Title:Plant disease detection using machine learning

INDEX

ABSTRACT :

Leaf diseases are a noteworthy risk to sustenance security, however their quick distinguishing proof stays troublesome in numerous parts of the world because of the non-attendance of the important foundation. Emergence of accurate techniques in the field of leaf-based image classification has shown impressive results. This product makes use of Random Forest in identifying between healthy and diseased leaf from the data sets created. Our proposed paper includes various phases of implementation namely dataset creation, feature extraction, training the classifier and classification. The created datasets of diseased and healthy leaves are collectively trained under Random Forest to classify the diseased and healthy images. For extracting features of an image, we use Histogram of an Oriented Gradient (HOG). Overall, using machine learning to train the large data sets available publicly gives us a clear way to detect the disease present in plants in a colossal scale.

INTRODUCTION :

Plants are sensitive to diseases especially the plant leaves as symptoms of the disease appear first on the leaves. Due to the bad impacts of plant diseases on the both the economy and the environment, the farmers should consider monitoring the crops in such a way that they may mitigate losses.The core purpose of this proposed system is not aiming only at detecting the plant diseases using the image processing technology, but it aims also at directing the user 12 (farmer) to use a mobile application in which he will upload the image and receive the type of disease infection along with a suggestion of needed pesticides. The digitalization of the agriculture field has known the intervention of the latest technologies namely the image processing. As a result, our system that is designed to be automated system is implemented using image processing technique using MATLAB. Farmers opt for monoculture due to its benefits that will be discussed in the coming sections.

EXISTING SYSTEM :

Leaf disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. They used SVM classifier which has MCS, used for detecting disease in wheat plant offline.

DISADVANTAGES :

- Required long training time.

PROPOSED SYSTEM :

By seeing the results, we can say that our proposal model SVM can predict leaf disease from plant. By adapting the method automatic location of disease can be more reliable by reducing the manual work. So that farmers can save their plants before losing.

To find out whether the leaf is diseased or healthy, certain steps must be followed. i.e., Preprocessing, Feature extraction, Training of classifier and Classification. Preprocessing of image, is bringing all the images size to a reduced uniform size.

ADVANTAGES :

- Manual work is reduced.
- Image processing helps in to get better results.
- Computationally efficient works well for images having good contrast.

MODULES :

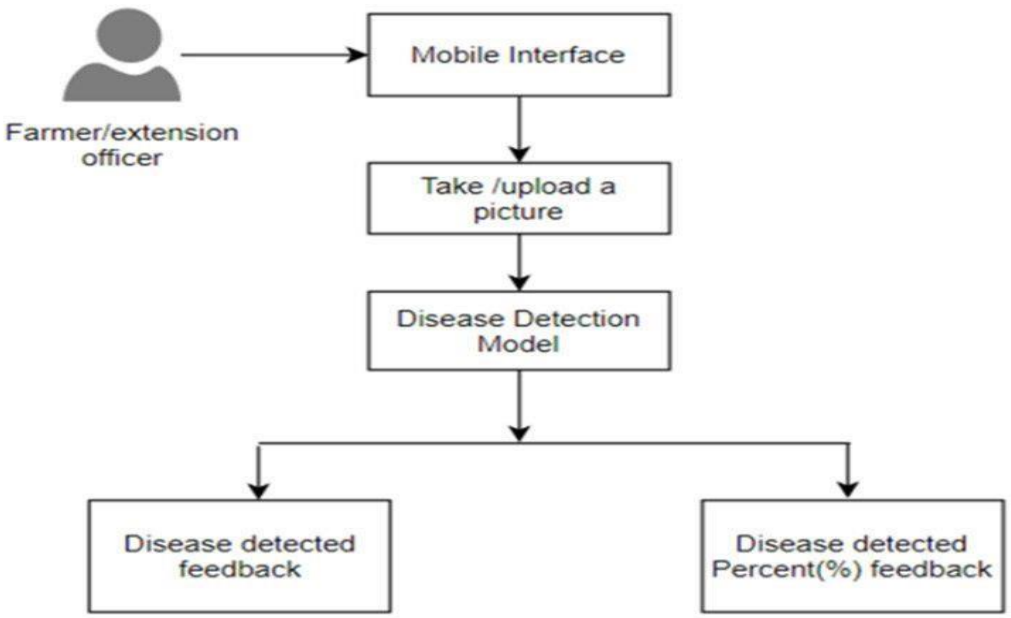
1. Preprocessing:

- Resize the images to a consistent input size suitable for the model.
- Normalize the pixel values to a common range to ensure consistent input scaling

2. Training:

- Train the CNN model using the training set, optimizing the model's parameters (Weights and biases) to minimize a suitable loss function (e.g., cross-entropy loss).

DIAGRAM :



IMPLEMENTATION

Libraries :

The libraries used for implementation are:

1.OpenCV

2.Django

3.Cmake

4.Asgiref

5.Pillow

6.Imutils

7.numpy

8.tkinter

9.Time

10.Logging

11.Datetime

SOURCE CODE

```
#Import necessary libraries from flask import Flask,
render_template, request import numpy as np import os from
tensorflow.keras.preprocessing.image import load_img from
tensorflow.keras.preprocessing.image import img_to_array from
tensorflow.keras.models import load_model

filepath =
'C:/Users/Madhuri/AppData/Local/Programs/Python/Python38/Tomato_Leaf_Diseas
e_Prediction/model.h5' model = load_model(filepath)
print(model)

print("Model Loaded Successfully")

def pred_tomato_dieas(tomato_plant):   test_image = load_img(tomato_plant,
target_size = (128, 128)) # load image    print("@@ Got Image for prediction")

  test_image = img_to_array(test_image)/255 # convert image to np array and normalize
  test_image = np.expand_dims(test_image, axis = 0) # change dimention 3D to 4D

result = model.predict(test_image) # predict diseased palnt or not
print('@@ Raw result = ', result)     pred = np.argmax(result,
axis=1)


print(pred)
if pred==0:
```

```
    return "Tomato - Bacteria Spot Disease", 'Tomato-Bacteria Spot.html'    elif
pred==1:
    return "Tomato - Early Blight Disease", 'Tomato-Early_Blight.html'   elif
pred==2:
    return "Tomato - Healthy and Fresh", 'Tomato-Healthy.html'   elif
pred==3:
    return "Tomato - Late Blight Disease", 'Tomato - Late_blight.html'   elif
pred==4:
    return "Tomato - Leaf Mold Disease", 'Tomato - Leaf_Mold.html'   elif
pred==5:
    return "Tomato - Septoria Leaf Spot Disease", 'Tomato - Septoria_leaf_spot.html'   elif
pred==6:
    return "Tomato - Target Spot Disease", 'Tomato - Target_Spot.html'   elif
pred==7:
 return "Tomato - Tomoato Yellow Leaf Curl Virus Disease", 'Tomato -
Tomato_Yellow_Leaf_Curl_Virus.html'   elif pred==8:
return "Tomato - Tomato Mosaic Virus Disease", 'Tomato -
Tomato_mosaic_virus.html          elif pred==9:
    return "Tomato - Two Spotted Spider Mite Disease", 'Tomato -
Twospotted_spider_mite.html'

# Create flask instance
app = Flask(_name_) #
render index.html page
@app.route("/", methods=['GET', 'POST']) def
home():
 return render_template('index.html')
# get input image from client then predict class and render respective .html page for solution
@app.route("/predict", methods = ['GET','POST'])
```

```
def predict():          if request.method ==
'POST':         file = request.files['image'] #
fet input               filename = file.filename
print("@@ Input posted = ", filename)

 file_path =
os.path.join('C:/Users/Madhuri/AppData/Local/Programs/Python/Python38/Tomato_
Leaf_Disease_Prediction/static/upload/', filename)        file.save(file_path)

      print("@@ Predicting class.....")              pred, output_page =
pred_tomato_dieas(tomato_plant=file_path)

 return render_template(output_page, pred_output = pred, user_image =  file_path)
# For local system & cloud

if _name_ == "_main_":
   app.run(threaded=False,port=8080)
```

Fig.6(A) Normal leaf

Fig.6(B) Diseased leaf 1

Fig.6(C) Diseased leaf 2

TESTING :

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying

all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

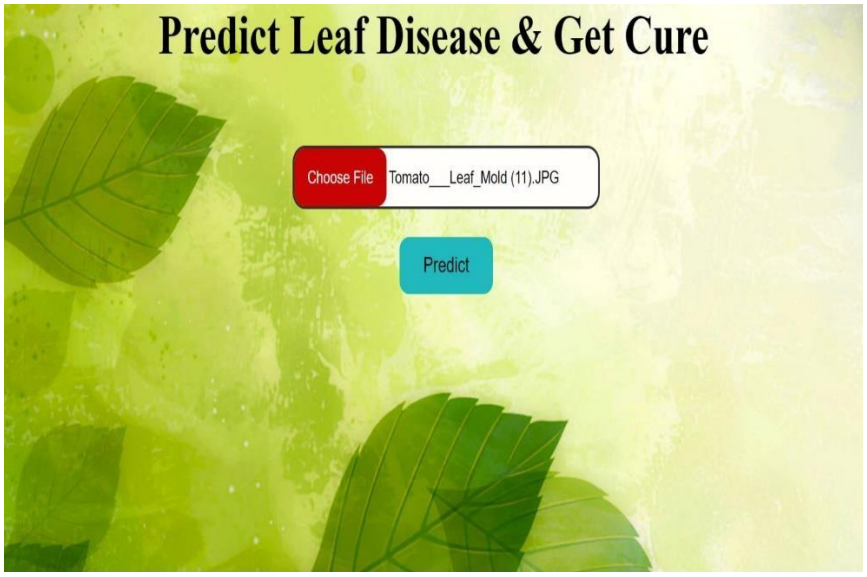
BLACK BOX TESTING :

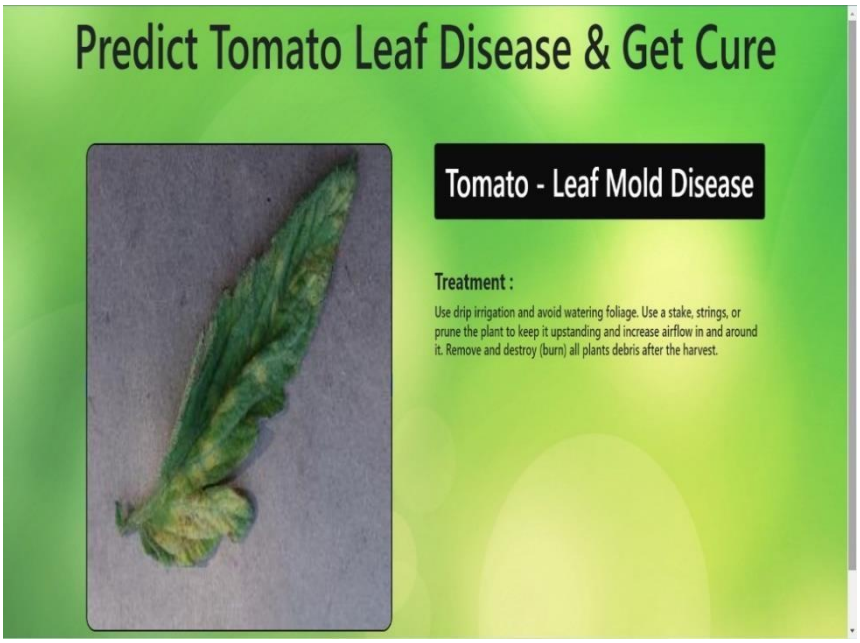
Black box testing is a software testing technique in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

WHITE BOX TESTING :

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the 25 applications, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing. It is one of two parts of the "box testing" approach of software testing. Its counter-part, black box testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "white box" was used because of the see-through box concept. The clear box or whitepox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings.

RESULTS :


Predict Leaf Disease & Get Cure
Choose File
Tomato___Leaf_Mold (11).JPG
Predict


Predict Tomato Leaf Disease & Get Cure
Tomato - Leaf Mold Disease
Treatment :
Use drip irrigation and avoid watering foliage. Use a stake, strings, or prune the plant to keep it upstanding and increase airflow in and around it. Remove and destroy (burn) all plants debris after the harvest.

CONCLUSION :

We concentrated on how an image from a particular dataset (a trained dataset) was used in the field and historical data sets to forecast the pattern of plant diseases. The following information regarding predicting plant leaf disease is provided by this. This method will cover the greatest variety of plant leaves, allowing farmers to learn about leaves that may have never been cultivated. By listing all potential plant leaves, it aids farmers in choosing which crop to grow. Additionally, this technology takes historical data production into account, giving the farmer information into market prices and demand for specific plants. Compared to AlexNet and GoogleNet LeNet got the best results in the experiment.

The objective of this algorithm is to recognize abnormalities that occur on plants in their greenhouses or natural environment. The image captured is usually taken with a plain background to eliminate occlusion. The algorithm was contrasted with other machine learning models for accuracy. Using Random forest classifier, the model was trained using 160 images of papaya leaves. The model could classify with approximate 70 percent accuracy. The accuracy can be increased when trained with vast number of images and by using other local features together with the global features.

FUTURE SCOPE :

The proposed model uses computer vision techniques including RGB conversion to gray, HE, Kmeans clustering, contour tracing is employed in preprocessing stage. The multiple descriptors Discrete Wavelet Transform, Principal Component Analysis and GLCM are used to extract the informative features of the leaf samples. The machine learning approaches such as SVM, K-NN and CNN are used to distinguish diseased or non-diseased leaf. The analysis of the proposed model is well suited for CNN machine learning classification technique with a desired accuracy compared to other state of the art method. In future, the model can be improved using fusion techniques for extraction of significant features and examined for other leaf samples of datasets.