Potato Disease Classification Dataset

Abstract

The potato plant disease detection dataset comprises 5,748 images of potato leaves categorized into three classes: Potato\_Early blight (1,000), potato\_Late Blight (1,000), Potato\_healthy(750). The dataset was collected from various open-access sources and integrated class-wise for comprehensive analysis. This dataset provides a robust foundation for training and evaluating convolutional neural networks in plant disease detection.

**Instructions:**

The dataset comprises labeled images organized into six categories, each representing different plant conditions or health states. These categories include Potato\_Early blight (1,000), potato\_Late Blight (1,000), Potato\_healthy(750) . A representative sample image is provided for each category. This dataset is structured to facilitate research and development in plant health diagnostics, particularly in applications involving machine learning and computer vision.

**Import data into tensorflow dataset object**

We will use image\_dataset\_from\_directory api to load all images in tensorflow dataset:

**Function to Split Dataset**

Dataset should be bifurcated into 3 subsets, namely:

1. Training: Dataset to be used while training
2. Validation: Dataset to be tested against while training
3. Test: Dataset to be tested against after we trained a model

**Building the Model**

**Creating a Layer for Resizing and Normalization**

Before we feed our images to network, we should be resizing it to the desired size. Moreover, to improve model performance, we should normalize the image pixel value (keeping them in range 0 and 1 by dividing by 256). This should happen while training as well as inference. Hence we can add that as a layer in our Sequential Model.

You might be thinking why do we need to resize (256,256) image to again (256,256). You are right we don't need to but this will be useful when we are done with the training and start using the model for predictions. At that time somone can supply an image that is not (256,256) and this layer will resize it

**Data Augmentation**

Data Augmentation is needed when we have less data, this boosts the accuracy of our model by augmenting the data.

**Model Architecture**

We use a CNN coupled with a Softmax activation in the output layer. We also add the initial layers for resizing, normalization and Data Augmentation.

**Compiling the Model**

We use adam Optimizer, SparseCategoricalCrossentropy for losses, accuracy as a metric

**Plotting the Accuracy and Loss Curves and loss, accuracy, val loss etc are a python list containing values of loss, accuracy etc at the end of each epoch**