Plant Disease Recognition: Comparison of CNNs and Transfer Learning Models

Pravalika Kuppireddy
University of Michigan–Dearborn
Email: pkuppire@umich.edu

Harika Kanala
University of Michigan–Dearborn
Email: harikams@umich.edu

***Abstract*—Plant disease recognition from leaf images is an important application of pattern recognition and neural networks, as early identification of disease can help reduce crop loss and improve agricultural decision making. In this project, we study a real-world plant disease dataset where images contain natural backgrounds, varying lighting conditions, and subtle disease symptoms. We begin by training convolutional neural networks (CNNs) from scratch to establish baseline performance and to understand common sources of classification errors. We then apply transfer learning using pretrained models such as VGG16, ResNet50, and EfficientNet, and compare frozen and fine-tuned training strategies. Through systematic experiments and analysis, we observe that pretrained models consistently generalize better than CNNs on this small dataset. Overall, this project demonstrates how model choice and training strategy interact with dataset characteristics in realistic image classification tasks.**

***Index Terms*—Plant disease detection, Convolutional Neural Networks, Transfer learning, VGG16, ResNet, EfficientNet, Image classification**

I. Introduction

Plant disease is a serious challenge to agricultural productivity, as infections can spread quickly and significantly reduce crop yield if not detected early. Traditionally, disease identification relies on visual inspection by farmers or agricultural experts, which can be time-consuming and subjective. Automated plant disease recognition using image-based methods has the potential to support this process by providing fast and consistent assessments based on visual patterns observed on leaves.

From a pattern recognition perspective, plant disease detection can be viewed as an image classification problem where the goal is to map visual features such as texture, color, and local lesions to specific disease categories. Convolutional neural networks (CNNs) are well suited for this task because they are capable of learning hierarchical feature representations directly from images. However, real-world plant images introduce additional challenges compared to controlled laboratory datasets. Outdoor images always include cluttered backgrounds, overlapping leaves, and varying illumination, which can introduce noise unrelated to disease symptoms.

These real-world conditions make classification more difficult, especially when disease symptoms are subtle or partially visible. For example, powdery images may appear as faint discoloration that is easily confused with natural leaf texture or lighting effects. As a result, even visually simple classification tasks can become challenging when models are trained on limited and noisy data.

In this project, we aim to understand how different modeling choices behave under such conditions. We first evaluate custom CNNs trained from scratch to establish baseline performance and to identify common failure modes. We then compare these results with transfer learning approaches using pretrained deep neural networks, which leverage features learned from large-scale image datasets. This project systematically evaluates custom CNNs and pretrained deep learning models to understand their behavior on a small, real-world dataset.

II. Related Work

Plant disease recognition using image-based pattern recognition techniques has gained significant attention due to its potential applications in agriculture and food security. Early studies demonstrated that convolutional neural networks (CNNs) are capable of learning discriminative visual features from leaf images, motivating their use for automated disease detection. However, subsequent research has shown that dataset characteristics such as background complexity, image quality, and dataset size play a critical role in determining model performance, especially in real-world scenarios.

A. CNN-based approaches for plant disease recognition

One of the most influential early works by Mohanty et al. [1] showed that deep CNNs can achieve very high classification accuracy on the PlantVillage dataset. In this dataset, leaf images are captured under controlled conditions with clean backgrounds and consistent lighting. Similar conclusions were reported by Sladojević et al. [2] and Ferentinos [3], where CNN architectures successfully learned disease-specific texture and color patterns.

Despite these promising results, later studies highlighted that CNNs trained on such controlled datasets often fail to generalize to real-world images. Field images typically contain background clutter, overlapping leaves, and varying illumination, which introduce noise unrelated to disease symptoms. As a result, CNNs may incorrectly rely on background cues rather than true pathological features.

Relevance to our work: Our dataset closely resembles real-world field conditions rather than controlled laboratory settings. During exploratory data analysis, we observed that many images contain complex green backgrounds and multiple

leaves. This explains why our baseline CNN models showed limited robustness and confusion between visually similar classes such as Healthy and Powdery.

B. Limitations of training CNNs from scratch on small datasets

Training CNNs from scratch is known to be data-intensive. When datasets are small, models are prone to overfitting and unstable validation behavior. Shorten and Khoshgoftaar [4] emphasized that while data augmentation can improve generalization, it cannot fully replace the diversity of a large dataset. This issue is particularly severe in fine-grained classification tasks, where class boundaries are subtle.

In plant disease recognition, diseases such as powdery mildew may appear faint or localized, leading to high intra-class variability. As noted by Too et al. [5], small datasets combined with subtle disease patterns significantly reduce the effectiveness of scratch CNNs.

Relevance to our work: Our dataset includes only 60 validation images, making reliable validation challenging. This limitation was reflected in fluctuating validation curves for our scratch CNN models, even after architectural improvements and augmentation. These observations motivated us to explore transfer learning approaches.

C. Transfer learning for plant disease classification

Transfer learning has emerged as an effective solution for small and noisy datasets. By leveraging models pretrained on large-scale datasets such as ImageNet, transfer learning provides robust low-level and mid-level features that can be adapted to new tasks. Too et al. [5] conducted a comparative study of multiple pretrained models and showed that transfer learning consistently outperforms CNNs trained from scratch for plant disease classification.

The success of transfer learning is attributed to the reuse of general visual features such as edges, textures, and shape primitives. Even though ImageNet does not contain plant disease images, these features transfer well because disease symptoms are largely texture-driven.

Relevance to our work: This insight directly guided our experimental design. We evaluated multiple pretrained architectures and observed faster convergence, smoother validation curves, and improved test accuracy compared to CNNs.

D. Choice of pretrained architectures

We selected three widely used pretrained architectures VGG, ResNet, and EfficientNet based on their complementary design and proven effectiveness in transfer learning.

VGG networks, introduced by Simonyan and Zisserman [6], use deep stacks of small convolutional filters and are known to perform well as generic feature extractors, especially for texture-based classification tasks. Despite their computational cost, VGG models remain popular in plant disease studies due to their stable fine-tuning behavior.

ResNet architectures proposed by He et al. [7] introduced residual connections that enable very deep networks to be trained effectively. These skip connections improve gradient flow and feature reuse, making ResNet particularly suitable for fine-tuning on small datasets.

EfficientNet, introduced by Tan and Le [8], employs compound scaling of network depth, width, and resolution. This balanced scaling allows EfficientNet models to achieve strong performance with fewer parameters, reducing overfitting risk on small datasets.

Relevance to our work: Our experiments confirmed these theoretical advantages. While all pretrained models outperformed baseline CNNs, fine-tuned ResNet50 and VGG achieved the best test performance, while EfficientNet demonstrated strong stability and efficiency.

E. Fine-tuning strategies for small and noisy datasets

Although transfer learning provides a strong initialization, improper fine-tuning can degrade performance. Aggressive fine-tuning with high learning rates may cause catastrophic forgetting, particularly when the target dataset is small. Best practices reported in the literature include staged training, low learning rates, early stopping, and freezing batch normalization layers [5], [9].

Batch normalization layers are especially sensitive because updating their statistics on small datasets can destabilize training. Ioffe and Szegedy [10] showed that batch normalization improves training stability, but later studies recommend freezing these layers during fine-tuning on limited data.

Relevance to our work: We followed these best practices by employing low learning rates, partial unfreezing of deeper layers, early stopping, and freezing batch normalization layers during fine-tuning. This "safe fine-tuning" strategy resulted in consistent validation performance and strong generalization.

Our work builds upon these findings by providing a systematic experimental comparison between scratch CNNs and multiple pretrained models under real-world conditions, explicitly analyzing how dataset characteristics influence model behavior. Rather than proposing a new architecture, our contribution lies in empirical insight, careful fine-tuning, and a clear pattern-recognition based analysis.

III. Dataset Description and EDA

A. Dataset Overview

We used the Plant Disease Recognition dataset from Kaggle [11]. The dataset is already organized into official train, validation, and test splits with a consistent folder structure: `Train/Train/<class>/*.jpg`, `Validation/Validation/<class>/*.jpg`, and `Test/Test/<class>/*.jpg`. The three classes are Healthy, Powdery, and Rust. In total, we scanned 1532 images and verified that all were readable RGB images. We also checked for corruption and exact duplicates using MD5 hashing and found none.

A useful property of this dataset is that it is balanced across classes in each split. The training set contains 458 Healthy, 430 Powdery, and 434 Rust images. The validation set is small but balanced with 20 images per class (60 total), and the test set

contains 50 images per class (150 total). Because the classes are balanced, we did not use class weighting or oversampling; instead, we used standard cross-entropy training and focused on model behavior and generalization.

B. Leaf Image Characteristics

Even though the dataset size is moderate, the images are high-resolution photographs. Most images are around 4000×2672 pixels, with a minimum near 2421×1728 and a maximum near 5184×3456. This immediately creates a practical constraint: training on raw resolution is expensive and unnecessary for early experiments, so resizing becomes an important design choice.

We also observed that most images share a similar aspect ratio (roughly 1.5), which suggests that standard resizing should not introduce extreme geometric distortion. At the same time, these are outdoor images with clutter. Many images contain multiple leaves, branches, and background greenery. Lighting varies due to sun angle, shadows, and reflections. These factors make the classification task more realistic and more challenging than controlled lab datasets.

C. EDA Observations Relevant to Modeling

We performed exploratory checks on file sizes, brightness, and RGB channel statistics to understand whether basic image-level factors might correlate with classes. File sizes were generally in a normal range for high-resolution photos (roughly a few hundred KB to under 2 MB), and we did not find any corrupted samples.

Brightness and color checks showed reasonable trends that match intuition: Rust images tend to look slightly darker on average, while Powdery images can appear brighter due to the whitish fungal layer. Across all classes, green is the dominant channel, which is expected for leaves. These trends are not strong enough to solve the problem by simple thresholds, but they signal that illumination differences exist and could confuse a model if not handled well.

Visual inspection was also important. Healthy leaves often have uniform green textures. Rust has more distinctive reddish/orange lesions and spots. Powdery mildew appears as white/gray areas with reduced contrast, but the visibility varies a lot across samples. In some images, Powdery symptoms are faint and can be confused with glare or natural texture. This observation helped explain why errors are likely to concentrate between Healthy and Powdery.

D. How EDA Guided Design Choices

The EDA influenced our pipeline in several ways. First, the high resolution pushed us to use resized inputs. We used 128×128 for CNN baselines to train quickly and to get a reliable reference point. We also used 224×224 because many pretrained models expect that input size and because higher resolution might help capture small texture cues.

Second, because the dataset is real-world and brightness varies, we used data augmentation on the training set (random flips, rotations, zoom, and brightness/contrast variations). The goal was not to artificially increase dataset size, but to reduce sensitivity to orientation and lighting differences. Third, since the task is texture-driven (spots, coatings, lesions), CNN feature extraction is a natural fit. Finally, the very small validation split (60 images total) made us cautious about over-interpreting validation curves; we relied more on consistent trends and final test performance than on noisy validation points.

IV. Baseline and Improved CNN Models

A. Why Start with custom CNNs?

Before using pretrained models, we built baseline and improved CNNs to understand how much the dataset supports learning from the ground up. This step matters because it reveals whether errors come from insufficient model capacity, from limited data, or from ambiguity in the images themselves. A baseline model also gives a fair reference point for later improvements and makes it easier to justify why transfer learning is needed.

B. Baseline CNN Behavior

Our baseline CNN used a simple stack of convolution and pooling blocks followed by a small dense classifier with dropout and L2 regularization. We trained it with standard callbacks (checkpointing, early stopping, and learning-rate reduction) to reduce overfitting risk.

At 128×128, the baseline achieved strong test performance and learned meaningful disease cues. The confusion matrix showed that Rust was usually classified correctly, while the main confusion was Powdery predicted as Healthy. This matches the EDA: Powdery symptoms can be subtle and resemble normal brightness variation. At 224×224, the baseline did not improve and slightly degraded. This was an important lesson: simply increasing resolution increases input complexity, and without changing model capacity or training strategy, a scratch CNN may not benefit from the extra detail.

C. Motivation for an Improved CNN

We then attempted an improved CNN because the baseline model still showed consistent confusion between Healthy and Powdery. Based on EDA and visual inspection, disease cues can be small texture patterns, so we expected that adding depth and stabilization mechanisms would help. The improved CNN introduced additional convolutional depth and batch normalization to stabilize training under lighting variation, while keeping regularization (dropout) to limit overfitting.

D. Improved CNN Observations

At 128×128, the improved CNN achieved similar accuracy to the baseline, but it shifted error patterns slightly. Model generally converged and produced a strong confusion matrix. At 224×224, the improved CNN performed worse and again showed increased powdery-to-Healthy confusion. This shows that higher resolution does not automatically improve scratch training when data is limited. In our case, the combination of small validation size and complex backgrounds likely made optimization less stable and increased the chance of learning spurious correlations.

E. Where custom CNNs Struggle

Across baseline and improved CNNs, the main pattern was consistent: Rust is easier due to clearer lesion cues, while Powdery vs Healthy is hard because symptoms are faint and can be masked by lighting. These observations set up the motivation for transfer learning: if we can start from a feature extractor that already captures robust texture and shape cues, the model may depend less on background correlations and generalize more reliably.

V. Transfer Learning Models

A. Why Transfer Learning for This Dataset

Transfer learning is well suited here because our dataset is not large and the validation split is especially small. A custom CNN must learn low-level features (edges and textures) and higher-level patterns entirely from our training images. In contrast, a pretrained backbone already has filters that respond to meaningful visual primitives. For leaf disease images, those primitives still matter because disease symptoms often appear as texture changes and local contrast patterns.

We selected three architectures that represent different design ideas: VGG16, ResNet50, and EfficientNetB0. We used two training stages for each model: (1) frozen feature extraction, where the backbone is fixed and only a small classifier head is trained, and (2) careful fine-tuning, where a subset of deeper layers is unfrozen with a small learning rate.

B. VGG (VGG16)

VGG was chosen because it is a simple and uniform convolutional architecture that often transfers well in texture-based tasks. Even though it has more parameters than some newer models, its layered structure makes it a strong feature extractor. In our setup, VGG was evaluated with a frozen backbone to measure how well ImageNet features transfer directly, and then with fine-tuning to allow deeper layers to adapt to leaf textures and disease-specific patterns.

C. ResNet50

ResNet50 uses residual connections, which support stable training of deeper networks and help gradients flow. We chose ResNet50 to study whether residual learning provides an advantage during fine-tuning on a small dataset. The frozen-backbone stage measures baseline transfer quality, and the fine-tuning stage allows deeper residual blocks to adjust to domain-specific cues without losing the general visual representation.

D. EfficientNetB0

EfficientNetB0 was chosen because it is parameter-efficient and often performs well when data is limited. Since overfitting is a concern in our dataset, a model that achieves good performance with fewer parameters is attractive. We expected EfficientNet to provide stable learning curves under frozen training and potentially modest improvements with careful fine-tuning.

E. Frozen vs Fine-Tuned Strategy

For all transfer learning models, the frozen-backbone stage is important because it isolates the benefit of generic features. Fine-tuning is then used to specialize the representation. Because our dataset is small, we avoided aggressive fine-tuning. We used low learning rates, partial unfreezing, and freezing of batch normalization statistics (where applicable) to prevent unstable updates.

VI. Experimental Setup

All experiments were implemented in TensorFlow/Keras using a consistent pipeline so comparisons are fair. We trained and evaluated models on the official dataset splits, keeping the test set fixed at 150 images (50 per class). We resized images to either 128×128 (CNN baselines) or 224×224 (scratch comparison and all pretrained models).

We used sparse categorical cross-entropy because labels are integer-encoded, and we used the Adam optimizer with standard learning-rate schedules. Data augmentation was applied only on the training set and included random flips, small rotations, zoom, and brightness/contrast variations. Validation and test images were not augmented. We used common callbacks: ModelCheckpoint to save the best model, EarlyStopping to limit overfitting, and ReduceLROnPlateau to reduce the learning rate when validation loss plateaued.

Because the validation set is small, validation curves can be noisy. For that reason, we treated validation metrics as guidance for early stopping and model selection, but we relied on consistent patterns across experiments and final test evaluation for conclusions. We report test accuracy and loss, along with confusion matrices and class-wise precision/recall/F1 as supporting evidence.

VII. Results

The key comparison is between custom CNNs and transfer learning models, and between frozen and fine-tuned transfer learning.

Figure 1 shows training curves for custom CNN models and highlights two practical observations: (1) CNN models can achieve strong training performance quickly, and (2) validation behavior is noticeably less stable, which is expected given the small validation set and real-world image complexity. The figure also supports the observation that increasing resolution alone did not produce better generalization for custom CNNs.

Figure 2 shows the frozen-backbone training curves for VGG, ResNet50, and EfficientNetB0. Compared to baseline CNNs, these models converge faster and generally show smoother validation behavior, suggesting that pretrained features provide a better starting point under limited data.

Figure 3 shows the fine-tuned training curves for the same pretrained models. Fine-tuning improved generalization for VGG and ResNet when done conservatively, and the curves remain relatively stable, which suggests that staged training helped avoid overfitting.

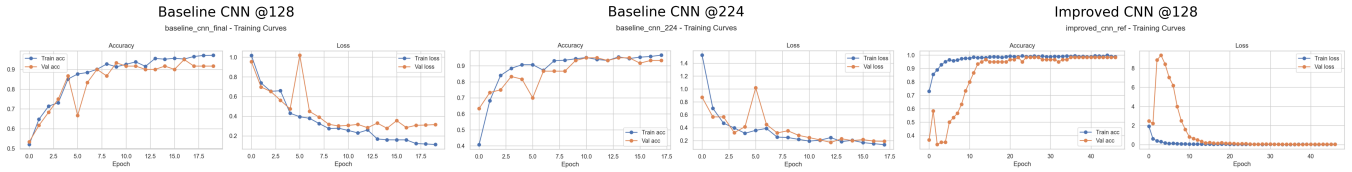

Fig. 1. Training and validation accuracy and loss curves for custom CNN models.

Fig. 2. Training curves for pretrained models with frozen backbones. Compared to baseline CNNs, these models converge faster and exhibit smoother validation behavior, indicating better feature generalization.

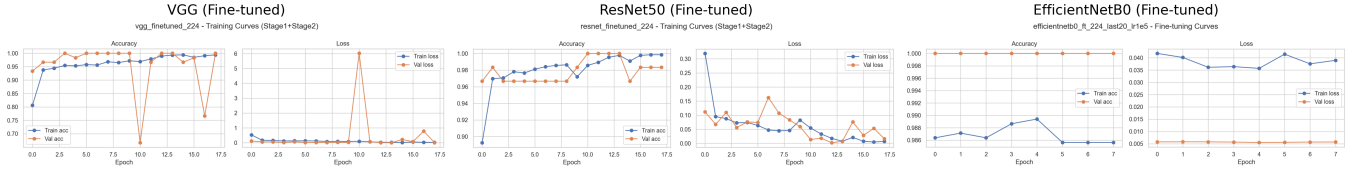

Fig. 3. Training curves for fine-tuned pretrained models. Fine-tuning selected deeper layers improves validation performance while maintaining stable training behavior.

TABLE I
TEST SET COMPARISON ACROSS ALL EXPERIMENTS. BEST RESULTS ARE BOLDED.

Model	Input	Training	Test Acc.	Test Loss
Baseline CNN	128	Scratch	**0.9467**	0.1975
Baseline CNN	224	Scratch	0.9267	0.2588
Improved CNN	128	Scratch	**0.9467**	0.3053
Improved CNN	224	Scratch	0.9133	1.0080
VGG16	224	Frozen	∼0.9600	0.1705
VGG16	224	**Fine-tuned**	**0.9800**	0.1526
EfficientNetB0	224	Frozen	0.9733	0.0785
EfficientNetB0	224	Fine-tuned	0.9733	0.0785
ResNet50	224	Frozen	0.9467	0.0759
ResNet50	224	**Fine-tuned**	**0.9800**	**0.0689**

To highlight class-level behavior, Figure 4 compares a representative baseline CNN confusion matrix with a best fine-tuned pretrained model confusion matrix. The main qualitative difference is reduced confusion between Healthy and Powdery in the transfer learning case, while Rust remains strong in both. This is consistent with our earlier EDA observations: Rust images are distinctive, while Powdery symptoms overlap more with Healthy due to brightness and texture variation.

Finally, Table I provides a single comparison table across all major experiments. We bold the best-performing models.

VIII. DISCUSSION

A. Baseline and improved CNNs behavior and what limits performance

Our baseline and improved CNNs achieved strong test accuracy, which indicates that the dataset contains clear visual signals for plant disease recognition. Across both architectures, the most consistent source of error was confusion between Healthy and Powdery leaves. This is reasonable given the nature of the task: early CNN layers capture general edges and textures, but separating subtle powdery coatings from normal leaf texture requires more fine-grained, disease-specific cues that can vary widely across real outdoor images.

Our EDA helps explain why this boundary is challenging. Powdery samples often appear slightly brighter, but brightness is not a reliable cue because outdoor lighting, shadows, and reflections vary substantially across images. In addition, backgrounds are cluttered and the leaf does not always occupy the full frame, so the model may sometimes pick up weak context correlations (background greenery, illumination patterns) that do not generalize perfectly to new images. These effects are most visible when comparing class-wise errors: Rust is typically easier due to distinctive spot patterns, while Powdery vs. Healthy remains the most ambiguous pair.

We also compared input resolutions (128×128 versus 224×224). In our experiments, the 128×128 models already performed very well, and the 224×224 versions produced similar overall trends with a small change in final test accuracy. Since the difference is modest and we did not exhaustively tune architectures and hyperparameters for each resolution, we

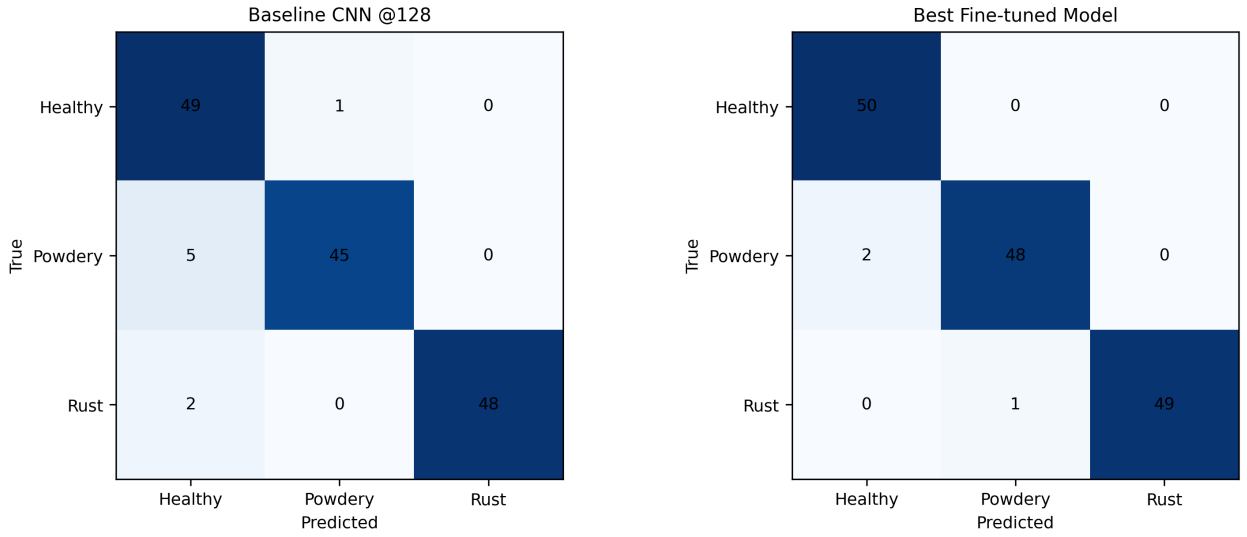

Fig. 4. Confusion matrix comparison between a baseline CNNs and the best fine-tuned pretrained model. Transfer learning significantly reduces inter-class confusion, particularly for visually similar disease categories.

treat this result as an observation rather than a strong conclusion. In practice, higher resolution can provide more texture detail, but it can also increase the difficulty of optimization and the sensitivity to nuisance factors unless model capacity, regularization, and training settings are tuned specifically for that input size.

B. Why Pretrained Models Worked Better

Transfer learning models performed better and more consistently, which is expected for small datasets. A pretrained backbone already contains filters that respond to edges, texture gradients, and shapes, learned from a very large variety of images. Even though ImageNet does not contain plant diseases, the low and mid-level features still transfer well because disease symptoms are texture-based and local.

In practical terms, pretrained models reduce the burden on our dataset. Instead of learning everything from scratch, the model learns only how to combine existing features into the three target classes. This explains the smoother convergence and more stable validation behavior observed for frozen-backbone training. The model is not trying to build a complete representation from limited images; it is adapting a strong representation to a narrower task.

C. Why Fine-Tuning Helped (and Why It Must Be Conservative)

Fine-tuning improved results for VGG and ResNet, while EfficientNet remained strong with little or no gain from fine-tuning in our experiments. Fine-tuning helps because the deeper layers of the backbone can adjust to domain-specific patterns, such as leaf vein textures, disease spot structures, and the particular color/contrast conditions of the dataset. However, with small datasets, fine-tuning must be cautious. If the learning rate is too high or too many layers are unfrozen, the model may overfit quickly or forget useful generic features.

We followed a approach: first train a classifier head with the backbone frozen, then unfreeze a subset of deeper layers and continue training with a small learning rate. This strategy keeps training stable and reduces the chance of large destructive updates. In our experiments, this approach improved generalization without introducing severe instability, which suggests the fine-tuning was safe for this dataset size.

D. Why Powdery vs Healthy Remains Difficult

Even for strong pretrained models, the hardest boundary remained Powdery vs Healthy. This is not only a model limitation; it is a data property. Powdery symptoms can be faint, localized, or partially hidden. Some images also contain glare, shadows, or natural texture patterns that resemble powdery patches. simply saying feature distributions overlap. When class separability is limited by the images themselves, even better models will still show errors in the same region.

Rust is easier because lesions often have distinctive color and spot patterns. This is why most models classified Rust reliably. The consistent error pattern across architectures suggests that future improvements should focus on reducing background influence and improving visibility of subtle powdery cues rather than simply using a larger model.

E. Key Takeaways from the Project

Several clear takeaways are from this study. First, even relatively simple CNNs trained from scratch can achieve strong performance when the dataset contains meaningful visual cues. However, their behavior is strongly influenced by dataset limitations, particularly when classes are visually similar and validation data is limited. In our case, this was most evident in the consistent confusion between Healthy and Powdery leaves.

Second, architectural complexity and input resolution alone are not sufficient to guarantee better performance. Increasing resolution or adding depth can change training dynamics, but

without careful tuning and sufficient data, these changes may yield only marginal gains. This highlights the importance of aligning model capacity, input representation, and dataset size rather than assuming that larger automatically means better.

Third, transfer learning proved to be the most reliable strategy for this task. Pretrained models consistently converged faster, showed more stable training behavior, and achieved higher test accuracy than scratch CNNs. Fine-tuning, when performed conservatively, further improved performance by allowing models to adapt to domain-specific features while preserving robust pretrained representations.

Overall, the results emphasize that model effectiveness in real-world image classification depends not only on architecture, but on how training strategy and dataset properties interact. The most effective solutions in this project were those that balanced representation strength, regularization, and practical constraints imposed by limited data.

IX. Challenges and Limitations

This project had several limitations that affected both training and interpretation of results.

First, the dataset is relatively small for deep learning, especially the validation set (60 images total). A few difficult images can noticeably change validation accuracy and loss, making the curves noisy and sometimes misleading. This forced us to be careful with early stopping decisions and to avoid over-tuning based on small validation fluctuations.

Second, overfitting risk was present throughout, particularly for deeper custom CNNs and for higher-resolution inputs. We used augmentation and dropout, but augmentation cannot fully replace diverse real samples. For fine-tuning, we used conservative learning rates and partial unfreezing to avoid unstable updates.

Third, visual similarity between Healthy and Powdery is an inherent challenge. In some cases, the correct label is not obvious even to humans, especially when powdery symptoms are weak or when lighting creates bright patches. This limits the maximum achievable performance without additional information or improved data.

Fourth, background clutter is a real issue. Unlike other datasets with clean backgrounds, our dataset includes outdoor context that can bias models. baseline and improved CNNs in particular may pick up accidental correlations. Transfer learning reduces this effect but does not remove it completely.

Finally, pretrained models require more compute and careful hyperparameter handling. Fine-tuning ResNet50, for example, is more time-consuming than training a small scratch CNN, and it demands more trial and error to keep training stable on a small dataset.

X. Conclusion and Future Work

In this project, we studied plant disease recognition as a pattern recognition task on a small, real-world dataset with cluttered backgrounds and subtle symptom patterns. We began with custom CNN baselines to understand how much the dataset supports direct feature learning and to identify common failure modes. Scratch CNNs achieved strong performance overall, but they consistently struggled most on the Healthy vs Powdery boundary.

We then evaluated transfer learning using VGG16, ResNet50, and EfficientNetB0 under frozen and fine-tuned training strategies. Pretrained models converged faster and generalized more reliably than custom CNNs. Fine-tuning improved performance further for VGG and ResNet when performed conservatively, while EfficientNet remained strong with minimal change between frozen and fine-tuned settings. Overall, the results support the practical conclusion that transfer learning is a better fit than training for small, realistic leaf image datasets, especially when validation data is limited.

For future work, the most valuable improvements would focus on data and robustness rather than only model size. One direction is to use larger and more diverse datasets or to expand the validation set for more reliable model selection. Another direction is segmentation or leaf localization to reduce background influence and force the model to focus on disease regions. We also think class-specific augmentation could help, especially augmentations that simulate mild powdery symptoms or varying illumination. Finally, it would be useful to explore lightweight deployment-friendly models (or quantized versions of EfficientNet-like architectures) for practical use on mobile or edge devices.

References

[1] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, 2016.

[2] S. Sladojević, M. Arsenović, A. Anderla, D. Culibrk, and D. Stefanović, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational Intelligence and Neuroscience*, 2016.

[3] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, 2018.

[4] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019.

[5] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, 2019.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[8] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. ICML*, 2019.

[9] Y. Bengio, A. Courville, and P. Vincent, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, 2012.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015.

[11] R. Rahman, "Plant disease recognition dataset," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/rashikrahmanpritom/plant-disease-recognition-dataset