UST, Zewail City
Spring 2025
DSAI 308

Deep Learning Project Report

“Crop Disease Detection”

By:
Layla Mohammad 202201906,
Nehal Mohamed 202202020.

UST, Zewail City
Spring 2025
DSAI 308

Project Topic: AI-Based Crop Disease Detection

Aim: Using images of Plant leaves, the model should detect/ classify whether this plant is Healthy, or affected with one of two different diseases; Powdery and Rusty introducing a multi-class (3 classes) classification problem..

Expected Outcome: Using CNN architecture model should be able to correctly classify plant images.

Data Set Used: Plant Disease Recognition Dataset

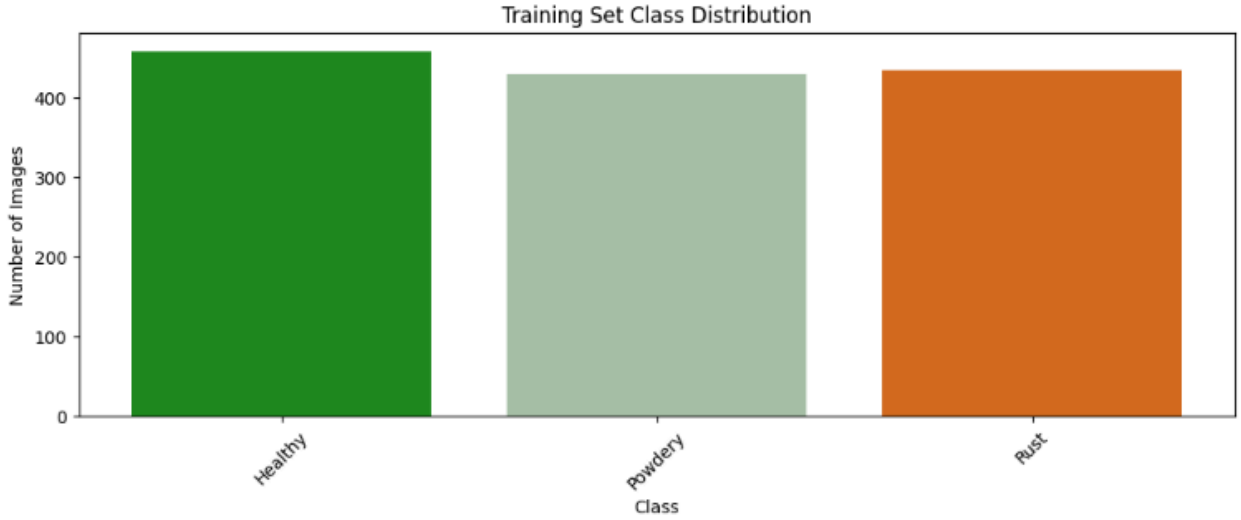
DataExploration:

- Data is already splitted into Train, validate and test folders
  1322 images for training, 60 images for validation and 150 images for testing.

- Each folder (set) contains Three Classes (Labels) describing leaf condition; Healthy, Powdery and Rust.

- Detailed Classes Distribution:
  Training:
  458 Healthy images
  430 Powdery images
  434 Rusty images
  Validation:
  20 Healthy images
  20 Powdery images
  20 Rusty images
  Testing:
  50 Healthy images
  50 Powdery images
  50 Rusty images

The following bar charts represents the number of images in each class in each folder (set):



Key observations:

1. Test Data and Validation data are perfectly balanced.
2. Although the number of images in each class is not exactly the same, the difference between the classes is not extreme allowing it to be considered balanced.

(There for, no need for imbalance handling before training)

Samples of Data:

1- Healthy

2- Powdery

3- Rusty

Preprocessing:

- Resize All images to 224 * 224 before passing it to the model
- Batch size chosen to be 32
  How many images will be loaded and processed together in one batch during training or evaluation.
  This is to ensure balance between memory usage and training speed.
- Normalised all images (Train, Test, Validation)
  Dividing pixel values by 255 to Convert them from[0, 255] to [0, 1].
  To make training faster and more stable

- **Data Augmentation** Applied for **Training** only
  To ensure more generalization in training process
  Not applied to Validation & test to keep them reflecting real-world data, not artificially modified versions.

Note That: While pipelining data for training
Training shuffled to: randomly shuffle images before each epoch to prevent the model from learning order bias.

Validation shuffled to: to make the model see a randomized subset of validation data between epochs. This helps provide a better overview of performance considering that validation data is small.

Testing kept without shuffling to: Keep the data in a fixed order to ensure consistent predictions. As shuffling test data may lead to mismatch predictions with labels during evaluation.

Model Architecture Explanation:

First Conv Layer:
64 filters of size 3×3 to detect 64 different features.
Padding='same': output size remains 224×224.
ReLU: (activation function) removes negative values
BatchNormalization: normalises activations to stabilize learning.
MaxPooling: reduce sample size to 112×112; (keeps strongest features)
Dropout: randomly deactivate 25% of neurons (to reduce overfitting)

Second Conv layer:

64 filters: to learn deeper and more complex features
Same processing as Layer 1
After MaxPooling: size is reduced to 56*56

256 filters: deeper feature detection
output after MaxPooling: 28*28*256

Flattening 28*28*256 will result in an extremely huge vector
So global Average pooling is used to reduce each 28*28 feature map to 1 value, resulting in an output vector containing 265.

Fully Connected Layers (Created 3 Dense layers for the 3 CNN Layers)

Followed the common numbering approach for neurons numbers (2 to the power __)

ReLU, batch normalisation and dropout were used for generalization and to reduce overfitting.

OutPut Layer:

3 neurons (one for each class)
Softmax Activation function for multiclass classification using probability.

Used Adam optimizer a it is common and work good with most cases and models
Used categorical_crossentropy as it is suitable for one hot encoded class, one having three or more classes.

To handle Overfitting:

Training Data Augmentation, Training and validation data shuffle, Batch Normalisation, Dropout were all applied, however the overfit problem improved a little bit, the overall model accuracy was not good so we decided to increase the number of epochs but to avoid falling again into the overfit problem we applied the next:

EarlyStopping to stop the training process when the model's performance on the validation data does not improve

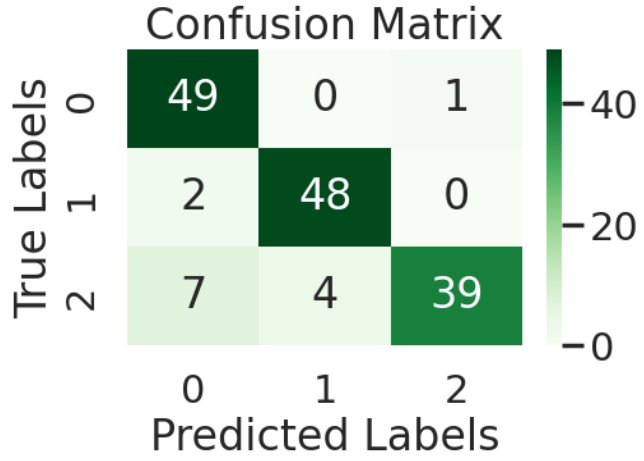
ReduceLROnPlateau Callback to reduce the learning rate when the model's performance on validation data is no longer improving.
(Lowering the learning rate helps to fine-tune the model more carefully as it approaches the optimal solution. This can help the model converge more smoothly and avoid overshooting the optimal weights.)

Results and Evaluation:

Accuracy: 0.9066666666666666
Recall: 0.9066666666666666
Precision: 0.9143015030946066
F1 Score: 0.9050835148874364

Classification Report:

	precision	recall	f1-score
0	0.84	0.98	0.91
1	0.92	0.96	0.94
2	0.97	0.78	0.87
accuracy			0.91
macro avg	0.91	0.91	0.91
weighted avg	0.91	0.91	0.91



UST, Zewail City
Spring 2025
DSAI 308