

Fertilizers Recommendation System For Disease Prediction

1 INTRODUCTION

1.1 Overview:

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.

An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

1.2 Purpose:

To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for crop disease prediction.

2 LITERATURE SURVEY

2.1 Existing problem:

Adequate mineral nutrition is central to crop production. However, it can also exert considerable Influence on disease development. Fertilizer application can increase or decrease development of diseases caused by different pathogens, and the mechanisms responsible are complex, including effects of nutrients on plant growth, plant resistance mechanisms and direct effects on the pathogen. The effects of mineral nutrition on plant disease and the mechanisms responsible for those effects have been dealt with comprehensively elsewhere. In India, around 40% of land is kept and grown using reliable irrigation technologies, while the rest relies on the monsoon environment for water. Irrigation decreases reliance on the monsoon, increases food security, and boosts agricultural production.

Most research articles use humidity, moisture, and temperature sensors near

the plant's root, with an external device handling all of the data provided by the sensors and transmitting it directly to an external display or an Android application. The application was created to measure the approximate values of temperature, humidity, and moisture sensors that were programmed into a microcontroller to manage the amount of water.

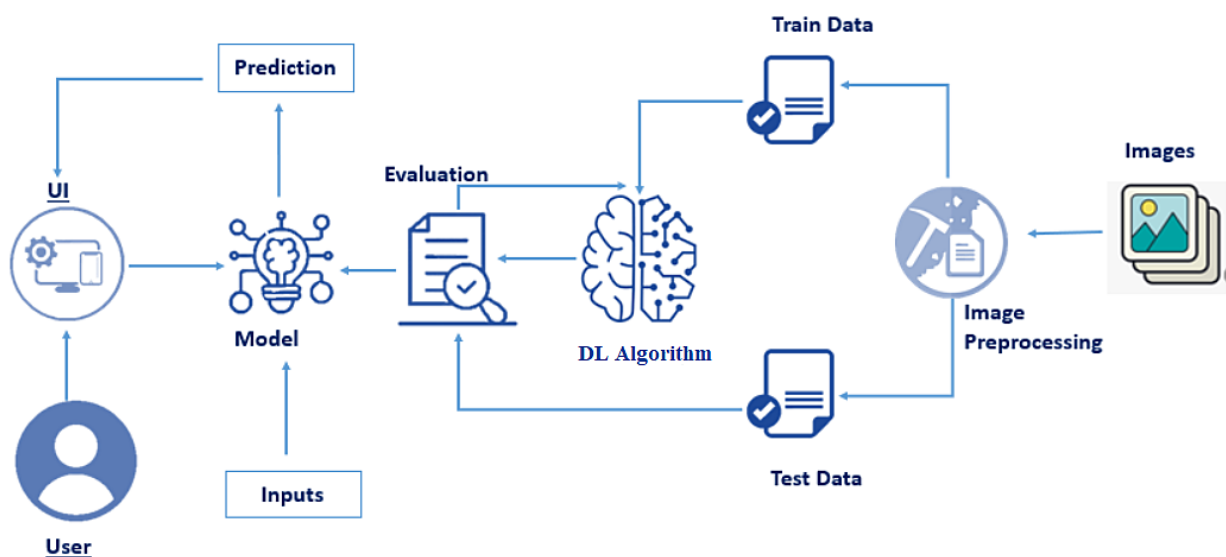
2.2 Proposed solution:

Web Application si built where :

1. Farmers interact with the portal build
2. Interacts with the user interface to upload images of diseased leaf
3. Our model built analyses the Disease and suggests the farmer with fertilizers are to be used.

3 THEORITICAL ANALYSIS

3.1 Block diagram :



3.2 Hardware / Software designing :

To complete this project you should have the following software and packages.

Softwares:

- Anaconda Navigator
- pycharm
- Visual studio code
- Jupiter notebook

- IBM watson studio

Packages:

- Tensor flow
- Keras
- Flask
- Numpy
- Pandas

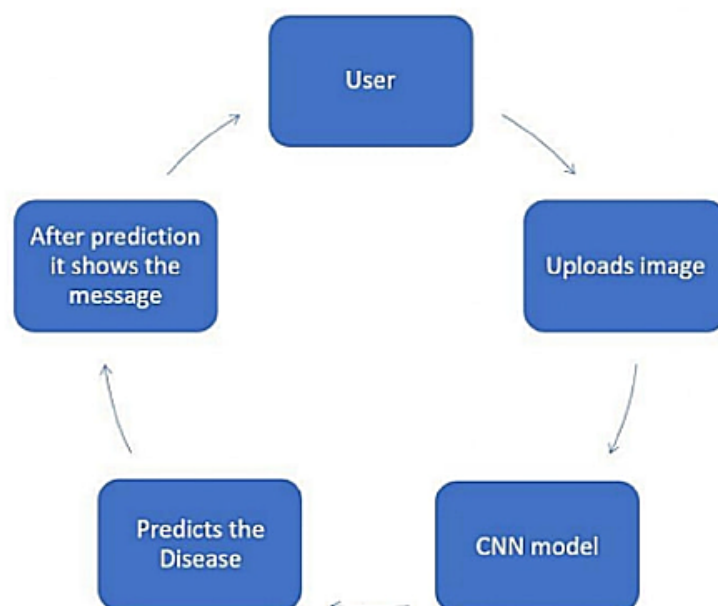
By using the above listed softwares and packages ,we build this application to take the input (image) from the farmer and detects whether the plant is infected or not. Here we use Deep learning techniques and give the out put to the user (Farmer).

4 EXPERIMENTAL INVESTIGATIONS :

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogenvarieties, changes in cultivation methods, and inadequate plant protection techniques.

An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant. Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

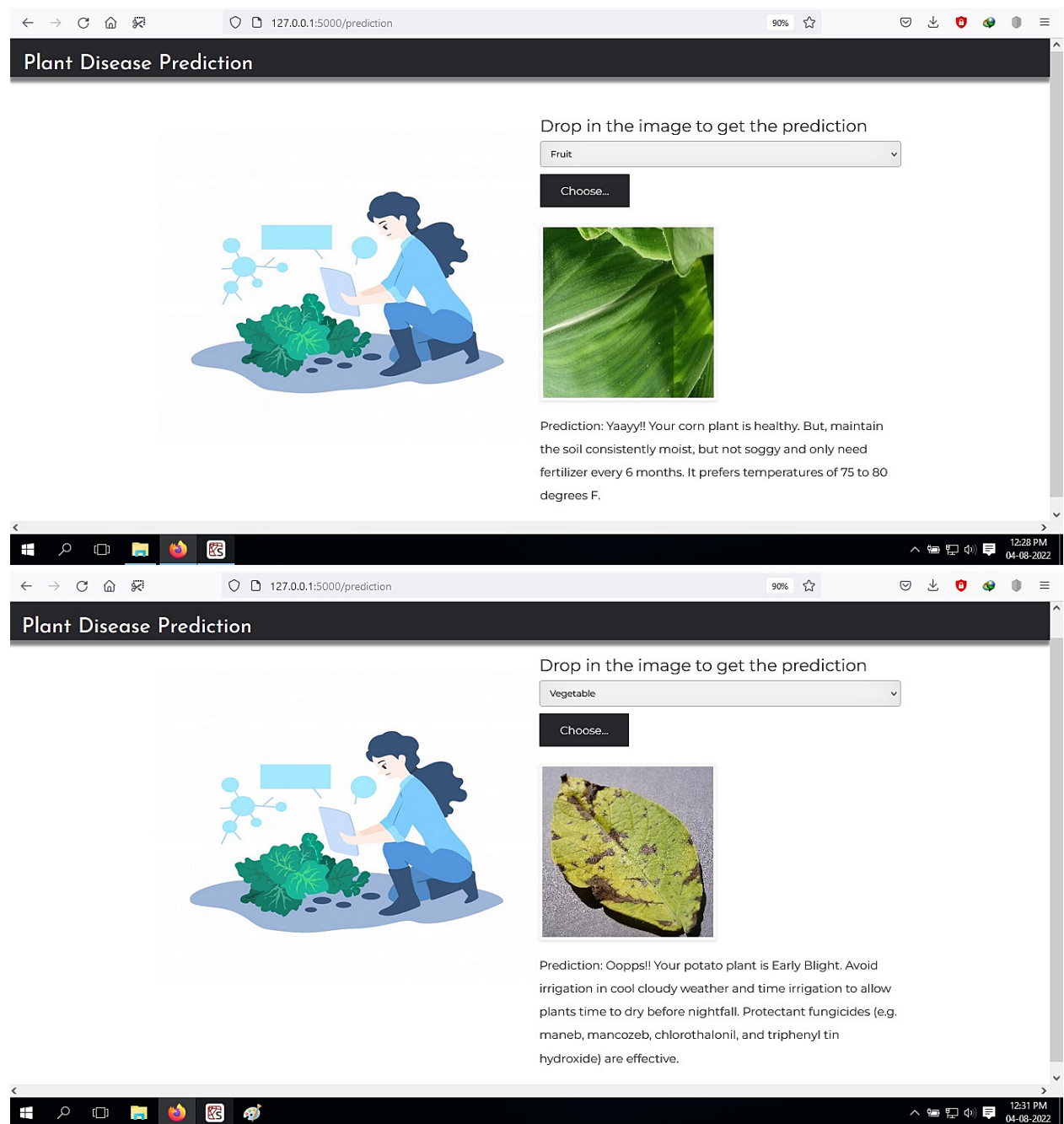
5 FLOWCHART



To accomplish the above task you must complete the below activities and tasks :

- Download the dataset.
- Classify the dataset into train and test sets.
- Add the neural network layers.
- Load the trained images and fit the model.
- Test the model.
- Save the model and its dependencies.
- Build a Web application using a flask that integrates with the model built.

6 RESULT



7 ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- The proposed model could predict the disease just from the image of a particular plant
- 2. Easy to use UI
- 3. Model has some good accuracy in detecting the plant just by taking the input as leaf

DISADVANTAGES:

- Prediction is limited to few plants as we haven't trained all the plants

8 APPLICATIONS

This web application can be used by farmers or users to check whether their plant is infected or not and can also show the remedy so that the user can take necessary precautions. These kind of web applications can be used in the agricultural sector as well as for small house hold plants.

9 CONCLUSION

Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality.

In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques. Usage of such applications could help the farmers to necessary precautions so that they dont face any loss as such.

10 FUTURE SCOPE

As of now we have just build the web application which apparently takes the input as an image and then predict the output. In the near future we can develop an application with computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf. This could make the project even more usable.

11 BIBILOGRAPHY

1. Usman Ahmed, Jerry Chun-Wei Lin, Gautam Srivastava, Youcef Djenouri, "A nutrient recommendation system for soil fertilization based on evolutionary computation", *Computers and Electronics in Agriculture*, vol. 189, 2021.
2. Guiling Sun, Xinglong Jia and Tianyu Geng, "Plant Disease Recognition Based on Image Processing Technology", *Journal of Electrical and Computer Engineering Volume*, 2018.

3. R Sujatha, Y Sravan Kumar and Garine Uma Akhil, "Leaf Disease Detection using Image Processing", *Journal of Chemical and Pharmaceutical Sciences*, vol. 10, no. 1, March 2017.
4. Rajneet Kaur and Manjeet Kaur, "A Brief Review on Plant Disease Detection using in Image Processing", *International Journal of Computer Science and Mobile Computing*, February 2017.
5. H. Al-Hiary, S. Bani-Ahmad, M. Reyalat, M. Braik and Z. ALRahamneh, "Fast and Accurate Detection and Classification of Plant Diseases", *International Journal of Computer Applications (0975 –8887)*, vol. 17, no. 1, March 2011.

APPENDIX

A. Source Code

Fruit Training

Image Preprocessing

```
In [1]: 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./255)

In [2]: x_train = train_datagen.flow_from_directory('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/train',target_size = (128,128))
x_test = test_datagen.flow_from_directory('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test',target_size = (128,128))

Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.

In [3]: x_train.class_indices

Out[3]: {'Apple__Black_rot': 0,
'Apple__healthy': 1,
'Corn_(maize)__Northern_Leaf_Blight': 2,
'Corn_(maize)__healthy': 3,
'Peach__Bacterial_spot': 4,
'Peach__healthy': 5}

In [4]: x_test.class_indices

Out[4]: {'Apple__Black_rot': 0,
'Apple__healthy': 1,
'Corn_(maize)__Northern_Leaf_Blight': 2,
'Corn_(maize)__healthy': 3,
'Peach__Bacterial_spot': 4,
'Peach__healthy': 5}
```

Importing Libraries

```
In [5]: from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

Initializing the model

```
In [6]: model=Sequential()
```

Adding CNN Layers

```
In [7]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
flatten (Flatten)	(None, 127008)	0
Total params: 896		
Trainable params: 896		
Non-trainable params: 0		

Adding Dense Layers

```
In [8]: model.add(Dense(40,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(6,activation='softmax'))
```

Compile the model

```
In [9]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Fit & Save the model

```
In [10]: model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
```

Epoch 1/10

C:\Users\SRT\AppData\Local\Temp\ipykernel_5992\1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

```
model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
673/673 [=====] - 102s 150ms/step - loss: 0.6724 - accuracy: 0.7981 - val_loss: 0.3418 - val_accuracy: 0.8826
```

Epoch 2/10

```
673/673 [=====] - 119s 177ms/step - loss: 0.2896 - accuracy: 0.9047 - val_loss: 0.3573 - val_accuracy: 0.8968
```

Epoch 3/10

```
673/673 [=====] - 93s 139ms/step - loss: 0.2227 - accuracy: 0.9246 - val_loss: 0.1992 - val_accuracy: 0.9383
```

Epoch 4/10

```
673/673 [=====] - 91s 135ms/step - loss: 0.1937 - accuracy: 0.9315 - val_loss: 0.2791 - val_accuracy: 0.9152
```

Epoch 5/10

```
673/673 [=====] - 103s 154ms/step - loss: 0.1696 - accuracy: 0.9456 - val_loss: 0.2423 - val_accuracy: 0.9217
```

Epoch 6/10

```
673/673 [=====] - 93s 138ms/step - loss: 0.1666 - accuracy: 0.9452 - val_loss: 0.1918 - val_accuracy: 0.9425
```

Epoch 7/10

```
673/673 [=====] - 101s 149ms/step - loss: 0.1907 - accuracy: 0.9335 - val_loss: 0.2243 - val_accuracy: 0.9353
```

Epoch 8/10

```
673/673 [=====] - 113s 168ms/step - loss: 0.1537 - accuracy: 0.9521 - val_loss: 0.1775 - val_accuracy: 0.9460
```

Epoch 9/10

```
673/673 [=====] - 102s 152ms/step - loss: 0.0958 - accuracy: 0.9662 - val_loss: 0.1119 - val_accuracy: 0.9650
```

Epoch 10/10

```
673/673 [=====] - 114s 169ms/step - loss: 0.1267 - accuracy: 0.9591 - val_loss: 0.1949 - val_accuracy: 0.9526
```

```
Out[10]: <keras.callbacks.History at 0x5ec20db8b0>
```

```
In [11]: model.save("fruit.h5")
```

Fruit Testing

```
In [9]: from tensorflow.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 [10]: model=load_model("fruit.h5")

In [12]: img=image.load_img('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test/Peach___Bacterial_spot/0bb77fcc-27ca-474f-beae-6b3519727cf9___Rutg._Ba

In [14]: x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)

In [22]: pred=np.argmax(model.predict(x),axis=1)
        1/1 [=====] - 0s 32ms/step

In [23]: pred
Out[23]: array([4], dtype=int64)

In [25]: index=['Apple_Black_rot','Apple_healthy','Corn_(maize)_Northern_Leaf_Blight','Corn_(maize)_healthy','Peach_Bacterial_spot','Peach_healthy']

In [27]: index[pred[0]]
Out[27]: 'Peach_Bacterial_spot'

In [28]: img=image.load_img('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/fruit-dataset/fruit-dataset/test/Apple___healthy/0c056dd8-2040-48cf-a9a8-53d8ae5662d0___RS_HL_7987.JPG')
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)
        pred=np.argmax(model.predict(x),axis=1)
        index[pred[0]]
        1/1 [=====] - 0s 31ms/step

Out[28]: 'Apple_healthy'
```

Vegetable Training

Image Preprocessing

```
In [1]: 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./255)

In [2]: x_train = train_datagen.flow_from_directory('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/Veg-dataset/Veg-dataset/train_set',target_size = (128,128),batch_size=8,class_mode='categorical')
        x_test = test_datagen.flow_from_directory('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/Veg-dataset/Veg-dataset/test_set',target_size = (128,128),batch_size=8,class_mode='categorical')
        Found 11386 images belonging to 9 classes.
        Found 3416 images belonging to 9 classes.

In [3]: x_train.class_indices
Out[3]: {'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}

In [4]: x_test.class_indices
Out[4]: {'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}
```

Importing Libraries

```
In [5]: from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Convolution2D
        from keras.layers import MaxPooling2D
        from keras.layers import Flatten
```

Initializing the model

```
In [6]: model=Sequential()
```

Adding CNN Layers

```
In [7]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(Flatten())
```


Adding Dense Layers

```
In [8]: model.add(Dense(200,activation='relu'))
model.add(Dense(100,activation='relu'))
model.add(Dense(50,activation='relu'))
model.add(Dense(9,activation='softmax'))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
flatten (Flatten)	(None, 127008)	0
dense (Dense)	(None, 200)	25401800
dense_1 (Dense)	(None, 100)	20100
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 9)	459
Total params: 25,428,305		
Trainable params: 25,428,305		
Non-trainable params: 0		

Compile the model

```
In [9]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Fit & Save the model

```
In [10]: model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
```

```
Epoch 1/10
C:\Users\SR\AppData\Local\Temp\ipykernel_2512\1582812018.py:1: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports generators.
  model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
1424/1424 [=====] - 441s 309ms/step - loss: 1.1204 - accuracy: 0.6246 - val_loss: 0.7601 - val_accuracy: 0.7550
Epoch 2/10
1424/1424 [=====] - 419s 294ms/step - loss: 0.6134 - accuracy: 0.7857 - val_loss: 0.4293 - val_accuracy: 0.8522
Epoch 3/10
1424/1424 [=====] - 395s 278ms/step - loss: 0.5117 - accuracy: 0.8220 - val_loss: 0.6385 - val_accuracy: 0.7971
Epoch 4/10
1424/1424 [=====] - 471s 331ms/step - loss: 0.4350 - accuracy: 0.8502 - val_loss: 0.4554 - val_accuracy: 0.8489
Epoch 5/10
1424/1424 [=====] - 409s 287ms/step - loss: 0.3891 - accuracy: 0.8682 - val_loss: 0.2624 - val_accuracy: 0.9095
Epoch 6/10
1424/1424 [=====] - 413s 290ms/step - loss: 0.3401 - accuracy: 0.8858 - val_loss: 0.3192 - val_accuracy: 0.8888
Epoch 7/10
1424/1424 [=====] - 387s 272ms/step - loss: 0.3229 - accuracy: 0.8905 - val_loss: 0.2865 - val_accuracy: 0.8975
Epoch 8/10
1424/1424 [=====] - 380s 267ms/step - loss: 0.2948 - accuracy: 0.9024 - val_loss: 0.5083 - val_accuracy: 0.8302
Epoch 9/10
1424/1424 [=====] - 387s 272ms/step - loss: 0.2707 - accuracy: 0.9076 - val_loss: 0.1508 - val_accuracy: 0.9491
Epoch 10/10
1424/1424 [=====] - 381s 267ms/step - loss: 0.2543 - accuracy: 0.9131 - val_loss: 0.1271 - val_accuracy: 0.9561
Out[10]: <keras.callbacks.History at 0x901f9b7fd0>
```

```
In [11]: model.save("vegetable.h5")
```

Vegetable Testing

```
In [1]: from tensorflow.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 [2]: model=load_model("vegetable.h5")
```

```
In [3]: img=image.load_img('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/Veg-dataset/Veg-dataset/test_set/Tomato___Late_blight/ce3f895d-e031-4ed9-ac40-7f6550894bff___GHLB_PS_Leaf.jpg')
x=img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x),axis=1)
index=['Pepper___bell___Bacterial_spot','Pepper___bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato___healthy','Tomato___Bacterial_spot','Tomato___Late_blight']
index[pred[0]]
```

```
1/1 [=====] - 0s 185ms/step
Out[3]: 'Tomato___Late_blight'
```

```
In [4]: img=image.load_img('C:/Users/SRT/Desktop/Project/Dataset Plant Disease/Veg-dataset/Veg-dataset/test_set/Potato___healthy/ff700844-68ad-4e99-8427-58a39c07f817___RS_HL_1860.JPG',target_size=(150,150))
x=img_to_array(img)
x=np.expand_dims(x,axis=0)
pred=np.argmax(model.predict(x),axis=1)
index[pred[0]]
```

```
1/1 [=====] - 0s 34ms/step
Out[4]: 'Potato___healthy'
```

Flask Integration

```
1 import requests
2 from tensorflow.keras.preprocessing import image
3 from tensorflow.keras.models import load_model
4 import numpy as np
5 import pandas as pd
6 import tensorflow as tf
7 from flask import Flask, request, render_template, redirect, url_for
8 import os
9 from werkzeug.utils import secure_filename
10 from tensorflow.python.keras.backend import set_session
11 app=Flask(__name__)
12 model=load_model("C:/Users/SRT/Desktop/Project/vegetable.h5")
13 model1=load_model("C:/Users/SRT/Desktop/Project/fruit.h5")
14 @app.route('/')
15 def home():
16     return render_template('home.html')
17
18 @app.route('/prediction')
19 def prediction():
20     return render_template('predict.html')
21
22 @app.route('/predict',methods=['POST'])
23
24 def predict():
25     if request.method == 'POST':
26
27         f = request.files['image']
28
29
30         basepath = os.path.dirname(__file__)
31         file_path = os.path.join(
32             basepath, 'Dataset Plant Disease', secure_filename(f.filename))
33         f.save(file_path)
34         img = image.load_img(file_path, target_size=(128, 128))
35         |
36         x = image.img_to_array(img)
37         x = np.expand_dims(x, axis=0)
38
39         plant=request.form['plant']
40         print(plant)
41
42         if(plant=="vegetable"):
43             preds = model.predict(x)
44             preds = np.argmax(preds)
45             print(preds)
46             df=pd.read_excel('precautions - veg.xlsx')
47             print(df.iloc[preds]['caution'])
48         else:
49             preds = model1.predict(x)
50             preds = np.argmax(preds)
51
52             df=pd.read_excel('precautions - fruits.xlsx')
53             print(df.iloc[preds]['caution'])
54
55
56         return df.iloc[preds]['caution']
57
58
59
60
61
62 if __name__ == "__main__":
63     app.run(debug=False)
```

IBM Training

```
In [1]: pwd

Out[1]: '/home/wsuser/work'
```

Image Preprocessing

```
In [21]: 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./255)

In [22]: import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

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 those credentials before you share the notebook.
client_0ef70f313fde4b659f04ac75d99b4f3c = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='LeEierGXhXUJFNO77KPXvpNUB8Et35CuQbse0gsct-D_',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth')),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

streaming_body_1 = client_0ef70f313fde4b659f04ac75d99b4f3c.get_object(Bucket='fertilizersrecommendationsystemfo-donotdelete-pr-dfzzgm9hsg1qkl', Key='Dataset Plant Disease.zip')['B

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

```
In [23]: from io import BytesIO
import zipfile
unzip=zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths=unzip.namelist()
for path in file_paths:
    unzip.extract(path)

In [24]: ls

'Dataset Plant Disease/'   fertilizer-recommendation.tgz

In [25]: x_train = train_datagen.flow_from_directory('/home/wsuser/work/Dataset Plant Disease/fruit-dataset/fruit-dataset/train',target_size = (128,128),batch_size=8,class_mode = 'categorical')
x_test = test_datagen.flow_from_directory('/home/wsuser/work/Dataset Plant Disease/fruit-dataset/fruit-dataset/test',target_size = (128,128),batch_size=8,class_mode = 'categorical')

Found 5384 images belonging to 6 classes.
Found 1686 images belonging to 6 classes.

In [26]: x_train.class_indices

Out[26]: {'Apple__Black_rot': 0,
'Apple__healthy': 1,
'Corn_(maize)__Northern_Leaf_Blight': 2,
'Corn_(maize)__healthy': 3,
'Peach__Bacterial_spot': 4,
'Peach__healthy': 5}

In [27]: x_test.class_indices

Out[27]: {'Apple__Black_rot': 0,
'Apple__healthy': 1,
'Corn_(maize)__Northern_Leaf_Blight': 2,
'Corn_(maize)__healthy': 3,
'Peach__Bacterial_spot': 4,
'Peach__healthy': 5}
```

Importing Libraries

```
In [28]: from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

Initializing the model

```
In [29]: model=Sequential()
```

Adding CNN Layers

```
In [30]: model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
flatten (Flatten)	(None, 127008)	0

=====

Total params: 896

Trainable params: 896

Non-trainable params: 0

=====

Adding Dense Layers

```
In [31]: model.add(Dense(40,activation='relu'))
model.add(Dense(20,activation='relu'))
model.add(Dense(6,activation='softmax'))
```

Compile the model

```
In [32]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Fit & Save the model

```
In [33]: model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)

/tmp/wsuser/ipykernel_164/1582812018.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(x_train,steps_per_epoch=len(x_train),validation_data=x_test,validation_steps=len(x_test),epochs=10)
Epoch 1/10
673/673 [=====] - 63s 93ms/step - loss: 0.6957 - accuracy: 0.7602 - val_loss: 0.2967 - val_accuracy: 0.8980
Epoch 2/10
673/673 [=====] - 63s 93ms/step - loss: 0.3288 - accuracy: 0.8884 - val_loss: 0.2402 - val_accuracy: 0.9098
Epoch 3/10
673/673 [=====] - 62s 93ms/step - loss: 0.2788 - accuracy: 0.9032 - val_loss: 0.2982 - val_accuracy: 0.8873
Epoch 4/10
673/673 [=====] - 61s 91ms/step - loss: 0.2336 - accuracy: 0.9242 - val_loss: 0.1606 - val_accuracy: 0.9448
Epoch 5/10
673/673 [=====] - 62s 92ms/step - loss: 0.2184 - accuracy: 0.9246 - val_loss: 0.1968 - val_accuracy: 0.9342
Epoch 6/10
673/673 [=====] - 61s 91ms/step - loss: 0.1876 - accuracy: 0.9342 - val_loss: 0.1621 - val_accuracy: 0.9395
Epoch 7/10
673/673 [=====] - 62s 91ms/step - loss: 0.1726 - accuracy: 0.9424 - val_loss: 0.2014 - val_accuracy: 0.9407
Epoch 8/10
673/673 [=====] - 62s 92ms/step - loss: 0.1507 - accuracy: 0.9510 - val_loss: 0.2241 - val_accuracy: 0.9324
Epoch 9/10
673/673 [=====] - 60s 89ms/step - loss: 0.1504 - accuracy: 0.9504 - val_loss: 0.1567 - val_accuracy: 0.9567
Epoch 10/10
673/673 [=====] - 60s 89ms/step - loss: 0.1182 - accuracy: 0.9593 - val_loss: 0.3144 - val_accuracy: 0.9247
Out[33]: <keras.callbacks.History at 0x7f3815c2b310>
```

```
In [34]: model.save("fruit.h5")
```

```
In [35]: ls
```

```
'Dataset Plant Disease'/  fertilizer-recommendation.tgz  fruit.h5
```

IBM Deployment

```
In [36]: !pip install watson-machine-learning-client

Requirement already satisfied: watson-machine-learning-client in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.391)
Requirement already satisfied: boto3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.18.21)
Requirement already satisfied: ibm-cos-sdk in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.11.0)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.26.7)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.3.3)
Requirement already satisfied: pandas in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (1.3.4)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2.26.0)
Requirement already satisfied: tqdm in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (4.62.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (2022.6.15)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from watson-machine-learning-client) (0.8.9)
Requirement already satisfied: botocore<1.22.0,>=1.21.21 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (1.21.41)
Requirement already satisfied: s3transfer<0.6.0,>=0.5.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.5.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from boto3->watson-machine-learning-client) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=1.21.21->boto3->watson-machine-learning-client) (1.15.0)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk->watson-machine-learning-client) (2.11.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (3.3)
Requirement already satisfied: charset-normalizer==2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->watson-machine-learning-client) (2.0.4)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas->watson-machine-learning-client) (1.20.3)
```

```
In [37]: from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"iREwpeSu_L4djBA1HwpwIBXYRxpYpgKakkFqTqSiHEXR"
}
```

```
In [38]: client=APIClient(wml_credentials)
```

```
In [39]: def guid_space_name(client,Fertilizer_deploy):
        space=client.spaces.get_details()
        return(next(item for item in space['resources'] if item['entity']['name']==Fertilizer_deploy))['metadata']['id']
```

```
In [40]: space_uid=guid_space_name(client,'Fertilizer_deploy')
print(space_uid)

72692168-1a03-4787-8b5b-8dc723713486
```

```
In [41]: client.set.default_space(space_uid)
```

```
Out[41]: 'SUCCESS'
```

```
In [51]: client.software_specifications.list(100)
```

```
In [52]: software_space_uid=client.software_specifications.get_uid_by_name('tensorflow_rt22.1-py3.9')
software_space_uid
```

```
Out[52]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'
```

```
In [44]: ls
'Dataset Plant Disease'/  fertilizer-recommendation.tgz  fruit.h5
```

```
In [47]: !tar -zcvf fertilizer-recommendation.tgz fruit.h5
fruit.h5
```

```
In [48]: ls
'Dataset Plant Disease'/  fertilizer-recommendation.tgz  fruit.h5
```

```
In [53]: model_details=client.repository.store_model(model='fertilizer-recommendation.tgz',
    meta_props={
        client.repository.ModelMetaNames.NAME:"CNN",
        client.repository.ModelMetaNames.TYPE:'tensorflow_2.7',
        client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_space_uid
    })
```

```
In [54]: model_id=client.repository.get_model_id(model_details)
```

```
In [55]: model_id
```

```
Out[55]: 'b8a5ddd2-c83f-4b7c-8f3b-4c159276ba1f'
```

```
In [56]: client.repository.download(model_id,'fertilizer.tar.gb')
Successfully saved model content to file: 'fertilizer.tar.gb'
Out[56]: '/home/wsuser/work/fertilizer.tar.gb'
```

```
In [57]: ls
'Dataset Plant Disease'/  fertilizer.tar.gb
fertilizer-recommendation.tgz  fruit.h5
```