Lab Report: **Plant Disease Classification Using CNN**

Ego Victor Chibueze
12401308
Advanced Programming
Instructor: Prof. Tobias Schaffer

11.07.2025

1 Introduction

This lab involved building a deep learning model capable of identifying plant diseases based on images of leaves. The dataset, a subset of the PlantVillage collection, was divided into labeled training samples and unlabeled test samples. The primary aim was to design a convolutional neural network (CNN) that could accurately classify plant diseases and generalize to new images.

2 Methodology

The project was developed in Python, utilizing TensorFlow and Keras. Starting from a basic CNN, several refinements were made to improve both accuracy and generalization.

2.1 Tools and Environment

- Language: Python 3.11
- Libraries: TensorFlow, Keras, NumPy, scikit-learn, matplotlib
- Platform: Google Colab
- Hardware: NVIDIA Tesla T4 GPU (via Colab)

2.2 Source Code

The complete implementation can be accessed via GitHub:

`https://github.com/Egovictorc/dit/tree/main/SUM_2025/Advanced_Programming`

2.3 Model Architecture

Here is the final version of the CNN used:

```
model = Sequential([
    Input(shape=(256, 256, 3)),

    Conv2D(32, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D((2, 2)),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),

    Dense(train_generator.num_classes, activation='softmax')
])
```

The model was compiled with the Adam optimizer and categorical crossentropy loss function. Training was performed over 9 epochs with a batch size of 32, using an 80/20 training-validation split. Image normalization was applied as the only preprocessing step.

3 Results

The training process yielded the following results:

- **Training Accuracy (final)**: 94.29%
- **Validation Accuracy (final)**: 92.81%
- **Training Loss (final)**: 0.1781
- **Validation Loss (final)**: 0.2873

Figure 1 shows the model's learning dynamics.

Accuracy Analysis:

- Training accuracy consistently improved, reaching close to 97%.
- Validation accuracy fluctuated between 91% and 93%, lacking a consistent upward trend.
- This suggests that although the model fit the training data well, it started to overfit during later epochs.

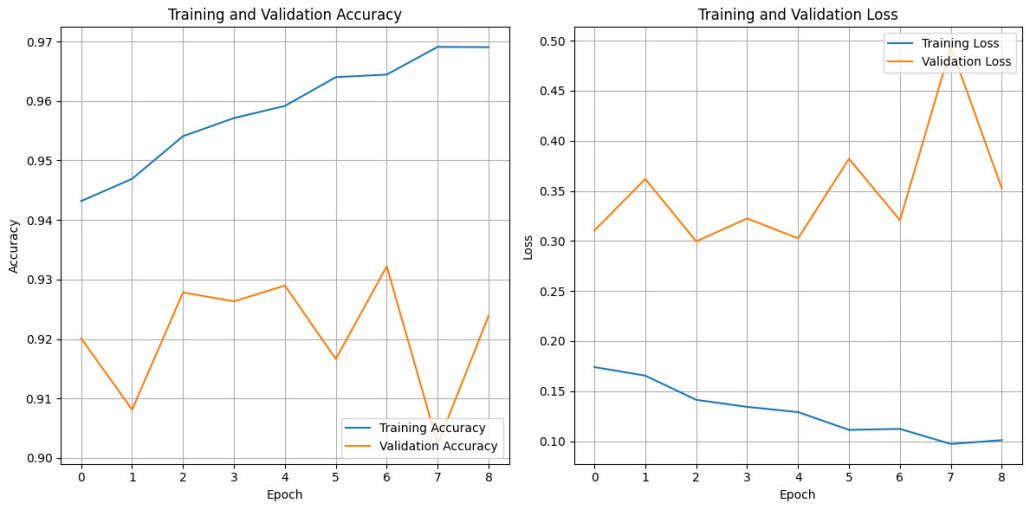

Figure 1: Left: Accuracy during training and validation. Right: Training and validation loss over 9 epochs.

Loss Analysis:

- Training loss steadily decreased, indicating strong learning on the training set.
- Validation loss varied and even increased towards the final epochs.
- This disparity implies overfitting, as the model starts to memorize rather than generalize.

3.1 Prediction Samples

The following are the model's predictions for the first ten test images:

Image	Predicted Disease
00000.JPG	Orange___Haunglongbing
00001.JPG	Corn___Northern_Leaf_Blight
00002.JPG	Peach___Bacterial_spot
00003.JPG	Grape___healthy
00004.JPG	Tomato___Tomato_Yellow_Leaf_Curl_Virus
00005.JPG	Grape___Black_measles
00006.JPG	Orange___Haunglongbing
00007.JPG	Tomato___Late_blight
00008.JPG	Orange___Haunglongbing
00009.JPG	Tomato___Septoria_leaf_spot

While the model demonstrated high accuracy on seen data, there is still room for generalization improvements.

4 Challenges, Limitations, and Observations

4.1 Challenges Encountered

- Fine-tuning the CNN architecture for better performance.
- Staying within Colab's runtime and memory constraints.
- Balancing model complexity and generalization capacity.

4.2 Error Analysis

- Initial results were poor (around 41% accuracy) due to shallow network depth.
- Similar-looking diseases likely caused classification errors.
- Training for more than 9 epochs didn't yield better validation results, indicating learning saturation.

4.3 Model Limitations

- No use of advanced augmentation (rotation, brightness, flip, etc.), which might affect real-world generalization.
- The model relied solely on a custom CNN instead of leveraging pretrained models like MobileNet or EfficientNet.
- The test set remained unlabeled, limiting the scope of evaluation.

5 Discussion

The architecture refinements — such as added convolutional layers and Dropout — significantly improved the network's learning ability. However, validation accuracy did not improve after a certain point, signaling overfitting.

Despite attempts to extend training, the model's performance reached a ceiling by the 9th epoch. The results suggest that improvements would more likely come from data-level strategies (augmentation) or model-level enhancements (transfer learning).

6 Conclusion

This lab demonstrated how convolutional neural networks can effectively identify plant leaf diseases from images. Without using pretrained models, the custom network achieved over 92% validation accuracy, indicating strong learning potential under constrained resources.

To further improve performance, strategies such as extensive data augmentation, use of pretrained models, and better regularization techniques are recommended. Nonetheless, this implementation sets a solid groundwork for practical applications in plant health monitoring systems.

7 References

- PlantVillage Dataset: `https://www.kaggle.com/datasets/emmarex/plantdisease`
- Keras Model Saving Guide: `https://keras.io/guides/serialization_and_saving/`
- Advanced Programming Course, Prof. Tobias Schaffer, DIT Summer 2025