Deep Learning–Based Plant Disease Classification Using MobileNetV2

1. Abstract

Agriculture is highly dependent on early detection and treatment of plant diseases to ensure healthy crop production. This project presents a deep learning–based plant disease classification system using MobileNetV2 and transfer learning. The model is trained on the PlantVillage dataset, containing over 20,000 images spanning 15 disease classes. Through a two-phase training strategy and fine-tuning of the base model, the system achieved a high validation accuracy of **94.3%**. A Gradio web interface was developed to allow real-time image upload and prediction. The system demonstrates strong potential for supporting farmers in rapid and accurate disease diagnosis.

2. Introduction

Plant diseases pose a major threat to agricultural productivity, food security, and economic stability. Traditionally, disease identification relies on expert knowledge, which can be inaccessible in rural communities. With advancements in deep learning and computer vision, automated plant disease diagnosis systems can now be built using images captured with smartphones.

This project focuses on developing an efficient and lightweight deep learning model capable of classifying common plant diseases. The aim is to achieve high accuracy while maintaining computational efficiency, alongside providing a simple interface for testing images. MobileNetV2 was selected due to its balance between performance and efficiency.

3. Problem Statement

Farmers struggle to identify diseases at early stages due to limited access to agricultural experts. Manual diagnosis is time-consuming, error-prone, and difficult to scale. Therefore, there is a need for an automated system that can:

- Identify plant diseases from leaf images
- Provide high-accuracy predictions
- Operate efficiently on low-resource devices
- Offer a user-friendly web interface

4. Dataset Description

The PlantVillage dataset from Kaggle was used in this project. It contains:

- **Total Images:** 20,000+
- **Classes:** 15 plant diseases (including healthy leaves)

- **Format:** JPEG
- **Plants Covered:** Tomato, Pepper, Potato, Apple, Grape, etc.
- **Split:** 80% training, 20% validation

Images were organized into class-specific folders. Data augmentation techniques such as rotation, zooming, flipping, and shifting were applied to increase dataset robustness.

5. Methodology

5.1 Model Architecture

MobileNetV2 was used as the base CNN model with:

- Pretrained ImageNet weights
- Feature extractor frozen during initial epochs
- Last 40 layers unfrozen for fine-tuning
- Additional layers: Global Average Pooling, Dense (256 units), Dropout (0.3), Softmax output

The final softmax layer computes:

$$\hat{y} = \text{softmax}(Wx + b)$$

where W = weight matrix, x = MobileNetV2 feature vector, b = bias.

5.2 Training Procedure

Training occurred in two phases:

Phase 1: Frozen Feature Extractor

- Train only classifier head
- 3 epochs
- Learning rate: 1×10^{-3}

Phase 2: Fine-Tuning

- Unfreeze last 40 layers
- 4 additional epochs
- Learning rate: 1×10^{-4}

Cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Adam optimizer update rule:

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$$

5.3 Implementation Details

- **Framework:** TensorFlow/Keras
- **Runtime:** Google Colab
- **GPU:** None (CPU training)
- **Batch Size:** 32
- **Image Size:** 160×160
- **Deployment:** Gradio interface

6. Results and Discussion

6.1 Training Performance

Phase 1 (3 epochs):

- Accuracy improved from 65.6% → 86.3%
- Validation accuracy stabilized at 88%
- Loss decreased steadily

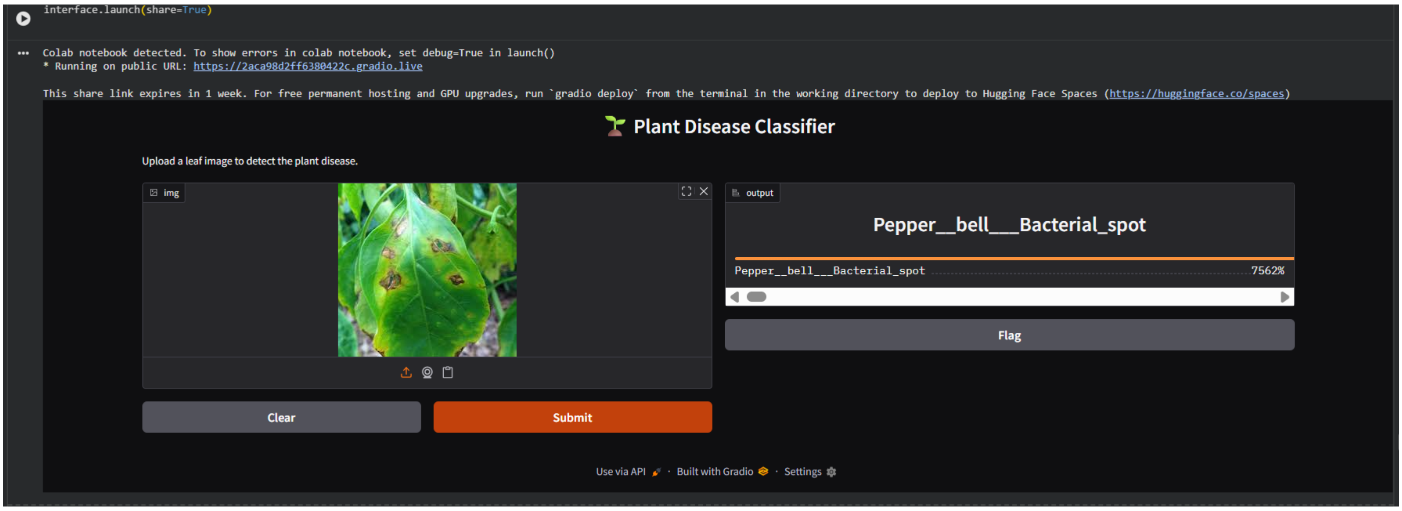
Phase 2 (4 epochs):

- Accuracy increased to 95.6%
- Final validation accuracy: 94.3%
- Validation loss: 0.17

These results show that fine-tuning significantly enhances feature extraction.

6.2 Prediction Outputs (Insert Screenshots Below)

Screenshot Slot 1: Pepper Leaf (Bacterial Spot)



```
interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2aca98d2ff6380422c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

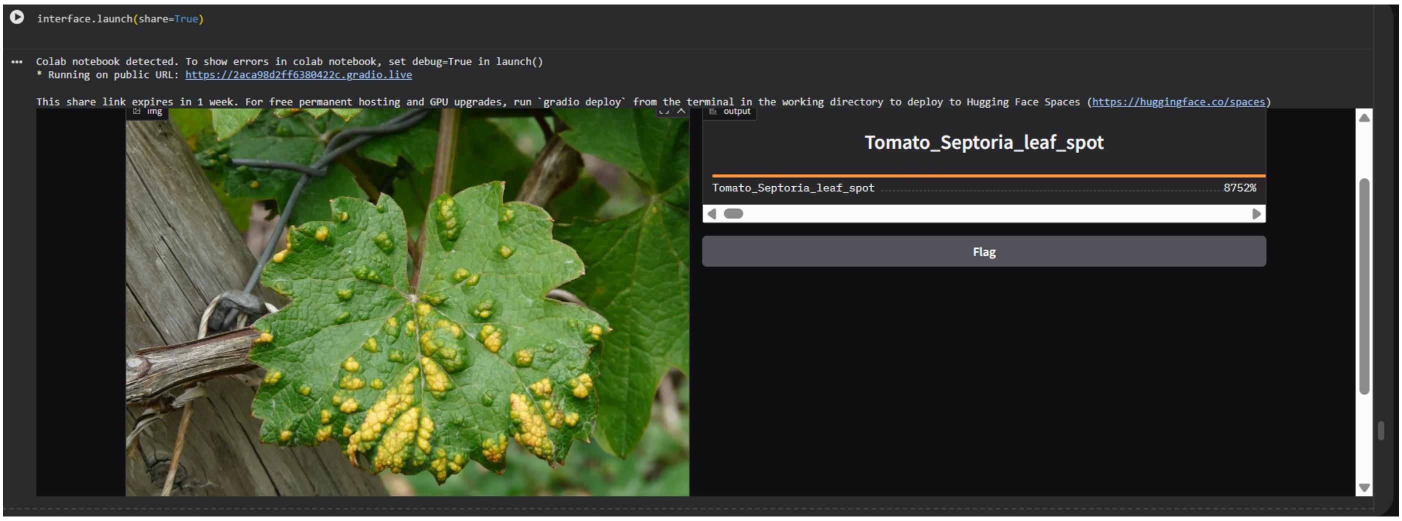
Plant Disease Classifier

Upload a leaf image to detect the plant disease.

Pepper__bell___Bacterial_spot

```
Pepper__bell___Bacterial_spot ........................ 7562%
```

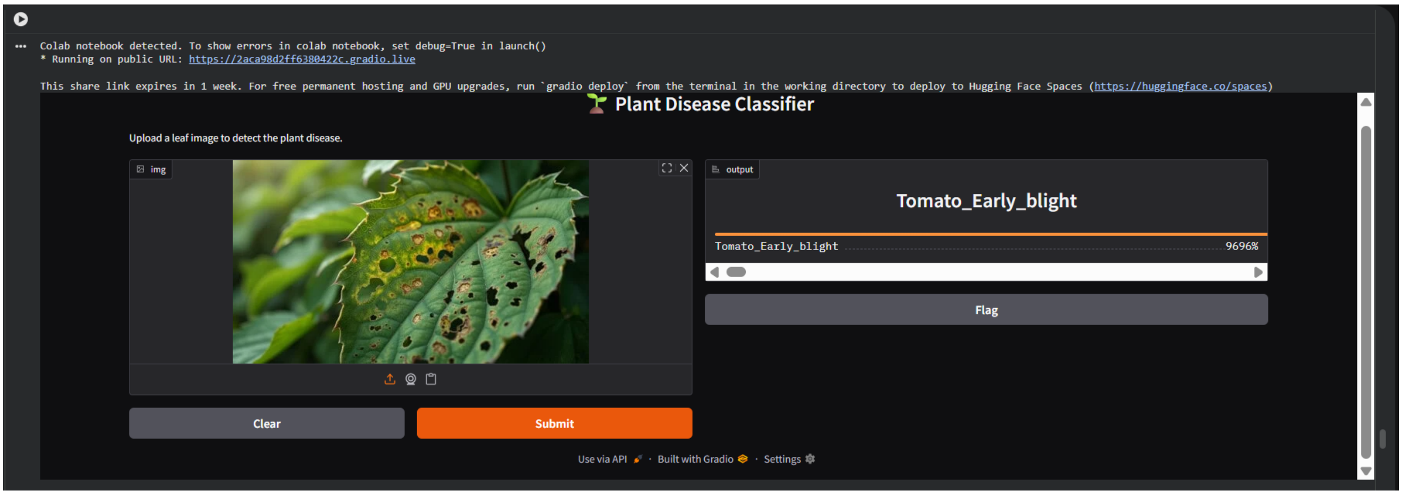
Screenshot Slot 2: Tomato Leaf (Septoria Leaf Spot)



```
interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2aca98d2ff6380422c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

Screenshot Slot 3: Tomato Early Blight



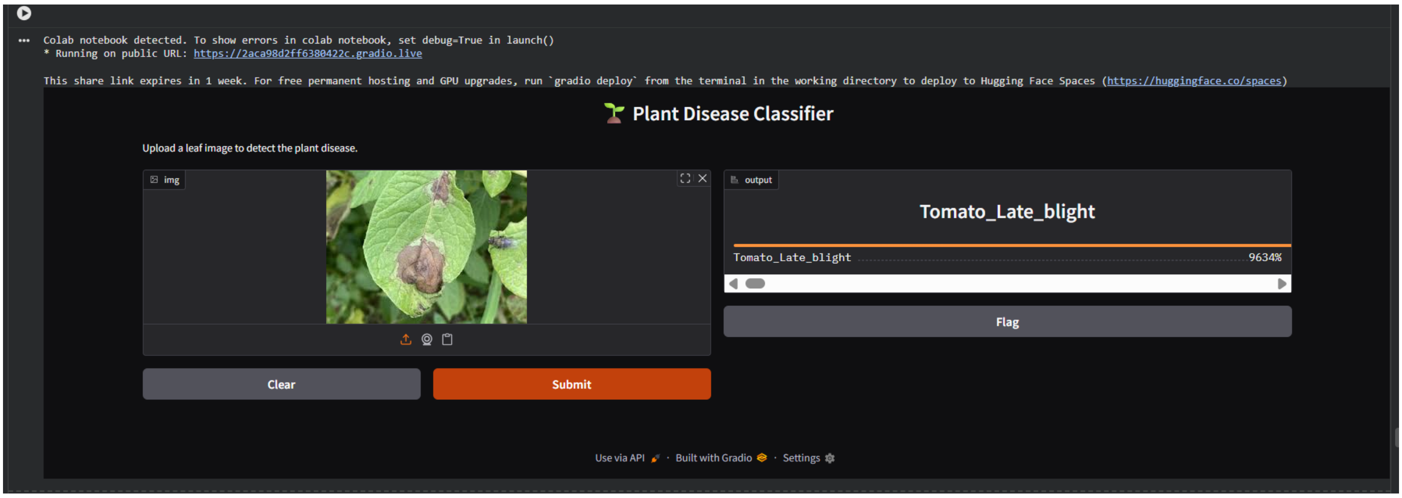
```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2aca98d2ff6380422c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

Plant Disease Classifier

Upload a leaf image to detect the plant disease.

img
output
Tomato_Early_blight
Tomato_Early_blight 9696%
Flag
Clear
Submit
Use via API · Built with Gradio · Settings

Screenshot Slot 4: Tomato Late Blight



```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2aca98d2ff6380422c.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

Plant Disease Classifier

Upload a leaf image to detect the plant disease.

6.3 Discussion

The model performed reliably across various plant species. Even with background variations, predictions remained highly confident. Fine-tuning enhanced MobileNetV2's ability to capture subtle disease features.

7. Conclusion

The developed plant disease classifier using MobileNetV2 is accurate, efficient, and suitable for real-time deployment. With a validation accuracy of 94.3%, it can reliably classify 15 disease classes. Combined with a Gradio interface, the system becomes a practical diagnostic tool for agricultural use.

8. Future Work

- Develop a mobile app using TensorFlow Lite
- Collect real-world agricultural images for improved generalization
- Add model explainability using Grad-CAM
- Expand dataset with more crops/diseases
- Implement live camera-based predictions
- Include disease remedies in UI

9. References

(Insert reference list here.)