**Title: Potato Disease Classification**

* Ravina Ingole

**Contents:**

1. [Executive Summary](#_1.Executive_Summary:)
2. [Problem Statement & Research Question](#_2.Problem_Statement_&)
3. [Data Preprocessing and Pipeline](#_3.Data_Preprocessing_and)
   1. [Image Classification](#_3.1._Image_Classification:)
   2. [Visualize some of the images from dataset](#_3.2._Visualize_some)
   3. [Dataset Split](#_3.3._Dataset_Split:)
   4. [Buffered Prefetching](#_Buffered_Prefetching:)
4. [Methods (Models, Experiments, Analysis) and Results](#_4._Methods_(Models,)
   1. [Methods](#_4.1_Methods:)
   2. [Cache, Shuffle, and Prefetch the Dataset:](#_4.2_Cache,_Shuffle,)
   3. [Building the Model](#_4.3_Building_the)
   4. [Normalization and resizing](#_4.4_Normalization_and)
   5. [Data Augmentation](#_4.5_Data_Augmentation:)
   6. [Model Architecture](#_4.3_Pruned_Decision)
   7. [Results](#_4.7_Results:)
5. [Conclusion](#_5.Conclusion:)
6. [Appendix](#_6.Appendix:)
   1. [Images of different types of Potatoes:](#_6.1_Images_of)
   2. [Internal Block of CNN](#_6.3_Internal_Block)
   3. [Max Pooling](#_6.4_Max_Pooling)

## **1.Executive Summary:**

This project involves deploying the Convolutional neural network (CNN) for image classification of potato leaves to identify potato disease. The ability of CNN to develop an internal representation of a two-dimensional image makes it best model for image classification This allows the model to learn position and scale in variant structures in the data, which is important when working with images. As a pre-elementary step, datasets are cache, shuffle and prefetched. Then to improve model performance layers for resizing and normalization is added. Model is made to learn the true characteristic of the dataset using training data and evaluated using test data to build a generalized model. This project involves use a CNN coupled with a Softmax activation in the output layer. This project have incorporated Adam Optimizer, SparseCategoricalCrossentropy for losses, accuracy as a metric to access the classifier performance in a better way and gain more insights about the classifier's prediction ability. We have also plotted accuracy and loss curves to understand losses with each epoch.

## **2.Problem Statement & Research questions:**

Farmers are facing lots of agricultural problem because of different diseases on plants in which two common diseases are early blight and late blight. If farmers detect these diseases early and treat them it can save lot of waste and prevent economy loss. Our goal is to detect type of diseases based different images by using image classification using CNN so that proper action can be taken.

## **Research questions:**

1. In this project, we will investigate if potato leaf belongs to early blight, late blight disease or it’s a healthy potato.
2. We will different methods to improve performance of model by applying different techniques like resizing, normalization, and data augmentation
3. We will use artificial neural network (ANN) – Convolutional Neural Network.
4. We will use Tensor flow for preprocessing, dividing datasets into classes and building CNN model.

## **3.Data Preprocessing and Pipeline:**

Dataset for the project was derived from [Kaggle](https://www.kaggle.com/datasets/arjuntejaswi/plant-village). Dataset originally had 3,152 records which includes 1000 images for early blight disease, 1000 for late blight and 152 images of healthy potatoes. As a pre-elementary step, we have built a TensorFlow pipelines to build image batch, image size, cache, shuffle, and prefect the dataset.

## **3.1. Image Classification:**

Dataset had 3,152 records before fitting images in CNN dataset it is divided into batches using TensorFlow. In this project image\_batch shape is (32, 256, 256, 3) and labels\_batch is [1 2 1 0 2 1 0 1 0 2 0 2 0 0 1 2 1 0 1 1 0 0 2 1 0 0 0 2 1 1 1 1]. According to observation, each element in the dataset is a tuple. First element is a batch of 32 elements of images. Second element is a batch of 32 elements of class labels.

A picture containing text

Description automatically generated

## **3.2. Visualize some of the images from dataset:**

Dataset is divided into early blight, late blight, and healthy potatoes. [Images](#_3.5_Boxplots:) were plotted to understand types of images for different diseases.

## **3.3. Dataset Split:**

Dataset have been 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

## **Buffered Prefetching:**

There are two important methods that this project has used when loading data.

* + To keep the images in memory after they have been loaded off disk during the first epoch cache() is used.
  + This will ensure that the dataset doesn't become a bottleneck when the model is being trained.
  + If the dataset is too large to fit into memory, this method can be used to create a performant on-disk cache.
  + prefetch() will overlap the data preprocessing and model execution while training.

## **4. Methods (Models, Experiments, Analysis) and Results:**

## **4.1 Methods:**

Post pre-processing, dataset was split into training, validation, and testing data (80%-10%-10% split). Idea was to build model using training data and evaluate the performance on unseen data i.e., test data to build a generalized model and dataset is again tested while training on validation dataset.

## **4.2 Cache, Shuffle, and Prefetch the Dataset:**

To increase the training accuracy, it’s important to shuffle the data well, In this project we have used ds.shuffle to shuffle records and shuffle\_files=True to get good shuffling behavior for larger datasets that are sharded into multiple files so that epoch won’t be truly randomized and read the shards in the same order. To catch a dataset either in memory or on local storage catch is used. Also, tf.data.Dataset class .prefetch() function is applied to produce a dataset that prefetches the specified elements from this given dataset.

## **4.3 Building the Model:**

Images should be resized to desired size before feeding our images to network to improve model performance, image pixel should be normalized by keeping them in range 0 and 1 and dividing by 256. This should happen while training as well as inference hence in our sequential model we can add that as a layer.

## **4.4 Normalization and Resizing:**

To make gradient descents converge faster we normalize training data by making sure that the various features have similar value ranges. This also helps to train model and decrease its learning speed.

Before imputing images to the CNN, all the images need to be resized to a fixed size so that neural networks receive input of the same size. The larger the fixed size, the less shrinking required. Less shrinking means less deformation of features and patterns inside the image.

## **4.5 Data Augmentation:**

Data augmentation is useful to improve performance and outcomes of machine learning models by forming new and different examples to train datasets. If the dataset in a machine learning model is rich and sufficient, the model performs better and more accurately.

In this project we have used data augmentation to increase the diversity and amount of training data by applying random transformation. This also helps to detect image if it’s placed in any orientation. Here is an example for digit image augmentation.

A screenshot of a game

Description automatically generated with low confidence

## **4.6 Model Architecture:**

[Image 6.2](#_6.2._Model_Architecture:) shows model architecture used for this project data is mainly divided into training and testing dataset, before putting images into neural network it is labeled and passed into training image set then various preprocessing technique is used to make dataset more relevant for training, features is extracted during training dataset which is then passed into convolutional neural network once model is trained on training dataset we use model prediction which helps to give classification results.

[Image 6.3](#_3.8_Bar_Plot) shows Internal block of CNN, in this project we have coupled CNN with a Softmax activation in the output layer. We also add the initial layers for resizing, normalization, and Data Augmentation. CNN is popular for image classification tasks. This model consists of resize, rescaling layer and 6 layers of Conv2D and max pooling which is then flattened to understand features of data and passed to fully connected layer to image classification. Here maxpooling is applied to down sampling strategy in CNN that helps to reduce over-fitting and reduce computational cost. [Image 6.4](#_3.9_Scatter_Box) shows a reference image for maxpooling.

## **4.7 Results:**

Accuracy, losses, value losses, value accuracy was calculated for 10 epoch to understand model performance. Below table shows the result for different epochs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Losses** | **Accuracy** | **Val\_loss** | **Val\_accuracy** |
| 1 | 77 | 68 | 77 | 69 |
| 2 | 49 | 82 | 41 | 83 |
| 3 | 35 | 87 | 52 | 81 |
| 4 | 25 | 91 | 20 | 91 |
| 5 | 21 | 92 | 16 | 94 |
| 6 | 15 | 94 | 18 | 92 |
| 7 | 15 | 94 | 34 | 85 |
| 8 | 14 | 94 | 0.116 | 95 |
| 9 | 11 | 95 | 25 | 90 |
| 10 | 10 | 96 | 9 | 97 |

# Table 4.1.1

Below image shows accuracy and losses for training, validation.

Chart, line chart, histogram

Description automatically generated

Output of prediction models:

A picture containing chart

Description automatically generated

Diagram

Description automatically generated with medium confidences

## **5.Conclusion:**

* Model was built using Convolutional neural network and accuracy of 97% was found out after 10 epochs.
* After each epoch losses were reduced, and accuracy was increasing.
* By increasing a greater number of epochs 100% accuracy can be obtained.
* Model was able to classify images correctly for different diseases.

## **6.Appendix:**

## **6.1 Images of different types of Potatoes:**

A collage of watermelons

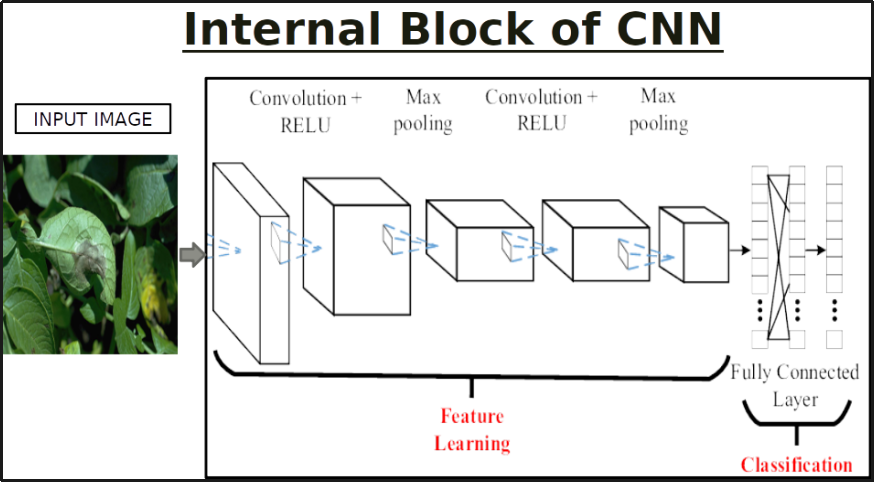
Description automatically generated with medium confidence

## **6.2. Model Architecture:**

Diagram

Description automatically generated

## **6.3 Internal Block of CNN**



## **6.4 Max Pooling**

A picture containing diagram

Description automatically generated