$

We also tested a Houdini-style loss — this function simultaneously uses stochastic probabilities and standard distance metrics or gradients:

$$L_{\text{Houdini}}(x,y) \;=\; P\big(s_y(x) - \max_{k\neq y} s_k(x) > \tau\big)\cdot \ell(x,y),$$

It helps to evaluate better both distance factors and predictions, and gives better weights also in difficult cases and even in black boxes. (In this case, l(x,y) = 1)

Metrics: top-1 accuracy, macro-F1 (unweighted mean over classes), and validation loss (ROC-AUC for summaries). The best checkpoint is selected by validation accuracy (ties by lower loss).

Metric definitions Precision $= \frac{TP}{TP+FP}$ (accuracy of positive predictions); Recall $= \frac{TP}{TP+FN}$ (sensitivity); $F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$. $F1_{\text{macro}}$ is the unweighted mean of classwise F1; $F1_{\text{weighted}}$ averages F1 with class-frequency weights. We select the best checkpoint by validation accuracy (ties by lower loss) and report accuracy, macro-F1, and validation loss (ROC–AUC is optional).

3.3 Loss Competition

We ran two loss setups on a 20% subset for 4 epochs: (A) a Houdini-style margin loss and (B) cross-entropy with class weights. Cross-entropy measures the fit between the true distribution p and the model prediction q:

Plant recognition (This happened because plant classification in our dataset is an easier task.) The model shows very high accuracy on both training and validation with both loss functions. That is a good sign that we don't have overfitting (we also verified that there is no data leakage). The loss is very low, which means the model detects patterns well.

The F1 score is close to 1, which shows we have a very good balance between precision and recall. This means the model works well with both rare and common classes. In the case of CrossEntropyLoss + ClassWeights, our ClassWeights were sufficient to fix the imbalance that we observed in Random Forest and Logistic Regression.

Disease recognition All results are lower and worse, but still not bad with both models. Houdini achieved a better loss, but CE+W gave a better F1 score.

3.4 Technology:

Google Colab; NVIDIA T4 GPU (16 GB VRAM, 15 GB usable) with high-RAM host (51 GB). Python 3.12; PyTorch 2.x, torchvision, scikit-learn; TensorFlow only for tf.data EDA; CUDA 12.x + cuDNN. Training: 224×224 inputs, batch 32, AdamW with cosine decay, up to 15 epochs with early stopping, class-weighted cross-entropy, fp32/AMP. Reproducibility: fixed class indices, saved best validation checkpoint, set random seed, logged package versions.

3.5 Augmentations and regularization

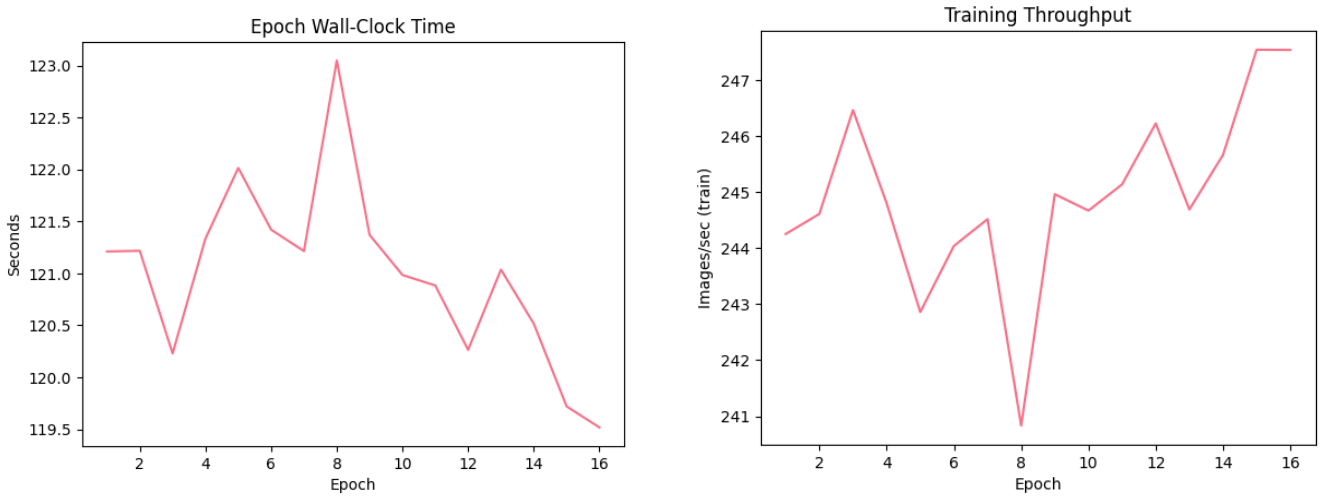
Inputs were resized/cropped to 224×224 and normalized to ImageNet statistics. We used mild augmentations (random resized crop, flips, $\pm 15^\circ$ rotation, light color jitter). To limit overfitting we applied early stopping, AdamW with weight decay and a cosine schedule, dropout in the head, DropPath (stochastic depth), and gradient clipping.

3.6 Training protocol

- **Full-data baseline:** fine-tune ResNet-18 for up to 15 epochs; pick best checkpoint by `val` accuracy; report once on `test`.
- **Short-budget sweep:** 20% stratified subset; identical pipeline; 5–7 epochs per model for quick comparison.

3.7 Optimization and regularization

We used learning-rate warm restarts to stabilize early epochs and help escape poor minima, increased epochs/patience, applied gradient clipping to prevent rare gradient spikes, and used DropPath (stochastic depth) to reduce reliance on specific paths. Mild augmentations (resize/crop to 224×224, flips, $\pm 15^\circ$ rotation, light color jitter) and AdamW with weight decay were applied throughout.



(a) Epoch wall-clock time.

(b) Training throughput (images/s).

Figure 3: Runtime characteristics on T4.

4 Model Competition

4.1 Plant recognition

Plant classification is easy in this dataset: we observed very high accuracy on train/val and low loss with both losses, with no signs of leakage (checked post-cleaning). Macro-F1 is close to 1, indicating a good precision–recall balance across both rare and common classes. With cross-entropy + class weights, imbalance effects seen with Random Forest and Logistic Regression were mitigated.

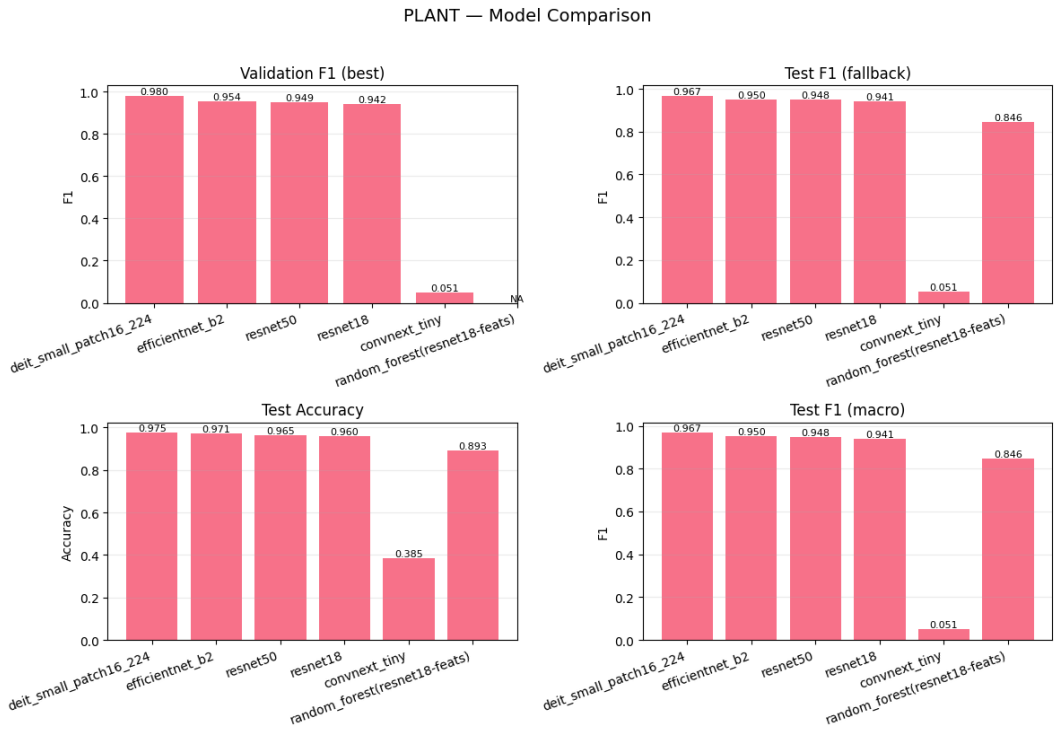


Figure 4: Caption

4.2 Disease recognition

Performance is lower than plants but still solid across models. Houdini achieved lower loss (better separation), while cross-entropy + class weights achieved higher macro-F1 (e.g., 0.867 vs. 0.844), which we prioritize for practical usefulness.

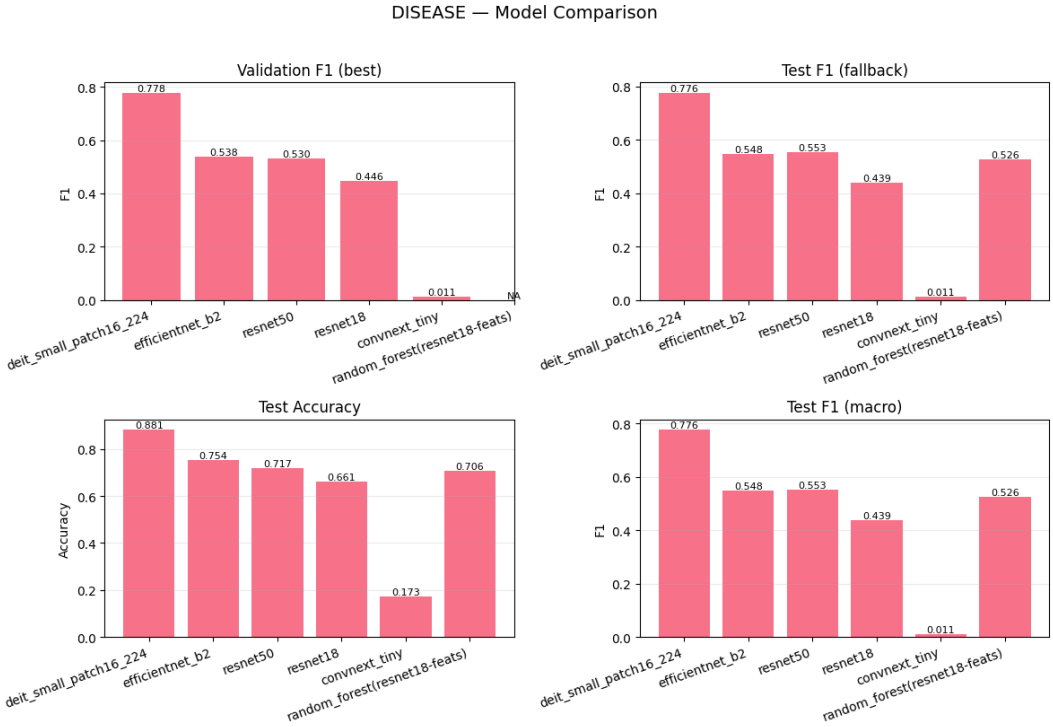


Figure 5: Caption

5 Best model

5.1 Selection and tuning

After the model competition, we chose DeiT-Small as the best model: it delivered the strongest accuracy with the most confident indicators. We also decided to combine two losses, since Houdini produced a lower validation loss (better separation), while cross-entropy with class weights achieved a higher macro-F1 (e.g., 0.867 vs. 0.844), which is more useful in practice. Concretely, we used a linear combination

$$L_{\text{final}} \;=\; (1-\lambda)\, L_{\text{CE}} \;+\; \lambda\, L_{\text{Houdini}},$$

with λ selected on the validation set.

Tuning steps:

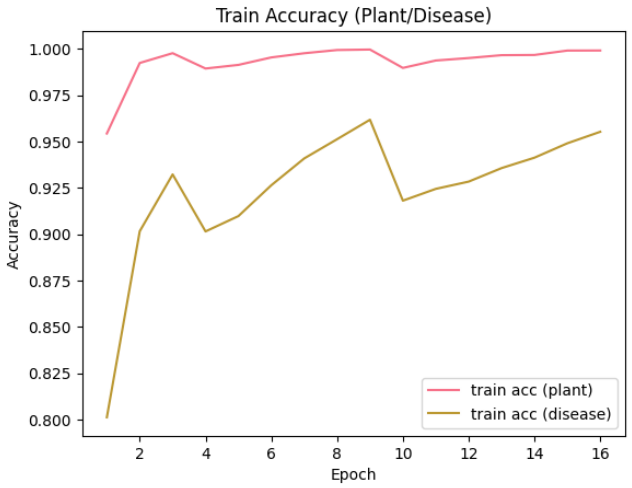
- Learning-rate warm restarts to stabilize early epochs and escape poor minima.
- Increased epochs and patience to allow more improvement.
- Gradient clipping to prevent rare gradient spikes.
- DropPath (stochastic depth) to reduce reliance on any single path and curb overfitting.

5.2 Testing

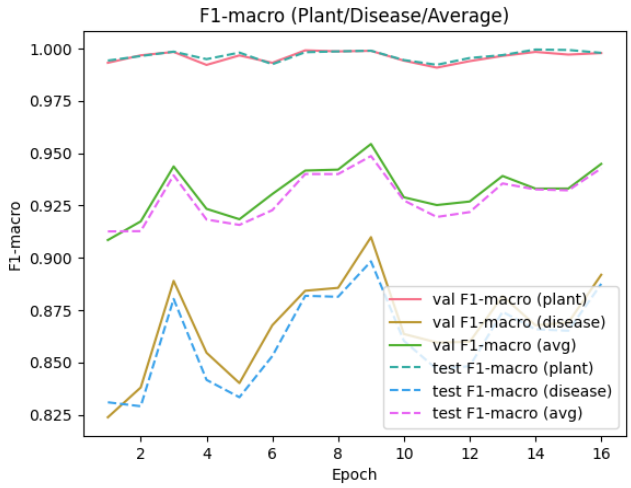
We trained on `train`, selected the checkpoint with the best validation accuracy (ties by lower loss), and evaluated once on the held-out `test` split. We report top-1 accuracy, macro-F1 (unweighted mean over classes), and loss; ROC–AUC is shown for summaries. Figures with learning dynamics and confusion/ROC (if available) are included in the Results section.

6 Results and conclusions

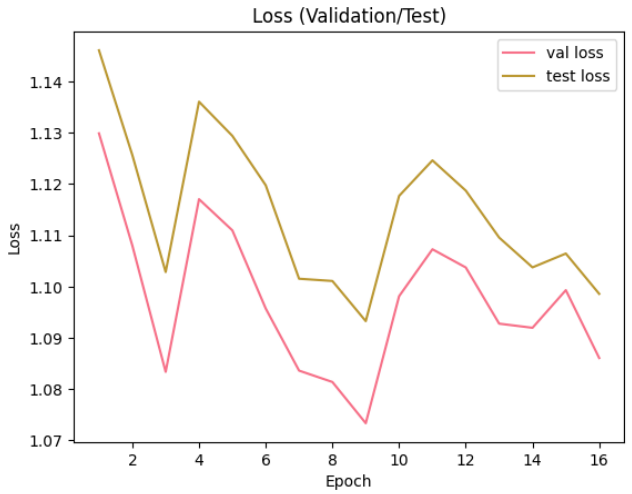
- Plant recognition is easy in this dataset (very high accuracy/F1, low loss); disease recognition is lower but solid.
- Loss comparison: Houdini achieves lower loss; cross-entropy + class weights achieves higher macro-F1 (e.g., 0.867 vs. 0.844). We prioritize macro-F1 for practical usefulness.
- Model choice: DeiT-Small provided the strongest overall accuracy in our sweep; ResNet-18 remains a fast, compact baseline with competitive results.
- Runtime: on a T4, DeiT-Small is feasible; throughput and epoch times are reported in the runtime figures.



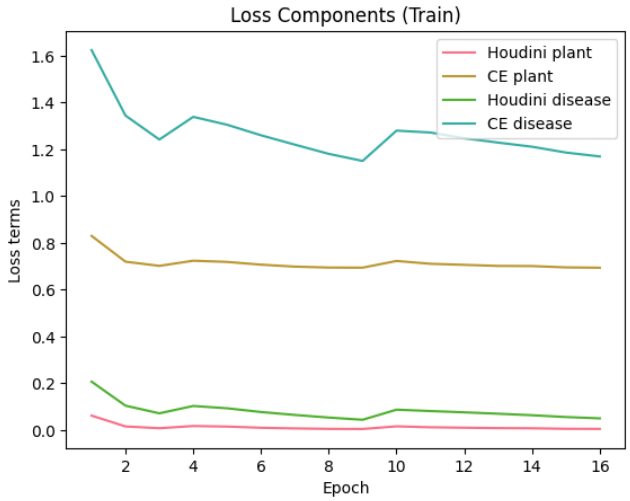
(a) Train accuracy (plant / disease).



(b) Macro-F1 on val / test.



(c) Validation / test loss.



(d) Train loss components (Houdini vs CE).

Figure 6: Learning dynamics for ResNet-18.

We can see in graph (a) that both plant accuracy and disease accuracy improve, and the accuracy is high enough (disease acc ≥ 0.92 and plant acc close to 1) after 16 epochs. The classification model seems efficient in predicting the image classes. Graph (b) shows that both validation and test F1 scores converge, which means the model is in a healthy state, and there is no detectable risk of overfitting. In graph (c), we see the combined loss looks nearly identical for val and test—that's another sign of a stable model. But the loss is still pretty high; this could reflect class imbalance in the dataset, which inflates the importance of some classes over others (and helps explain why F1-macro(disease) is lower than F1-macro(plant)).

Plants vs. diseases. Plant recognition is easier (very high accuracy/F1, low loss); disease recognition is lower but solid. **Loss comparison:** Houdini achieved lower loss (better separation), while cross-entropy with class weights achieved higher macro-F1 (e.g., 0.867 vs. 0.844), which we prioritize. **Model comparison:** DeiT-Small

reached the strongest accuracy in our sweep; ResNet-18 remains a fast, compact baseline with competitive results.

6.1 Metric rationale

We report top-1 accuracy for overall correctness, macro-F1 to balance rare and common classes, and validation loss for optimization diagnostics; best checkpoints are selected by validation accuracy (ties by lower loss).

Model choice. In the short-budget sweep, DeiT-Small produced the strongest disease-results overall; ResNet-18 remains a compact, fast baseline with competitive performance.

References

[1] asheniranga, "Leaf disease dataset combination," https://www.kaggle.com/datasets/asheniranga/leaf-disease-dataset-combination, 2023, kaggle dataset.

[2] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[3] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[4] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] M. Waskom, "Seaborn: Statistical data visualization," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.592845

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] "Torchvision: Models and datasets for computer vision," https://pytorch.org/vision/stable/, accessed 2025-09-28.

[9] R. Wightman, "Pytorch image models," 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4414861

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *USENIX OSDI*, 2016.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[13] "Pillow (pil fork) documentation," https://pillow.readthedocs.io/, accessed 2025-09-28.

[14] J. Buchner, "Imagehash," https://pypi.org/project/ImageHash/, 2013, perceptual hashing utilities.

[15] C. da Costa-Luis *et al.*, "tqdm: A fast, extensible progress bar for python and cli," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.595120

[16] "kagglehub: Simple access to kaggle datasets," https://github.com/Kaggle/kagglehub, 2023.

[17] "Google colaboratory," https://colab.research.google.com, 2024.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[19] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[20] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 11 976–11 986.

[21] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 10 347–10 357.

[22] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[23] ——, "SGDR: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations (ICLR)*, 2017.

[24] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision (ECCV)*, 2016.

[25] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[26] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[28] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

7 Addendum

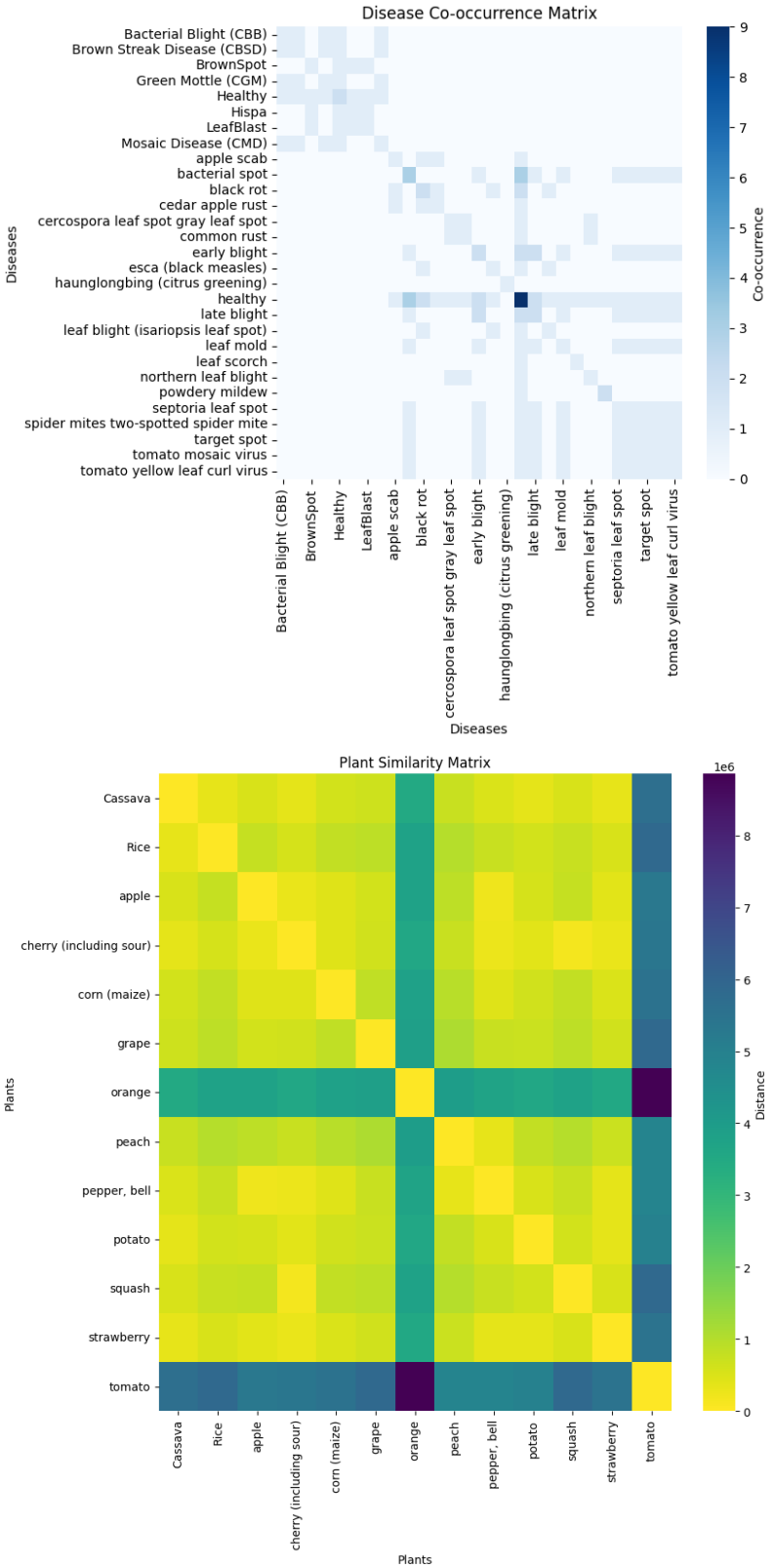


Figure 7: Caption

Figure 8: Caption