Deep Learning for Plant Disease Identification and Treatment

Introduction

In India, agriculture is the primary source of food. Numerous plant diseases can arise during the crop cycle and have an impact on the yield. A critical component of crop management is the identification and categorization of plant diseases at the leaf level, which aids in the early diagnosis and detection of plant diseases. Farmers have traditionally classified plant diseases by hand, utilising techniques including manual inspection and field observations. There is a need for an intelligent system that can automatically classify photos based on plant leaf diseases because the current method is laborious and prone to errors.

Plants are crucial for food production, but diseases cause significant losses. Deep learning techniques detect plant diseases using image attributes, identifying infection stages, and suggesting potential treatments if available.

Problem Statement

"Identify the illness that a tomato plant leaf contains and provide a remedy (if possible), given an image showing the disease" is the issue statement that will be addressed.  
That is, if we receive a tomato plant leaf snapshot from a farmer that illustrates a crop disease, we must inform them of the name of the disease that the leaf is infected with and the remedy for early diagnosis.

Proposed Solution

The creation of a deep learning model that can recognise, categorize, and state the stage of infection in tomato plant leaf photos according to the illness they carry is the suggested answer to this issue. In this project, we will make use of an eight-layered convolution network called AlexNet.

Description of the Dataset

For this project, we will use the [Tomato Leaf Disease Detection](https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf)dataset available on Kaggle.

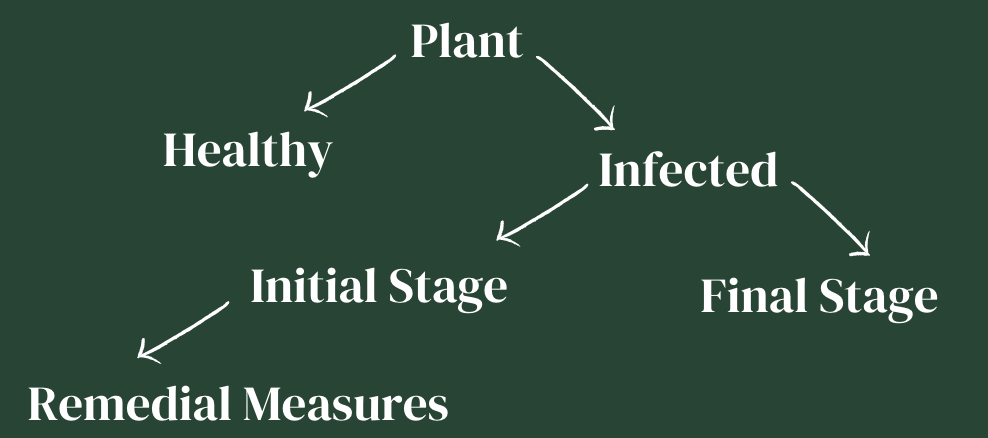
The data has different types of diseases for tomato leaves.  
Here is the list:

* Tomato\_mosaic\_virus
* Target\_Spot
* Bacterial\_spot
* Tomato\_Yellow\_Leaf\_Curl\_Virus
* Late\_blight
* Leaf\_Mold
* Early\_blight
* Spider\_mites Two-spotted\_spider\_mite
* Tomato\_\_\_healthy
* Septoria\_leaf\_spot

The dataset consists of test and training images for the 10 classes mentioned above.

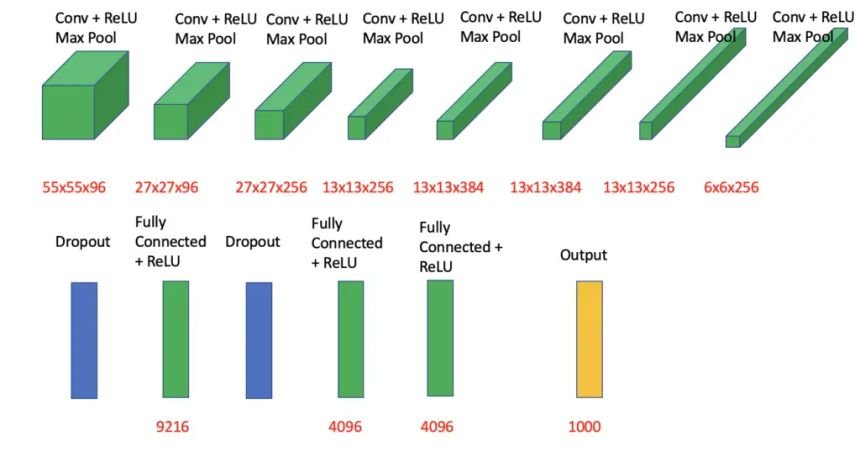
Methodology

First, we'll determine whether or not the tomato plant has an illness. We shall ascertain the disease's stage if the plant is affected. If it's initially detected, corrective action will be given to preserve the yield.



Architecture:

AlexNet, a CNN architecture, was selected as the 2012 LSVRC competition champion. Research teams test their algorithms on a large dataset of annotated photos (ImageNet) in the Large Scale Visual Recognition Challenge, where the goal is to improve accuracy on a variety of visual recognition tasks. More than 1.2 million photos are used for training, 50,000 for validation, and 150,000 for testing. The model creators enforced a fixed size of 256×256 pixels on each image by deleting the middle 256×256 patch.  
Eight convolutional layers make up AlexNet's architecture; there are three ANN layers and five convolutional layers total. A max pooling layer comes after each convolution layer. The architecture of it is simple to comprehend. It involves overlapping max pooling and ReLu activation.



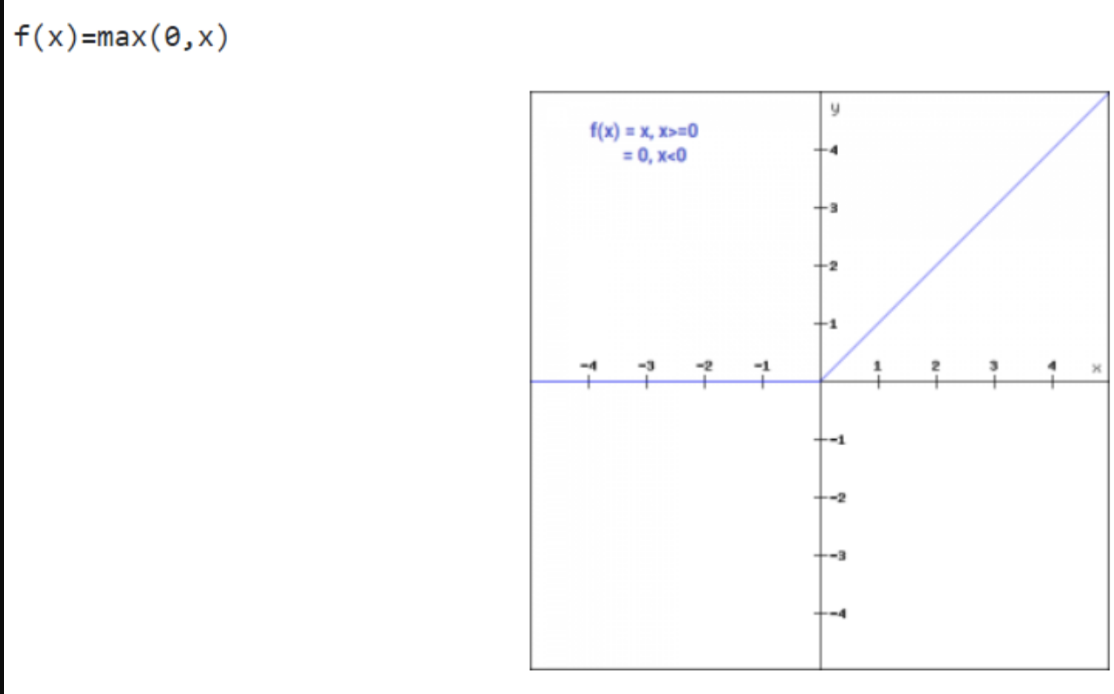
Features of AlexNet:

* Kernels in the 2nd, 4th, and 5th convolution layers are interlinked to those kernels in the previous layers that lie on the same GPU.
* The neurons present in the fully connected layers are interconnected with each other.
* ReLu activation is applied to the output of every convolution layer and in the simple artificial neural network structure.

ReLu Activation in AlexNet:

Usually, the sigmoid or tanh function is used as an activation function in deep learning networks. However, AlexNet employs a non-linear activation function called Rectified Linear Unit activation. The former activation functions saturate quickly, and training on GPUs using these functions is not easy, so non-linearity in ReLu helps with effective learning.

The vanishing gradient descent problem present when using sigmoid and tanh functions is not experienced during ReLu activation.



Local Response Normalization in AlexNet:

For neural networks that employ nonlinear activation functions, normalisation is an essential component. Since the unbounded activation outputs of nonlinear activation functions are not bounded like those of linear activation functions, we utilise normalisation to limit them. Normalisation of local responses facilitates generalisation. ReLU was selected as the activation function in place of the then-common tanh and sigmoid, which resulted in the AlexNet design using "local response normalisation" (LRN).

Apart from the previously described rationale, the application of LRN was suggested to encourage lateral inhibition. In neuroscience, a neuron's capacity to reduce the activity of its neighbours is a concept. The lateral inhibition function in DNNs is utilised for local contrast augmentation, whereby the layers that follow are stimulated locally by the greatest pixel values. Using LRN, a non-trainable layer, the pixel values of a feature map of a local neighbourhood are square-normalized.

Coding Steps:

The steps involved in this project are as follows:

* Import Libraries
* Define batch specifications
* Load the dataset
* Building the AlexNet architecture
* Compile the model
* Training and validation
* Saving the model
* Testing the model with a test image

## Code Implementation

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import tensorflow as tf*

*from tensorflow.keras import layers*

*from time import perf\_counter*

*import os*

*import seaborn as sns*

*from PIL import Image*

*from PIL import ImageEnhance*

*from skimage.io import imread*

*import matplotlib.pyplot as plt*

*import os, random, pathlib, warnings, itertools, math*

*warnings.filterwarnings("ignore")*

*import tensorflow.keras.backend as K*

*from sklearn.metrics import confusion\_matrix*

*from tensorflow.keras import models*

*from tensorflow.keras.models import Model*

*from tensorflow.keras.models import load\_model*

*from tensorflow.keras.preprocessing import image*

*from keras.preprocessing.image import load\_img,img\_to\_array*

*from tensorflow.keras.layers import Dense, Dropout, Flatten, Input, LeakyReLU*

*from tensorflow.keras.layers import BatchNormalization, Activation, Conv2D*

*from tensorflow.keras.applications import ResNet101V2*

*from tensorflow.keras.models import Sequential*

*from tensorflow.keras.preprocessing.image import ImageDataGenerator*

*from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D, Dense, Dropout*

*K.clear\_session()*

*from google.colab import drive*

*drive.mount('/content/drive')*

*# Path to the dataset in Google Drive*

*data\_dir = '/content/drive/MyDrive/tomato/train'*

*# Create the training dataset*

*# Data generators*

*train\_datagen = ImageDataGenerator(*

*rescale=1./255,*

*rotation\_range=40,*

*width\_shift\_range=0.2,*

*height\_shift\_range=0.2,*

*shear\_range=0.2,*

*zoom\_range=0.2,*

*horizontal\_flip=True,*

*fill\_mode='nearest'*

*)*

*training\_data = train\_datagen.flow\_from\_directory(*

*data\_dir,*

*target\_size=(250, 250),*

*batch\_size=100,*

*class\_mode='categorical' # Ensure this matches your loss function*

*)*

*# Path to the dataset in Google Drive*

*data\_dir\_val = '/content/drive/MyDrive/tomato/val'*

*# Create the validating dataset*

*valid\_datagen = ImageDataGenerator(rescale=1./255)*

*validation\_data = valid\_datagen.flow\_from\_directory(*

*data\_dir\_val,*

*target\_size=(250, 250),*

*batch\_size=100,*

*class\_mode='categorical' # Ensure this matches your loss function*

*)*

*target\_names = list(training\_data.class\_indices.keys())*

*print(target\_names)*

*from tensorflow.keras.models import Sequential*

*from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Activation, Dropout, Flatten, Dense*

*from tensorflow.keras.optimizers import Adam*

*from tensorflow.keras.preprocessing.image import ImageDataGenerator*

*# Define the model*

*model = Sequential()*

*# 1st Convolutional Layer*

*model.add(Conv2D(filters=96, input\_shape=(250, 250, 3), kernel\_size=(11, 11), strides=(4, 4), padding='valid'))*

*model.add(Activation('relu'))*

*model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2), padding='valid'))*

*model.add(BatchNormalization())*

*# 2nd Convolutional Layer*

*model.add(Conv2D(filters=256, kernel\_size=(11, 11), strides=(1, 1), padding='valid'))*

*model.add(Activation('relu'))*

*model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2), padding='valid'))*

*model.add(BatchNormalization())*

*# 3rd Convolutional Layer*

*model.add(Conv2D(filters=384, kernel\_size=(3, 3), strides=(1, 1), padding='valid'))*

*model.add(Activation('relu'))*

*model.add(BatchNormalization())*

*# 4th Convolutional Layer*

*model.add(Conv2D(filters=384, kernel\_size=(3, 3), strides=(1, 1), padding='valid'))*

*model.add(Activation('relu'))*

*model.add(BatchNormalization())*

*# 5th Convolutional Layer*

*model.add(Conv2D(filters=256, kernel\_size=(3, 3), strides=(1, 1), padding='valid'))*

*model.add(Activation('relu'))*

*model.add(MaxPooling2D(pool\_size=(2, 2), strides=(2, 2), padding='valid'))*

*model.add(BatchNormalization())*

*# Flattening*

*model.add(Flatten())*

*# 1st Dense Layer*

*model.add(Dense(4096))*

*model.add(Activation('relu'))*

*model.add(Dropout(0.4))*

*model.add(BatchNormalization())*

*# 2nd Dense Layer*

*model.add(Dense(4096))*

*model.add(Activation('relu'))*

*model.add(Dropout(0.4))*

*model.add(BatchNormalization())*

*# Output Layer*

*model.add(Dense(10))*

*model.add(Activation('softmax'))*

*# Summarize the model*

*model.summary()*

*optimizer = Adam(learning\_rate=1e-4)*

*model.compile(optimizer=optimizer, loss='categorical\_crossentropy', metrics=['accuracy'])*

*model.fit(*

*training\_data,*

*validation\_data=validation\_data,*

*epochs=10*

*)*

*model.save("AlexNetModel1.hdf5")*

*import imageio*

*import tensorflow*

*from tensorflow import keras*

*import numpy as np*

*from tensorflow.keras.preprocessing.image import img\_to\_array, load\_img*

*from tensorflow.keras.models import load\_model*

*from PIL import Image*

*target\_names = ['Tomato\_\_\_Bacterial\_spot', 'Tomato\_\_\_Early\_blight', 'Tomato\_\_\_Late\_blight', 'Tomato\_\_\_Leaf\_Mold', 'Tomato\_\_\_Septoria\_leaf\_spot', 'Tomato\_\_\_Spider\_mites Two-spotted\_spider\_mite', 'Tomato\_\_\_Target\_Spot', 'Tomato\_\_\_Tomato\_Yellow\_Leaf\_Curl\_Virus', 'Tomato\_\_\_Tomato\_mosaic\_virus', 'Tomato\_\_\_healthy']*

*def run(source = None):*

*model = tensorflow.keras.models.load\_model('/content/drive/MyDrive/AlexNetModel.hdf5')*

*img = imageio.imread(source)*

*img = Image.fromarray(img).resize((64, 64))*

*x = img\_to\_array(img)*

*x = np.expand\_dims(img, axis=0)*

*x = x/255 #STANDARDIZATION*

*prediction = model.predict(x)*

*print("Predicted Image is:",target\_names[np.argmax(prediction)])*

*run(source=r'/content/drive/MyDrive/tomato/val/Tomato\_\_\_Bacterial\_spot/01a3cf3f-94c1-44d5-8972-8c509d62558e\_\_\_GCREC\_Bact.Sp 3396.JPG')*

## References

## <https://www.analyticsvidhya.com/blog/2023/02/plant-disease-classification-using-alexnet/>

## <https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf>

## <https://search.brave.com/images?q=alexnet>