

**FERTILIZERS RECOMMENDATION SYSTEM  
FOR DISEASE PREDICTION**

**PROJECT REPORT**

Submitted by

**Team ID: PNT2022TMID34154**

Maneesha M S (960219106085),

Karthika D L (960219106079),

Devi Narayani A N (960219106058),

Bijon Sathya S (960219106052)

In partial fulfillment of the award of the degree

Of

**BACHELOR OF ENGINEERING**

In

**ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**ARUNACHALA COLLEGE OF ENGINEERING  
FOR WOMEN**

## 1. INTRODUCTION

**1.1 Overview** In this project, two datasets name fruit dataset, and the vegetable dataset are collected. The collected datasets are trained and tested with deep learning neural network named Convolutional Neural Networks (CNN). First, the fruit dataset is trained and then tested with CNN. It has 6 classes and all the classes are trained and tested. Second, the vegetable dataset is trained and tested. The software used for training and testing datasets in Python. All the Python codes are first written in Jupyter notebook supplied along with Anaconda Python and then the codes are tested in the IBM cloud. Finally, a web-based framework is designed with the help of Flask a Python library. There are 2 HTML files are created in the templates folder along with their associated files in the static folder. The Python program 'app.py' used to interface with these two webpages is written in Spyder-Anaconda python and tested.

**1.2 Purpose** This project is used to test the fruit and vegetable samples and identify the different diseases. Also, this project recommends fertilizers for predicted diseases.

## 2. LITERATURE SURVEY

**2.1 Existing problem** Indumathi proposed a method for leaf disease detection and suggest fertilizers to cure leaf diseases. But the method involves less number of train and test sets which results in poor accuracy. Pandi Selvi proposed a simple prediction method for a soil-based fertilizer recommendation system for predicted crop diseases. This method gives less accuracy and prediction. Shiva reddy proposed an IoT-based system for leaf disease detection and fertilizer recommendation which is based on Machine Learning techniques and yields less than 80 percent accuracy.

**2.2 Proposed solution** In this project work, a deep learning-based neural network is used to train the collected datasets and test the same. The deep learning-based neural network is CNN which gives more than 90% classification accuracies. By increasing the number of dense layers and by modifying hyperparameters such as the number of epochs, and batch size, the accuracy rate can be increased from 95% to 98%.

## 3. THEORETICAL ANALYSIS

### 3.1 Block diagram

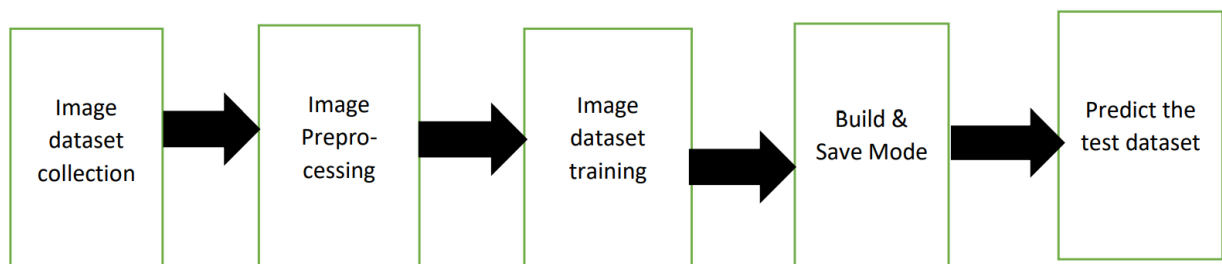


Figure.3.1. Block Diagram of the project

The block diagram of the entire project is shown in Fig.3.1. First step is the image dataset collection followed by image preprocessing. The third step is the training of image datasets by initializing different hyperparameters. Then build the model and save the model file in .h5 format. The final stage is the testing of existing or new datasets using the trained model.

### 3.2 Hardware/Software designing

The software used for training and testing the dataset is Python. The Jupyter notebook (Notebook of IBM cloud also) is used for python programming. The neural network used for training and testing the model is Convolutional Neural Network (CNN).

The CNN has following layers:

- Convolutional layer (32x32 kernel (3x3))
- Max-pool layer (kernel(2x2))
- Flatten layer
- Dense layer (different layers with different size)
- Drop out layer (optional)
- Final output dense layer(size 6x1 for fruit dataset and 9x1 for Vegetable dataset)

In the preprocessing step, images are normalized to 1 and then resized to 128x128. The images are arranged in different batch sizes. Then train set and test set are formed from the collected datasets. In order to do the above steps in Python, the following Python libraries must be imported before starting the process:

- NumPy
- TensorFlow
- Keras
- Matplotlib (optional for data visualization)

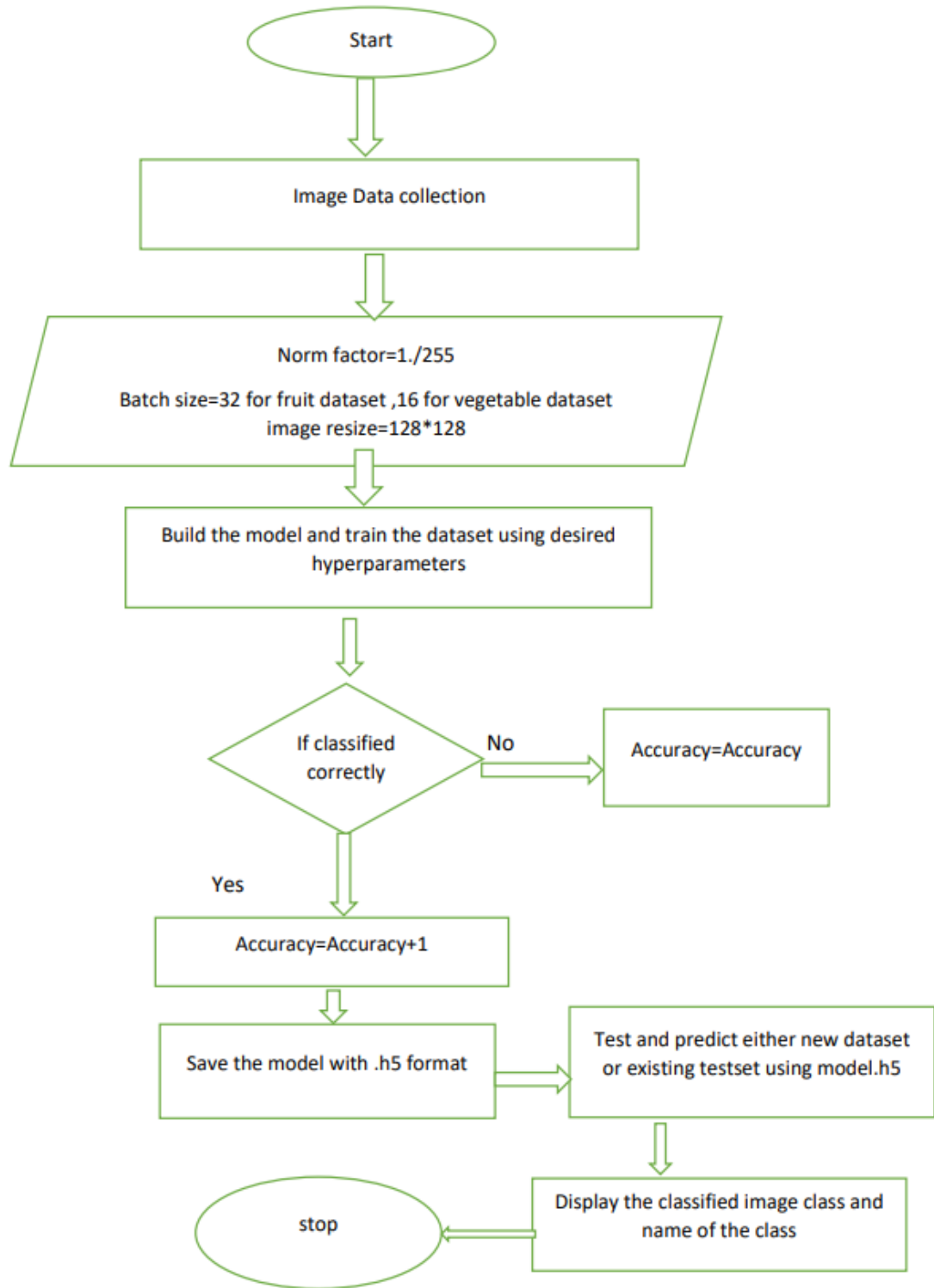
The following activation functions used in the CNN training:

- RELU at the end of convolution layer and Max Pool layer
- SoftMax at the end of output dense layer
- For testing the dataset argmax is used, its an optional

#### **4. EXPERIMENTAL INVESTIGATIONS**

The analysis made while working on the solution The batch sizes are varied and tested. For different batch sizes, the CNN gives different accuracies. The batch size determines the number of iterations per epoch. Another important hyperparameter is the number of epochs. This determines accuracy and it has a high influence on accuracy compared to other hyperparameters. The accuracy can be varied from 80% to 90% in the vegetable dataset and 95% to 98% in the case of the fruit dataset by increasing the number of epochs. The size of the test dataset and train dataset also has a very high influence on accuracies. The accuracy can be increased by using more images in the training dataset. The computational time for model building is increased when the size of the training dataset increases and also the number of epochs increases. The batch size of the training dataset and test dataset also play a vital role in computational time. The Neural Network complexity is increased when more number of convolutional layers increases. If the number of layers increases, better accuracy results will obtain. At the same increasing the number of layers in CNN leads to more training time and also requires more time to build a model. The model .h5 size depends on the size of the train datasets. But the memory requirement depends on the size of the training dataset and CNN architecture complexity.

#### **5. FLOWCHARTS**



## 6. RESULTS

The final findings(output) of the project are given below in the form of a screenshot:

Training and Testing of Fruit dataset

The image displays two screenshots of a Jupyter Notebook titled "Fruit-Training" running on a local host. The notebook is in a Python 3 (ipykernel) environment.

**Top Screenshot:**

- In [40]:** `pred = model.predict_classes(x)`  
A warning message is shown: `WARNING:tensorflow:From <ipython-input-40-f93bd039d5d>:1: Sequential.predict_classes (from tensorflow.python.keras.engine.sequential) is deprecated and will be removed after 2021-02-01. Instructions for updating: Please use instead: "np.argmax(model.predict(x), axis=-1)", if your model does multi-class classification (e.g. if it uses a 'softmax' last-layer activation)." "(model.predict(x) > 0.5).astype("int32")". If your model does binary classification (e.g. if it uses a 'sigmoid' last-layer activation).`
- In [41]:** `pred`  
**Out[41]:** `array([1], dtype=int64)`
- In [42]:** `Index = ['Apple__Black_rot', 'Apple__healthy', 'Corn_(maize)__Northern_leaf_Blight', 'Corn_(maize)__healthy', 'Peach__Bacterial', 'Peach__healthy']`
- In [43]:** `print('the given image belongs to-', Index[pred[0]])`  
Output: `the given image belongs to- Apple__healthy`
- Section Header:** **Test Apple Black Rot class images**
- In [54]:** `img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test/Apple__Black_rot/0f3d65f4-4.jpg')`  
Output: `To call: x=image_img to array(img)`

**Bottom Screenshot:**

- Section Header:** **Fit the Model**
- In [28]:** `model.fit_generator(x_train, steps_per_epoch=100, validation_data=x_test, validation_steps=10, epochs=1)`  
A warning message is shown: `WARNING:tensorflow:From <ipython-input-28-0de952bde51e>:1: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version. Instructions for updating: Please use Model.fit, which supports generators.`  
Output shows training progress:  
`Epoch 1/3  
100/100 [=====] - 465s 3s/step - loss: 1.3844 - accuracy: 0.4589 - val_loss: 107.1929 - val_accuracy: 0.0071  
Epoch 2/3  
100/100 [=====] - 414s 2s/step - loss: 0.6387 - accuracy: 0.7534 - val_loss: 62.9415 - val_accuracy: 0.0143  
Epoch 3/3  
100/100 [=====] - 382s 2s/step - loss: 0.4579 - accuracy: 0.8283 - val_loss: 142.3666 - val_accuracy: 0.7356`
- Out[28]:** `<tensorflow.python.keras.callbacks.History at 0x0205d040>`
- Section Header:** **Save the Model**
- In [29]:** `model.save("fruit.h5")`
- In [30]:** `Is`

localhost:8888/notebooks/Vegetable-Training.ipynb

Channel content - Y...

jupyter Vegetable-Training Last Checkpoint: Last Thursday at 4:15 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[21]: 711.825

### Fit the Model

```
In [27]: model.fit_generator(x_train, steps_per_epoch=80, validation_data=x_test, validation_steps=27, epochs=20)

WARNING:tensorflow:From <ipython-input-27-bc2f09aa13c3>:1: Model.fit_generator (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/20
80/80 [=====] - 223s 3s/step - loss: 2.0686 - accuracy: 0.2193 - val_loss: 199.9088 - val_accuracy: 0.3472
Epoch 2/20
80/80 [=====] - 188s 2s/step - loss: 1.6082 - accuracy: 0.3792 - val_loss: 200.8022 - val_accuracy: 0.2708
Epoch 3/20
80/80 [=====] - 197s 2s/step - loss: 1.4385 - accuracy: 0.4522 - val_loss: 189.9319 - val_accuracy: 0.3819
Epoch 4/20
80/80 [=====] - 192s 2s/step - loss: 1.2958 - accuracy: 0.5323 - val_loss: 278.2356 - val_accuracy: 0.4607
Epoch 5/20
80/80 [=====] - 208s 2s/step - loss: 1.1856 - accuracy: 0.5688 - val_loss: 267.2704 - val_accuracy: 0.4213
Epoch 6/20
80/80 [=====] - 233s 3s/step - loss: 1.0660 - accuracy: 0.6128 - val_loss: 308.8078 - val_accuracy: 0.3149
```

localhost:8888/notebooks/Vegetable-Training.ipynb

Channel content - Y...

jupyter Vegetable-Training Last Checkpoint: Last Thursday at 4:15 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

### Test the Model

```
In [32]: from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np

In [33]: model = load_model("vegetable.h5")

In [38]: index=["Pepper_bell__Bacterial_spot", "Pepper_bell__healthy", "Potato__Early_blight", "Potato__Late_blight", "Potato__healthy"]
```

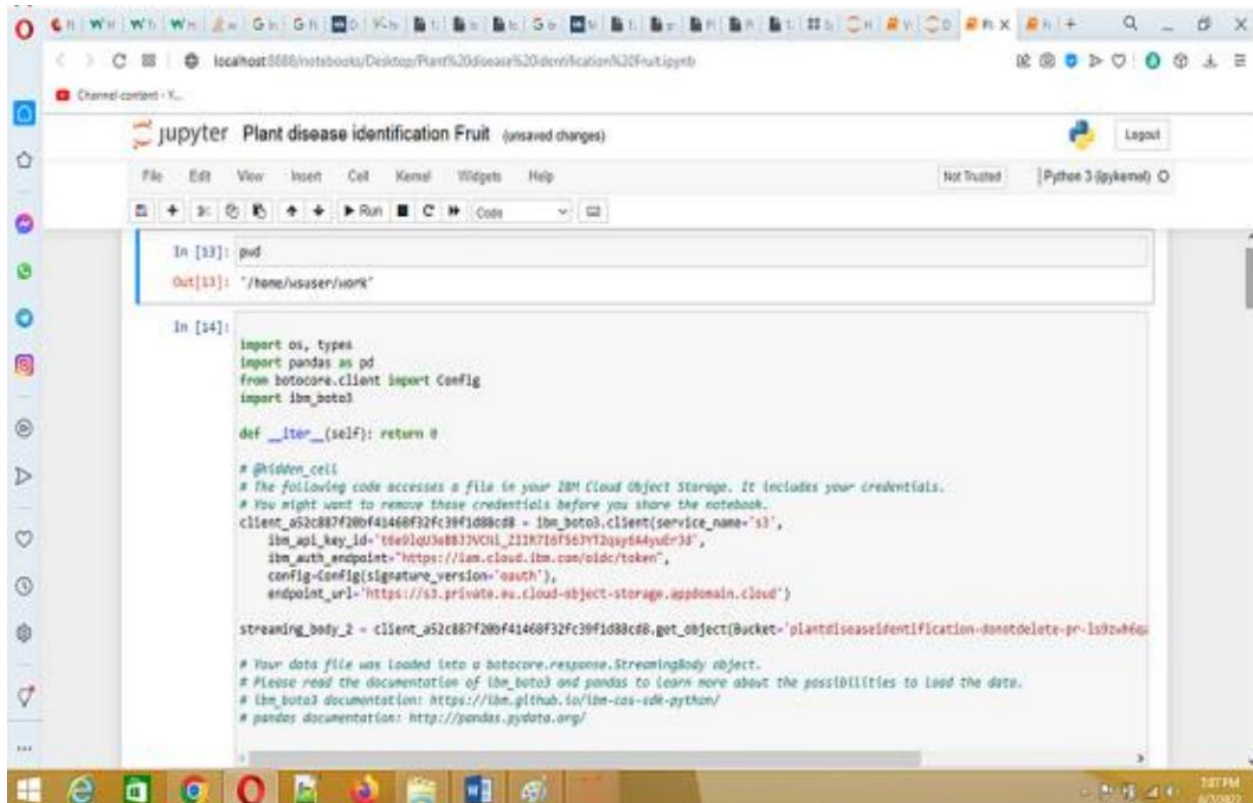
### Test Pepper Bell Bacterial Spot Class Images

```
In [36]: img = image.load_img('E:/IDM_MY_COURSE/Project/Dataset Plant Disease/veg-dataset/veg-dataset/test_set/Pepper_bell__Bacterial_spot')

In [39]: x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
pred = model.predict_classes(x)
print('the given image belongs to-',index[pred[0]])

the given image belongs to- Pepper_bell__Bacterial_spot
```





localhost8080/notebooks/Desktop/Plant%20disease%20identification%20fruit.ipynb

Channel context - X...

jupyter Plant disease identification Fruit (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel)

In [13]: pwd

Out[13]: '/home/vsuser/work'

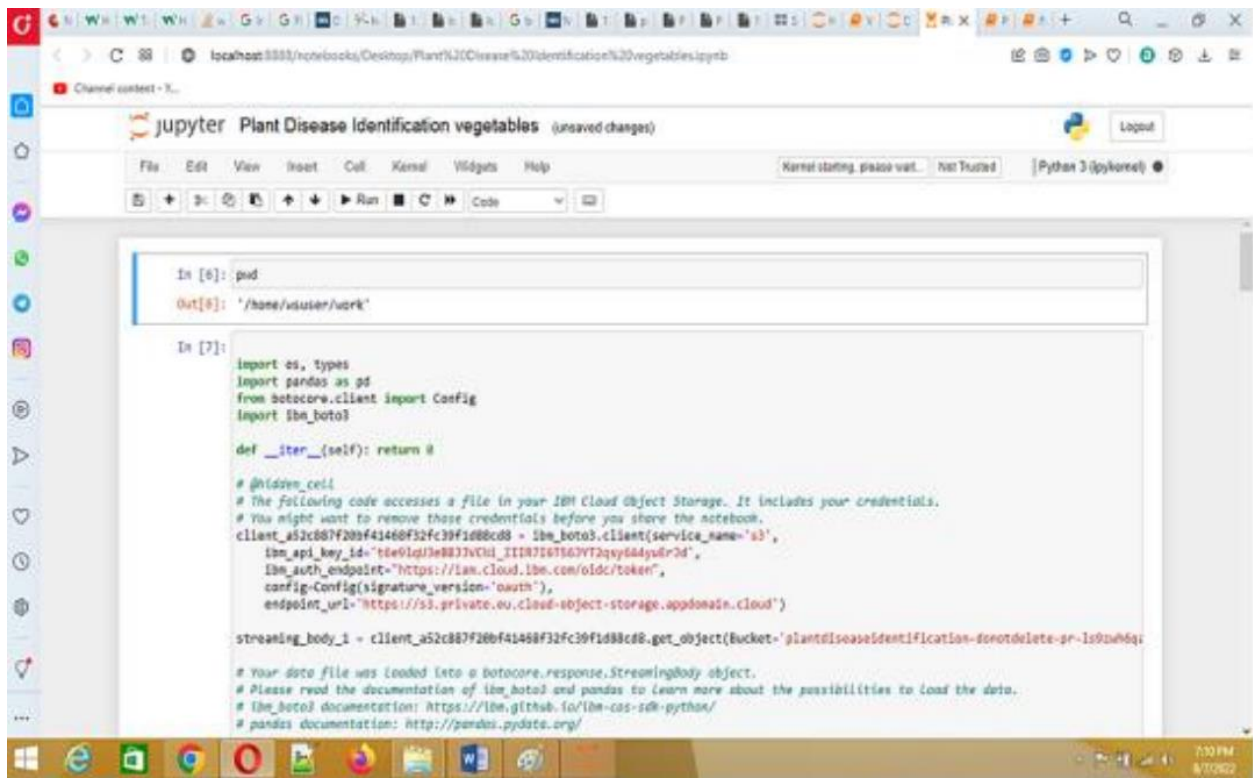
In [14]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_botocore

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove these credentials before you share the notebook.
client_a52c887f28b41468f32fc39f1d88cd8 = ibm_botocore.client(service_name='s3',
    ibm_api_key_id='t6e9lq3e8837vCld_222K7l67563Y72qy644ydr3d',
    ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

streaming_body_2 = client_a52c887f28b41468f32fc39f1d88cd8.get_object(Bucket='plantdiseaseidentification-donotdelete-pr-1s9zw66q
```



localhost8080/notebooks/Desktop/Plant%20disease%20identification%20vegetables.ipynb

Channel context - X...

jupyter Plant Disease Identification vegetables (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Kernel starting, please wait. Not Trusted Python 3 (ipykernel)

In [6]: pwd

Out[6]: '/home/vsuser/work'

In [7]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_botocore

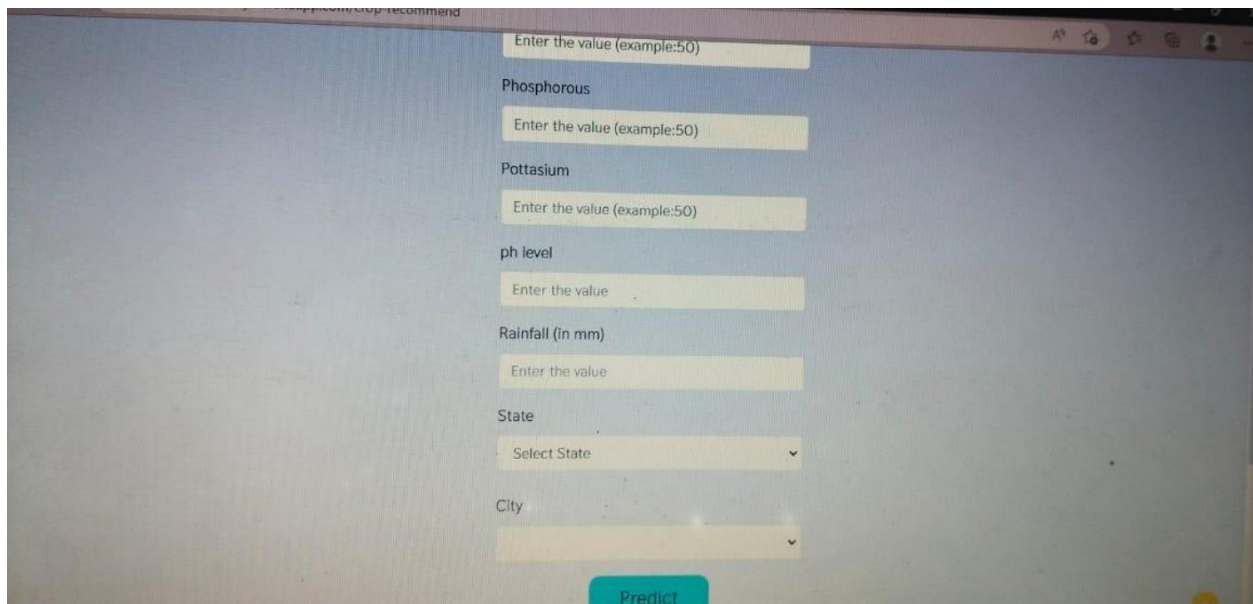
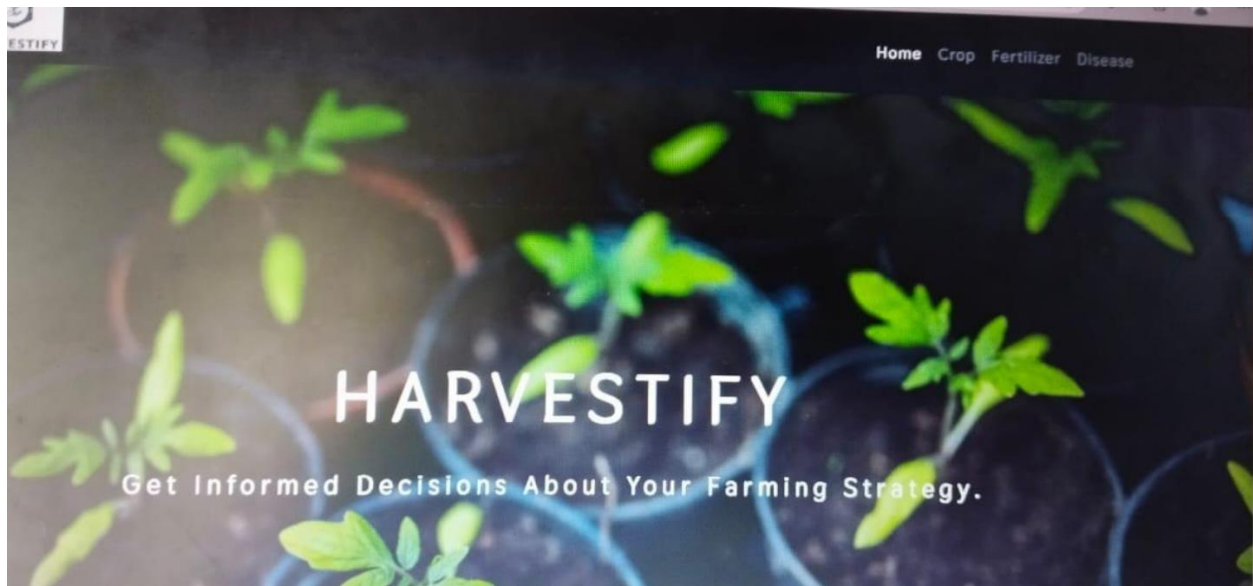
def __iter__(self): return 0


# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove these credentials before you share the notebook.
client_a52c887f28b41468f32fc39f1d88cd8 = ibm_botocore.client(service_name='s3',
    ibm_api_key_id='t6e9lq3e8837vCld_222K7l67563Y72qy644ydr3d',
    ibm_auth_endpoint='https://iam.cloud.ibm.com/oidc/token',
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client_a52c887f28b41468f32fc39f1d88cd8.get_object(Bucket='plantdiseaseidentification-donotdelete-pr-1s9zw66q
```



## OUTPUT

A screenshot of the Harvestify website's prediction form. The form is titled "Crop recommendation" and contains several input fields. The first field is labeled "Enter the value (example:50)". Below it are fields for "Phosphorous" and "Pottasium", both with the placeholder text "Enter the value (example:50)". The next field is labeled "ph level" with the placeholder "Enter the value". This is followed by a field for "Rainfall (in mm)" with the placeholder "Enter the value". Below these are two dropdown menus: "State" with the option "Select State" and "City". At the bottom of the form is a green button labeled "Predict".



HomeCropFertilizerDisease

## Get informed advice on fertilizer based on soil

Nitrogen

Enter the value (example:50)

Phosphorous

Enter the value (example:50)

Pottasium

Enter the value (example:50)


Crop you want to grow

Select crop

Predict

Harvestify - Disease Detection

https://harvestify.herokuapp.com/disease-predict



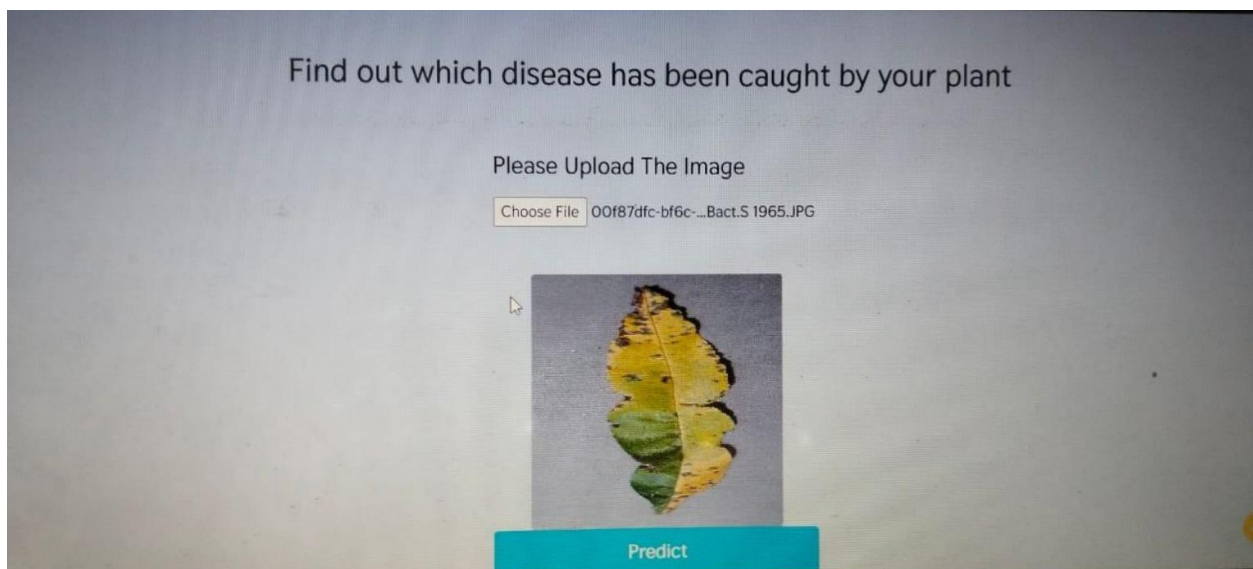
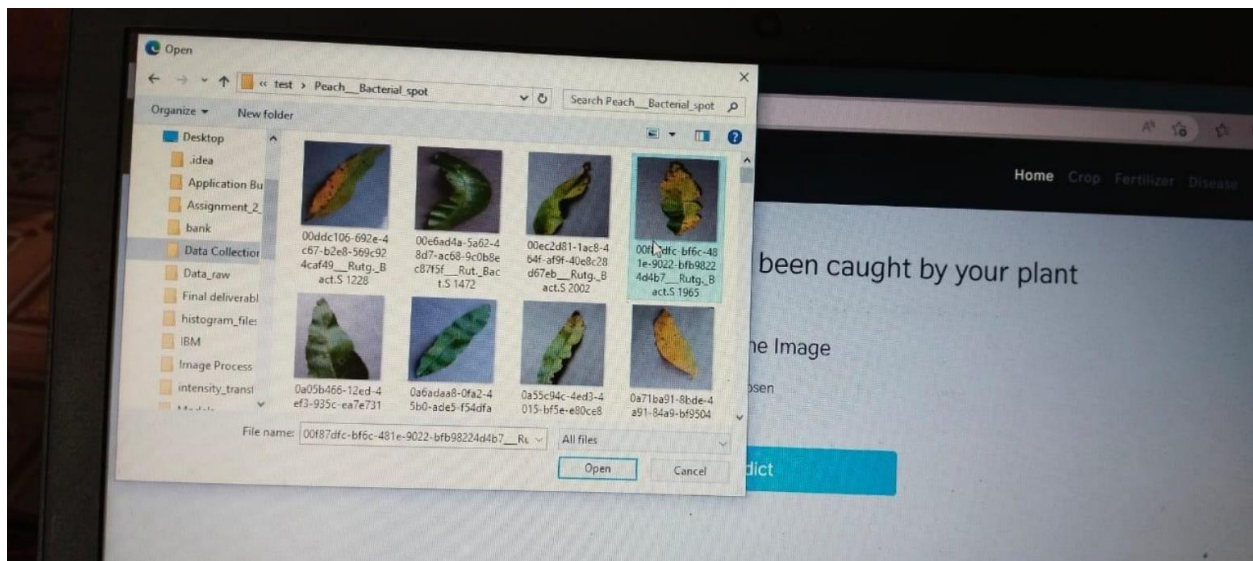
HomeCropFertilizerDisease

## Find out which disease has been caught by your plant

Please Upload The Image

Choose FileNo file chosen

Predict





Crop: Peach

Disease: Bacterial Spot

Cause of disease:

1. The disease is caused by four species of *Xanthomonas* (*X. euvesicatoria*, *X. gardneri*, *X. perforans*, and *X. vesicatoria*). In North Carolina, *X. perforans* is the predominant species associated with bacterial spot on tomato and *X. euvesicatoria* is the predominant species associated with the disease on pepper.
2. All four bacteria are strictly aerobic, gram-negative rods with a long whip-like flagellum (tail) that allows them to move in water, which allows them to invade wet plant tissue and cause infection.

How to prevent/cure the disease

1. The most effective management strategy is the use of pathogen-free certified seeds and disease-free transplants to prevent the introduction of the pathogen into greenhouses and field production areas. Inspect plants very carefully and reject infected transplants- including your own!
2. In transplant production greenhouses, minimize overwatering and handling of seedlings when they are wet.
3. Trays, benches, tools, and greenhouse structures should be washed and sanitized between seedlings crops.
4. Do not spray, tie, harvest, or handle wet plants as that can spread the disease.

## 7.ADVANTAGES & DISADVANTAGES

List of advantages

- The proposed model here produces very high accuracy of classification.
- Very large datasets can also be trained and tested.

- Images of very high can be resized within the proposed itself. List of disadvantages
- For training and testing, the proposed model requires very high computational time.
- The neural network architecture used in this project work has high complexity.

## **8. APPLICATIONS**

1. The trained network model is used to classify the image patterns with high accuracy.
2. The proposed model not only used for plant disease classification but also for other image pattern classification such as animal classification.
3. This project work application involves not only image classification but also for pattern recognition.

## **9.CONCLUSIONS**

The model proposed here involves image classification of fruit datasets and vegetable datasets. The following points are observed during model testing and training:

- The accuracy of classification increased by increasing the number of epochs.
- For different batch sizes, different classification accuracies are obtained.
- The accuracies are increased by increasing more convolution layers.
- The accuracy of classification also increased by varying dense layers.
- Different accuracies are obtained by varying the size of kernel used in the convolution layer output.
- Accuracies are different while varying the size of the train and test datasets.

## **10. FUTURE SCOPE**

The proposed model in this project work can be extended to image recognition. The entire model can be converted to application software using python to exe software. Real-time image classification, image recognition, and video processing are possible with the help OpenCV python library. This project work can be extended for security applications such as figure print recognition, iris recognition, and face recognition.

## **11.BIBLIOGRAPHY**

[1]. R Indumathi Leaf Disease Detection and Fertilizer Suggestion", IEEE International Conference on System, Computation, Automation, and Networking (SCAN), 29-30 March 2019, DOI: 10.1109/ICSCAN.2019.8878781.

[2]. P. Pandi Selvi, P. Poornima, "Soil Based Fertilizer Recommendation System for Crop Disease Prediction System", International Journal of Engineering Trends and Applications (IJETA) – Volume 8 Issue 2, Mar-Apr 2021.

[3]. H Shiva reddy, Ganesh hedge, Prof. DR Chinnaya3, "IoT based Leaf Disease Detection and Fertilizer Recommendation", International Research Journal of Engineering and Technology (IRJET), Volume: 06 Issue: 11, Nov 2019, e-ISSN: 2395- 0056.

## APPENDIX

### A. Source Code (Jupyter notebook python code)

fruit.ipynb (due to limited page size the code vegetable.ipynb uploaded in github)

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]: pwd
```

```
# In[2]: cd E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-dataset
```

```
# # Apply ImageDataGenerator functionality to Train and Test set
```

```
# # Preprocessing # In[3]: from keras.preprocessing.image import
```

```
ImageDataGenerator train_datagen =
```

```
ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True) test_datagen = ImageDataGenerator(rescale=1)
```

```
# In[4]: pwd # In[5]: x_train =
```

```
train_datagen.flow_from_directory('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruitdataset/train', target_size=(128,128), batch_size=32, class_mode='categorical')
```

```
# In[6]: x_test=test_datagen.flow_from_directory('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test', target_size=(128,128), batch_size=32, class_mode='categorical') # # Import the models
```

```
# In[7]: from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense, Convolution2D, MaxPool2D, Flatten
```

```
# # Initializing the models 10
```

```
# In[8]: model=Sequential()
```

```
# # Add CNN Layers
```

```
# In[9]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

```
# In[10]: x_train.class_indices
```

```

## Add Pooling layer

# In[11]: model.add(MaxPool2D(pool_size=(2,2)))

## Add Flatten layer # In[12]: model.add(Flatten())

## Add Dense Layer

# In[21]: model.add(Dense(40, kernel_initializer='uniform',activation='relu'))
model.add(Dense(20, kernel_initializer='random_uniform',activation='relu'))

## Add Output Layer # In[24]: model.add(Dense(6,activation='softmax',
kernel_initializer='random_uniform'))

## Compile the model # In[25]:
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy']) #
In[26]: len(x_train)

# In[27]: 5384/32

## Fit the Model

# In[28]:

model.fit_generator(x_train,steps_per_epoch=168,validation_data=x_test,validation_steps=52,epochs=3)

## Save the Model

# In[29]: model.save("fruit.h5")

# In[30]: ls

## Test the Model

# In[32]: from keras.preprocessing import image from tensorflow.keras.preprocessing.image
import img_to_array from tensorflow.keras.models import load_model import numpy as np

# In[33]: model = load_model("fruit.h5")

## Test Apple_Healthy Class images

# In[37]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Apple__healthy/00fca0da-2db3-
481bb98a9b67bb7b105c__RS_HL_7708.JPG',target_size=(128,128)) 11

# In[39]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0)

# In[40]: pred = model.predict_classes(x)

# In[41]: pred

```



```

# In[45]: index
=['Apple___Black_rot','Apple___healthy','Corn_(maize)___Northern_Leaf_Blight','Corn_(
maize)___healthy','Peach___Bacterial_spot','Peach___healthy'] # In[46]: print('the given
image belongs to=',index[pred[0]])

# # Test Apple Black Rot class images # In[54]: img =
image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-
dataset/test/Apple___Black_rot/0f3d45f4-e121-42cda5b6-be2f866a0574___JR_FrgE.S
2870.JPG',target_size=(128,128))

# In[55]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]]) # # Test Corn
Northern leaf Blight class images

# In[56]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant
Disease/fruitdataset/fruitdataset/test/Corn_(maize)___Northern_Leaf_Blight/00a14441-7a62-
4034-bc40-b196aeab2785___RS_NLB_3932.JPG',target_size=(128,128))

# In[57]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

# # Test Corn Healthy class images # In[58]: img =
image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant Disease/fruitdataset/fruit-
dataset/test/Corn_(maize)___healthy/0a68ef5a-027c41ae-b227-159dae77d3dd___R.S_HL
7969 copy.jpg',target_size=(128,128))

# In[59]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]]) # # Test Peach
Bacterial spot class images

# In[60]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Peach___Bacterial_spot/00ddc106-692e4c67-b2e8-
569c924caf49___Rutg._Bact.S_1228.JPG',target_size=(128,128)) 12

# In[61]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

# # Test Peach Healthy class images

# In[62]: img = image.load_img('E:/IBM_MY_COURSE/Project/Dataset Plant
Disease/fruitdataset/fruit-dataset/test/Peach___healthy/1a07ce54-f4fd-41cfb088-
144f6bf71859___Rutg._HL_3543.JPG',target_size=(128,128))

# In[63]: x=image.img_to_array(img) x=np.expand_dims(x,axis=0) pred =
model.predict_classes(x) print('the given image belongs to=',index[pred[0]])

```