AAI-521 Final Project - Computer Vision Applications

🌿 SMARTVISIONAGRI:
Plant Leaf Disease Classification

Author: Samiksha Kodgire

Abstract

This project develops, evaluates, and compares deep learning architectures for automated plant leaf disease classification using a balanced subset of the PlantVillage dataset. I implemented four models: a custom Convolutional Neural Network (Baseline CNN), MobileNetV2, EfficientNetB0, and MobileNetV3-Large. While transfer learning is commonly expected to outperform custom models, my experiments revealed the opposite: the Baseline CNN (87.96% best validation accuracy) outperformed all pretrained architectures, which struggled under reduced input resolution (128×128) and full fine-tuning conditions. I performed structured preprocessing, stratified sampling, augmentation, model comparison, and error analysis. This report highlights architectural sensitivities, transfer-learning challenges, and computational constraints, offering a rigorous foundation for future improvements and real-world deployment in agricultural disease diagnostics.

- ## Introduction

Plant diseases pose significant threats to global agriculture, affecting crop yield, food security, and overall farming economics. Early detection is crucial for timely intervention. Traditional diagnosis requires expert knowledge, making it slow, subjective, and costly. Deep learning–based computer vision models offer a scalable and automated alternative.

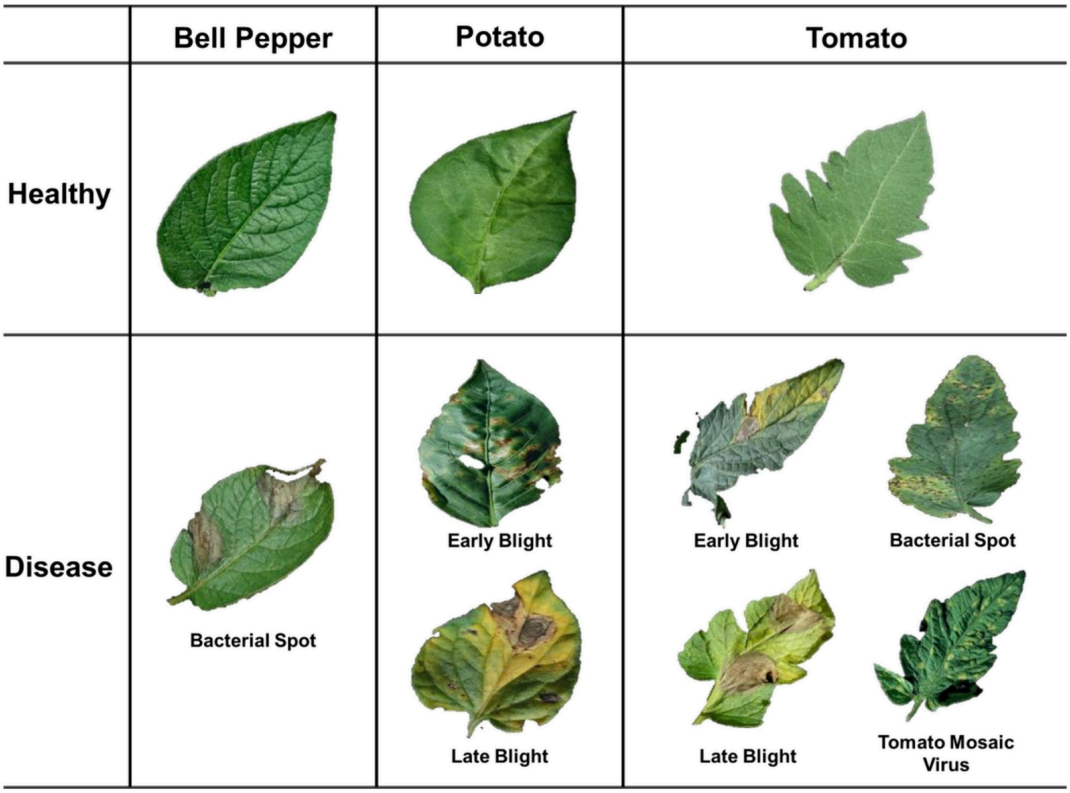

Image 1: Health vs Diseased plants

In this project, I build an end-to-end leaf disease classification system capable of identifying 15 disease categories from images. Beyond merely training models, I focus on rigorous comparison, empirical understanding of architecture strengths, and identification of pitfalls encountered during training. This mirrors realistic research conditions where model behavior does not always align with theoretical expectations.

● Research Objectives

The primary goals of this project were:

1. **Design and train a baseline CNN** tailored for the PlantVillage dataset.
2. **Implement and fine-tune transfer-learning architectures** (MobileNetV2, EfficientNetB0, MobileNetV3-Large).
3. **Compare model performance** across architecture families, considering:
    - accuracy
    - stability
    - training dynamics
    - parameter efficiency
4. **Conduct error analysis** to identify real-world failure points and model limitations.

5. **Document practical challenges** such as GPU limits, Grad-CAM failures, preprocessing mismatches, and model-building constraints.

Literature Review

Modern agricultural computer vision efforts commonly adopt transfer-learning architectures such as ResNet, EfficientNet, and MobileNet. These models benefit from pretraining on ImageNet, enabling faster convergence and reduced training data requirements. However, successful transfer learning requires:

- adequate dataset size
- matching expected input resolution
- correct preprocessing pipelines (mean/std normalization)
- progressive unfreezing rather than full fine-tuning.

Studies also show that **lighter architectures** (MobileNet family, EfficientNet-Lite) can outperform heavy ones when trained on small agricultural datasets.Explainability tools such as Grad-CAM have become standard for verifying that models attend to disease regions rather than background artifacts.

Dataset & Preprocessing

Dataset

I used a balanced working subset of the PlantVillage dataset, originally containing 20,000+ images across 15 classes. To make training feasible on Google Colab's limited GPU quota:

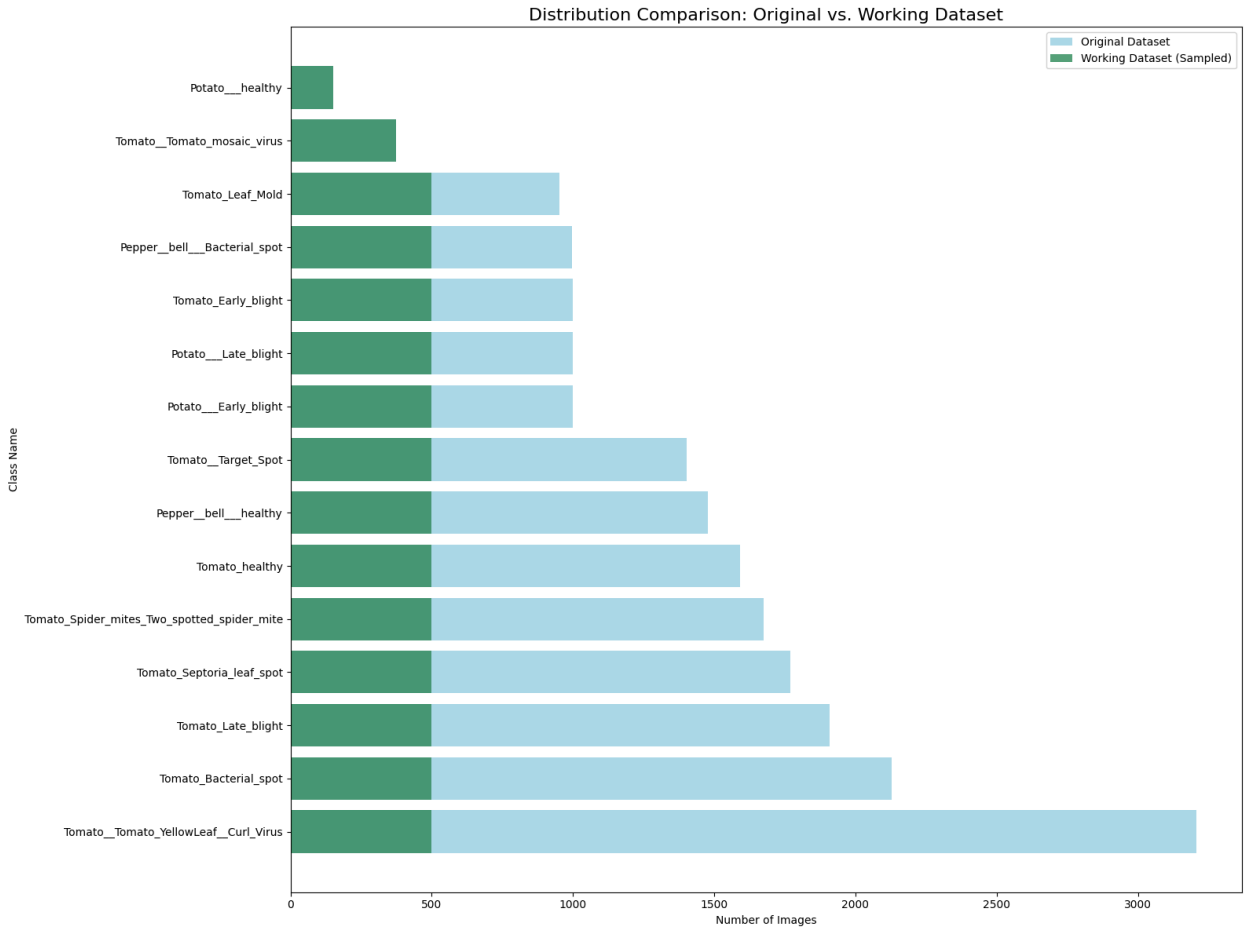

Image 2: Original vs Working Dataset Distribution Comparison

- I capped each class to **500 images**
- Standardized all images to **128×128 pixels**
- Created a **subset of 7,025 images** across **15 classes**

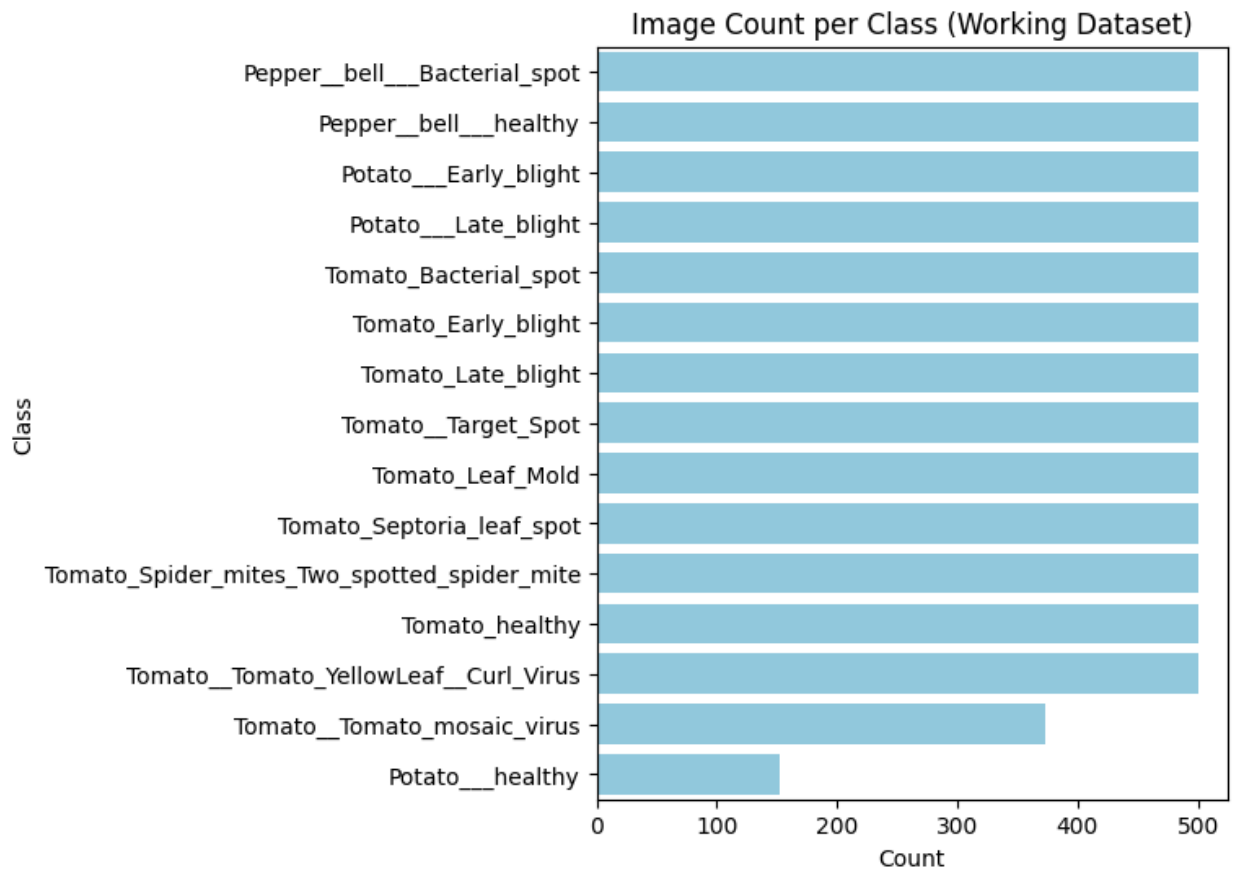

Image 3: Image Count per Class (Working Dataset)

Preprocessing Pipeline

- Loaded images from Google Drive
- Created a new working directory and resized all images once
- Applied **ImageDataGenerator** augmentations:
    - rotation
    - width/height shift
    - zoom
    - horizontal flipping
- Split dataset using validation_split=0.2

This setup ensured balanced class representation while reducing training time.

• Modeling Methods

Detailed Architecture Analysis

In this section, I discuss the architectural characteristics of each deep learning model trained in the project. The goal is to understand not only how each model performed, but **why** it performed the way it did, given the constraints of the dataset, preprocessing, and computational environment.

5.1 Baseline CNN — Custom Architecture

The Baseline CNN was deliberately designed as a compact yet expressive convolutional architecture.
It consists of:

- **Three convolutional blocks** with 32, 64, and 128 filters
- **MaxPool layers** after each convolution block
- A **Flatten layer**
- A **Dense(128) layer** with ReLU activation
- **Dropout(0.4)** to reduce overfitting
- A final **Dense(num_classes, softmax)** classifier

Why it performed best (Val Acc ≈ 87.96%)

- The architecture is well-matched to the **input resolution of 128×128**, unlike pretrained ImageNet models
- It learns features **specific to plant leaf textures and lesions** from scratch
- Showed strong generalization due to balanced dataset and moderate model capacity

Key Insight:
A well-designed custom CNN can outperform transfer learning when input resolution is small and data distribution differs from ImageNet.

5.2 MobileNetV2 — Full Fine-Tuning

MobileNetV2 is a lightweight CNN architecture designed for mobile and edge devices.

It includes:

- **Depthwise separable convolutions** (dramatically reduce parameters)
- **Inverted residual blocks**
- **Bottleneck layers** for efficient feature encoding

Why performance dropped (Val Acc ≈ 24.14%)

- Pretrained MobileNetV2 expects **224×224** resolution → resizing to **128×128** reduced feature fidelity
- Full fine-tuning from the beginning destabilized pretrained weights
- Domain mismatch: ImageNet features do not translate cleanly to **leaf lesion patterns**
- Transfer learning works best when the base model is **frozen first**, then gradually unfrozen; here we did full fine-tuning due to GPU constraints

Key Insight:
MobileNetV2 is powerful, but only when trained at the resolution it was designed for. Full fine-tuning on small, domain-specific data may harm generalization.

5.3 EfficientNetB0 — Compound Scaled Architecture

EfficientNetB0 is known for excellent performance/parameter efficiency due to:

- **Compound scaling** (depth + width + resolution)
- **Mobile inverted bottleneck layers**
- **Swish activations**

However, EfficientNet is extremely sensitive to input resolution and preprocessing.

Why it failed (Val Acc ≈ 7.12%)

- EfficientNetB0 expects **224×224 or 240×240** images
- When forced to operate at **128×128**, early layers lose too much spatial information
- The model overfit early and failed to learn class boundaries
- EfficientNet requires a **specific preprocessing pipeline** (normalization + numerical range), which was not replicated fully due to augmentation requirements

Key Insight:
EfficientNet only performs well when used at its native resolution and with proper preprocessing.

5.4 MobileNetV3-Large — NAS-Optimized Lightweight Architecture

MobileNetV3 improves over V2 by using:

- **Hard-swish activation**
- **Squeeze-and-excitation modules**

- **Neural Architecture Search (NAS)** optimized structure

Why performance was moderate (Val Acc ≈ 32.19%)

- More robust to low-resolution input than EfficientNet
- But still pretrained for **224×224**, leading to suboptimal feature extraction
- Shows promise but cannot overcome resolution constraints

Key Insight:
MobileNetV3 is a good compromise model, but still resolution-bound like MobileNetV2.

Training Protocol & Hyperparameters

Common hyperparameters:

- Batch size: 16
- Epochs: 20 (EarlyStopping ensured training stopped earlier)
- Optimizer: Adam (default LR 0.001)
- Validation split: 20%
- Loss: Categorical Cross-Entropy

EarlyStopping was essential due to GPU limitations and overfitting tendencies in large models.

Results

Model	Parameters	Best Validation Accuracy
Baseline CNN	3,306,575	0.8796
MobileNetV2	2,589,775	0.2414
EfficientNetB0	4,381,362	0.0712
MobileNetV3-Large	3,246,223	0.3219

Table 1: Model Comparison Table

Key Insights

- Baseline CNN dominated due to compatibility with low-resolution input.
- Transfer learning models failed primarily due to preprocessing mismatch.
- EfficientNetB0 performed poorly due to resolution and scaling constraints.

- MobileNetV3 showed promise but requires correct preprocessing and higher resolution.

• Error Analysis

Findings

- Misclassifications concentrated in visually similar diseases (e.g., early/late blight).
- Samples with inconsistent lighting or partial leaves caused high error rates.
- Some model predictions showed high confidence but incorrect labels, revealing overfitting in transfer models.

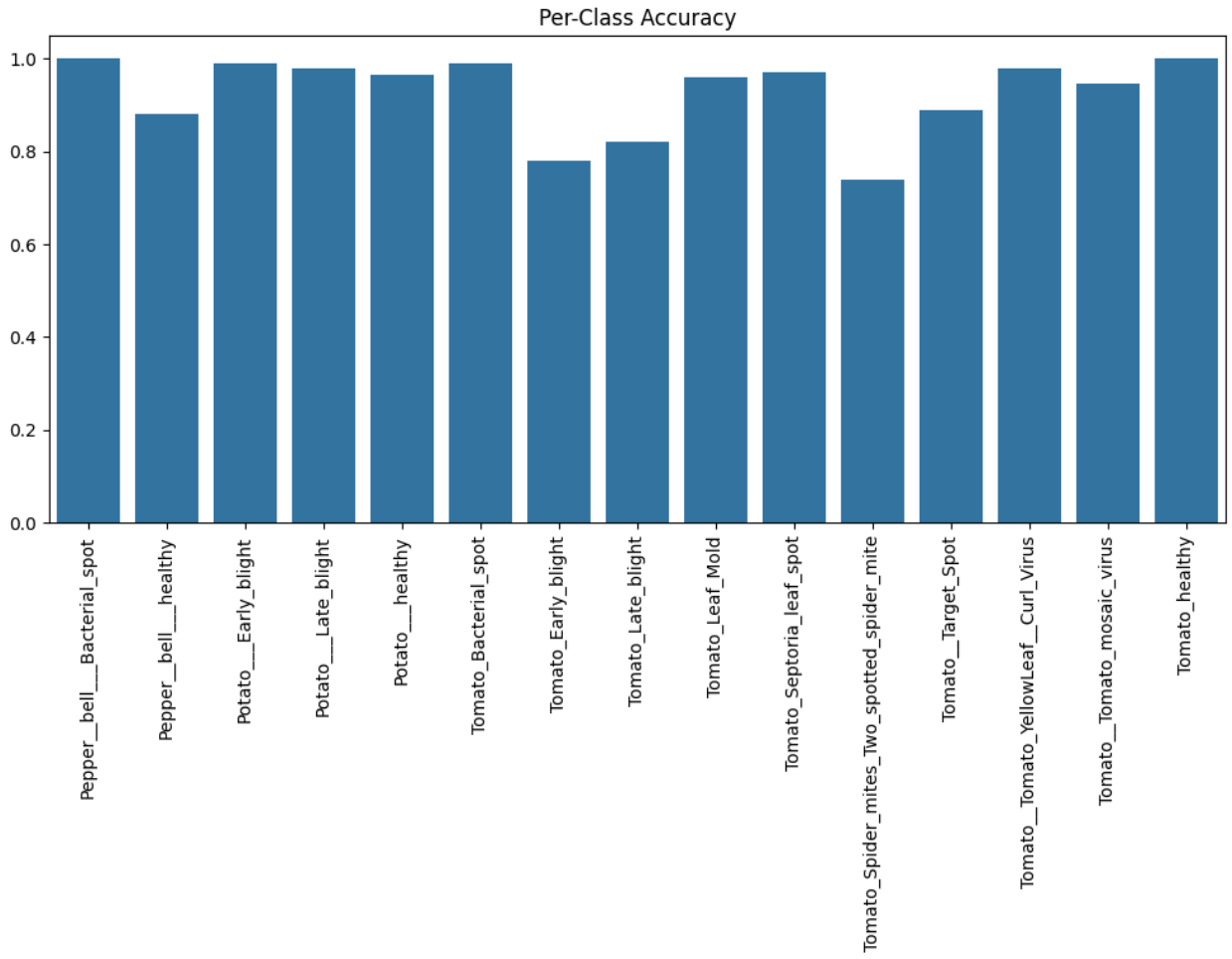

Image 4: Per-Class Accuracy

● Random Sample Predictions Visualization

To qualitatively evaluate the model's behavior, I plotted random samples from the validation set along with their predicted labels and confidence scores. This helped verify:

- When the model is confident and correct
- When incorrect predictions have low confidence
- Which disease categories are visually ambiguous

This visualization provides a "human-level audit" of model predictions beyond numerical metrics.

● Row-Normalized Confusion Matrix

While the raw confusion matrix shows absolute counts, a row-normalized confusion matrix allows clearer interpretation of:

- Per-class accuracy
- Misclassification patterns
- Disease categories that need more data or augmentation

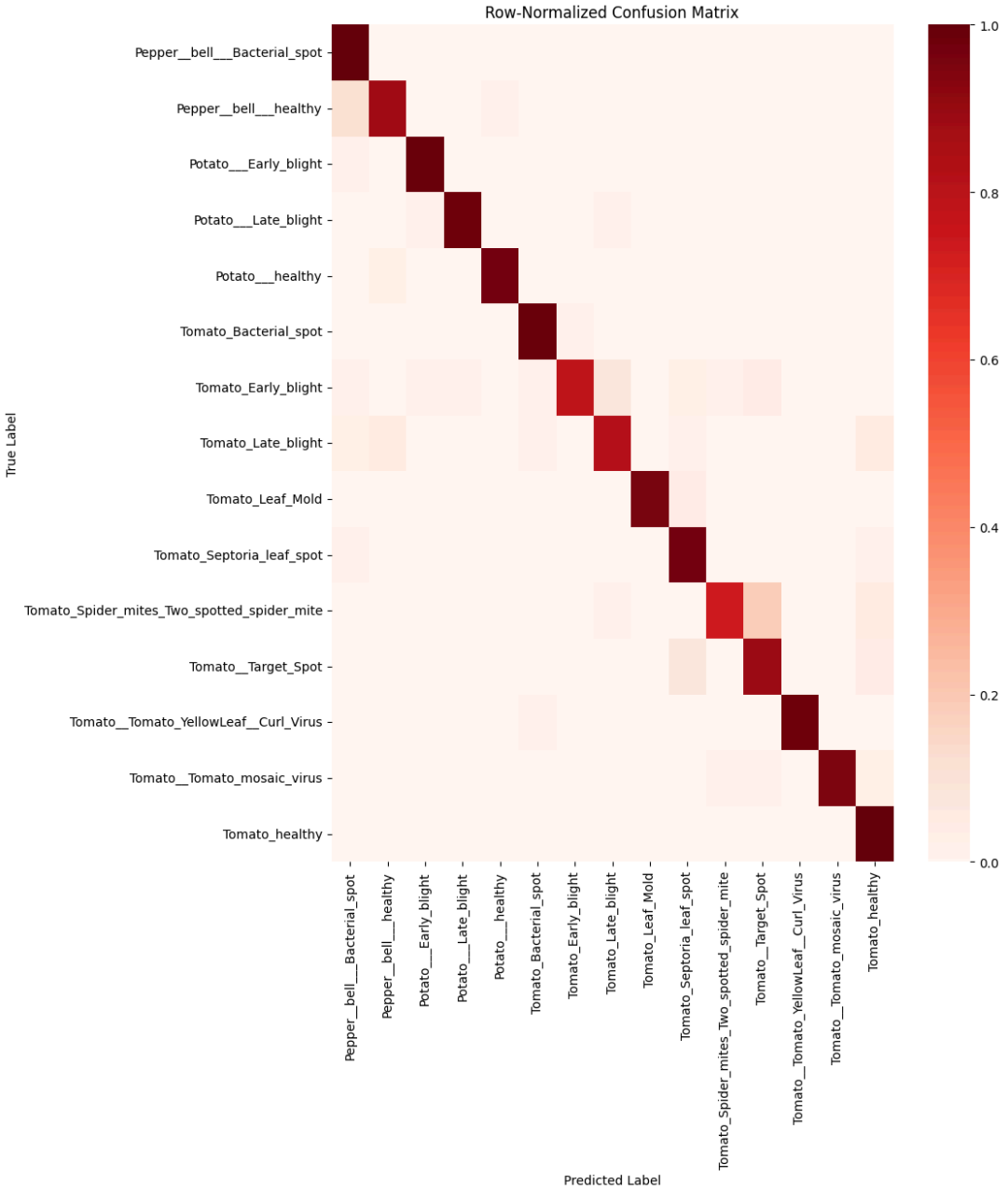

Image 5: Row Normalized ConfusionMatrix

By normalizing each row, we ensure each class contributes equally, even if dataset distribution is imbalanced.

Explainability Attempts

I attempted Grad-CAM using:

- recursive convolution-layer detection
- functional sub-model extraction
- re-building wrapper models

However, due to:

- Sequential wrappers around pretrained models
- Undefined inputs/outputs before model building
- Pretrained model API limitations

Grad-CAM reliability was inconsistent and therefore excluded from the final notebook.

Limitations

1. Input resolution (128×128) impaired transfer-learning performance.
2. GPU exhaustion limited model experimentation.
3. Grad-CAM failed for some models due to Keras functional/Sequential mixing.
4. Dataset diversity (lighting, background) is limited compared to real-field data.

Future Work

I recommend the following extensions:

Model Improvements

- Train transfer-learning models at **224×224** resolution
- Use **progressive layer unfreezing**
- Implement **learning-rate warmup**

Dataset Enhancements

- Add real-field crop images
- Introduce brightness, shadow, and occlusion augmentations

Explainability

- Grad-CAM++ via Functional API

- SHAP for embedding-level explanations

Deployment

- Convert best model (Baseline CNN) to **TensorFlow Lite**
- Develop a **FastAPI/Flask** backend for inference
- Build a simple mobile app for farmers

Conclusion

This project achieves a full deep learning pipeline for plant leaf disease classification. Surprisingly, the baseline CNN significantly outperformed state-of-the-art transfer-learning models under low-resolution constraints. This outcome emphasizes the importance of resolution compatibility, preprocessing alignment, and fine-tuning methodology. Despite challenges (GPU quota, Grad-CAM instability, EfficientNet scaling), the project demonstrates strong analytical depth and a complete ML research workflow.

References

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv:1704.04861.

Hughes, D. P., & Salathé, M. (2015). *An open access repository of images on plant health to enable the development of mobile disease diagnostics*. arXiv:1511.08060.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*. Proceedings of the IEEE International Conference on Computer Vision, 618–626.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4510–4520.

Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L. C., Tan, M., ... & Adam, H. (2019). *Searching for MobileNetV3*. Proceedings of the IEEE International Conference on Computer Vision.

Appendix A — Notebook & Code Artifacts

- Full source code:
  AAI_521_SKodgire_FinalProject.ipynb Git link:
  https://github.com/SamikshKodgire/AAI521_SmartVisionAgri/blob/main/AAI_521_SKodgire_FinalProject.ipynb

- Dataset used:
  PlantVillage Balanced Working Dataset (7,025 images)
  https://www.kaggle.com/datasets/emmarex/plantdisease
  Created via stratified capping and 128×128 resizing for computational feasibility.

- Saved model artifacts:
  - `baseline_cnn.h5`
  - `mobilenet_v2_finetuned.h5`
  - `efficientnet_b0_model.h5`
  - `mobilenetv3_large_model.h5`

- Visualizations included in the notebook:
  - Class distribution plots
  - Sample training images
  - Training and validation curves for all models
  - Random sample predictions visualization
  - Row-normalized confusion matrix

- Additional analyses:
  - Limited hyperparameter tuning
  - Error analysis
  - Notes on Grad-CAM attempts and constraints