# An Automated Process to Detect and Help Cure Plant Diseases

## 1 INTRODUCTION

### *1.1 Overview*

Agriculture is one of the key sectors worldwide. It has immensely developed over the years and use of new technologies and equipment have replaced almost all the traditional methods of farming. Still it is observed that plant diseases directly affect the production in agriculture sector. Moreover, plant health and food safety are directly related. Identification of diseases and taking necessary precautions have the utmost importance and as these processes are generally done through bare eyes, some scientific method must be employed to help the farmers in dealing with this identification of plant diseases and cure of the same.

Convolutional Neural Networks (CNNs) over the years have been proven to be a great model in image classification. In this project, CNN models for few types of fruits and vegetables have been created that can identify whether a given image of a leaf is healthy or infected with high accuracy. Moreover, a web application using flask is also designed to integrate the model, where a user can upload a leaf image for those chosen fruits and vegetables and the model can predict whether it is healthy or not by observing some features present in the leaves and then can suggest precautionary measures.

For training the CNN, images of fruit and vegetable leaves are used belonging to 6 and 9 categories respectively. A total of 7070 images of fruit leaves and 14802 vegetable leaf images were used to train and validate the model. The results shows high training and validation accuracies with very low losses.

### *1.2 Purpose*

This application would automate the process of plant disease identification which is generally done manually till date. CNN application to image classification and disease prediction may help to save a lot of time and cost. This automatic screening tool would contribute to more sustainable agricultural practices and greater food production.

# 2 LITERATURE SURVEY

## *2.1 Existing Problem*

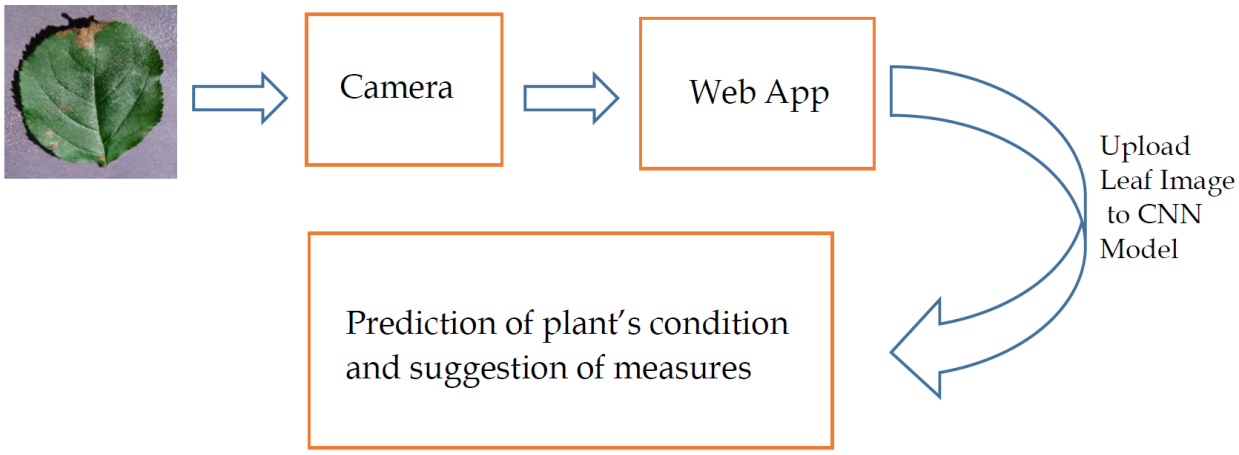
Traditionally farmers used to rely on visual examination to study infections in plants. Presently some molecular techniques are used to identify various diseases. It is primarily divided into two main broad categories: serological assays and nucleic acid based approaches. Currently polymerase chain reaction (PCR), immunofluorescence (IF), fluorescence in-situ hybridization (FISH), enzyme-linked immunosorbent assay (ELISA), flow cytometry (FCM) and gas chromatography-mass spectrometry (GC-MS) are some of the direct detection methods. Indirect methods include thermography, fluorescence imaging and hyper spectral techniques.

## *2.2 Proposed Solution*

This project proposes a deep learning Convolutional Neural Network (CNN) that is trained using images of healthy as well as infected leaves. CNN has proved to be greatly successful in image classification. This is done by the various convolutional layers of varying kernel thickness for identifying different features in leaves. One of the great advantages of using CNNs is that the hyper parameters like batch size, drop out, number of epochs and data augmentation techniques, number of convolutional & hidden layers, kernel thickness can be easily tuned to get high accuracy or low loss levels for prediction. This may lead to save a lot of time, money and resources.

# 3 THEORITICAL ANALYSIS

## *3.1 Block Diagram*



### *3.2 Hardware/Software*

A 64 bit PC loaded with Anaconda, Python3, Spyder and flask applications. Python libraries like Tensorflow, Numpy, Pandas must be installed and imported to compile and execute the program.

## 4 EXPERIMENTAL INVESTIGATIONS

Traditionally farmers used to rely on visual examination to study infections in plants. Presently some molecular techniques are used to identify various diseases. It is primarily divided into two main broad categories: serological assays and nucleic acid based approaches. Currently polymerase chain reaction (PCR), immunofluorescence (IF), fluorescence in-situ hybridization (FISH), enzyme-linked immunosorbent assay (ELISA), flow cytometry (FCM) and gas chromatography-mass spectrometry (GC-MS) are some of the direct detection methods. Indirect methods include thermography, fluorescence imaging and hyper spectral techniques.

### *2.2 Proposed Solution*

Two different models namely fruit-training and vegetable-training is created for detecting infected fruit leaves and vegetable leaves respectively. As data augmentation may improve the accuracy of the models, ImageDataGenerator is utilized for image augmentation with different features as given below:

rescale = 1.0/255, shear_range = 0.2,
rotation_range = 10, fill_mode='nearest',
width_shift_range=0.2, height_shift_range=0.2,
horizontal_flip= True, vertical_flip=False,
brightness_range=[0.8,1.2],
zoom_range= 0.2

**Fruit Model:**

The fruit model is trained for following 6 classes with 5384 images and different 1686 images are used for validation purpose.

Fruit Classes:

'Apple: Black Rots',
'Apple: Healthy',
'Corn: Northern Leaf Blight',
'Corn: Healthy',
'Peach: Bacterial Spots',
'Peach: Healthy'

The model is created with a Sequential layer followed by different sets of Convolution2D and MaxPool2D layers. After several trials and visualizing training loss, training accuracy, validation loss and validation accuracy, the following combination is adopted:

```
Convolution2D(16, (3,3), input_shape = (128,128,3), activation='relu', padding='same')
MaxPool2D(pool_size= (2,2))
Convolution2D(32, (3,3), input_shape = (128,128,3), activation='relu')
MaxPool2D(pool_size= (2,2))
Convolution2D(64, (3,3), input_shape = (128,128,3), activation='relu')
MaxPool2D(pool_size= (2,2))
```

Only the first Convolution2D layer has the zero padding and a Flatten layer is added after the last MaxPool2D layer. Adding more convolution layers did not increase the accuracy of the model but increased the time complexity of the problem.

For creation of neural network, two hidden Dense layers with 40 and 20 units respectively are added with “relu” activation function and lastly an output layer with “softmax” activation function is added. A checkpoint is created so that the model weights that would be considered the best can be loaded into the model after completion of the training. Finally, the model is trained with 30 epochs with “adam” optimizer.

**Vegetable Model:**

The vegetable model is trained for following 9 classes with 11386 images and different 3416 images are used for validation purpose.

Fruit Classes:

```
'Pepper: Bacterial Leaft Spots',
'Pepper: Healthy',
'Potato: Early Blight',
'Potato: Late Blight',
'Potato: Healthy',
'Tomato: Bacterial Spots',
'Tomato: Late Blight',
'Tomato: Leaf Molds',
'Tomato: Septoria Leaf Spot'
```

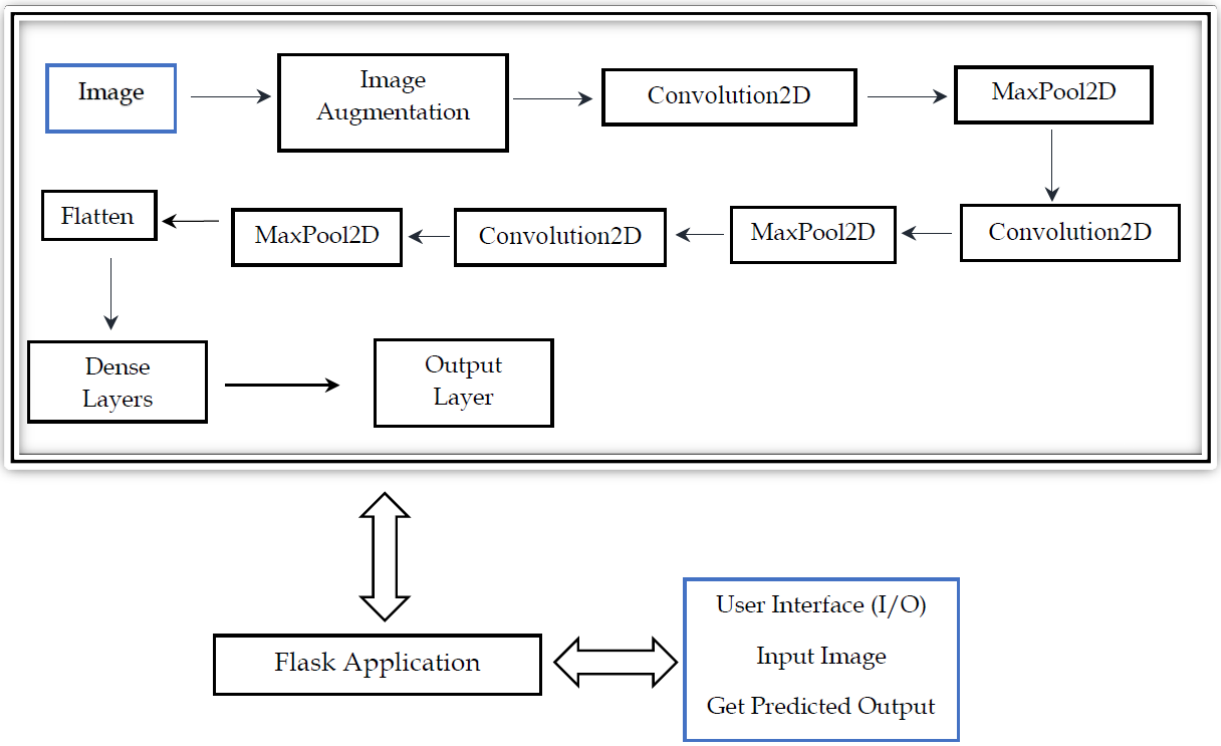
Like the previous model, this vegetable model is also created with a Sequential layer followed by different sets of Convolution2D and MaxPool2D layers. After several trials and visualizing training loss, training accuracy, validation loss and validation accuracy, the following combination is adopted followed by a Flatten layer:

```
Convolution2D(16, (3,3), input_shape = (128,128,3), activation='relu')
MaxPool2D(pool_size= (2,2))
```

```
Convolution2D(32, (3,3), input_shape = (128,128,3), activation='relu')
MaxPool2D(pool_size= (2,2))
Convolution2D(64, (3,3), input_shape = (128,128,3), activation='relu')
MaxPool2D(pool_size= (2,2))
```

For creation of neural network, four hidden Dense layers with 400, 300, 200 and 100 units respectively are added with “relu” activation function and lastly an output layer with “softmax” activation function is added. In between two consecutive layers, a Dropout layer is also added to avoid over-fitting of the model. A checkpoint is created so that the model weights that would be considered the best can be loaded into the model after completion of the training. Finally, the model is trained with 30 epochs with “adam” optimizer.

## 5 FLOWCHART




## 6 RESULT

The fruit model not only shows very low training loss of 0.068 with a high accuracy of 97%, but also a little loss of 0.06 is achieved for validation set so that the corresponding accuracy reaches to 98%. On the other hand, a training accuracy of 95% (with loss of 0.17) is obtained for the vegetable model where the validation accuracy is as high as 98% with validation loss of 0.058. Both the models are saved with the best weights.

In order to use the models as web application, a flask-based application is created by writing code in Spyder. The following cautions are used for fruit and vegetable model respectively:

| **Fruits** |
| --- |
| Oopps!! Your apple plant is infected by Black Rots. This infection is a fungal infection. To control balck rot, remove the cankers by pruning at least 15 inches below the end and burn or bury them. Treating the sites with the antibiotic streptomycin or a copper-based fungicide will be helpful. |
| Yaayy!! Your apple plant is healthy. But, maintain the soil pH of 6.0 to 7.0 for healthy growth. Avoid planting apples in a low spot where cold air or frost can settle. |
| Oopps!! Your corn plant is infected by Northern Leaf Blight. The primary management strategy to reduce the incidence and severity of NCLB is planting resistant products. Using fungicides is also helpful. |
| Yaayy!! Your corn plant is healthy. But, maintain the soil consistently moist, but not soggy and only need fertilizer every 6 months. It prefers temperatures of 75 to 80 degrees F. |
| Oopps!! Your peach plant is infected by Bacterial Spots. This is a difficult disease to control when environmental conditions favor pathogen spread. Compounds for the treatment include copper, oxytetracycline (Mycoshield and generic equivalents), and syllit+captan; however, repeated applications are typically necessary for even minimal disease control. |
| Yaayy!! Your peach plant is healthy. But, you should have deep sandy soil that ranges from a loam to a clay loam for healthy growth. Poor drainage in the soil will kill the root system of growing peach trees, so make sure the soil is well drained. Growing peach trees prefer a soil pH of around 6.5. |

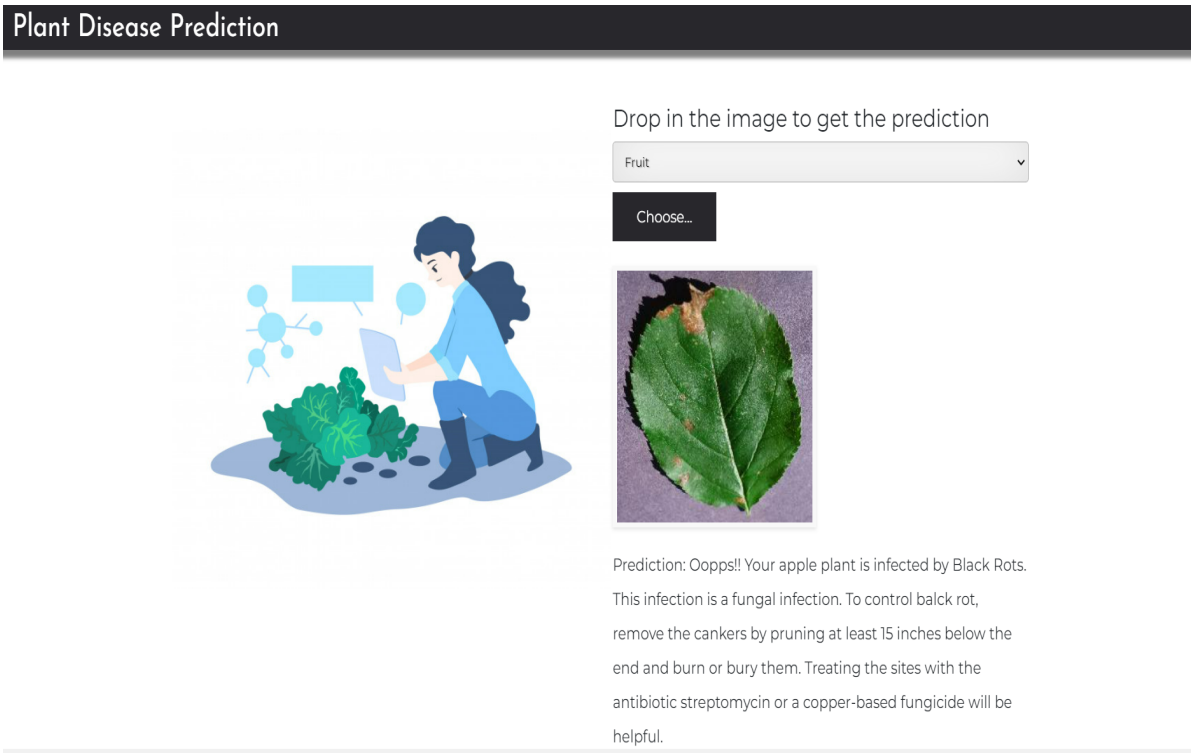
| **Vegetables** |
| --- |
| Oopps!! Your pepper plant is infected by Bacterial Leaft Spot. The disease cycle can be stopped by using the Sango formula for disinfectants. Bleach treatment and hot water treatment is also helpful. |
| Yaayy!! Your pepper plant is healthy. But, take the necessary precautions like, putting the plant where it gets at least 10 hours of direct sunlight. Keep soil evenly moist for good growth. Peppers need well draining soil that is rich and loamy, but avoid too much nitrogen in the soil. Too much nitrogen can cause plenty of leaves and little to no peppers. Your soil should have a pH between 6.0 and 6.5. |
| Oopps!! Your potato plant is Early Blight. Avoid irrigation in cool cloudy weather and time irrigation to allow plants time to dry before nightfall. Protectant fungicides (e.g. maneb, |

| |
|---|
| mancozeb, chlorothalonil, and triphenyl tin hydroxide) are effective. |
| Oopps!! Your potato plant is Late Blight. The late blight can be effectively managed with prophylactic spray of mancozeb, cymoxanil+mancozeb or dimethomorph+mancozeb. |
| Yaayy!! Your potato plant is healthy. But, take the necessary precautions like, putting the plant where it gets at least 10 hours of direct sunlight. Potatoes do best in well-drained and fertile soil. Maintain the pH between 5.0 and 5.5. Keep soil evenly moist for good growth. Do not add large amounts of organic matter to the soil as it may contribute to potato scab, a disease that frequently infects potatoes. |
| Oopps!! Your tomato plant is effected by bacterial spots. To protect the uninfected plants remove the infected leaves and bury or burn them s there is no cure for this infection. To prevent future infections plant pathogen-free seeds or transplants to prevent the introduction of bacterial spot pathogens on contaminated seed or seedlings. |
| Oopps!! Your tomato plant is late blight. Early treatment for this disease is needed. Fungicides like e Daconil fungicides from GardenTech brand prevent, stop, and control late blight and more than 65 types of fungal disease. Planting resistant cultivars and watering the plants early in the mornings help to prevent this infection. |
| Oopps!! Your tomato plant has leaf molds. Watering the plants early in the mornings help them to get sufficient time to dry out. Fungicidal sprays mostly calcium chloride based sparys help in getting rid of leaf molds. |
| Oopps!! Your tomato plant is infected by Septoria leaf spot. Removing the infected leaves immediately will curb the spread of infection. Organic and chemical fungicides with chlorothalonil are effective in treatment. |

After the models deployed as web application, different fruit and vegetable leaves are uploaded and predictions are observed. A few predictions are given below:

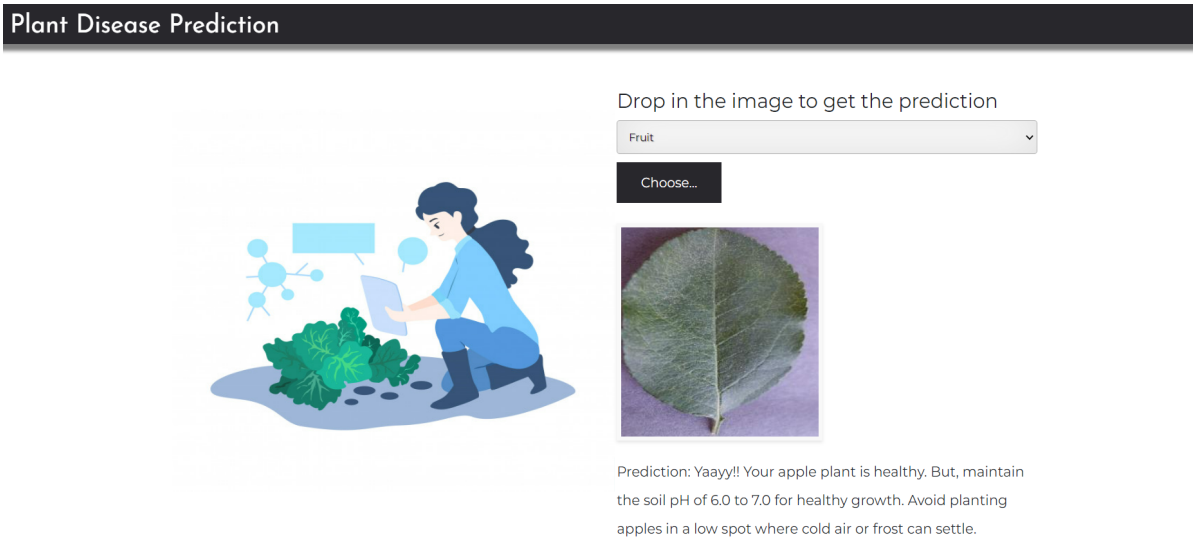
Uploaded Image Class: 'Apple: Black Rots'

Image with Prediction & Caution:



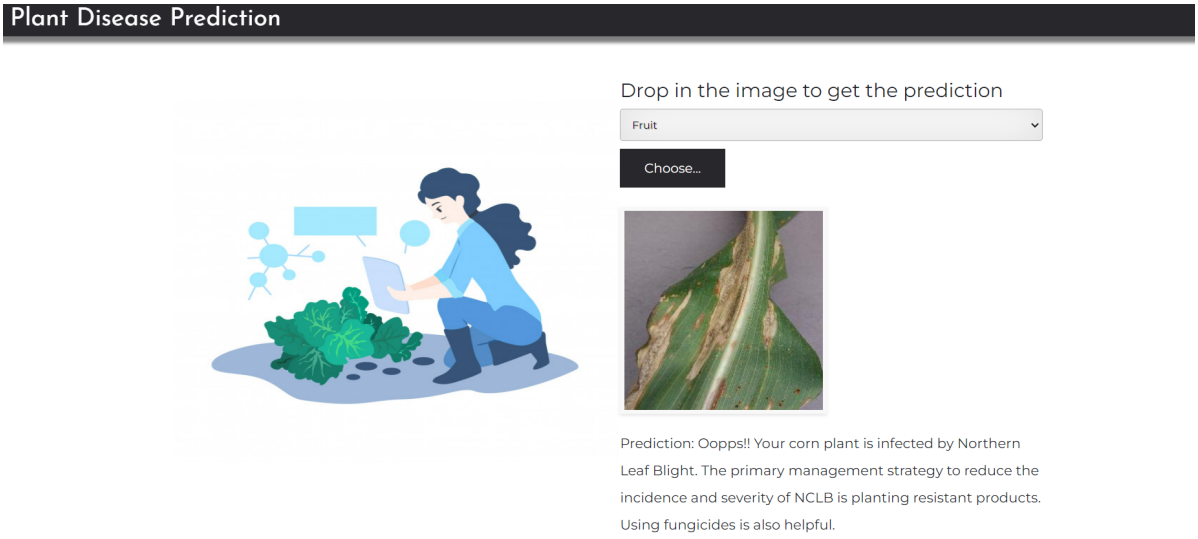

Uploaded Image Class: 'Apple: Healthy'

Image with Prediction & Caution:



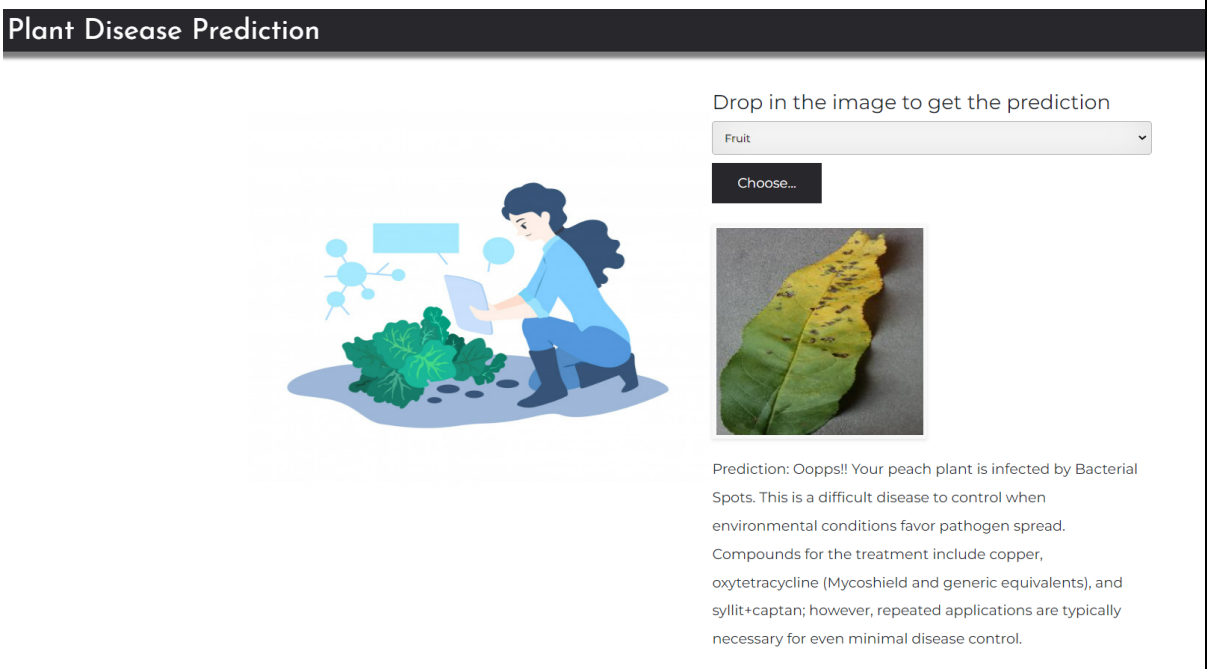
Uploaded Image Class: 'Corn: Northern Leaf Blight'

Image with Prediction & Caution:



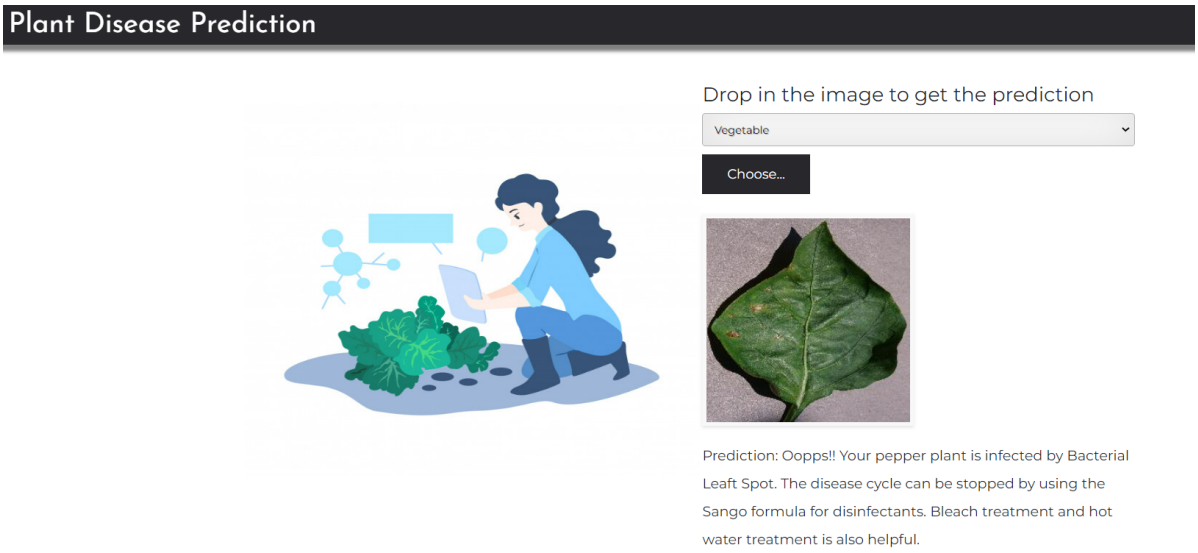

Uploaded Image Class: 'Peach: Bacterial Spots'

Image with Prediction & Caution:



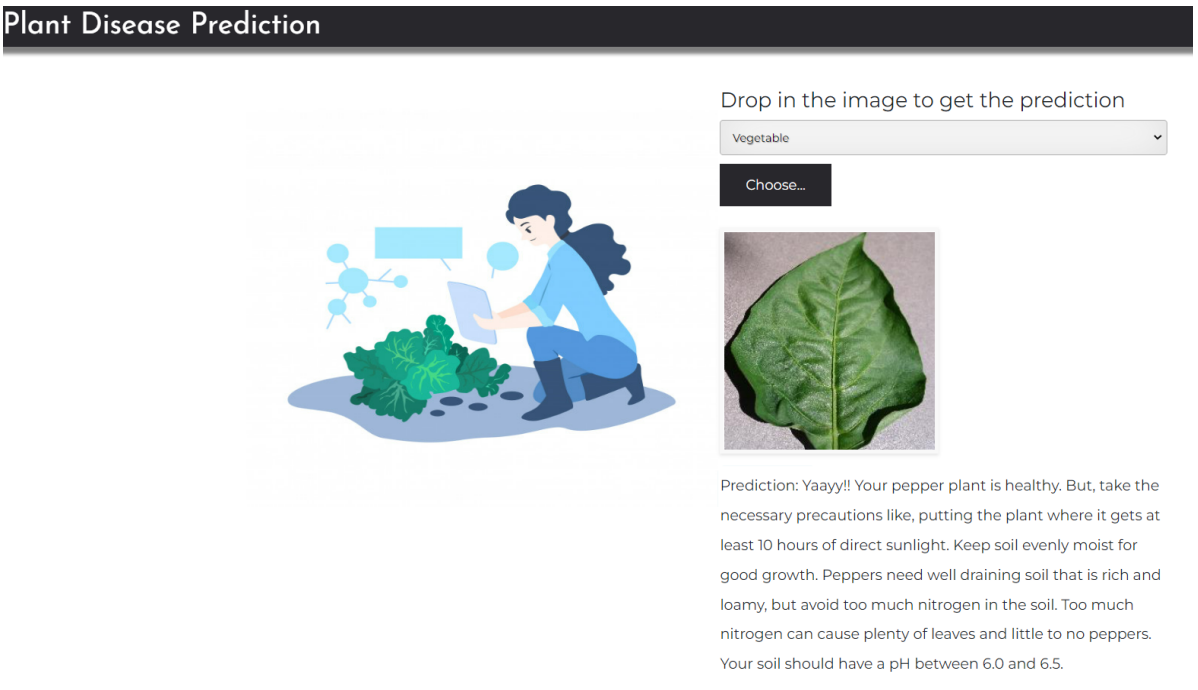
Uploaded Image Class: 'Pepper: Bacterial Leaft Spots'

Image with Prediction & Caution:



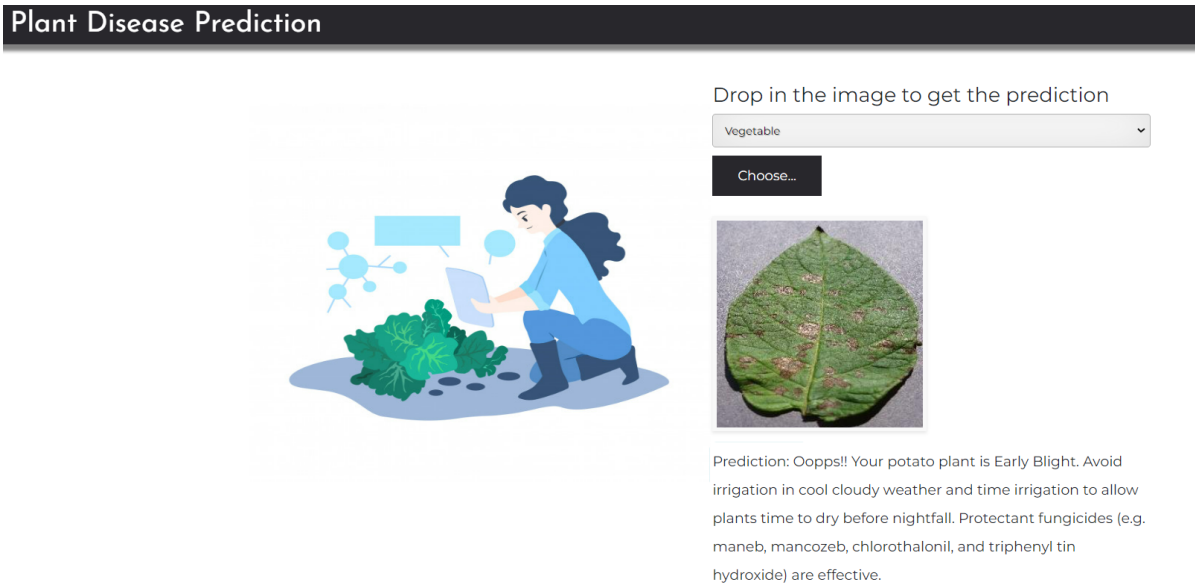

Uploaded Image Class: 'Pepper: Healthy'

Image with Prediction & Caution:



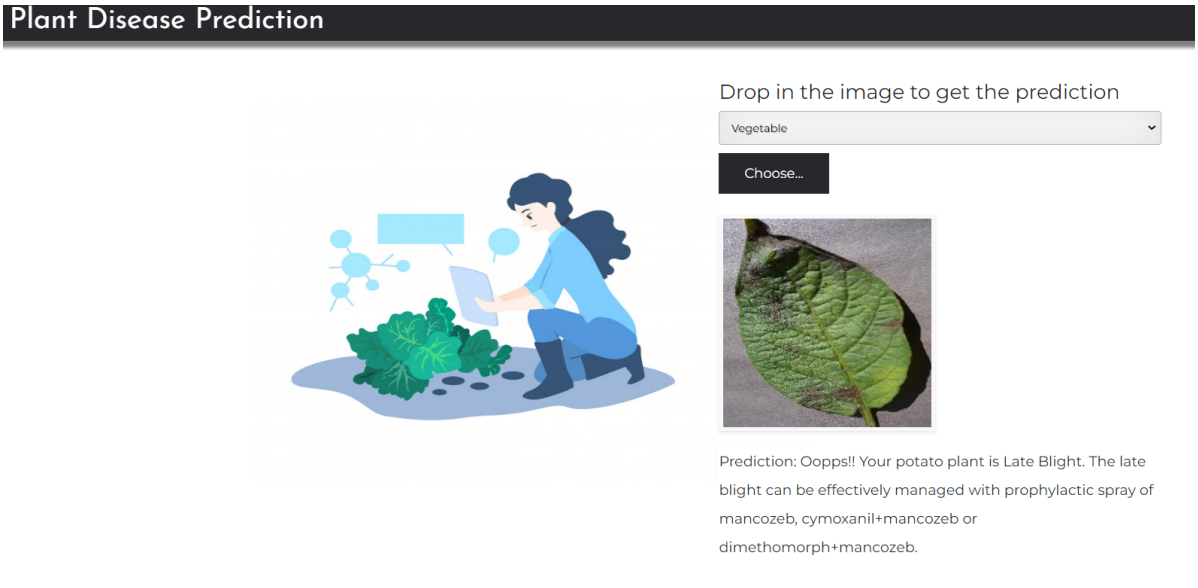
Uploaded Image Class: 'Potato: Early Blight'

Image with Prediction & Caution:



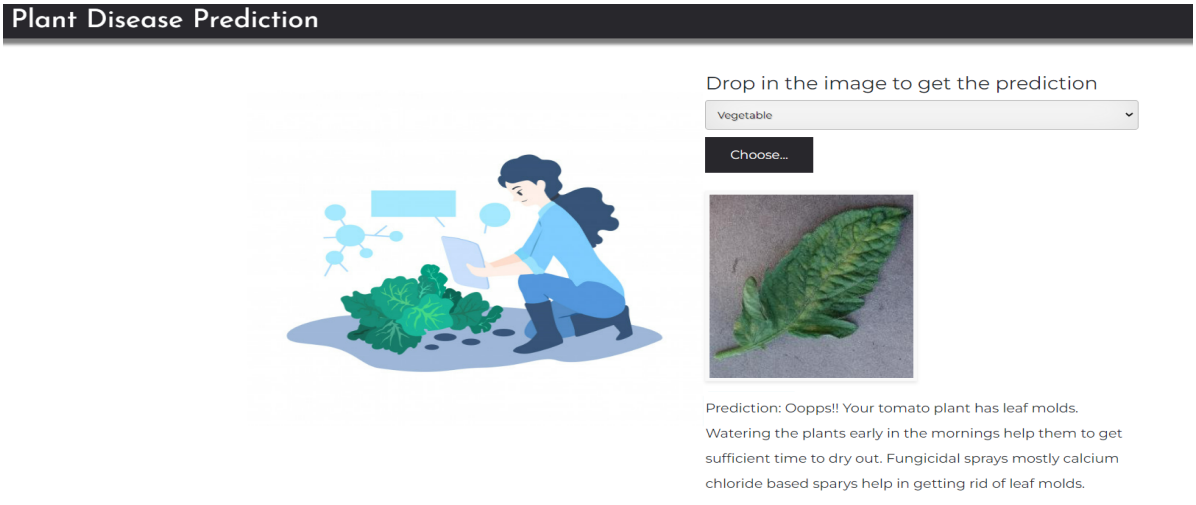

Uploaded Image Class: 'Potato: Late Blight'

Image with Prediction & Caution:



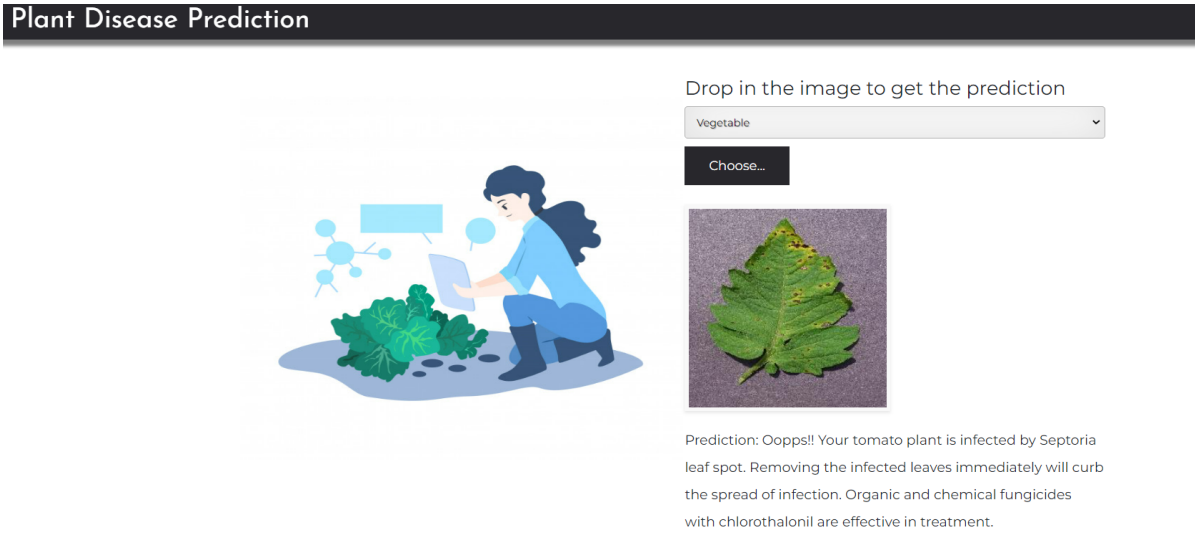
Uploaded Image Class: 'Tomato: Leaf Molds'

Image with Prediction & Caution:




Uploaded Image Class:

Image with Prediction & Caution:




# 7 CONCLUSION

As the accuracy of both the fruit and vegetable models is very high, the models successfully predict the condition of the given plant through the image of the plant uploaded through the web application. Therefore, it is expected that if the application is deployed in online portal, the user will get a proper prediction of the state of the plant and utilize the suggestion given in the form of caution.

## 8 FUTURE SCOPE

- Different combinations of Convolutional layers with different layer thickness can be tested to observe its effect on model accuracy.
- Other features of ImageDataGenerator can be utilized for image augmentation to regularize the training process.
- The effect of batch sizes of both the training set and validation set on model accuracy can be studied.
- Use of Transfer Learning

## APPENDIX

### Code for Fruit-Training Model:

# Preprocess the Images

```
# Required library for image data augmentation:
from keras.preprocessing.image import ImageDataGenerator

# Creation of Instance of the ImageDataGenerator class for train and test:
train_imagen = ImageDataGenerator(rescale = 1.0/255,
                                  shear_range = 0.2,
#                                 rotation_range = 10, fill_mode='nearest',
#                                 width_shift_range=0.2,
height_shift_range=0.2,
                                  horizontal_flip= True,
vertical_flip=False,
                                  brightness_range=[0.8,1.2],
                                  zoom_range= 0.2)
test_imagen = ImageDataGenerator(rescale = 1.0/255)

# Import Data
train_path = r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant
Disease\fruit-dataset\fruit-dataset\train"
test_path = r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant
Disease\fruit-dataset\fruit-dataset\test"
x_train = train_imagen.flow_from_directory(train_path, target_size=
(128,128), class_mode= "categorical", batch_size= 32)
x_test = test_imagen.flow_from_directory(test_path, target_size= (128,128),
```

```
class_mode= "categorical", batch_size= 32)
print("Categories with lebel = ", x_train.class_indices)
print("Length of Train Data = ", len(x_train), "," "Length of Test Data =
", len(x_test))
```

```
Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.
Categories with lebel =  {'Apple___Black_rot': 0, 'Apple___healthy': 1,
'Corn_(maize)___Northern_Leaf_Blight': 2, 'Corn_(maize)___healthy': 3,
'Peach___Bacterial_spot': 4, 'Peach___healthy': 5}
Length of Train Data =  169 ,Length of Test Data =  53
```

# Model Building for Fruit Disease Prediction

```
# Import Libraries for Model Training:
# Import the libraries that are required to initialize the neural network
layer,
# and create and add different layers to the neural network model.
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPool2D, Flatten
from keras.layers import Dense

# Model Construction:
model = Sequential()
model.add(Convolution2D(16, (3,3), input_shape = (128,128,3),
activation='relu', padding='same'))
model.add(MaxPool2D(pool_size= (2,2)))
model.add(Convolution2D(32, (3,3), input_shape = (128,128,3),
activation='relu'))
model.add(MaxPool2D(pool_size= (2,2)))
model.add(Convolution2D(64, (3,3), input_shape = (128,128,3),
activation='relu'))
model.add(MaxPool2D(pool_size= (2,2)))
model.add(Flatten())
model.summary()
```

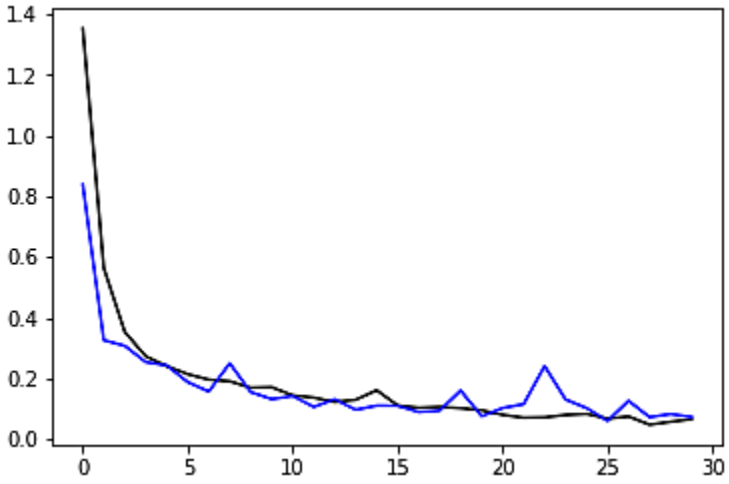
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 16)      448

 max_pooling2d (MaxPooling2D  (None, 64, 64, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 62, 62, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 31, 31, 32)       0
```

```
 2D)

 conv2d_2 (Conv2D)            (None, 29, 29, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)        0
 2D)

 flatten (Flatten)            (None, 12544)             0

=================================================================
Total params: 23,584
Trainable params: 23,584
Non-trainable params: 0
_________________________________________________________________
```

```
# Add Dense Layers in Model:
model.add(Dense(units=40,  kernel_initializer = "uniform", activation=
"relu"))            # 1st Hidden Layer
model.add(Dense(units=20, kernel_initializer = "random_uniform",
activation= "relu"))      # 2nd Hidden Layer
model.add(Dense(units=6, kernel_initializer = "random_uniform",
activation= "softmax"))    # Output Layer

# Creating Checkpoint:
from keras.callbacks import ModelCheckpoint
checkpoint_filepath =
r"C:\Users\ASUS\Documents\python\IBM_Project\checkpoints"
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_loss',
    mode='min',
    save_best_only=True,
    verbose =1)

# Model Compilation & Training:
model.compile(optimizer= "adam", loss= "categorical_crossentropy",
metrics= ['accuracy'])
history = model.fit(x_train, steps_per_epoch= len(x_train)-1,
                    validation_data= x_test, validation_steps=
len(x_test)-1,
                    callbacks=[model_checkpoint_callback], epochs = 30);
Epoch 1/30
168/168 [==============================] - ETA: 0s - loss: 1.3521 -
accuracy: 0.4695
Epoch 1: val_loss improved from inf to 0.83816, saving model to
```

```
C:\Users\ASUS\Documents\python\IBM_Project\checkpoints
168/168 [==============================] - 36s 210ms/step - loss: 1.3521 -
accuracy: 0.4695 - val_loss: 0.8382 - val_accuracy: 0.6881
...
...
Epoch 26/30
168/168 [==============================] - ETA: 0s - loss: 0.0681 -
accuracy: 0.9763
Epoch 26: val_loss improved from 0.07598 to 0.06067, saving model to
C:\Users\ASUS\Documents\python\IBM_Project\checkpoints
168/168 [==============================] - 36s 212ms/step - loss: 0.0681 -
accuracy: 0.9763 - val_loss: 0.0607 - val_accuracy: 0.9790
...
Epoch 30/30
168/168 [==============================] - ETA: 0s - loss: 0.0663 -
accuracy: 0.9776
Epoch 30: val_loss did not improve from 0.06067
168/168 [==============================] - 35s 211ms/step - loss: 0.0663 -
accuracy: 0.9776 - val_loss: 0.0735 - val_accuracy: 0.9730
```

```
# The model weights (that are considered the best) are loaded into the
model.
model.load_weights(checkpoint_filepath)

# Saving Model
model.save("fruit.h5")

# Visualization of Training and Validation Losses
import matplotlib.pyplot as plt
train_loss = history.history["loss"]
val_loss = history.history["val_loss"]
train_acc = history.history["val_accuracy"]
val_acc = history.history["val_accuracy"]
plt.plot(range(30), train_loss, 'k', range(30), val_loss, 'b')
#plt.plot(range(30), train_acc, range(30), val_acc)
```




# Testing of Model:

```
# Import Libraries for Model Testing:
from keras.models import load_model
from keras.preprocessing.image import image_utils
import numpy as np

# Load Model:
model = load_model("fruit.h5")

# Test Image Path:
testimg_folder_basic =
r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant Disease\fruit-
dataset\fruit-dataset\test"
testimg_folder_sub = "\Corn_(maize)___Northern_Leaf_Blight"
testimg = "\\8c674c27-dcc1-4ce3-a3e7-669f9dd3521b___RS_NLB 3867.jpg"
testimg_path = testimg_folder_basic + testimg_folder_sub  + testimg
# Categories:
categories =  ['Apple: Black Rots', 'Apple: Healthy',
               'Corn: Northern Leaf Blight','Corn: Healthy',
               'Peach: Bacterial Spots', 'Peach: Healthy']

# Load Test Image:
test_img = image_utils.load_img(testimg_path ,target_size = (128,128))
# Convert image to array:
test_img = image_utils.img_to_array(test_img)
# Expand Dimention
test_img = np.expand_dims(test_img, axis = 0)

# Final Prediction:
prediction = model.predict(test_img)
print("")
```

```
print("Image category = ", categories[np.argmax(prediction)])
1/1 [==============================] - 0s 77ms/step

Image category =  Corn: Northern Leaf Blight
```

## Code for Vegetable-Training Model:

# Preprocess the Images for Vegetable Training

```
# Required library for image data augmentation:
from keras.preprocessing.image import ImageDataGenerator

# Creation of Instance of the ImageDataGenerator class for train and test:
train_imagen = ImageDataGenerator(rescale = 1.0/255,
                                  shear_range = 0.2,
                                  rotation_range = 10, fill_mode='nearest',
                                  width_shift_range=0.2,
height_shift_range=0.2,
                                  horizontal_flip= True,
vertical_flip=False,
                                  brightness_range=[0.8,1.2],
                                  zoom_range= 0.2)
test_imagen = ImageDataGenerator(rescale = 1.0/255)

# Import Data
train_path = r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant
Disease\Veg-dataset\Veg-dataset\train_set"
test_path = r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant
Disease\Veg-dataset\Veg-dataset\test_set"
x_train = train_imagen.flow_from_directory(train_path, target_size=
(128,128), class_mode= "categorical", batch_size= 32)
x_test = test_imagen.flow_from_directory(test_path, target_size= (128,128),
class_mode= "categorical", batch_size= 32)
print("Categories with lebel = ", x_train.class_indices)
print("Length of Train Data = ", len(x_train), "," "Length of Test Data =
", len(x_test))
Found 11386 images belonging to 9 classes.
Found 3416 images belonging to 9 classes.
Categories with lebel =  {'Pepper,_bell___Bacterial_spot': 0,
'Pepper,_bell___healthy': 1, 'Potato___Early_blight': 2,
'Potato___Late_blight': 3, 'Potato___healthy': 4,
'Tomato___Bacterial_spot': 5, 'Tomato___Late_blight': 6,
'Tomato___Leaf_Mold': 7, 'Tomato___Septoria_leaf_spot': 8}
```

```
Length of Train Data =  356 ,Length of Test Data =  107
```

# Model Building for Vegitable Disease Prediction

```
# Import Libraries for Model Training:
# Import the libraries that are required to initialize the neural network layer,
# and create and add different layers to the neural network model.
from keras.models import Sequential
from keras.layers import Convolution2D, MaxPool2D, Flatten
from keras.layers import Dense, Dropout

# Model Construction:
model = Sequential()
model.add(Convolution2D(16, (3,3), input_shape = (128,128,3), activation='relu'))
model.add(MaxPool2D(pool_size= (2,2)))
model.add(Convolution2D(32, (3,3), input_shape = (128,128,3), activation='relu'))
model.add(MaxPool2D(pool_size= (2,2)))
model.add(Convolution2D(64, (3,3), input_shape = (128,128,3), activation='relu'))
model.add(MaxPool2D(pool_size= (2,2)))

model.add(Flatten())
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 max_pooling2d (MaxPooling2D  (None, 63, 63, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten (Flatten)           (None, 12544)             0
```

```
=================================================================
Total params: 23,584
Trainable params: 23,584
Non-trainable params: 0
_________________________________________________________________
```

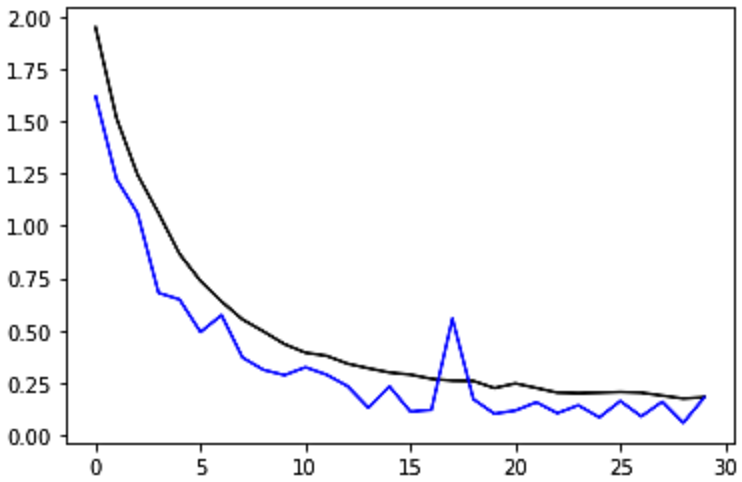
```
# Add Dense Layers in Model:
model.add(Dense(units=400, kernel_initializer = "uniform", activation=
"relu"))          # 1st Hidden Layer
# model.add(Dropout(0.1))
model.add(Dense(units=300, kernel_initializer = "uniform", activation=
"relu"))          # 2nd Hidden Layer
model.add(Dropout(0.1))
model.add(Dense(units=200, kernel_initializer = "uniform", activation=
"relu"))          # 3rd Hidden Layer
model.add(Dropout(0.2))
model.add(Dense(units=100, kernel_initializer = "uniform", activation=
"relu"))          # 4th Hidden Layer
model.add(Dropout(0.5))
model.add(Dense(units=9, activation= "softmax"))
# Output Layer
```

```
# Creating Checkpoint:
from keras.callbacks import ModelCheckpoint
checkpoint_filepath =
r"C:\Users\ASUS\Documents\python\IBM_Project\checkpoints"
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_loss',
    mode='min',
    save_best_only=True,
    verbose =1)
```

```
# Model Compilation & Training:
model.compile(optimizer= "adam", loss= "categorical_crossentropy",
metrics= ['accuracy'])
history = model.fit(x_train, steps_per_epoch= len(x_train)-1,
                    validation_data= x_test, validation_steps=
len(x_test)-1,
                    callbacks=[model_checkpoint_callback], epochs = 30);
Epoch 1/30
355/355 [==============================] - ETA: 0s - loss: 1.9498 -
accuracy: 0.2524
Epoch 1: val_loss improved from inf to 1.62067, saving model to
```

```
C:\Users\ASUS\Documents\python\IBM_Project\checkpoints
355/355 [==============================] - 94s 262ms/step - loss: 1.9498 -
accuracy: 0.2524 - val_loss: 1.6207 - val_accuracy: 0.3821
...
Epoch 29: val_loss improved from 0.08391 to 0.05782, saving model to
C:\Users\ASUS\Documents\python\IBM_Project\checkpoints
355/355 [==============================] - 85s 238ms/step - loss: 0.1739 -
accuracy: 0.9469 - val_loss: 0.0578 - val_accuracy: 0.9791
Epoch 30/30
355/355 [==============================] - ETA: 0s - loss: 0.1805 -
accuracy: 0.9461
Epoch 30: val_loss did not improve from 0.05782
355/355 [==============================] - 84s 236ms/step - loss: 0.1805 -
accuracy: 0.9461 - val_loss: 0.1793 - val_accuracy: 0.9399
```

```
# The model weights (that are considered the best) are loaded into the
model.
model.load_weights(checkpoint_filepath)

# Saving Model
model.save("vegetable.h5")

# Visualization of Training and Validation Losses
import matplotlib.pyplot as plt
train_loss = history.history["loss"]
val_loss = history.history["val_loss"]
train_acc = history.history["val_accuracy"]
val_acc = history.history["val_accuracy"]
plt.plot(range(30), train_loss, 'k', range(30), val_loss, 'b')
#plt.plot(range(30), train_acc, range(30), val_acc)
```




# Testing of Model:

```
# Import Libraries for Model Testing:
from keras.models import load_model
from keras.preprocessing.image import image_utils
import numpy as np

# Load Model:
model = load_model("vegetable.h5")

# Test Image Path:
testimg_folder_basic =
r"C:\Users\ASUS\Documents\python\IBM_Project\Dataset Plant Disease\Veg-
dataset\Veg-dataset\test_set"
testimg_folder_sub = "\Tomato___Late_blight"
testimg = "\\b8b5b34a-6856-4c79-9681-047c1aa47ae0___RS_Late.B 6222.jpg"
testimg_path = testimg_folder_basic + testimg_folder_sub  + testimg
# Categories:
categories =  ['Pepper: Bacterial Leaft Spots', 'Pepper: Healthy',
               'Potato: Early Blight', 'Potato: Healthy', 'Potato: Late
Blight',
               'Tomato: Bacterial Spots', 'Tomato: Late Blight', 'Tomato:
Leaf Molds', 'Tomato: Septoria Leaf Spot']

# Load Test Image:
test_img = image_utils.load_img(testimg_path ,target_size = (128,128))
# Convert image to array:
test_img = image_utils.img_to_array(test_img)
# Expand Dimention
test_img = np.expand_dims(test_img, axis = 0)


# Final Prediction:
prediction = model.predict(test_img)
print("")
print("Image category = ", categories[np.argmax(prediction)])
1/1 [==============================] - 0s 17ms/step

Image category =  Tomato: Late Blight
```

## Code for Flask Application:

```
import numpy as np
import pandas as pd
import os
```

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask, render_template, request

app = Flask(__name__)

model_1 = load_model('fruit.h5')
model_2 = load_model('vegitable.h5')

# Home Page:
@app.route('/')
def home():
    return render_template("home.html")

# Prediction Page:
@app.route('/prediction')
def prediction():
    return render_template("predict.html")

@app.route('/predict', methods=['POST'])
def upload():
    if request.method == 'POST':
        f=request.files['image']
        basepath = os.path.dirname(__file__)
        filepath = os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)

        img = image.load_img(filepath, target_size = (128,128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        plant = request.form['plant']
        print(plant)
```

```
        # Prediction for Fruits or Vegitables:
        if(plant == 'fruit'):
            # Prediction for Fruits:
            preds = np.argmax(model_1.predict(x), axis=1)
            #print(preds)
            df = pd.read_excel('precautions - fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            # Prediction for Vegitables:
            preds = np.argmax(model_2.predict(x), axis=1)
            #print(preds)
            df = pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']


if __name__ == '__main__':
    app.run(debug=False)
```