PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MINISTRY OF POST AND TELECOMMUNICATIONS

UNIVERSITY OF ABDELHAMID MEHRI – CONSTANTINE 2
IN COLLABORATION WITH ALGÉRIE TÉLÉCOM

Internship Report in Artificial Intelligence

Image Classification using Convolutional Neural Network in PyTorch

Supervised by:
Mr. CHIRANE Merouane
AI Engineer, Algérie Télécom

Prepared by:
MEDOUSE AbdelKarim
Master 2 – Data Science and Artificial Intelligence
Université Abdelhamid Mehri – Constantine 2

July 2025

Abstract

In this project, we tackled the problem of plant disease classification using deep learning. The objective was to build a model capable of identifying plant diseases from leaf images.

We trained two models: a Convolutional Neural Network (CNN) from scratch and a ResNet18 model using transfer learning. We also introduced class-weighted loss to address class imbalance. Evaluation was done using accuracy, confusion matrices, and ROC curves.

The results showed that both models achieved high accuracy, with transfer learning offering better efficiency. This work reflects a real-world AI pipeline — exploring data, testing models, handling imbalances, and evaluating performance systematically.

Project GitHub Repository: `https://github.com/kamydev/plant-disease-detection`

Contents

1. Introduction

Plant diseases can ruin crops fast, and catching them early makes a big difference. Instead of manually checking thousands of leaves, we wanted to see how well deep learning could handle the job using just images.

In this project, we used the **PlantVillage dataset**, which contains thousands of labeled images of healthy and diseased plant leaves. Our goal was to build a model that can look at a leaf image and predict if it's healthy or what disease it has.

We approached the problem in two ways:

- First, we built a CNN from scratch and trained it directly on the dataset.
- Then, we used transfer learning with `ResNet18` to see if a pretrained model could do better with less training time.

We also tested improvements like weighted loss to deal with class imbalance.

This report walks through how we explored the data, trained and evaluated our models, and compared the results. The focus was not just on performance but also on learning how to solve a real-world deep learning problem step by step.

2. Dataset & Exploration

We used the **PlantVillage** dataset from Kaggle, which contains over 50,000 images of plant leaves across different crops and disease types. Each image is labeled with the disease name or marked as healthy.

For this project, we focused on a filtered version of the dataset containing **15 classes**, including various tomato, potato, and pepper leaf diseases. After unzipping the dataset manually, we placed it in a local folder under `data/PlantVillage`, where each subfolder represents a class.

We ran an initial count of images per class to check for imbalances. The results showed that the dataset is not evenly distributed some classes have thousands of images, while others have less than 200.

Dataset Stats:

- Total images: 26,839
- Number of classes: 15
- Smallest class: `Potato__healthy` (152 images)
- Largest class: `Tomato__Tomato_YellowLeaf__Curl_Virus` (3209 images)

We visualized the class distribution using a bar plot (Figure 1) and displayed a few sample images to verify class structure and image quality.

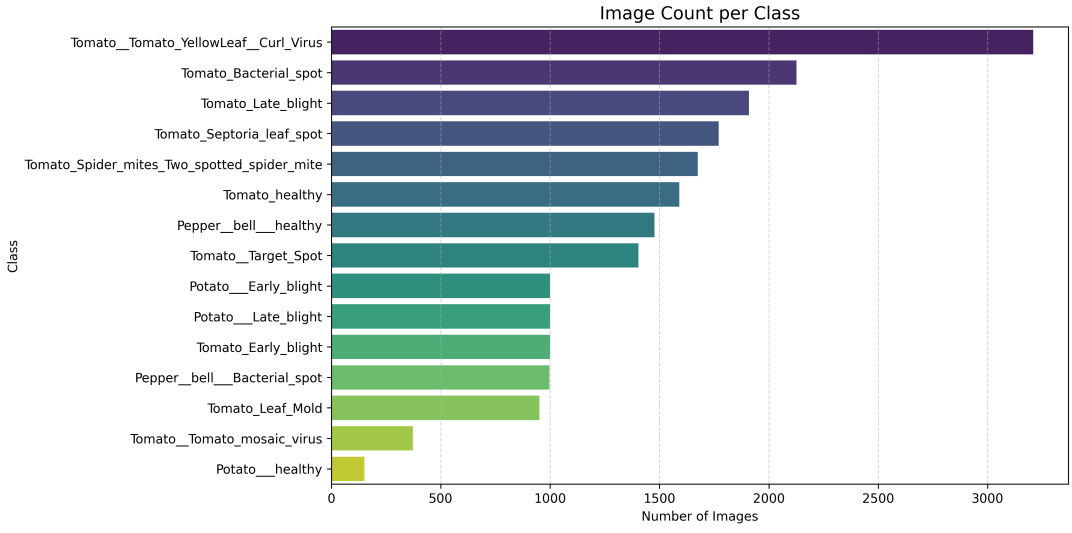

Figure 1: Number of images per class in the PlantVillage dataset.

Understanding this imbalance helped us later when applying weighted loss functions to improve model fairness across classes.

3. Preprocessing

Before training any models, we applied standard image preprocessing steps to ensure the data was clean and compatible with deep learning pipelines.

Image Size and Format

All images were resized to **224×224 pixels**, which is the standard input size expected by many CNN architectures including ResNet18. Most images were originally 256×256, so resizing helped standardize the input without losing significant detail.

Normalization

We normalized the pixel values of all images using the mean and standard deviation computed from the entire dataset. This helps the model converge faster during training.

- `mean = [0.4591, 0.4753, 0.4116]`
- `std = [0.1812, 0.1573, 0.1957]`

Normalization scales pixel values into a similar range, which makes gradient updates more stable and consistent.

Class Mapping

Each class was automatically assigned a numerical index by `ImageFolder`. We verified the mappings to ensure correct label association. This was essential for interpreting evaluation results later.

Comment on Data Quality

Most images were clean and well-centered on the leaf. No additional cleaning or filtering was needed at this stage.

These preprocessing steps were applied consistently for both training and validation images to avoid data leakage.

4. Modeling

We trained two different models to classify plant diseases from images:

- A Convolutional Neural Network (CNN) built from scratch
- A pretrained ResNet18 model using transfer learning

The goal was to compare their performance and learn how architectural choices and training strategies affect results.

1. CNN From Scratch

We implemented a basic CNN with three convolutional layers, each followed by max pooling and ReLU activation. This was followed by two fully connected layers. The input size was set to `3×224×224`, and the final output layer had 15 units (one per class).

- **Loss Function:** CrossEntropyLoss
- **Optimizer:** Adam with a learning rate of 1e-4
- **Training Time:** 3 hours on CPU for 10 epochs

Addressing Class Imbalance

To reduce bias caused by uneven class sizes, we experimented with a **weighted loss function**. Each class was given a weight inversely proportional to its frequency in the training set. This helped the model pay more attention to underrepresented classes.

2. Transfer Learning with ResNet18

We loaded a pretrained ResNet18 model and replaced its final layer to output 15 classes. Only the final layers were fine-tuned, which made training much faster.

- **Training Time:** 30 minutes for 5 epochs
- **Accuracy:** Comparable or better than the custom CNN, with significantly less training time

This approach showed that transfer learning is highly effective when working with limited data and computational resources.

Both models were trained and evaluated using the same loaders, preprocessing steps, and metrics for a fair comparison.

5. Evaluation

After training both models, we evaluated their performance using the validation set. The key metrics used were:

- **Accuracy:** Overall correct predictions over total samples.
- **Confusion Matrix:** To see how well the model distinguishes between classes.
- **ROC Curves (Multi-class):** To visualize model confidence and separability per class.

CNN From Scratch Results

- **Validation Accuracy:** 94.55%
- **Training Time:** 3 hours
- **Observation:** Very good performance overall, but some misclassifications between visually similar diseases.

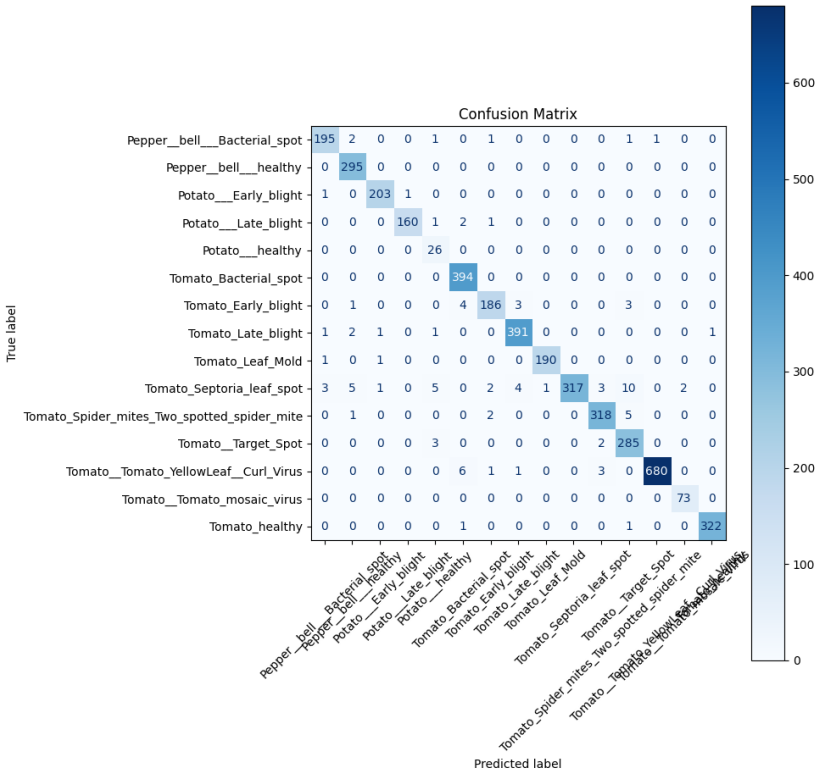

Figure 2: Confusion matrix – CNN from scratch

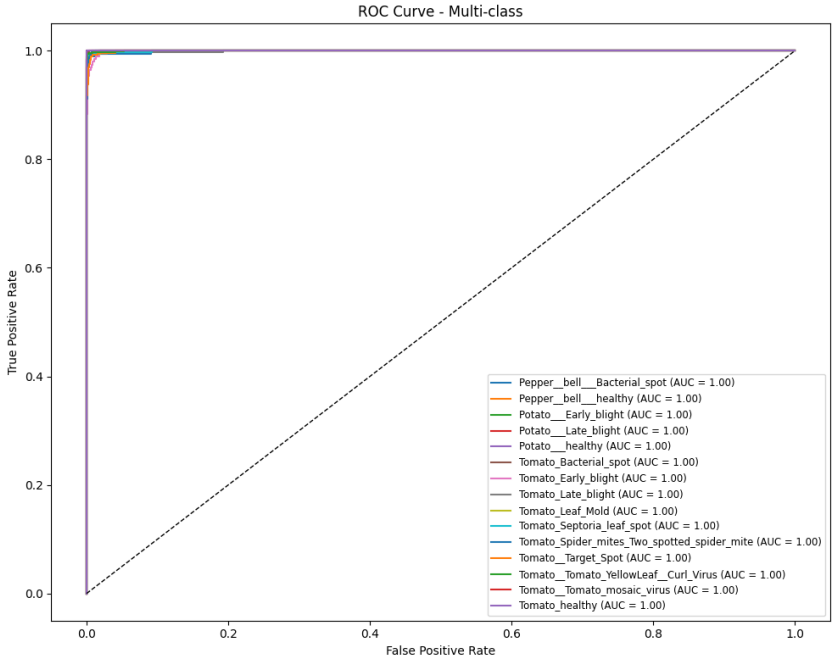

Figure 3: Multi-class ROC curve – CNN from scratch

Transfer Learning (ResNet18) Results

- **Validation Accuracy:** Comparable to CNN, sometimes higher on certain classes
- **Training Time:** 30 minutes (5 epochs)
- **Observation:** Faster convergence with strong performance; especially good for small datasets

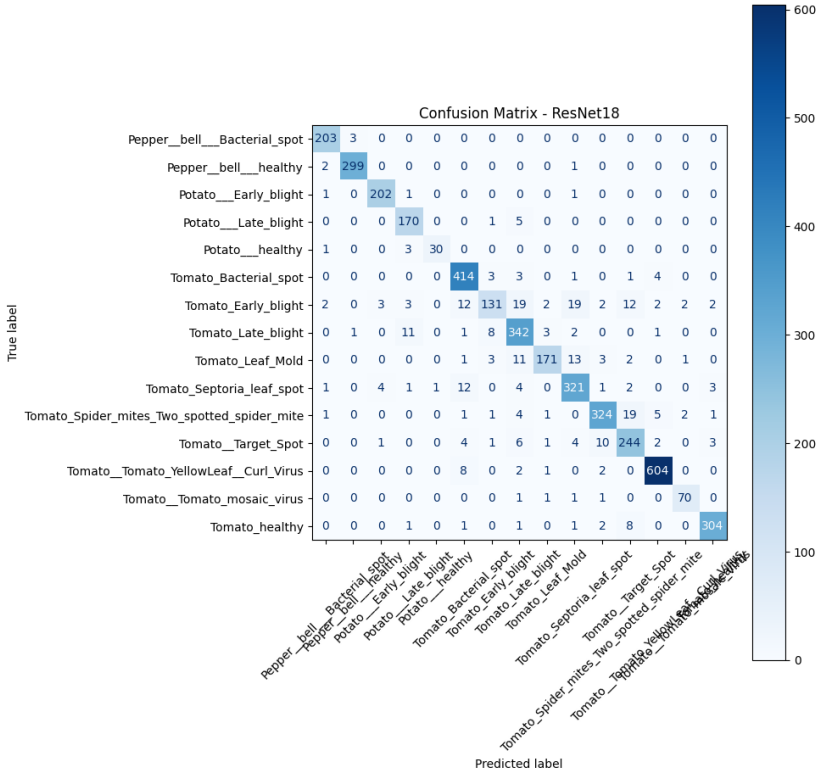

Figure 4: Confusion matrix – ResNet18

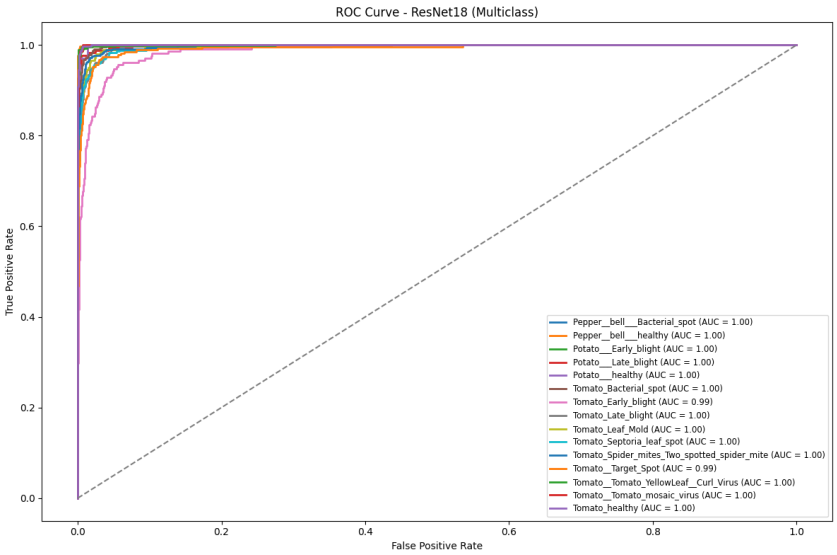

Figure 5: Multi-class ROC curve – ResNet18

Key Takeaways

- Both models achieved high accuracy, above 94%.

- CNN from scratch required more training time but performed well.
- ResNet18 with transfer learning trained faster with comparable results.
- Confusion matrices revealed that most errors were between visually similar leaf diseases.

6. Improvement: Weighted Loss Function

To address the class imbalance observed during data exploration, we implemented a **Weighted Cross-Entropy Loss** for our custom CNN model.

By assigning higher loss penalties to underrepresented classes, the model's sensitivity to rare disease types improved. While overall accuracy remained comparable, validation performance became more balanced across all classes — reducing bias toward dominant classes like tomato leaf diseases.

This simple yet effective change demonstrates how minor architectural or training adjustments can significantly impact real-world deep learning outcomes.

7. Conclusion

This project showed how deep learning can be used to detect plant diseases from leaf images with solid performance.

We started with a CNN built from scratch, handled class imbalance using a weighted loss function, and then compared results with a pretrained ResNet18 using transfer learning.

- Both models reached high accuracy (above 94%).
- Transfer learning with ResNet18 trained much faster and performed well.
- Weighted loss improved performance on underrepresented classes.

The full pipeline — from preprocessing to evaluation — reflects a practical, real-world approach to AI projects: start simple, measure results, improve strategically.

References

- PlantVillage Dataset: `https://www.kaggle.com/datasets/emmarex/plantdisease`
- PyTorch Documentation: `https://pytorch.org/docs/stable/index.html`
- Transfer Learning with ResNet: `https://pytorch.org/vision/stable/models.html`