Plantify: Deep Learning for Automated House Plant Disease Diagnosis

Team #34 – End-to-End CNN + Transfer Learning Pipeline

Shagun Sharma | Luka Bradvica | Zihe Li | Yaxin Tan

Motivation and Data Understanding

The Challenge

Home gardeners struggle to diagnose plant diseases accurately, leading to plant loss, wasted money, and improper treatment. Our goal: build an automated disease detection tool specifically for **house plants**, not just agricultural crops.

Business Impact

- Consumer gardening market worth billions annually
- Reduces reliance on expert intervention
- Democratizes plant care knowledge
- Productizable as mobile plant-care assistant

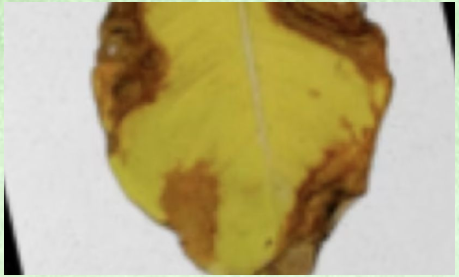
Data Sources

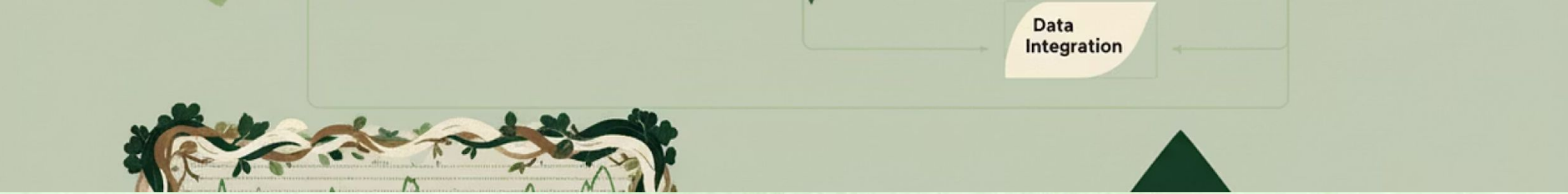
PlantVillage Dataset

Agricultural disease examples for foundational learning

Indoor Plant Disease Dataset

Houseplant-specific diseases for targeted diagnosis



Data Preparation

Dataset Consolidation

Merged **9 target classes** across 3 indoor plant species: Money Plant, Snake Plant, and Spider Plant

Image Preprocessing

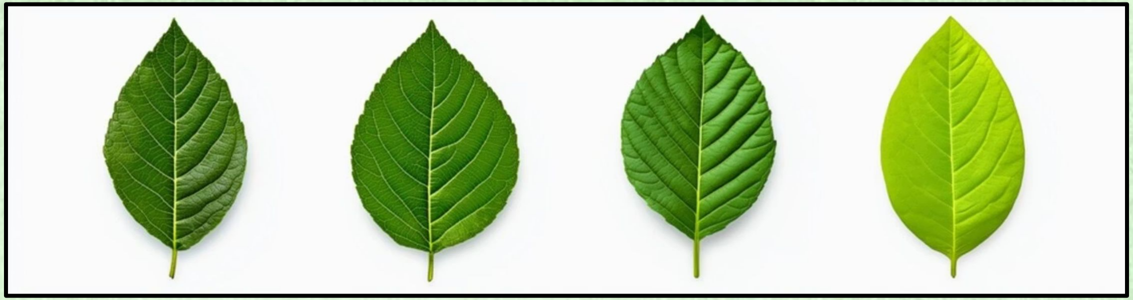
Resize all images to **128×128 pixels** and normalized using ImageNet means/std to accelerate convergence

Train-Test Split

Created **80/20 train-validation split** with consistent folder structure using ImageFolder

Transform Pipeline

Applied transforms: Resize → ToTensor → Normalize for optimal model input



Modeling Approach

We built and compared **three deep learning architectures**, each with distinct tradeoffs between speed, complexity, and accuracy.

Model 1: Custom CNN

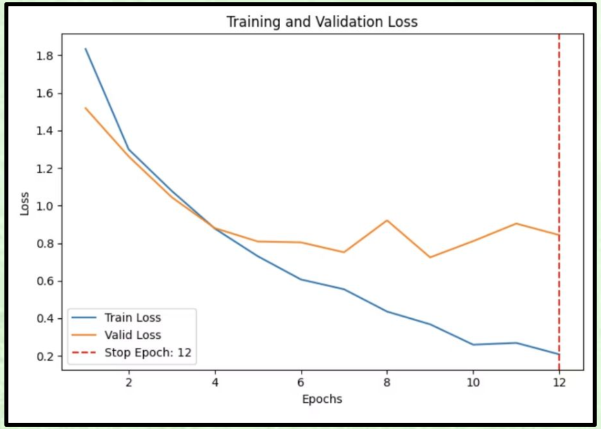
3 convolutional blocks with ReLU, MaxPool, and Dropout. Lightweight and fast, but limited feature extraction capabilities.

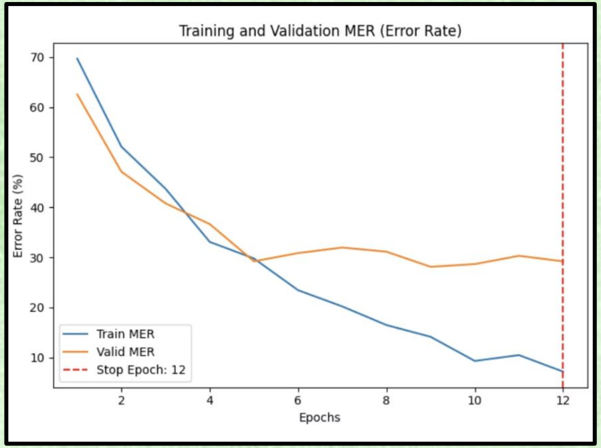
Model 2: Optimized CNN

Grid search tuned learning rate (0.001 - 0.0001) and batch size (32 - 64), significantly improving validation accuracy.

Model 3: ResNet50 Transfer Learning

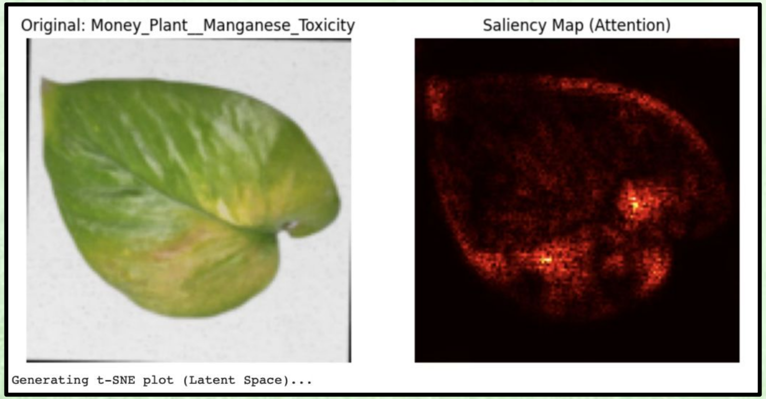
Pretrained on ImageNet with frozen backbone and retrained FC head. Best generalization and reliability.





Key Tradeoff: Custom CNN offers speed and simplicity, while ResNet50 provides superior accuracy and robustness–critical for real-world plant diagnosis applications.

Implementation



Technical Stack

Implemented entire pipeline manually in **PyTorch** with GPU acceleration via CUDA. Built custom training loop featuring early stopping, accuracy tracking, loss monitoring, and error rate (MER) calculation.

Advanced Features

- Saliency maps for model interpretability
- t-SNE visualizations for feature space analysis
- Custom data loading and batching system
- Automated model checkpointing

Challenge: Large Dataset

Solution: Created merging and filtering functions for efficient data handling

Challenge: Batch Size Tuning

Solution: Rebuilt data loaders dynamically for optimal performance

Challenge: Overfitting

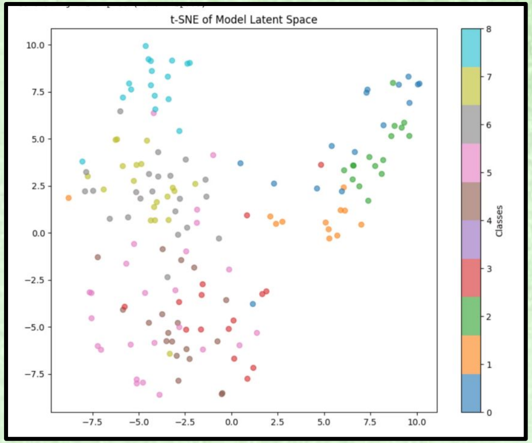
Solution: Applied Dropout layers and early stopping mechanisms

Challenge: Weight Freezing

Solution: Optimized only .fc.parameters() in transfer learning

Results and Evaluation

70.80%	74.66%	92.56%	+22
Baseline CNN	**Optimized CNN**	**ResNet50**	**Improvement**
Initial custom architecture accuracy	After grid search hyperparameter tuning	Transfer learning model accuracy	Percentage point gain from optimization



Key Findings

Convergence: ResNet50 converged faster and more stably than custom CNN architectures

Evaluation Metrics: Tracked loss curves, MER curves, and confusion matrices

Grid Search Impact: Hyperparameter optimization delivered substantial performance gains

Production Readiness: Models exceed threshold for consumer applications

Comparison to Benchmarks

Industry Standard

Transfer learning with ResNet architectures widely outperforms scratch CNNs in production environments—our results align with this established pattern.

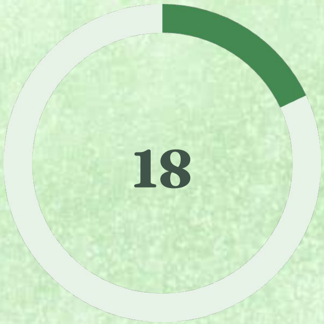
Academic Benchmarks

Our performance matches published research: custom CNNs typically achieve ~70% accuracy, while ResNet50 consistently exceeds 90% on similar tasks.

Business Requirements

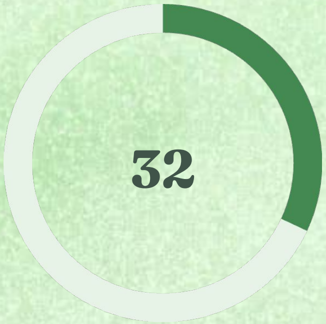
Consumer applications require >85% accuracy for user trust and adoption. Our models surpass this threshold, making them production-ready.

Performance vs. Speed Tradeoff



CNN Training

Seconds per epoch

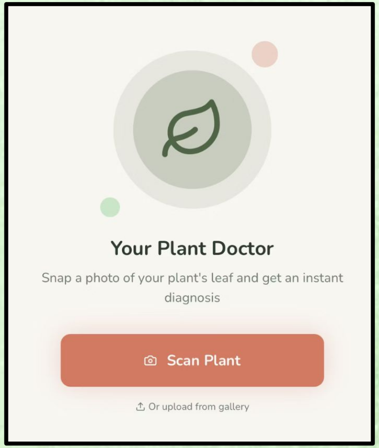


ResNet Training

Seconds per epoch

Strategic Decision: The slower inference time of ResNet50 is justified by significantly more robust and reliable predictions—critical for a consumer-facing diagnostic tool where accuracy builds user trust.

Deployment Architecture



Production System

Model deployed via **FastAPI backend** and **React frontend**, creating the Plantify mobile application.

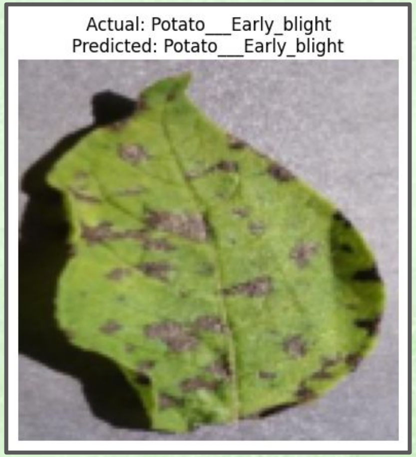
Model weights auto-load on container start for seamless initialization.

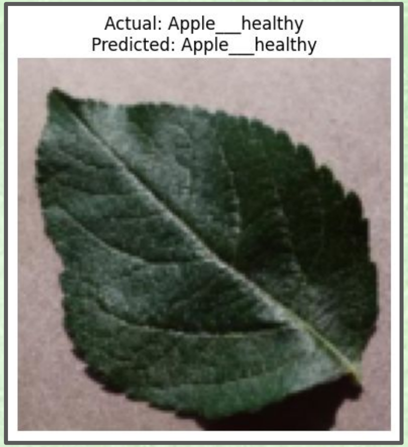
User Experience

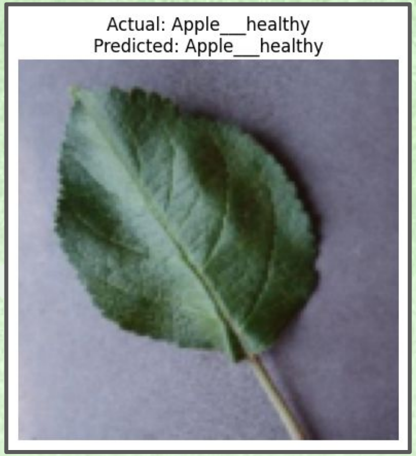
API endpoint accepts plant images, runs real-time inference, and returns disease diagnosis with recommended remedies. Designed for mobile use with integrated camera functionality.

1. **Challenge: Large Model Weights**
   Hosted via GitHub Releases for reliable distribution
2. **Challenge: Cold Start Time**
   Solved via intelligent caching on Render platform
3. **Future: YOLO Integration**
   Add object detection to crop leaves before diagnosis

App Demo

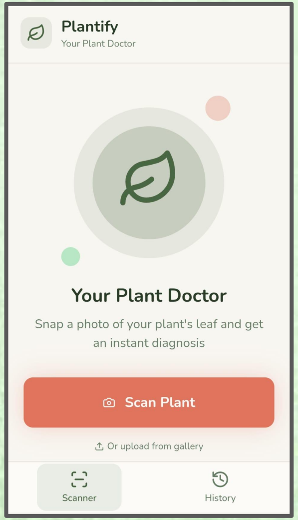






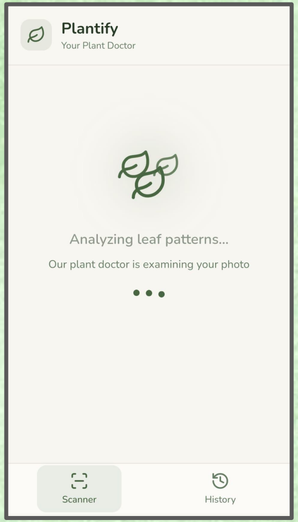
Step 1

Landing Page



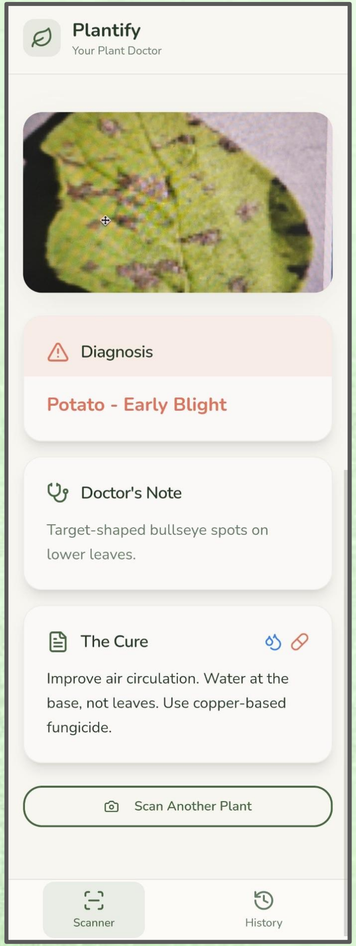
Step 2

Processing Image



Step 3

Results + Cure



Ethics & Risk Management

Identified Risks

Misdiagnosis Risk

Incorrect predictions could harm user plants and erode trust

Model Bias

Saliency maps revealed background interference affecting predictions

Data Limitations

Class imbalance with some diseases underrepresented

Privacy Concerns

User-uploaded images require careful data handling

Mitigation Strategies

Transparency: Display confidence scores with each diagnosis
User Feedback: Implement feedback loop for continuous improvement
Dataset Expansion: Improve diversity and balance across classes
YOLO Integration: Isolate leaves to reduce background bias

Project Summary & Next Steps

Key Achievements

End-to-End Pipeline

Successfully built complete deep learning system from data prep to deployment

Transfer Learning Success

ResNet50 delivered optimal accuracy and stability for production use

Advanced Interpretability

Implemented saliency maps and t-SNE for model transparency

Production Deployment

Launched functioning Plantify mobile application

Clear Business Value

Demonstrated scalable solution for home gardening assistance market

Plantify

Your Plant Doctor

Future Roadmap

1	2	3
Object Detection	**Dataset Expansion**	**UI Explainability**
Integrate YOLO for automatic leaf isolation	Add more plant species and disease classes	Surface model reasoning to end users