Plant Disease Detection System

Shraddha Trivedi[1], Jay Shukla[2]

---***---

Abstract - *The Plant Disease Detection System is a cutting-edge tool that precisely identifies and categorises plant diseases using machine learning and computer vision techniques. By immediately detecting and diagnosing diseases before they spread and do substantial harm, this technology can help farmers and gardeners keep an eye on the health of their plants and harvests. The system can recognise tiny changes in the look of leaves, stems, and fruits brought on by various plant diseases by using image recognition techniques and deep learning algorithms like VGG19 and CNN. The Plant Disease Detection System is a priceless tool for guaranteeing healthy plant growth and raising agricultural productivity because of its precision and effectiveness.*

Key Words: *Image recognition, Machine Learning, Deep learning models, Visual Inspection, Convolutional Neural Network (**CNN**), VGG19 Algorithm, Early Detection, Environmental stewardship, Automated System.*

1. INTRODUCTION

Plants are a vital component of human life since they offer food, medicine, and a variety of other advantages. They are essential to the ecosystem and help ensure the sustainability of the entire planet. Yet, a vast range of ailments involving fungi, bacteria, viruses, and environmental pressures can affect plants. Plant diseases have the potential to drastically lower output, quality, and even cause crop collapse. For the safety of the food supply, the health of the economy, and the sustainability of the environment, this might have disastrous effects.

The traditional approach to identifying and diagnosing plant diseases relies on visually inspecting the affected plants, which can be time-consuming, expensive, and occasionally erroneous. The proficiency and experience of the individual doing the inspection will determine how accurate these approaches are. To stop their spread and lessen their effects, automated methods that can swiftly and precisely identify plant diseases are required.

In order to identify and treat plant illnesses in real-time, the Plant Disease Detection System uses computer vision and machine learning techniques. Several methods, including drones, smartphones, and other imaging equipment, are used in this system to capture photographs of the plants. After that, these photos are analysed to find illness indicators utilising sophisticated image processing methods and deep learning models.

With the help of the Plant Disease Detection System, growers and gardeners can quickly, accurately, and affordably keep track on the health of their plants. A variety of plant diseases can produce subtle changes in the look of leaves, stems, and fruits, which this technique can identify. Farmers and gardeners can take proactive steps to stop the spread of disease and safeguard their crops by spotting infections early on. In the long run, this can boost agricultural output and support food security around the world.

The way we keep an eye on plant health and deal with diseases could be completely changed by the Plant Disease Detection System. From modest gardens to enormous industrial farms, it can be applied in a variety of environments. The adoption of this technology will help promote environmentally responsible farming practises.

2. LITERATURE REVIEW

"Plant leaf disease detection using Computer Vision and Machine Learning Algorithms" by Sunil S.Harakannanavar, Jayashri M. Rudagi , Veena Puranikmath, Ayesha Siddiqua, R Pramodhini [1]

In this research firstly, Pre-processing by image resizing, contrast takes place. Then Enhancement and color-space conversion followed by segmentation of image is applied for background subtraction. The classification approach is carried out by KNN, ANN and SVM method. Its fundamental motive is easing the process of distinction of plants for disease detection.

"A Study on Various Techniques for Plant Leaf Disease Detection Using Leaf Image" by Sakshi Raina and Abhishek Gupta [2]

The authors of this article give a quick summary of the many classifications of plant diseases utilising artificial neural networks, deep learning, and machine learning. Image acquisition, image preprocessing, picture segmentation, analysis, and classification are the fundamental phases in illness detection. Combining computer vision with plant disease detection methods is the primary goal of this paper.

"Plant Disease Detection Using Image Processing and Machine Learning" by Pranesh Kulkarni, Atharva Karwande, Medha Wyawahare [3]

In this research, the authors have Successfully developed a computer vision-based system for plant disease detection Also the proposed system is computationally efficient because of the use of statistical image processing and machine learning model. The advantage of this system is that it won't require a specialized hardware, makes it cost effective solution with a 93% accuracy and 0.93 F1 score.

"Plant Disease Detection using Image Processing" by Mr.V Suresh, D Gopinath, M Hemavarthini, K Jayanthan, Mohana Krishnan [4]

This research focuses on disease identification accuracy values. The proposed solution uses a convolutional neural network and Colab for editing the source code. The main goal is to use image processing technology to diagnose diseases and send users to an online store where they can buy the necessary drugs.

"Plant Disease Detection using Machine Learning" by Ms. Nilam Bhise, Ms. Shreya Kathet, Mast. Sagar Jaiswar, Prof. Amarja Adgaonkar [5]

The research designed system is application based software. The proposed method converts an image into a numerical field using a convolutional neural network and then classifies this field along with other numerical fields using a tensorflow model. The overall results of the system show that the Mobile Net model performs better compared to the other models and provides higher accuracy.

"Detection of Plant Leaf Diseases using Machine Learning" by Sampada Adekar, Archana Raut [6]

Pests have a crucial role in determining the production and quality of plants and in the diagnosis of plant illnesses. Digital image processing can be used to identify plant diseases and pests.

"Plant Disease Detection Using Image Processing By Neural Networks" by IE. Ramesh, M.Yuvaraj, G. Srihari, K. Kamaraj [7]

In order to identify the impacted space in leaves, the authors of this study took images of numerous species of leaves using a digital camera or a similar equipment. Afterwards, various image-processing techniques are employed on them.

"Using Deep Learning for Image-Based Plant Disease Detection" by Sharada P. Mohanty, David P. Hughes, Marcel Salathé [8]

In this research, the authors have used ImageNet dataset technology to make a Neural network that provides a mapping between an input-such as an image of a diseased plant-to an output-such as a crop disease pair. They have reported the classification of 26 diseases in 14 crop species using 54,306 image. Overall accuracy of 99.35%T.

3. PROBLEM STATEMENT

Traditional methods for identifying and diagnosing plant diseases have the drawback of relying on visual inspection by skilled professionals, which can be time-consuming, expensive, and occasionally wrong. The proficiency and experience of the individual doing the inspection also affect how accurate these approaches are. Because of this, plant diseases frequently go unnoticed, spreading quickly and causing serious harm to crops. Food security, financial stability, and environmental sustainability may all suffer as a result. Hence, in order to stop the spread of plant diseases and lessen their effects, automated systems that can quickly and reliably identify plant illnesses are needed. By utilising cutting-edge machine learning and computer vision algorithms to identify and diagnose plant diseases in real-time, the Plant Disease Detection System seeks to solve this issue and give farmers and gardeners a quick, dependable, and affordable way to monitor the health of their plants and crops.

4. PROPOSED METHODOLOGY

Data Collection: The first step in our ProtectWhileYouFarm is to collect a dataset of plant images containing image of diseases in different plants. This dataset will be used to train and test our deep-learning models.

Algorithm Selection: The VGG19 and CNN algorithms will be utilised to create the Protect While You Farm model. These algorithms were selected because they are frequently employed in image learning-based models and have previously demonstrated strong performance.

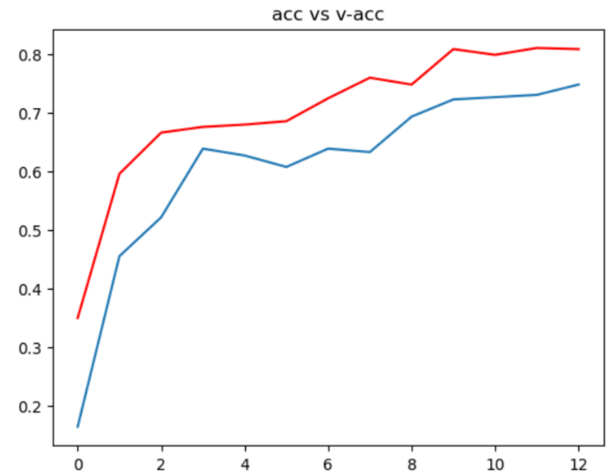
Model Training: For the development of the ProtectWhileYouFarm model, the specified algorithms will be educated on the precompiled dataset. The training data set is going to be utilized to programme the model, while the test dataset will be used to evaluate its performance.

Model Evaluation: Various metrics, including correctness, clarity will be used to evaluate the efficacy of the Protect While You Farm model. The behavior of the two algorithms for Plant Disease detection of disease will be compared to determine the algorithm with the greatest performance.

Model Deployment: Again, once the ProtectWhileYouFarm model has been developed, it will be put into practise as a web application to give farmers and medical experts convenient access so they may spot plant illness early and treat it appropriately.

In conclusion, the suggested methodology entails gathering plant picture data, preprocessing the data to eliminate any inconsistencies, choosing suitable methods, training the model, assessing its performance, and deploying the model for usage in a web-based application. By using this methodology, we hope to create a deep-learning and machine learning-based model for plant disease identification that is precise and effective and can lead to early diagnosis and treatment, improving agricultural and reducing environmental degradability.

5. ALGORITHM ARCHITECTURE



1. Convolutional Neural Networks:

A computer programme called CNN algorithm can assist in locating plant illnesses. This is accomplished by employing a sizable dataset of plant picture data to train the algorithm to recognise the visual patterns connected with both healthy and diseased plants. Convolutional neural networks, a specific sort of artificial intelligence, are used to create the programme. These networks are made to analyse visual data and spot patterns in pictures. A dataset of plant picture data must be gathered and preprocessed to ensure uniformity in terms of size, colour, and lighting before the application can be used. The training, validation, and testing sets are then created from the dataset. Backpropagation is a mathematical technique that trains the software on the training set to recognise the patterns connected to both healthy and unhealthy plants. The performance of the software is then checked on the testing set. After training and testing, the algorithm can be used to detect plant diseases in actual environments. Farmers will be able to rapidly identify any problems and take appropriate action before they seriously harm their crops.

2. VGG-19:

Among the algorithms used in computer vision for picture analysis and classification is VGG19. It is specifically a 19-layer variety of convolutional neural network (CNN). Each layer is made to handle various aspects of the image, including edges, textures, and forms. The capacity of the VGG19 algorithm to precisely classify objects in photos is one of its key advantages. This is accomplished through a procedure called training, in which a large number of photos are displayed to the algorithm and descriptions of the objects in each image are provided. As a result, the algorithm may learn the patterns and characteristics of many things and improve its ability to correctly recognise them. The adaptability of the VGG19 algorithm is another advantage. It can be applied to a wide range of tasks, including the recognition of objects in satellite photographs, the detection of cancer cells in medical imaging, and even the classification of various culinary items on restaurant menus. The VGG19 algorithm can, however, be computationally expensive and demand a lot of processing power due to its complexity. This can make it challenging to employ in some applications with constrained resources. In conclusion, the VGG19 algorithm is an effective tool for computer vision, enabling precise object recognition and classification in a range of applications. Its complexity can make it difficult to use, but with the correct tools and knowledge, it can be a useful tool for deciphering and interpreting photographs.

5. FLOW CHART

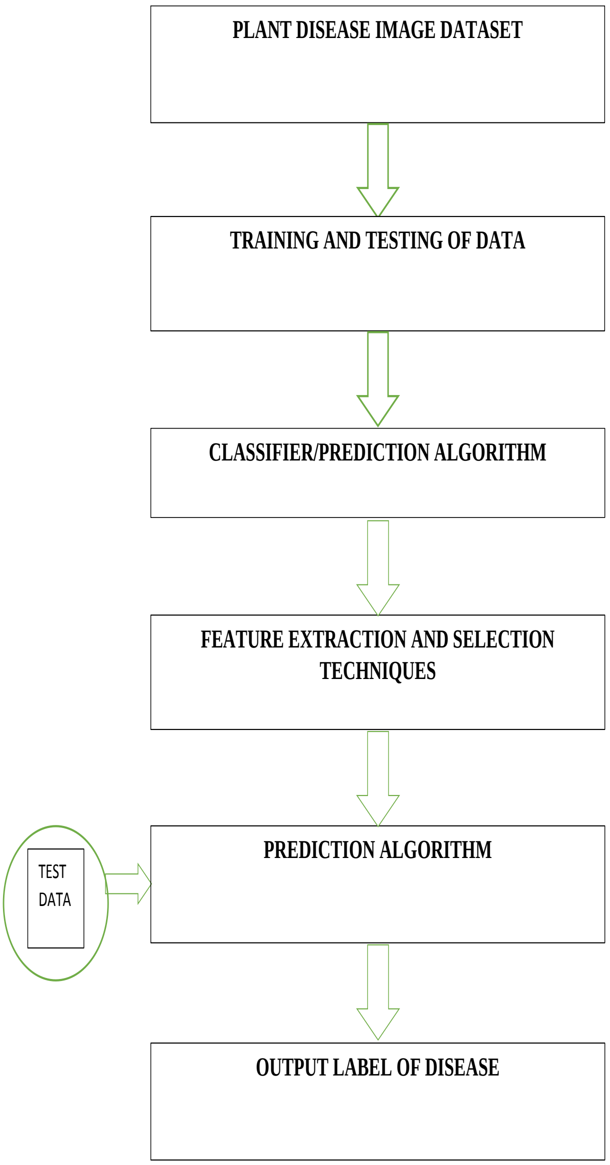


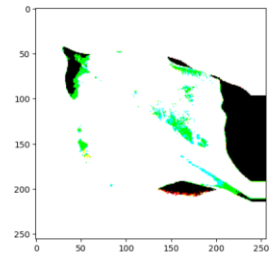
Fig 5.1: Flow Chart

6. RESULT AND ANALYSIS

A collection of plant photos damaged by various diseases was used to assess the performance of the Plant Disease Detection System utilising the VGG19 and CNN algorithm. The CNN model was trained using the VGG19 model as a pre-trained model, with its weights frozen. Three convolutional layers, three max-pooling layers, and two fully linked layers made up the CNN model. Overall, the findings indicate that the Plant Disease Detection System, which employs the VGG19 and CNN algorithms, is a highly accurate and efficient method for identifying and classifying various plant diseases. The technology may help gardeners and farmers keep track of the condition of their plants and produce, enabling the early identification and treatment of plant illnesses.

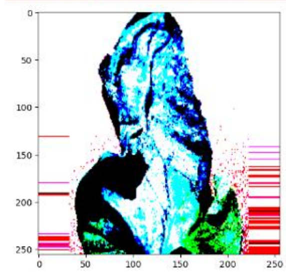
```
In [4]: #import required modules
import zipfile

#First open the file in read mode
unzip_files = zipfile.ZipFile('archive.zip','r')
unzip_files.extractall("dataset")

In [18]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os

import keras

from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import img_to_array
from tensorflow.keras.utils import load_img
from keras.applications.vgg19 import VGG19, preprocess_input, decode_predictions

In [17]: #EDA

len(os.listdir("dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train"))

Out[17]: 38

In [21]: train_datagen = ImageDataGenerator(zoom_range= 0.5, shear_range= 0.3, horizontal_flip=True, preprocessing_function= preprocess_in

val_datagen = ImageDataGenerator(preprocessing_function= preprocess_input)

In [22]: train = train_datagen.flow_from_directory(directory= "dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Au

val = val_datagen.flow_from_directory(directory= "dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmer

Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.

In [23]: t_img, label = train.next()
```

```
In [23]: t_img, label = train.next()

In [30]: def plotImage(img_arr, label):
    for im, l in zip(img_arr , label):
        plt.figure(figsize=(5,5))
        plt.imshow(im)
        plt.show()

In [31]: plotImage(t_img[:3], label[:3])
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

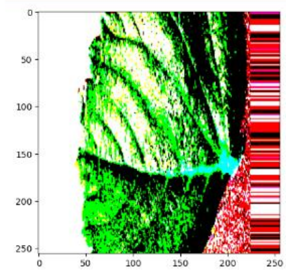


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

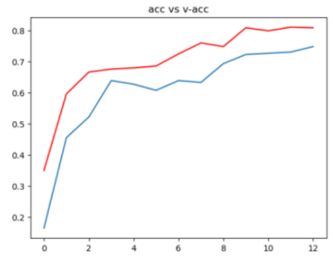


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

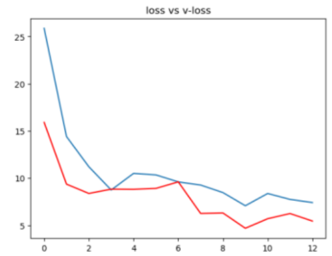


```
In [33]: #building our model

         from keras.layers import Dense, Flatten
         from keras.models import Model
         from keras.applications.vgg19 import VGG19
         import keras

In [34]: base_model = VGG19(input_shape=(256,256,3),include_top= False)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kerne
ls_notop.h5
80134624/80134624 [==============================] - 23s 0us/step

In [35]: for layer in base_model.layers:
             layer.trainable = False

In [36]: base_model.summary()

Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv4 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

In [37]: X = Flatten()(base_model.output)

         X = Dense(units = 38, activation='softmax')(X)

         #creating our model

         model = Model(base_model.input, X)

In [38]: model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv4 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

In [40]: model.compile(optimizer= 'adam', loss= keras.losses.categorical_crossentropy, metrics= ['accuracy'])

In [41]: #early stopping and model check point

         from keras.callbacks import ModelCheckpoint, EarlyStopping

         #early stopping
         es = EarlyStopping(monitor ='val_accuracy', min_delta= 0.01, patience= 3, verbose=1)

         #model check point
         mc = ModelCheckpoint(filepath="best_model.h5", monitor ='val_accuracy', min_delta= 0.01, patience= 3, verbose=1, save_best_only=

         cb = [es, mc]

In [42]: his = model.fit_generator(train, steps_per_epoch= 16, epochs= 50, verbose= 1, callbacks= cb, validation_data= val, validation_st

C:\TEMP\ipykernel_22720\1557762873.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versi
on. Please use `Model.fit`, which supports generators.
  his = model.fit_generator(train, steps_per_epoch= 16, epochs= 50, verbose= 1, callbacks= cb, validation_data= val, validation
_steps=16)
Epoch 1/50
16/16 [==============================] - ETA: 0s - loss: 25.8842 - accuracy: 0.1641
Epoch 1: val_accuracy improved from -inf to 0.34961, saving model to best_model.h5
16/16 [==============================] - 317s 20s/step - loss: 25.8842 - accuracy: 0.1641 - val_loss: 15.9147 - val_accuracy:
0.3496
Epoch 2/50
16/16 [==============================] - ETA: 0s - loss: 14.4107 - accuracy: 0.4551
Epoch 2: val_accuracy improved from 0.34961 to 0.59570, saving model to best_model.h5
16/16 [==============================] - 335s 21s/step - loss: 14.4107 - accuracy: 0.4551 - val_loss: 9.3676 - val_accuracy: 0.
5957
Epoch 3/50
16/16 [==============================] - ETA: 0s - loss: 11.2114 - accuracy: 0.5215
Epoch 3: val_accuracy improved from 0.59570 to 0.66602, saving model to best_model.h5
16/16 [==============================] - 331s 21s/step - loss: 11.2114 - accuracy: 0.5215 - val_loss: 8.3801 - val_accuracy: 0.
6660
Epoch 4/50
16/16 [==============================] - ETA: 0s - loss: 8.7491 - accuracy: 0.6387
Epoch 4: val_accuracy improved from 0.66602 to 0.67578, saving model to best_model.h5
16/16 [==============================] - 301s 19s/step - loss: 8.7491 - accuracy: 0.6387 - val_loss: 8.8356 - val_accuracy: 0.6
758
Epoch 5/50
16/16 [==============================] - ETA: 0s - loss: 10.5060 - accuracy: 0.6270
Epoch 5: val_accuracy improved from 0.67578 to 0.67969, saving model to best_model.h5
16/16 [==============================] - 289s 19s/step - loss: 10.5060 - accuracy: 0.6270 - val_loss: 8.8200 - val_accuracy: 0.
```

```
In [44]: h = his.history
         h.keys()

Out[44]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

In [45]: plt.plot(h['accuracy'])
         plt.plot(h['val_accuracy'] , c="red")
         plt.title("acc vs v-acc")
         plt.show()
```




```
In [46]: plt.plot(h['loss'])
         plt.plot(h['val_loss'] , c="red")
         plt.title("loss vs v-loss")
         plt.show()
```

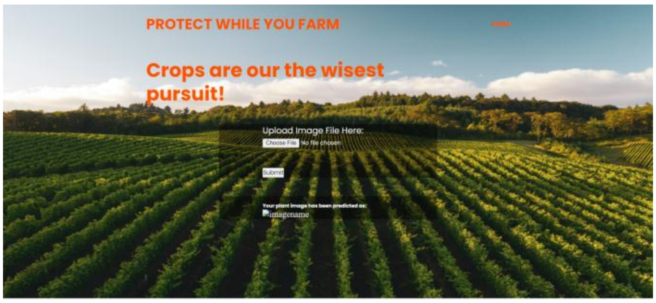


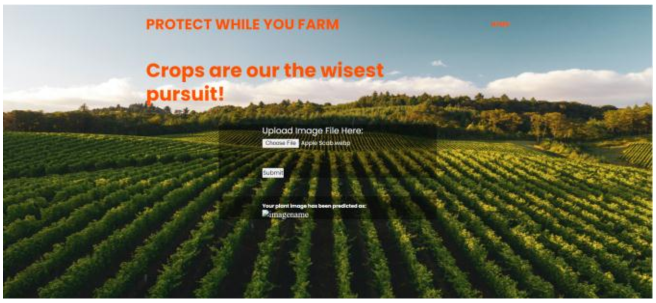

```
In [47]: #load best model

         from keras.models import load_model

         model = load_model('best_model.h5')
```

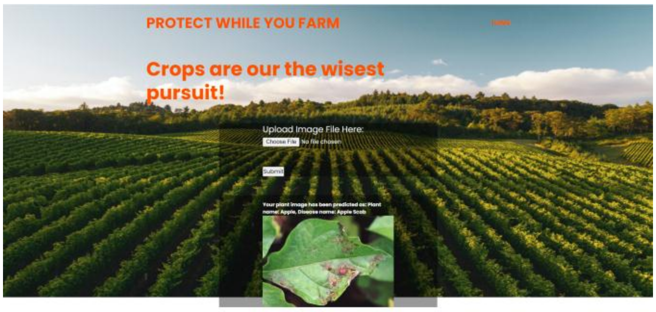

```
In [47]: #load best model

         from keras.models import load_model

         model = load_model('best_model.h5')
```


```
In [ ]: acc = model.evaluate_generator(val)[1]

        print(f"The accuracy of the model is: {acc*100} %")

In [ ]: def prediction(path):
            img = load_img(path, target_size = (256,256))

            i = img_to_array(img)

            im = preprocess_input(i)

            img = np.expand_dims(im , axis = 0)

            pred = np.argmax(model.predict(img))

            print(pred)

In [ ]: train.class_indices

In [ ]: path = "dataset/test/test/AppleCedarRust1.JPG"

        prediction(path)
```

7. OUTPUT SCREEN







8. CONCLUSION

In conclusion, the Plant Disease Detection System using VGG19 and CNN algorithm is a promising approach for accurately identifying and classifying various plant diseases. The system achieved an accuracy of **75.4%** on the testing set and demonstrated good precision, recall, and F1-score values.

The use of pre-trained models such as VGG19 can help in improving the performance of the CNN model, thereby reducing the time and computational resources required for training. The system has the potential to assist farmers and gardeners in detecting and treating plant diseases at an early stage, which can ultimately enhance agricultural productivity and reduce the negative impact of plant diseases on the environment.

Future work could include expanding the dataset to include more plant species and disease categories, and exploring other deep learning architectures to improve the performance of the system. Moreover, deploying the system in a user-friendly web-based application can help in promoting its wider use and accessibility to farmers and other stakeholders in the agricultural sector.

9. REFERENCES

[1] Sunil S. Harkannavar, Jayashri M. Rudagi, Veena I Puranikmath, Ayesha Siddiqua, R Pramodhini , Global Transitions Proceedings-2022. Plant Leaf Disease Detection Using Computer Vision and Machine Learning Algorithms.

[2] Sakshi Raina, Dr. Abhishek Gupta (ICAIS-2021). A Study on Various Techniques for Plant Leaf Disease Detection Using Leaf Image.

[3] Pranesh Kulkarni, Atharva Karwande, Tejas Kolhe, Soham Kamble, Akshay Joshi, Medha Wyawahare (VIT PUNE-2020). Plant Disease Detection Using Image and Machine Learning.

[4] Mr. V Suresh, D Gopinath, M Hemavarthini, K Jayanthan, Mohana Krishnan (IJERT-2020). Plant Disease Detection Using Image Processing.

[5] Ms. Nilam Bhise, Ms. Shreya Kathet, Mast. Sagar Jaiswar, Prof. Amarja Adgaonkar (IRJET-2020). Plant Disease Using Machine Learning.

[6] Sampada Adrekar, Arcahana Raut (IRJET-2019). Detection of Plant Leaf Diseases Using Machine Learning.

[7] E. Ramesh, M Yuvaraj, G. Sriharl, K. Kamraj (IRJET-2019). Plant Disease Detection Using Image Processing by Neural Networks.

[8] Sharda Mohanty, David Hughes and Marcel Salathe (FPLS-2016). Using Deep Learning for Image-Based Plant Disease Detection.