29.07.2025 **[INTERN NAME: Annapoorani P]**

Plant Disease Detection Project Report

1. Introduction and Project Overview

This project focuses on developing a deep learning model for plant disease classification using the Plant Village dataset. The primary goal is to accurately identify various plant diseases from images, which can significantly aid in early detection and management of crop health. The implementation leverages both TensorFlow and Keras, employing a convolutional neural network (CNN) architecture.

2. Dataset and Preprocessing

This project utilizes the Plant Village dataset, a widely recognized collection of plant leaf images categorized by species and disease status. The notebook indicates that images are handled with a target size of 224x224 pixels size and 3 color channels (RGB).

Data Organization and Splitting:

Custom functions are used to create a pandas dataframe from the dataset, mapping image file paths to their respective labels (disease categories). The dataset is then split into training, validation, and test sets from model selection (sci-kit learn library). A stratification strategy is applied based on the 'labels' to ensure that each split maintains the same proportion of classes as the original dataset. The split ratio is 80% for training, 10% for validation, and 10% for testing.

Image Data Generation and Augmentation:

Image Data Generator from Keras is employed to efficiently load and preprocess images in batches.

- **Training Data:** Augmented using horizontal flip is used to introduce variability and improve model generalization.
- **Validation and Test Data:** Not augmented, using a simple scalar preprocessing function that effectively performs no additional scaling beyond what's inherent in the Image Data Generator's default behaviour.
- Images are resized to 224x224 pixels and processed as RGB (3 channels).
- Batch sizes are configurable, with a specific calculation for the test batch size to ensure efficient processing of the entire test set.

3. Model Architecture

The notebook implements a sequential CNN model, characterized by multiple convolutional layers followed by pooling, batch normalization, and dropout layers to prevent overfitting. The specific architecture appears to be a custom-built CNN rather than a pre-trained model.

Key layers observed in the structure:

- **Convolutional Layers (Conv2D):** Extract features from the images.
- **MaxPooling Layers (MaxPooling2D):** Reduce spatial dimensions, making the model more robust to variations in image position.
- **Batch Normalization (BatchNormalization):** Stabilize and accelerate training.

- **Activation Layers (Activation):** ReLU activation is used after convolutional layers.
- **Dropout Layers (Dropout):** Regularization technique to reduce overfitting.
- **Flatten Layer (Flatten):** Converts the 2D feature maps into a 1D vector.
- **Dense Layers (Dense):** Fully connected layers for classification.
- **Output Layer:** A final Dense layer with softmax activation is used for multi-class classification, with the number of units equal to the number of disease classes.
- The model is compiled with the Adam optimizer and uses categorical_crossentropy as the loss function, which is appropriate for multi-class classification.

4. Training and Evaluation

The training process incorporates a custom callback (MyCallback) designed to enhance training stability and performance.

Custom Callback Features:

- **Learning Rate Adjustment:** Dynamically reduces the learning rate based on either training accuracy (if below a threshold) or validation loss (if above threshold). This helps the model converge effectively.
- **Early Stopping:** Monitors for lack of improvement over a specified patience, with an additional stop_patience to halt training if the learning rate has been adjusted multiple times without significant improvement.
- **Best Weights Restoration:** Saves and restores the model weights corresponding to the best observed performance during training (either highest training accuracy or lowest validation loss).
- **Interactive Control:** Allows the user to halt training or extend epochs during the process.

Performance Metrics:

The model's performance is evaluated using standard metrics such as accuracy and loss for both training and validation sets. The function (named plot_training) visualizes these metrics over epochs, clearly indicating the best epoch for both accuracy and loss. While the provided snippets do not include the final reported accuracy, the notebook's titled "Plant Village Disease Classification" with overall accuracy of 99.6% suggests a high level of performance was achieved, indicating the effectiveness of the chosen architecture and training methodology.

5. Conclusion

The project demonstrates a robust approach to plant disease classification using deep learning. The well-structured data preprocessing, custom CNN architecture, and sophisticated custom callback for training optimization contribute to achieving high accuracy in classifying plant diseases. This solution holds significant promise for practical applications in agriculture.

****