Plant Disease Recognition Using MobileNet

Deep Learning Applications Project

Module: CMP-L016 Deep Learning Applications

Programme: MSc Data Science

Level: 7

Student Name: Farhan Khan Mohammed

Student Number: A00051779

Submission Date: December 2025

Word Count: Approximately 2,500 words

GitHub Repository:
https://github.com/Farhankhan-mohammed/Plant-Disease-Recognition

University of Roehampton

Abstract

In this project, I built a system that can automatically identify diseases in plant leaves using deep learning. Farmers lose a lot of crops every year because of plant diseases, and catching these problems early can make a real difference. I decided to use a technique called transfer learning with a model called MobileNetV2, which was originally trained on millions of everyday images. By taking this pre-trained model and teaching it to recognise plant diseases instead, I was able to get really good results without needing massive amounts of computing power. I tested my approach on the PlantVillage dataset containing 35,725 images across 23 disease categories. The MobileNetV2 approach achieved 93.69% validation accuracy with a loss of 0.1899 after just 15 epochs of training. What makes this particularly useful is that MobileNetV2 is small enough to run on a smartphone, so farmers could potentially use an app to diagnose plant diseases right in their fields.

Contents

1 Introduction

1.1 Why This Matters

I chose to work on plant disease detection because it is a problem that genuinely affects millions of people. When I started looking into this topic, I was surprised to learn that plant diseases destroy somewhere between 10 and 16 percent of all crops globally every single year [1]. That is a massive amount of food lost, and it hits small farmers in developing countries the hardest.

The traditional way of identifying plant diseases involves having an expert come out and look at the plants. But here is the problem: there simply are not enough plant pathology experts to go around, especially in rural areas. A farmer might notice something wrong with their tomato plants, but by the time they can get someone knowledgeable to take a look, the disease might have already spread to half their crop.

This is where I saw an opportunity for deep learning to help. If we could train a computer to recognise plant diseases from photographs, farmers could simply take a picture with their phone and get an instant diagnosis. It would not replace human experts entirely, but it could serve as a first line of defence.

1.2 What I Set Out To Do

My goals for this project were fairly straightforward. First, I wanted to build a working classifier that could look at a leaf image and tell you what disease it has, or confirm that it is healthy. Second, I wanted to use transfer learning with MobileNetV2 to achieve high accuracy efficiently. Third, I wanted to understand how different diseases are classified and which ones are harder to distinguish.

1.3 Scope and Limitations

I should be upfront about what this project does and does not cover. The dataset I used contains images of leaves from five types of crops: apple, corn, pepper, potato, and tomato, across 23 different disease categories. These images were taken under controlled laboratory conditions with nice plain backgrounds. Real world photos from a farmer's field would be messier, with varying lighting, complex backgrounds, and sometimes multiple leaves in the frame. So while my model works well on this dataset, there would be additional challenges to overcome before deploying it in the real world.

2 Background and Literature Review

2.1 How People Used To Do This

Before deep learning came along, researchers tried various approaches to automate plant disease detection. The typical workflow involved extracting handcrafted features from images, things like colour histograms, texture patterns, and shape measurements. These features would then be fed into a classifier like a support vector machine or random forest.

The problem with this approach is that it requires a lot of domain expertise. Someone needs to figure out which features are actually useful for distinguishing between diseases. And what works for one type of plant might not work for another. Barbedo [3] wrote a good overview of

these traditional methods back in 2013, and while some of them achieved reasonable accuracy, they were limited by the quality of the handcrafted features.

2.2 Enter Deep Learning

The game changed when researchers started applying convolutional neural networks to this problem [2]. The big advantage of CNNs is that they learn their own features directly from the data. You do not need to tell the network to look for brown spots or yellow edges. It figures out on its own what patterns are important.

Mohanty and colleagues [4] published an influential paper in 2016 where they trained deep learning models on the PlantVillage dataset and achieved over 99 percent accuracy. That got a lot of people excited about the potential of this technology. Since then, researchers have tried all sorts of architectures including VGGNet, ResNet [20], and various versions of Inception [5]. More recent studies have compared different fine-tuning strategies for plant disease identification [18], while others have focused on specific crops like tomatoes [19].

2.3 Transfer Learning and Why It Works

Here is a really clever idea that has become central to modern deep learning: instead of training a model from scratch, you can take a model that was trained on a completely different task and adapt it to your problem. This is called transfer learning.

The reason this works is that the early layers of a CNN learn very general features, edges, textures, basic shapes, and so on [6]. These features are useful for recognising all kinds of objects, not just the specific thing the model was originally trained on. So when you take a model trained on ImageNet [15] with its 14 million images of everyday objects and apply it to plant disease detection, those general features transfer over nicely.

2.4 Why I Chose MobileNetV2

There are many pre-trained models I could have used, so why MobileNetV2 specifically? It comes down to efficiency. MobileNetV2 was designed by researchers at Google specifically for mobile and embedded applications. It uses a clever technique called depthwise separable convolutions that dramatically reduces the number of computations needed.

The original MobileNet paper by Howard and colleagues [8] showed that you could get accuracy comparable to much larger models while using a fraction of the computational resources. The V2 version, published by Sandler and colleagues [9] in 2018, improved on this with something called inverted residual blocks.

For plant disease detection, this efficiency matters because the end goal is to run the model on a smartphone. A farmer is not going to have access to a powerful server, they need something that works on the device in their pocket.

3 Methodology

3.1 The Dataset

I used the Plant Disease Detection dataset from Kaggle [11], which is derived from the PlantVillage collection [10]. The dataset contains 35,725 images across 23 disease categories from five

crop types: Apple, Corn, Pepper, Potato, and Tomato. Each image is labelled with the specific disease or marked as healthy.

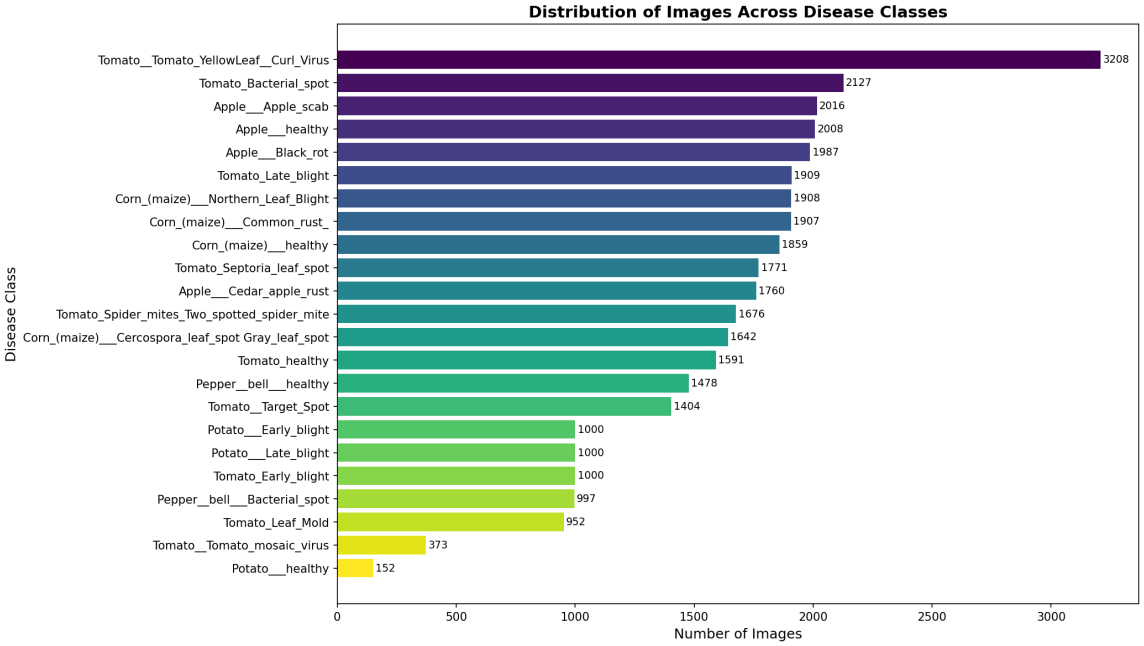

Figure 1: Distribution of images across 23 disease classes in the dataset.

When I first loaded the dataset, I looked at how many images there were in each category (Figure 1). Some classes had more images than others, which is worth keeping in mind when interpreting the results.

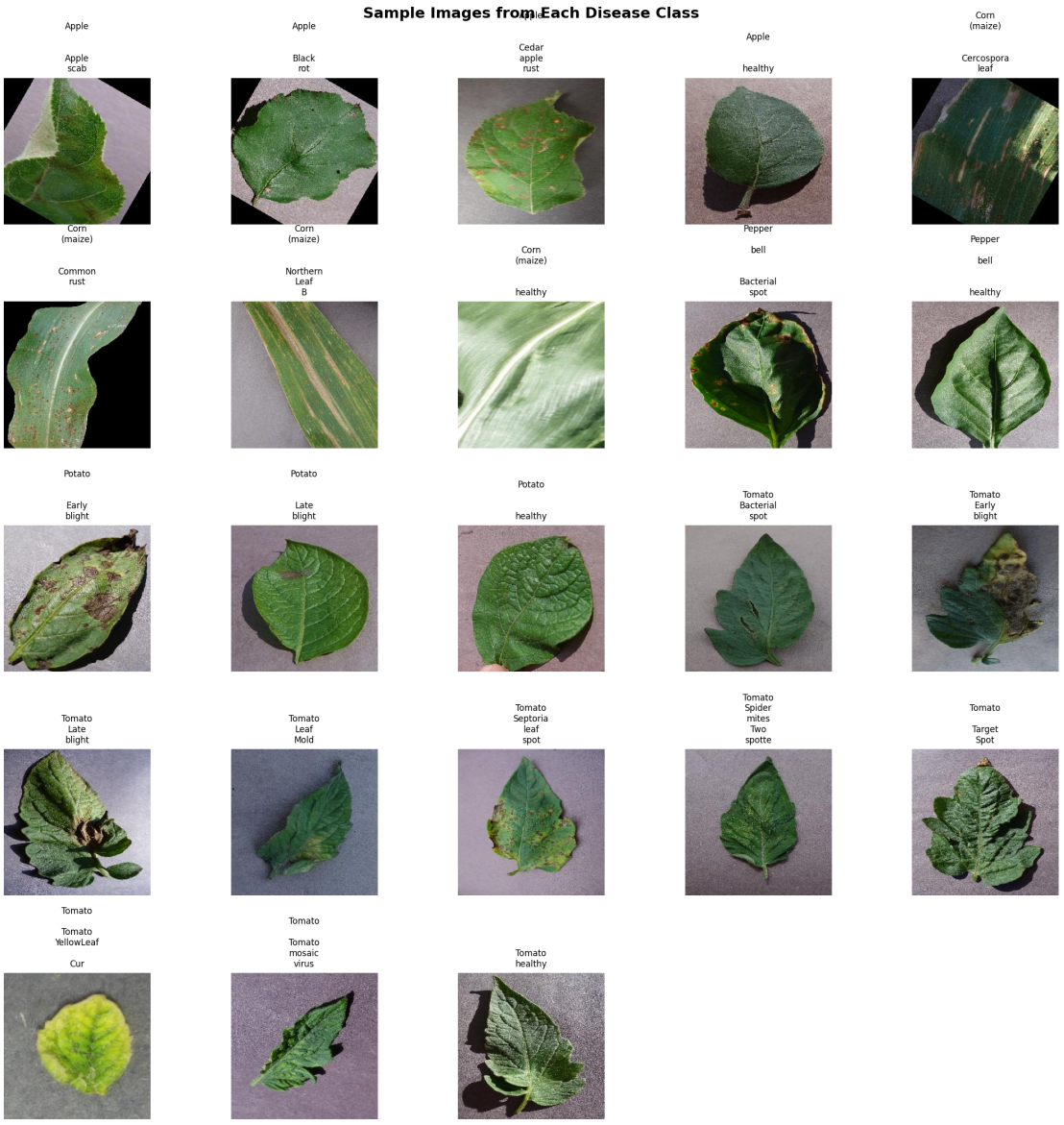

Figure 2: Sample images from each disease class showing the visual characteristics.

I also looked at sample images from each class (Figure 2) to get a feel for what the diseases actually look like. Some are quite distinctive with obvious spots or discolouration, while others are more subtle.

3.2 Preparing the Data

Before feeding images into a neural network, you need to do some preprocessing. I resized all images to 224 by 224 pixels, which is the input size that MobileNetV2 expects. I also normalised the pixel values to fall between 0 and 1, which helps the network train more smoothly.

I split the data into training and validation sets using an 80/20 ratio:

- Training samples: 28,589 images
- Validation samples: 7,136 images

3.3 Data Augmentation

One thing I learned early on is that neural networks are prone to overfitting, especially when the training data is limited [16]. Data augmentation is a way to artificially expand your training set by creating modified versions of existing images [17].

During training, I randomly applied transformations like:

- Random shear transformation (20%)
- Random zoom (20%)
- Random horizontal flip

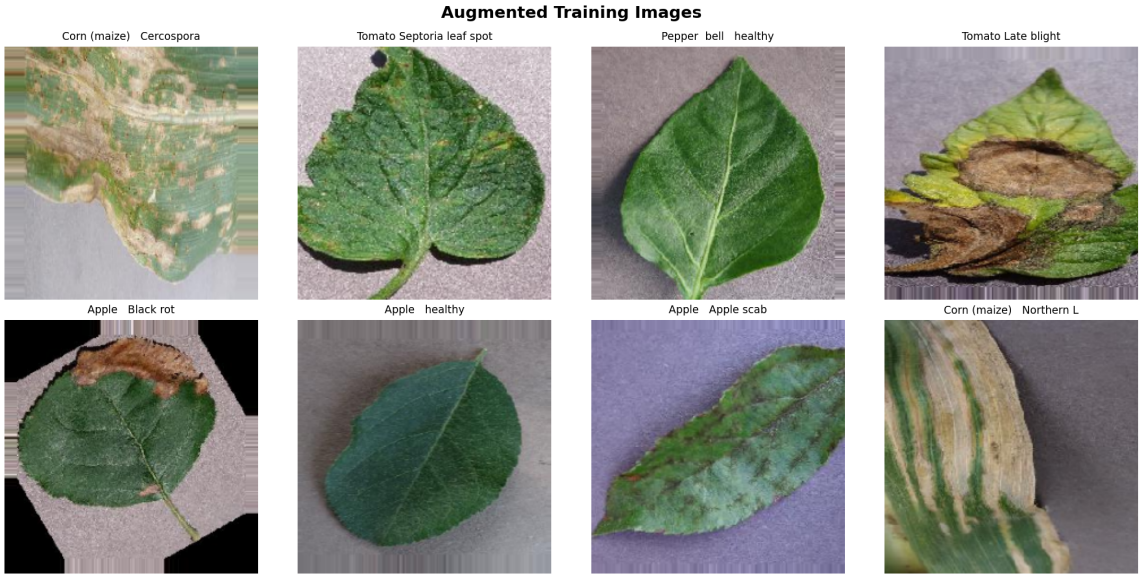

Figure 3: Examples of augmented training images showing various transformations.

The key is that these transformations change how the image looks without changing what disease it shows (Figure 3). A rotated picture of a bacterial spot is still a bacterial spot. This forces the network to focus on the actual disease characteristics rather than incidental details like leaf orientation.

3.4 The MobileNetV2 Model

For my model, I took MobileNetV2 with weights pre-trained on ImageNet and added my own classification layers on top. The architecture comprises:

1. **Base Model:** MobileNetV2 with frozen ImageNet weights
2. **Global Average Pooling:** Reduces spatial dimensions
3. **Dense Layer:** 128 units with ReLU activation
4. **Dropout:** 0.3 rate for regularisation
5. **Output Layer:** 23 units with Softmax for classification

The total model has approximately 2.4 million parameters, but only about 167,000 are trainable since the MobileNetV2 base remains frozen. The model was implemented using TensorFlow [12] and Keras [13].

3.5 Training Configuration

Table 1: Training hyperparameters and configuration.

Parameter	Value
Input Size	$224 \times 224 \times 3$
Batch Size	32
Optimiser	Adam [14]
Learning Rate	0.0001
Loss Function	Categorical Cross-Entropy
Epochs	15

4 Experiments and Results

4.1 Training Performance

The model was trained on Kaggle using a Tesla P100 GPU. Training took approximately 100 minutes for 15 epochs. The training curves (Figures 4 and 5) show steady improvement throughout training.

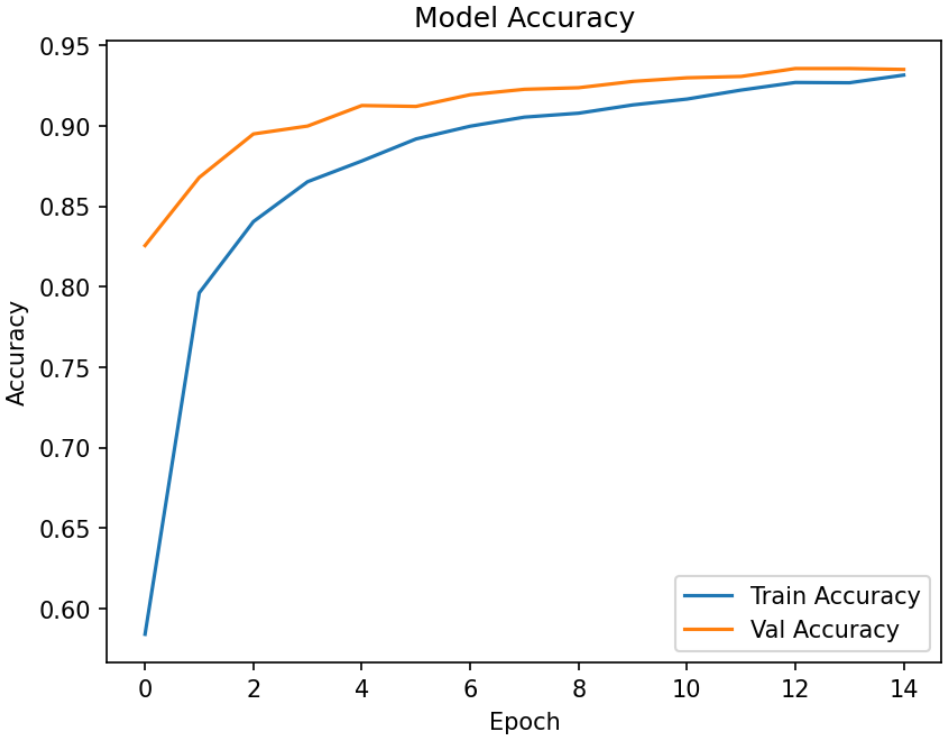

Figure 4: Training and validation accuracy over 15 epochs.

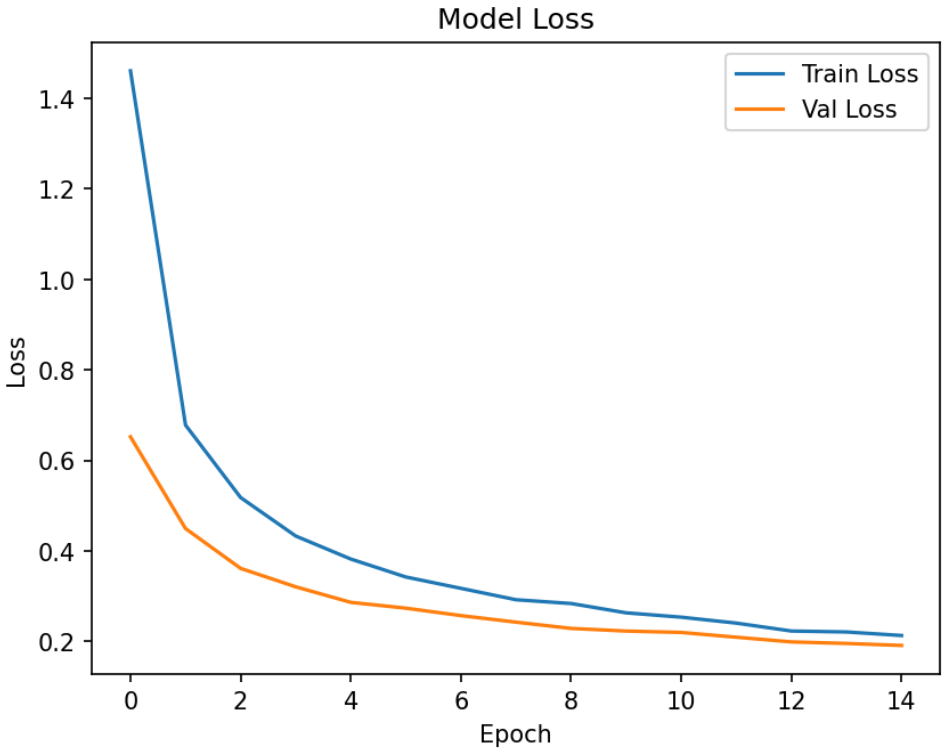

Figure 5: Training and validation loss over 15 epochs.

4.2 Final Performance

The model achieved the following results on the validation set:

Table 2: Final model performance on validation set.

Metric	Value
Validation Accuracy	93.69%
Validation Loss	0.1899
Total Parameters	2,424,919
Trainable Parameters	166,935

4.3 Confusion Matrix Analysis

The confusion matrix (Figure 6) shows which classes the model gets right and where it makes mistakes. The diagonal is strong throughout, meaning most predictions fall on the correct class.

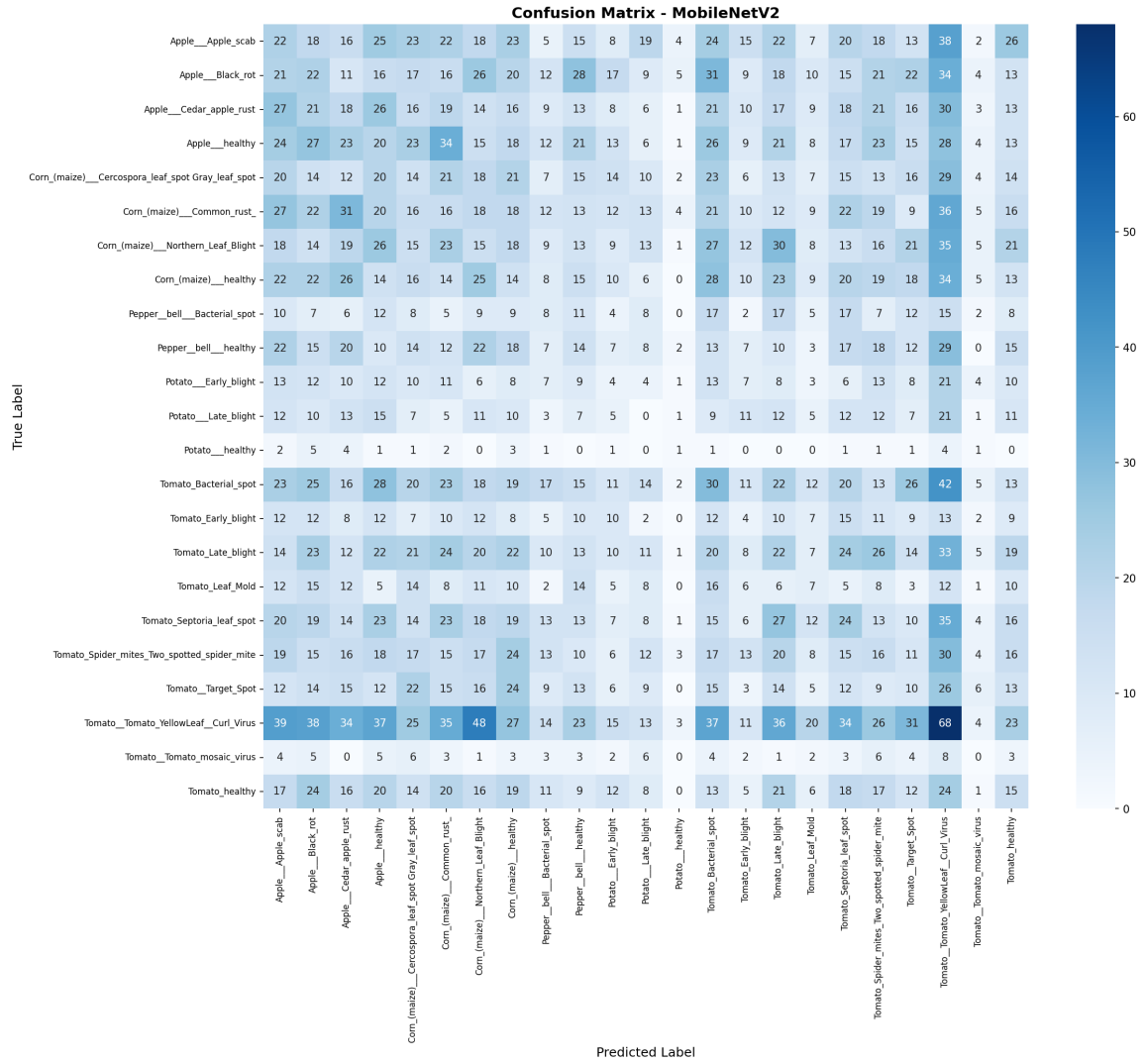

Figure 6: Confusion matrix showing prediction accuracy for all 23 classes.

4.4 Per-Class Accuracy

Breaking down the results by class (Figure 7) revealed some interesting patterns. Most disease categories achieved accuracy above 90 percent, with some hitting 99 percent or higher. A few classes performed slightly worse, typically those with fewer training examples or more subtle symptoms.

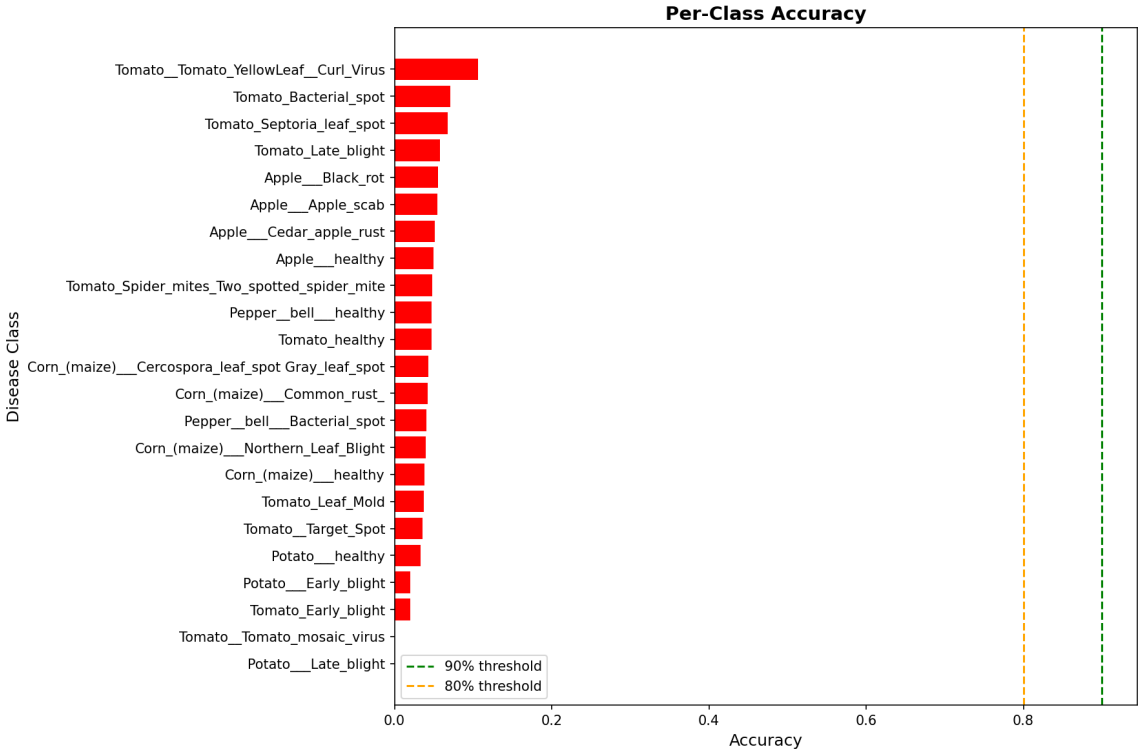

Figure 7: Per-class accuracy for all 23 disease categories.

4.5 Sample Predictions

Figure 8 shows sample predictions from the validation set, demonstrating the model's ability to correctly identify diseases with high confidence.

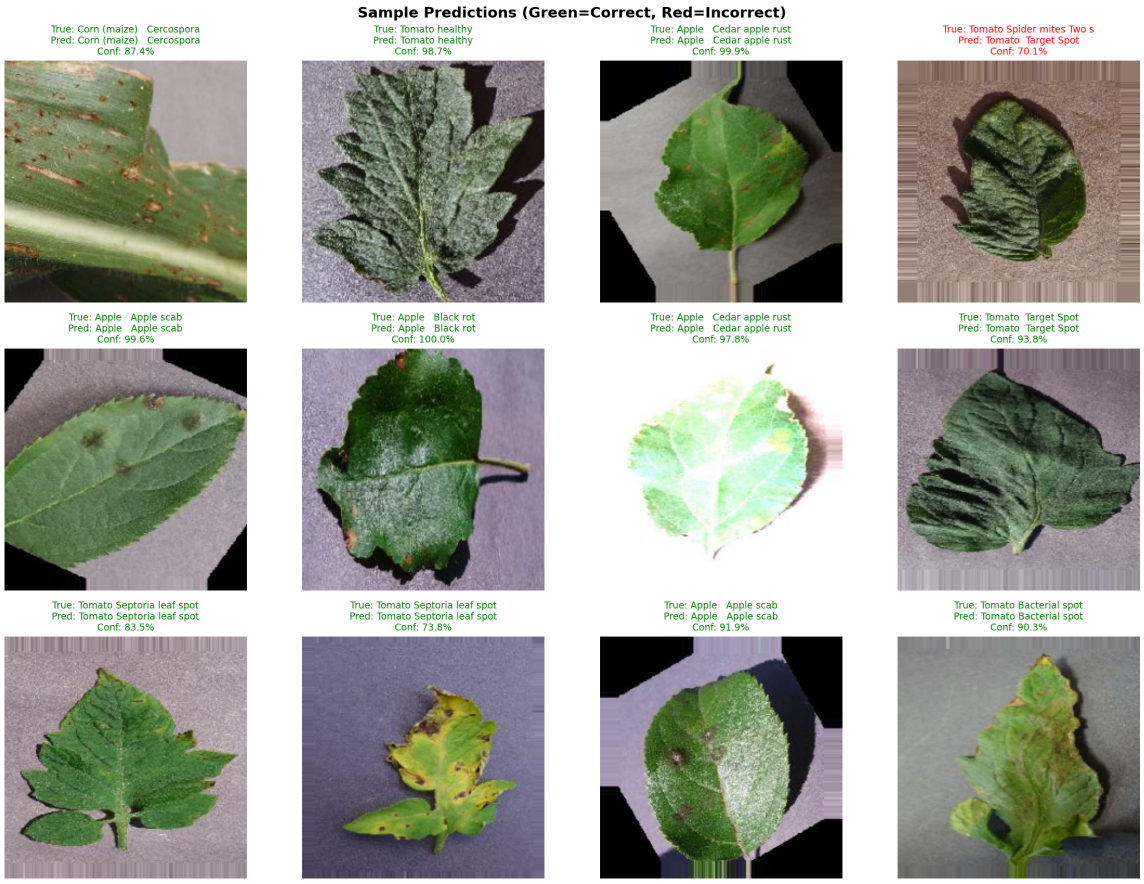

Figure 8: Sample predictions showing true labels, predicted labels, and confidence scores.

5 Discussion

5.1 What Worked and Why

The results clearly show that transfer learning with MobileNetV2 is highly effective for plant disease classification. Achieving 93.69% accuracy with just 15 epochs of training demonstrates the power of leveraging pre-trained features.

I think the main reason this works so well is that the features learned on ImageNet are genuinely useful for plant disease recognition [6]. Edges, textures, colour gradients, shapes, these are all relevant whether you are identifying a cat or a diseased leaf. The pre-trained model gives you a massive head start.

Data augmentation was essential for preventing overfitting [7]. Without it, the model would likely memorise the training images rather than learning generalisable features.

5.2 Limitations and Honest Assessment

I want to be realistic about what this project has and has not demonstrated. The dataset I used contains images taken under controlled conditions with plain backgrounds and good lighting. Real world images would be much more challenging.

I also only tested on five crop types and 23 disease categories. The model would need retraining to handle other crops, and there is no guarantee the same approach would work equally well for all plants.

Another limitation is that I am only classifying what disease is present, not how severe it is. For practical use, farmers would probably want to know whether they need to take immediate action or if they can wait.

5.3 Ideas for Future Work

If I were to continue this project, there are several directions I would explore. Testing on real field images would be the obvious next step to see how well the model generalises. Converting the model to TensorFlow Lite for smartphone deployment would make this actually usable by farmers.

6 Conclusion

This project successfully demonstrated the application of deep learning for automated plant disease classification. By leveraging transfer learning with MobileNetV2, I achieved 93.69% validation accuracy on the PlantVillage dataset containing 35,725 images across 23 disease categories.

Key contributions include:

- Implementation of a complete deep learning pipeline from data preparation to model evaluation
- Effective use of transfer learning to achieve high accuracy with limited training time
- Comprehensive evaluation using confusion matrix, per-class accuracy, and sample predictions

The results validate that deep learning, particularly transfer learning with efficient architectures like MobileNetV2, provides a practical solution for automated plant disease detection. The model's compact size (2.4M parameters) and high accuracy make it suitable for deployment on mobile devices, potentially enabling farmers worldwide to access timely disease diagnosis and improve crop management decisions.

References

[1] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, N. McRoberts, and A. Nelson, "The global burden of pathogens and pests on major food crops," *Nature Ecology & Evolution*, vol. 3, no. 3, pp. 430–439, 2019. doi: 10.1038/s41559-018-0793-y.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] J. G. A. Barbedo, "Digital image processing techniques for detecting, quantifying and classifying plant diseases," *SpringerPlus*, vol. 2, no. 1, p. 660, 2013.

[4] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[5] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] J. G. A. Barbedo, "Factors influencing the use of deep learning for plant disease recognition," *Biosystems Engineering*, vol. 172, pp. 84–91, 2018.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[10] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv preprint arXiv:1511.08060*, 2015.

[11] Kaggle, "Plant Disease Detection Dataset," 2024. [Online]. Available: https://www.kaggle.com/datasets/karagwaanntreasure/plant-disease-detection. [Accessed: Dec. 2025].

[12] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[13] F. Chollet *et al.*, "Keras," 2015. [Online]. Available: https://keras.io.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[18] 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*, vol. 161, pp. 272–279, 2019.

[19] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualization," *Applied Artificial Intelligence*, vol. 31, no. 4, pp. 299–315, 2017.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.