Plant Leaf Disease Detection Report

AI–ML Assignment

Group Number: G13

Group Members:

Mainak Bose (Roll No. 23EC8082)
Ayush Rai (Roll No. 23EC8081)
Ramala Venkata Vamsi (Roll No. 23EC8080)

Dataset Name: PlantVillage Leaf Disease Dataset

Department of Electronics and Communication Engineering
National Institute of Technology, Durgapur

November 11, 2025

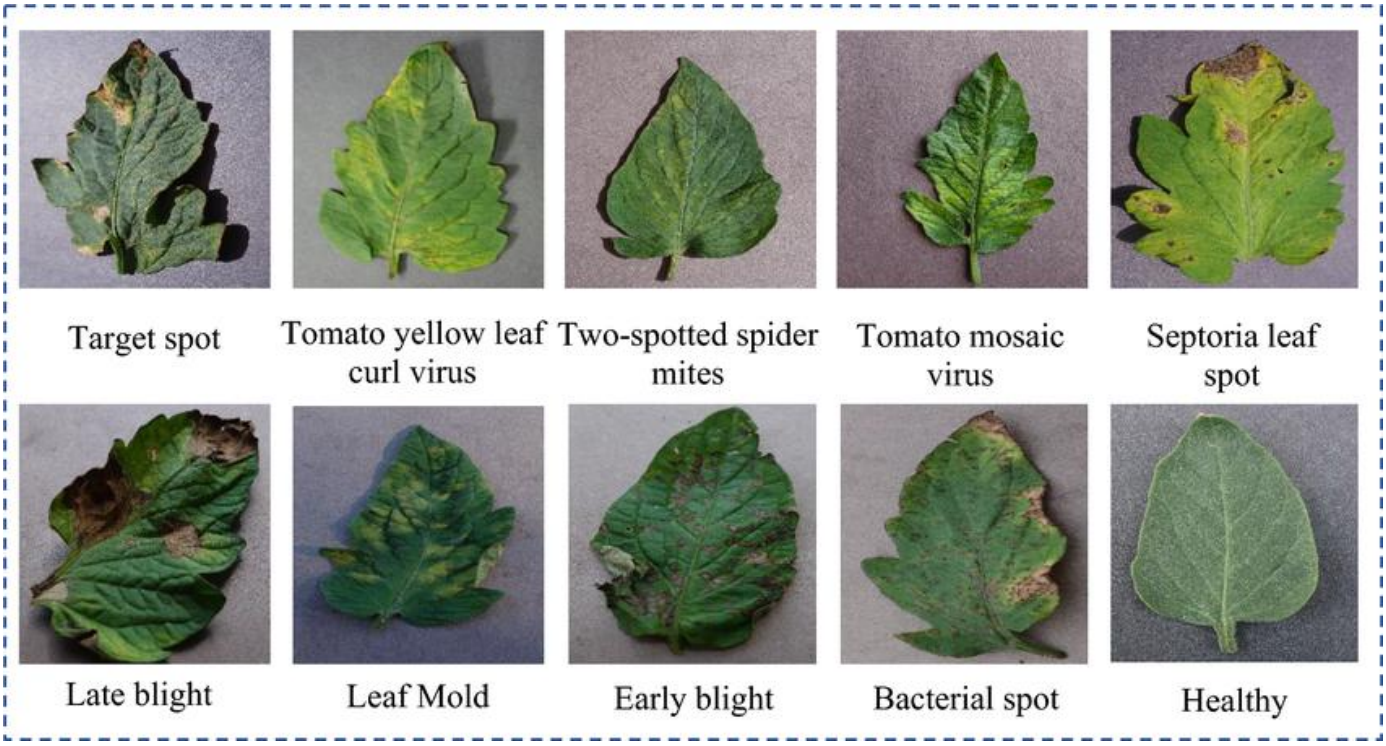
Contents

Plant Leaf Disease Detection

Dataset Description:

The PlantVillage dataset is a well-known benchmark in agricultural image classification. It contains 16,516 training and 4,122 validation images of plant leaves across 15 different classes. These classes include both healthy and diseased states
for multiple plants such as tomato, potato, and bell pepper. The images are taken under uniform lighting and background conditions to ensure consistency.

The dataset covers a wide range of common diseases such as bacterial spots, early blight, late blight, mold, septoria leaf spot, spider mites, and various viral infections.
This makes it an excellent resource for developing and benchmarking deep learning models for automated plant disease detection.



Each image was preprocessed by resizing and normalizing pixel values between 0 and 1. The balanced nature of the dataset across 15 classes ensures that the model learns robustly across multiple plant types.

Found 16516 images belonging to 15 classes.
Found 4122 images belonging to 15 classes.
📚 Number of classes: 15

📂 Classes in the Dataset:
01. Pepper__bell___Bacterial_spot (index: 0)
02. Pepper__bell___healthy (index: 1)
03. Potato___Early_blight (index: 2)
04. Potato___Late_blight (index: 3)
05. Potato___healthy (index: 4)
06. Tomato_Bacterial_spot (index: 5)
07. Tomato_Early_blight (index: 6)
08. Tomato_Late_blight (index: 7)
09. Tomato_Leaf_Mold (index: 8)
10. Tomato_Septoria_leaf_spot (index: 9)
11. Tomato_Spider_mites_Two_spotted_spider_mite (index: 10)
12. Tomato__Target_Spot (index: 11)
13. Tomato__Tomato_YellowLeaf__Curl_Virus (index: 12)
14. Tomato__Tomato_mosaic_virus (index: 13)
15. Tomato_healthy (index: 14)

Data Augmentation

Data augmentation played a critical role in improving the generalization capability of the models.
Techniques such as rotation, width and height shift, horizontal flipping, zooming, and brightness variation were applied dynamically during training.
This helped simulate real-world variations like leaf orientation, camera angle, and lighting changes, which are common in field conditions.

Augmentation ensures the model doesn't simply memorize training data but learns to detect diseases based on true feature patterns.

```python
IMG_SIZE = (128, 128)   # Reduce to fit in 2GB VRAM (MX130)
BATCH_SIZE = 16          # Keep small batch size to prevent OOM

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=25,
    width_shift_range=0.15,
    height_shift_range=0.15,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Example of Data Augmentation

Model Training and Evaluation using Normal CNN

Train and validation data split:

```python
train_gen = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

val_gen = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

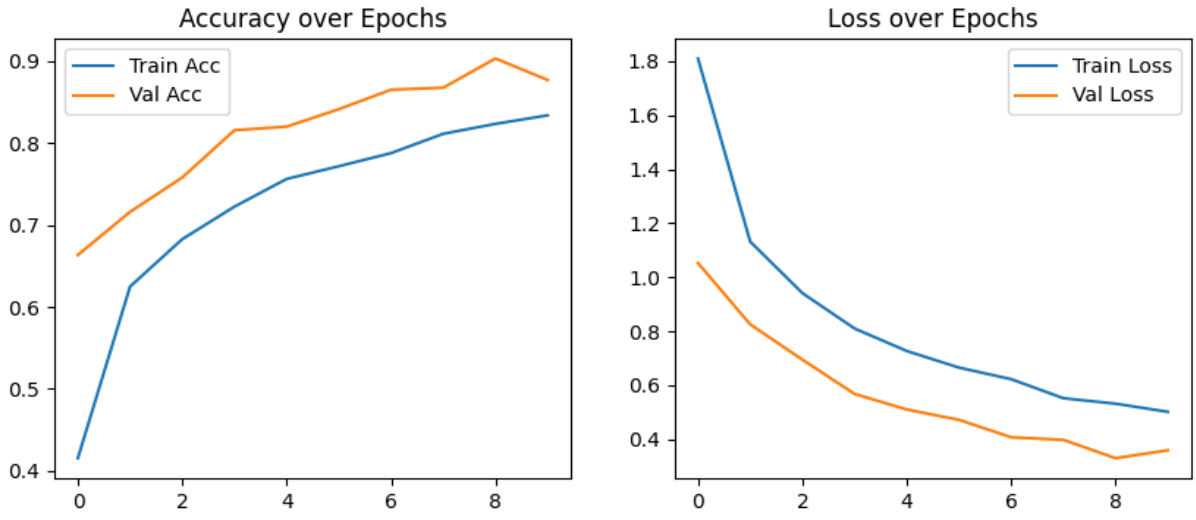
A Convolutional Neural Network (CNN) was built from scratch using several convolutional, pooling, and fully connected layers.
Model structure:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 128)               3211392

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 15)                1935

=================================================================
Total params: 3,306,575
Trainable params: 3,306,575
Non-trainable params: 0
```

This model was trained for 10 epochs on all 15 classes, achieving a validation accuracy of approximately 87.89 percent.

```
# ------------------------------------------------
# Compile the Model
# ------------------------------------------------
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# ------------------------------------------------
# Train the Model
# ------------------------------------------------
EPOCHS = 10

start = time.time()
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    verbose=1
```

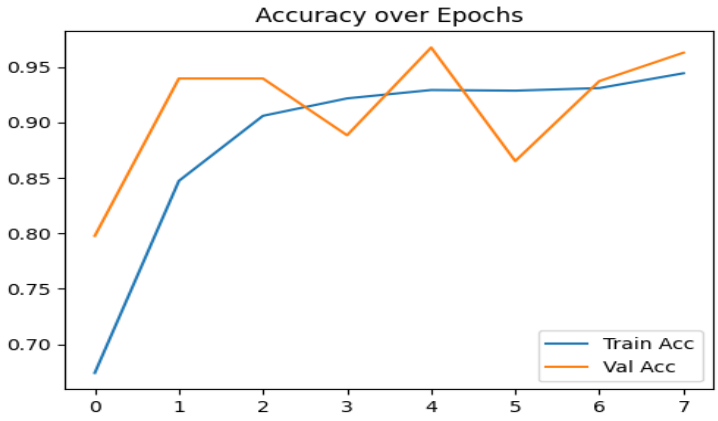
The training accuracy and validation accuracy increased steadily, while the training and validation losses decreased uniformly — a sign of effective learning without significant overfitting.

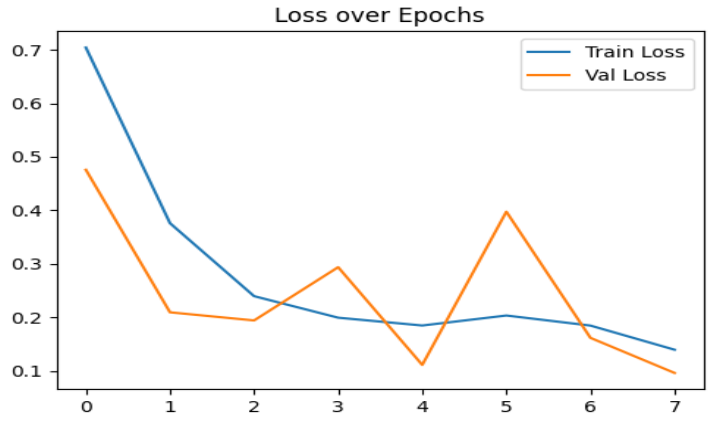


The architecture leveraged ReLU activation, dropout layers for regularization, and Adam optimizer for adaptive learning rate adjustment.

When trained on only three classes (Potato: Early Blight, Late Blight, and Healthy), the model achieved an impressive 96.05 percent validation accuracy.

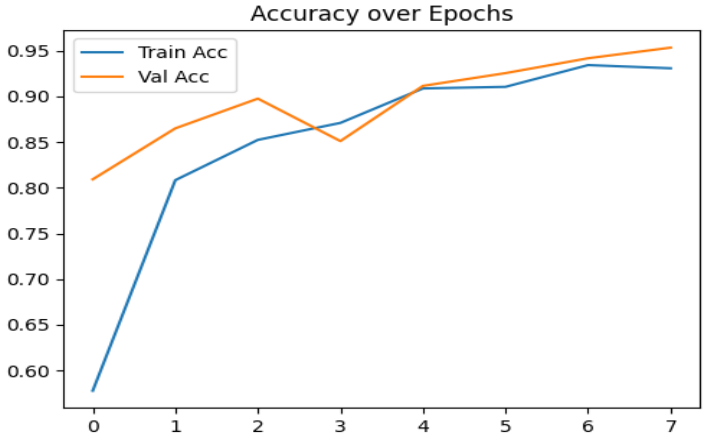
```
Found 1722 images belonging to 3 classes.
Found 430 images belonging to 3 classes.

Classes found: 3
01. Potato___Early_blight (index: 0)
02. Potato___Late_blight (index: 1)
03. Potato___healthy (index: 2)

Image counts per class:
00. Potato___Early_blight          -> 1000 images
01. Potato___Late_blight           -> 1000 images
02. Potato___healthy               ->  152 images

Total images: 2152
```

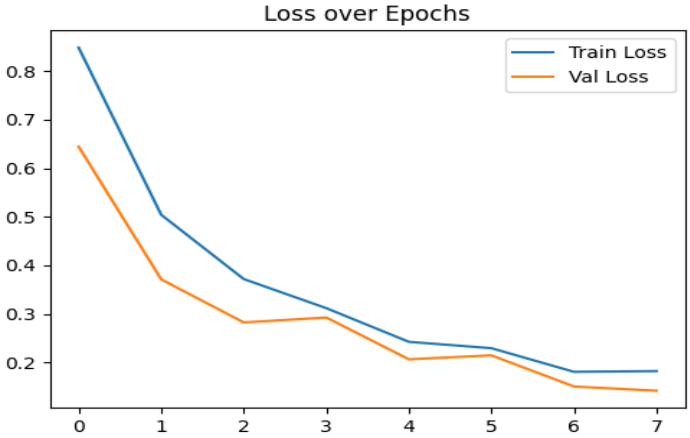




This highlights how reducing class complexity allows the network to converge faster and achieve higher precision.

Further optimization was achieved by lowering the learning rate to 1e-4, which reduced accuracy fluctuations and improved convergence stability.





Transfer Learning (EfficientNetB0, MobileNet, ResNet)

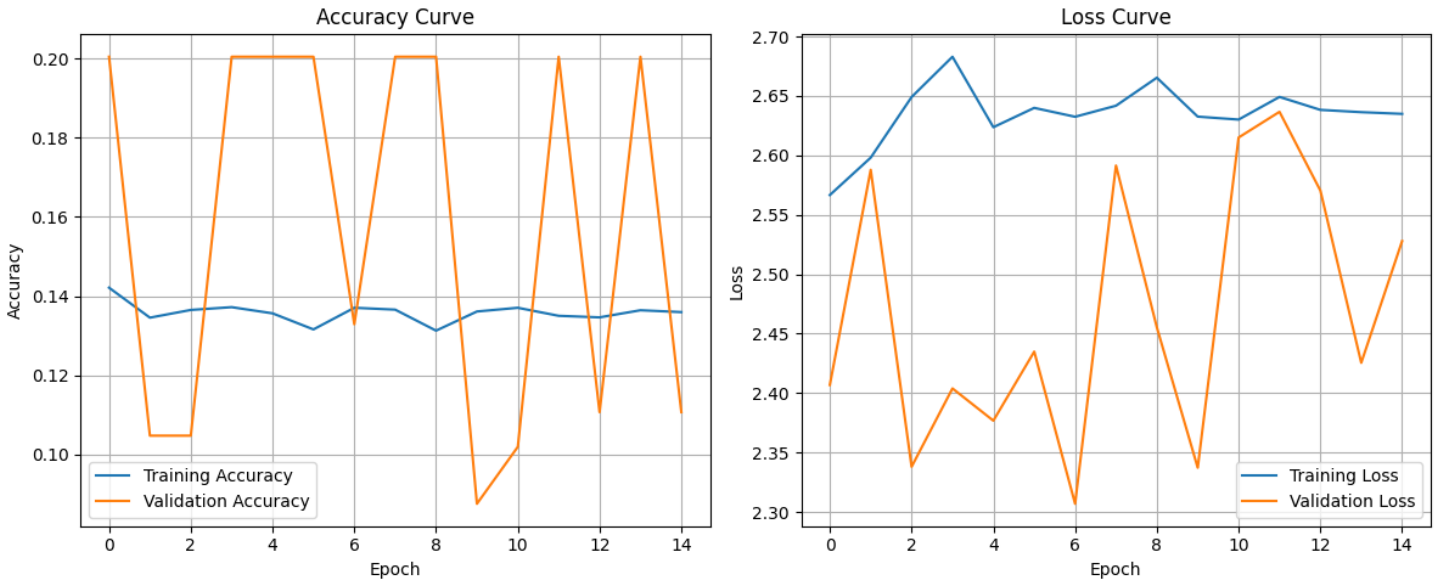
To improve performance and reduce training time, transfer learning was applied using three popular pretrained architectures - EfficientNetB0, MobileNetV2, and ResNet50V2 - initialized with ImageNet weights. Only the final classification layers were retrained on the plant disease dataset.

- **EfficientNetB0**: Showed limited adaptability to the dataset.

```
base_model = EfficientNetB0(weights='imagenet',include_top=False,
input_shape=img_size + (3,))
base_model.trainable = False  # Freeze pretrained layers
```

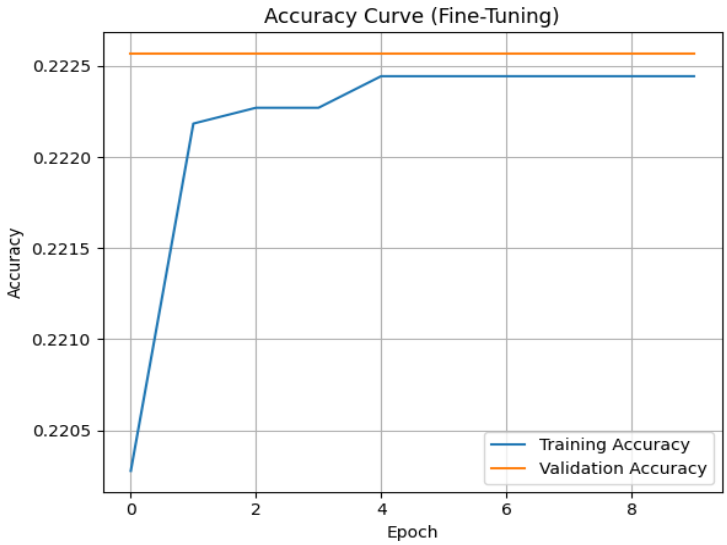
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb0 (Functional)  (None, 7, 7, 1280)       4049571

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1280)              0

 dense (Dense)               (None, 3)                 3843

=================================================================
Total params: 4,053,414
Trainable params: 3,843
Non-trainable params: 4,049,571
_________________________________________________________________
```

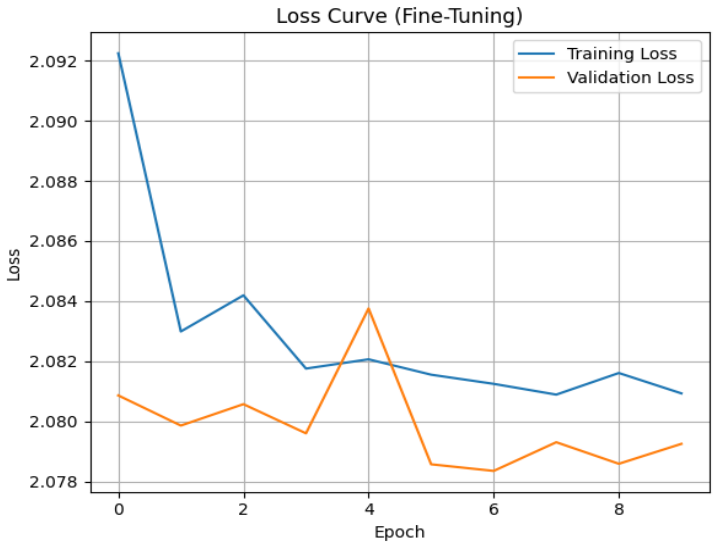
```
✅ Using 10 classes: ['Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_healthy', 'Tomato_Late_blight', 'Tomato_Leaf_Mold', 'Tomato_Septoria_leaf_spot', 'Tom
ato_Spider_mites_Two_spotted_spider_mite', 'Tomato__Target_Spot', 'Tomato__Tomato_mosaic_virus', 'Tomato__Tomato_YellowLeaf__Curl_Virus']
Found 12813 images belonging to 10 classes.
```



Even after fine-tuning (unfreezing the last 50 layers), accuracy improvements were minimal.

```
print("\nBuilding fine-tuning model...")
base_model = EfficientNetB0(weights='imagenet', include_top=False,
input_shape=img_size + (3,))


# 1. Unfreeze the base model
base_model.trainable = True


# 2. Freeze all layers *except* the last 50
unfreeze_last_n_layers = 50
freeze_until_layer = len(base_model.layers) - unfreeze_last_n_layers


print(f"Total layers in base_model: {len(base_model.layers)}")
print(f"Freezing all layers up to layer {freeze_until_layer}...")


for layer in base_model.layers[:freeze_until_layer]:
    layer.trainable = False
```





- **MobileNetV2:** With its lightweight structure (approximately 8.7 MB) and efficient depthwise separable convolutions, MobileNetV2 achieved stable and fast convergence.

Dataset balancing:

```
if "CHANGE/ME" in original_dataset_dir:
    print("="*50)
    print("ERROR: Please update the 'original_dataset_dir' variable")
    print("       to point to your unzipped 'PlantVillage' folder.")
    print("="*50)
    sys.exit()

print(f"Checking for balanced dataset at: {balanced_dir}")
if not os.path.exists(balanced_dir):
    print(f"Creating balanced dataset at: {balanced_dir}")
    os.makedirs(balanced_dir, exist_ok=True)
    for class_name in selected_classes:
        src_class_dir = os.path.join(original_dataset_dir, class_name)
```

```
        dst_class_dir = os.path.join(balanced_dir, class_name)
        os.makedirs(dst_class_dir, exist_ok=True)

        all_images = os.listdir(src_class_dir)
        random.shuffle(all_images)

        files_to_copy = all_images[:images_per_class] if len(all_images) >= images_per_class else all_images

        for img_file in files_to_copy:
            shutil.copy(os.path.join(src_class_dir, img_file),
                        os.path.join(dst_class_dir, img_file))
    print("✅ Balanced dataset created.")
else:
    print("✅ Balanced dataset already exists.")
```

Model structure for stage 1:

```
Building model with mobilenet base...
--- Model Summary (Stage 1: Head Training) ---
Model: "functional_1"
```

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dropout_1 (Dropout)	(None, 1280)	0
dense_1 (Dense)	(None, 9)	11,529

```
Total params: 2,269,513 (8.66 MB)
Trainable params: 11,529 (45.04 KB)
Non-trainable params: 2,257,984 (8.61 MB)
```

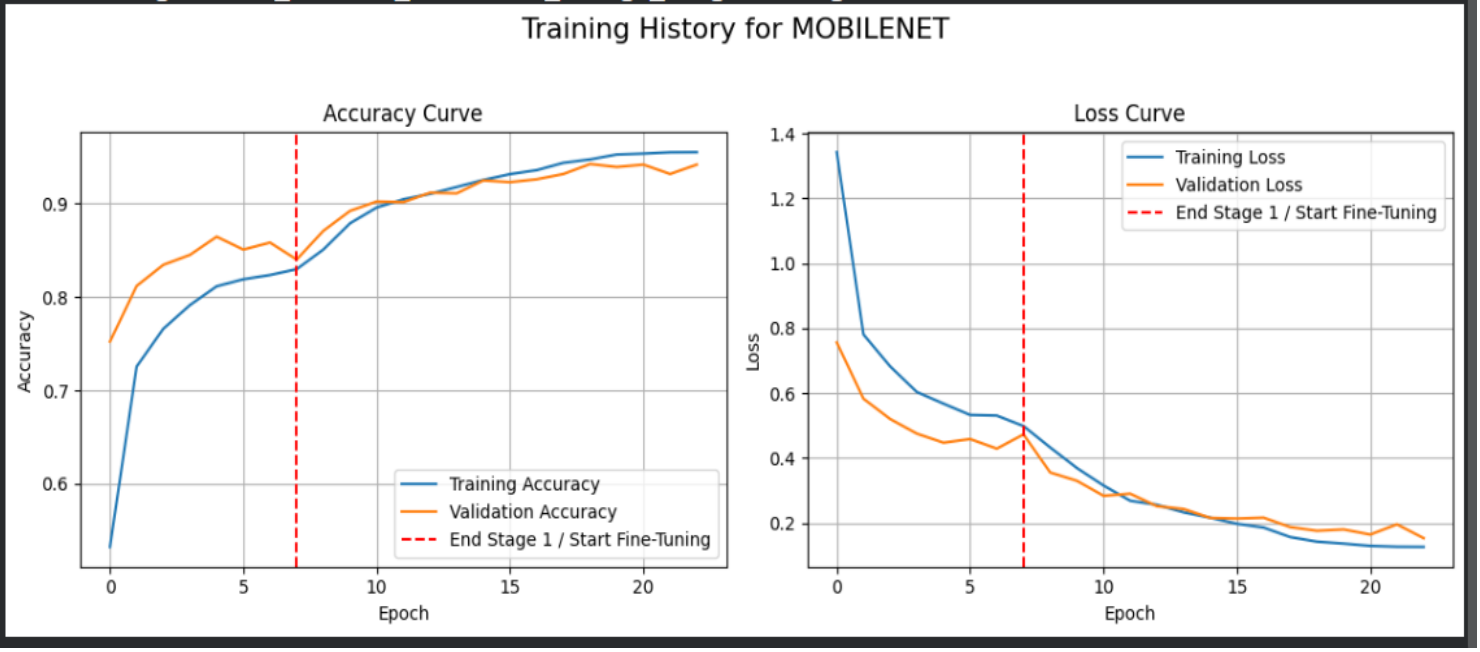
Model structure for stage 2:

```
--- Re-compiling for STAGE 2 (Fine-Tuning) ---
Total layers in base_model: 154
Freezing all layers up to layer 104...
--- Model Summary (Stage 2: Fine-Tuning) ---
Model: "functional_1"
```

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dropout_1 (Dropout)	(None, 1280)	0
dense_1 (Dense)	(None, 9)	11,529

```
Total params: 2,269,513 (8.66 MB)
Trainable params: 1,845,961 (7.04 MB)
Non-trainable params: 423,552 (1.62 MB)
```

```
✅ Final Validation Accuracy for mobilenet: 94.54%
✅ Final Validation Loss for mobilenet: 0.1613
Downloading tomato_disease_mobilenet_2stage_weights.weights.h5...
```



- **ResNet50V2:** With a deeper architecture (approximately 90 MB), ResNet showed strong feature extraction capabilities. However, it required more computational power and training time.

Dataset Balancing:

```
if "CHANGE/ME" in original_dataset_dir:
    print("="*50)
    print("ERROR: Please update the 'original_dataset_dir' variable")
    print("       to point to your unzipped 'PlantVillage' folder.")
    print("="*50)
    sys.exit()

print(f"Checking for balanced dataset at: {balanced_dir}")
if not os.path.exists(balanced_dir):
    print(f"Creating balanced dataset at: {balanced_dir}")
    os.makedirs(balanced_dir, exist_ok=True)
    for class_name in selected_classes:
        src_class_dir = os.path.join(original_dataset_dir, class_name)
```

```
        dst_class_dir = os.path.join(balanced_dir, class_name)
        os.makedirs(dst_class_dir, exist_ok=True)

        all_images = os.listdir(src_class_dir)
        random.shuffle(all_images)

        files_to_copy = all_images[:images_per_class] if len(all_images) >= images_per_class else all_images

        for img_file in files_to_copy:
            shutil.copy(os.path.join(src_class_dir, img_file),
                        os.path.join(dst_class_dir, img_file))
    print("✅ Balanced dataset created.")
else:
    print("✅ Balanced dataset already exists.")
```

Model structure for stage 1:

```
--- Model Summary (Stage 1: Head Training) ---
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
resnet50v2 (Functional)	(None, 7, 7, 2048)	23,564,800
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 9)	18,441

```
 Total params: 23,583,241 (89.96 MB)
 Trainable params: 18,441 (72.04 KB)
 Non-trainable params: 23,564,800 (89.89 MB)

--- Preparing for Model Training ---
--- Starting Model Training: STAGE 1 (Training Head) for 8 epochs ---
```

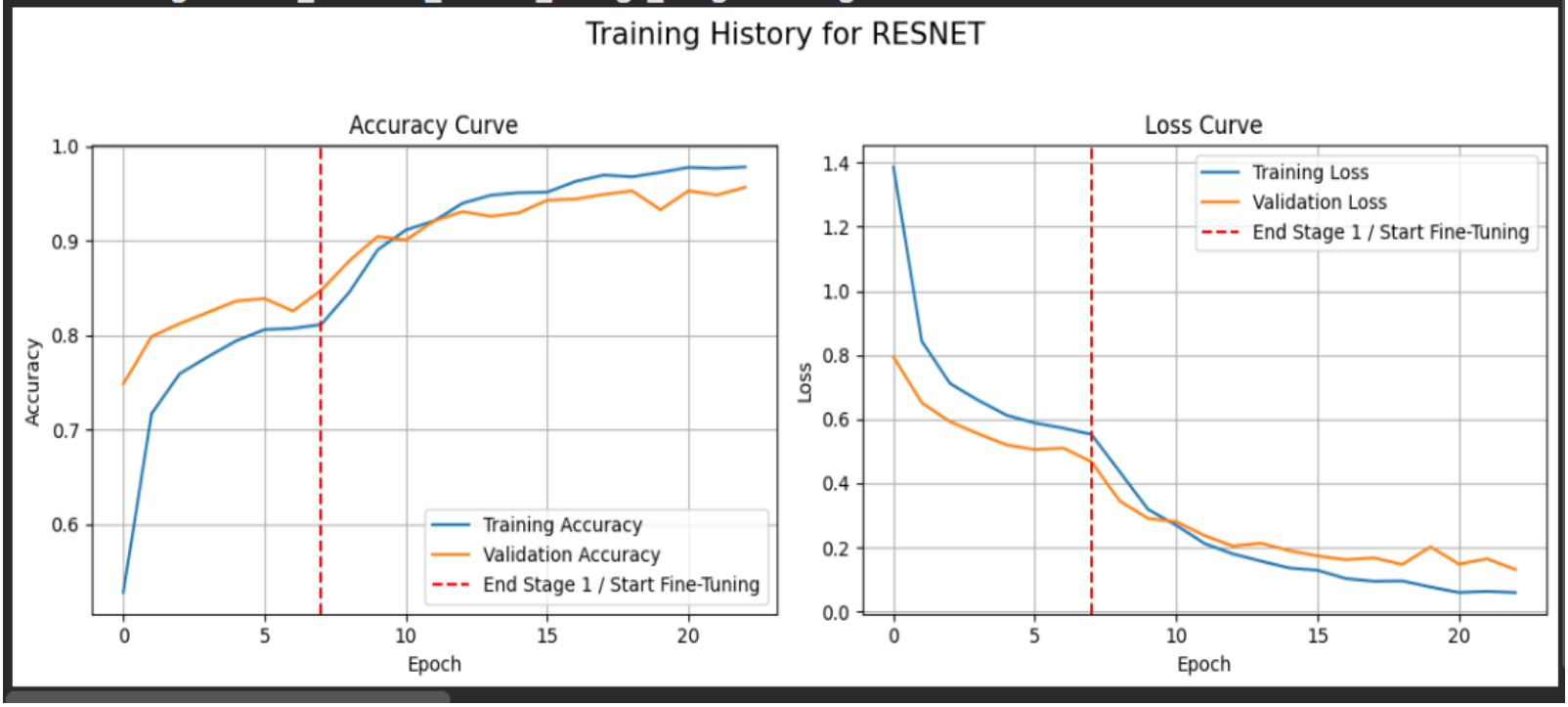
Model structure for stage 2:

```
--- Re-compiling for STAGE 2 (Fine-Tuning) ---
Total layers in base_model: 190
Freezing all layers up to layer 140...
--- Model Summary (Stage 2: Fine-Tuning) ---
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
resnet50v2 (Functional)	(None, 7, 7, 2048)	23,564,800
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 9)	18,441

```
 Total params: 23,583,241 (89.96 MB)
 Trainable params: 16,347,145 (62.36 MB)
 Non-trainable params: 7,236,096 (27.60 MB)

--- Starting Model Training: STAGE 2 (Fine-Tuning) for 15 epochs ---
```

```
✅ Final Validation Accuracy for resnet: 95.80%
✅ Final Validation Loss for resnet: 0.1409
Downloading tomato_disease_resnet_2stage_weights.weights.h5...
```



MobileNet vs ResNet inference:

```python
# Build model and load weights
    model_resnet = build_model('resnet', num_classes)
    weights_resnet = "tomato_disease_resnet_2stage_weights.h5"
    if not os.path.exists(weights_resnet):
        raise FileNotFoundError(f"Missing file: {weights_resnet}")

    model_resnet.load_weights(weights_resnet)
    print("ResNet50V2 weights loaded.")
```

```python
  # Build model and load weights
    model_mnet = build_model('mobilenet', num_classes)
    weights_mnet = "tomato_disease_mobilenet_2stage_weights.h5"
    if not os.path.exists(weights_mnet):
        raise FileNotFoundError(f"Missing file: {weights_mnet}")

    model_mnet.load_weights(weights_mnet)
    print("MobileNetV2 weights loaded.")
```

```
--- FINAL PERFORMANCE COMPARISON ---
-----------------------------------------
| Model       | Val. Accuracy | Val. Loss |
|-------------|---------------|-----------|
| MobileNetV2 | 93.94%        | 0.1602    |
| ResNet50V2  | 95.13%        | 0.1641    |
-----------------------------------------
```

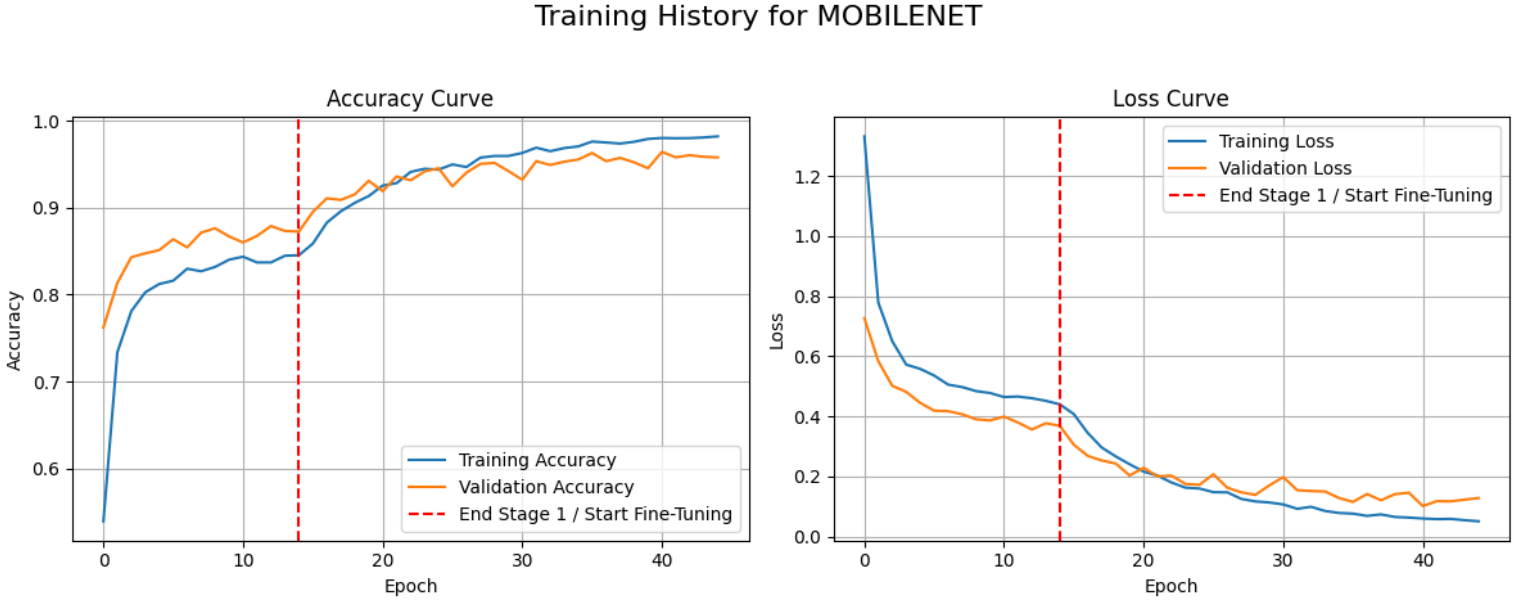
Bonus: Running the models for 15 + 30 epochs:

Resnet:

```
✅ Final Validation Accuracy for resnet: 95.98%
✅ Final Validation Loss for resnet: 0.1441
💾 Model saved successfully to tomato_disease_resnet_2stage_weights.h5!
```

MobileNet:

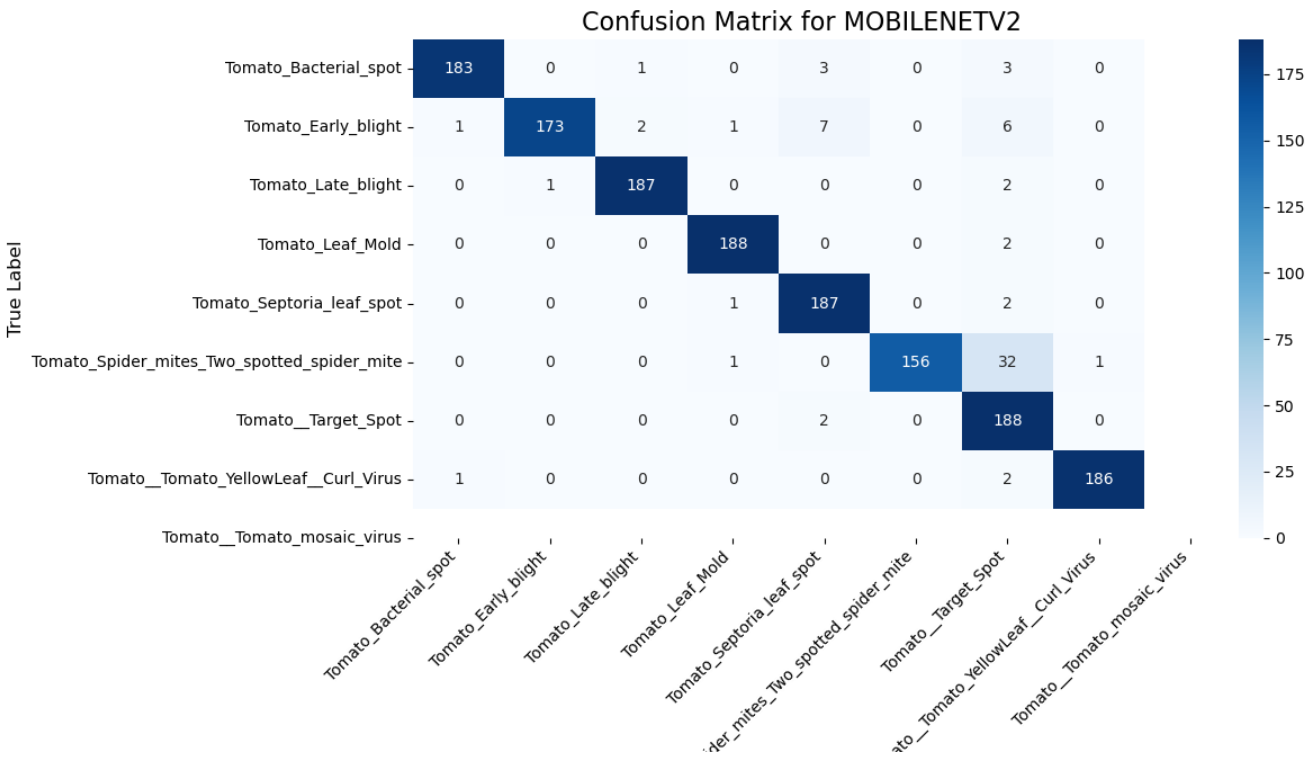
```
✅ Final Validation Accuracy for mobilenet: 95.67%
✅ Final Validation Loss for mobilenet: 0.1323
Downloading tomato_disease_mobilenet_2stage_weights.weights.h5.
```

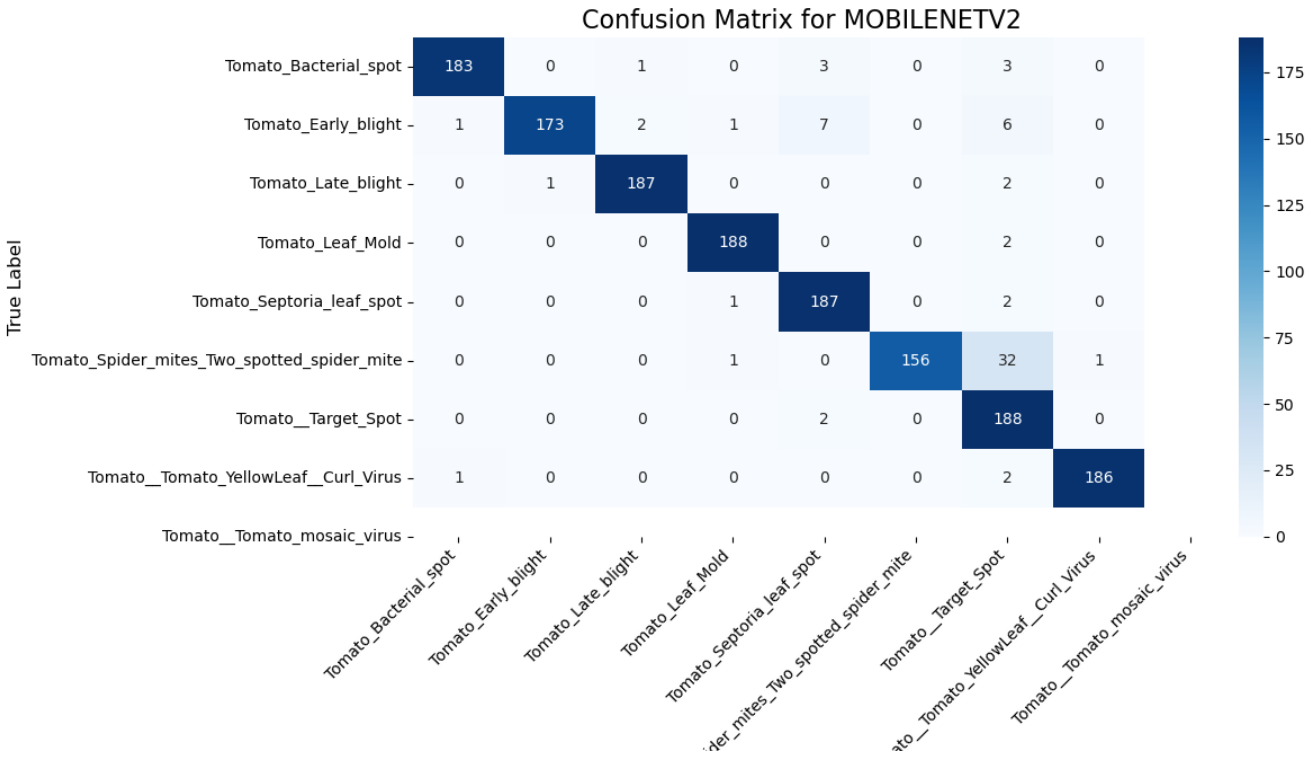


Experiments running these models for 15 and 30 epochs showed diminishing returns after 20 epochs,
indicating early convergence, signifying that these models (MobileNet & ResNet) are not much scalable.

Overall, MobileNet achieved the best trade-off between performance and computational efficiency.

📈 Confusion Matrix Analysis





To further diagnose performance, these matrices provide a class-by-class breakdown for the top two models.

- **Overall Performance:** The strong, dark-blue diagonal in both matrices visually confirms the high overall accuracy, representing correctly classified images.
- **MobileNetV2:** This model performed well but showed one significant area of confusion: **32 instances** of "Tomato_Spider_mites" were incorrectly predicted as "Tomato_Target_Spot".
- **ResNet50V2:** This model's largest single error was smaller, misclassifying **13 instances** of "Tomato_Target_Spot" as "Tomato_Spider_mites".

Key Insight: Both models struggle most with the distinction between **"Tomato_Target_Spot"** and **"Tomato_Spider_mites"**. However, ResNet50V2's errors are less concentrated (a worst error of 13 vs. 32 for MobileNet). This helps explain its slightly higher validation accuracy and reinforces the report's main conclusion: ResNet is marginally more accurate, but MobileNet offers the best trade-off for efficiency.

Insights and Conclusion

The comparative analysis of CNN and transfer learning models leads to the following insights:

1. Custom CNN models can perform exceptionally well when trained on a well-preprocessed dataset with balanced classes and appropriate hyperparameters.
2. Transfer learning models like MobileNet and ResNet are effective for rapid training and deployment, especially in low-data regimes.
3. EfficientNetB0 may require domain-specific fine-tuning or larger datasets to perform competitively on agricultural images.
4. MobileNet emerges as the best candidate for real-world applications due to its light architecture and good accuracy balance.
5. Further enhancements can be achieved by combining ensemble learning, attention mechanisms, or Grad-CAM-based explainability methods.

In conclusion, deep learning offers a powerful and scalable approach for early plant disease detection, which can significantly improve agricultural productivity and sustainability.