A Project Report

on

Plant Disease Detection (FarmDoc)

Submitted By:

STUTI UPADHYAY (19BT04056)

IN

COMPUTER SCIENCE AND ENGINEERING

Guided by:

Dr. Saurabh Shah,

Professor and Dean, School of Technology.

Academic Year: 2020-21.

GSFC University
School of Technology
Computer Science and Engineering

CANDIDATES' DECLARATION

We, the students of Computer Science & Engineering hereby declare that the project report entitled " Plant Disease Detection " is our own work conducted under the supervision of the guide Dr. Saurabh Shah for the subject **Fundamentals of AI & ML (BTCS405)** in Semester IV, Academic Year 2020-21 .

We further declare that to the best of our knowledge that the report contains the original work of the project carried out as the partial fulfillment of the assignment submission.

Stuti Upadhyay **19BT04056** **Sign:** Stuti

Date: 22nd May, 2021.

CERTIFICATE

This is to certify that the project entitled **"** ***Plant Disease Detection*** **"** is a bonafide report of the work carried out by ***Stuti Upadhyay (19BT04056)*** subject in Semester IV of Computer Science & Engineering under the guidance and supervision of **Dr. Saurabh Shah** for the partial fulfillment of assignment submission.

To the best of my knowledge and belief, this work embodies the work of candidates themselves, have duly completed, fulfills the requirement of assignment submission and is up to the standard in respect of content, presentation and language.

Date: 22nd May, 2021

Dr. Saurabh Shah
Project Guide

GSFC University
School of Technology
Computer Science and Engineering

Abstract

Our idea is to build an app for farmers to detect the disease in the crops/plants and then provide the farmers it’s remedies like pesticides, fertilizers, etc. with day-to-day dose type progress. And information on various government schemes in the same app which help farmers to get benefits that the government is providing to the farmers. In the same app the farmers will get the information on modern techniques of farming and the app will also recommend to farmers what type of crops should be cultivated according to the soil.

INDEX

Introduction

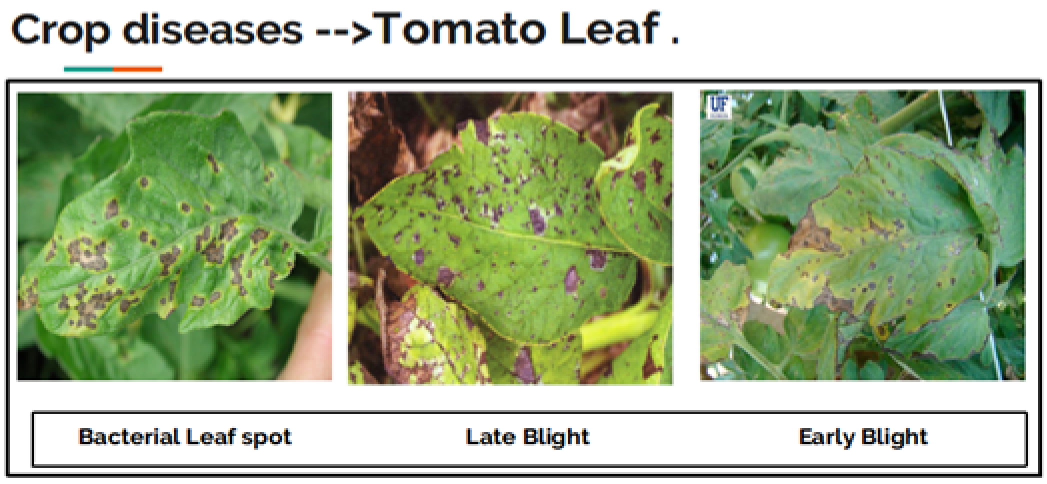
Image is a collection of pixels or dots which are stored in a rectangular array. Each individual pixel is having a certain kind of color. We can measure the size of the image by counting the no of pixels in that particular image. Different types of images are there such as Black and White and Grey scale images. Both types vary from each other .In black and white images each dot or pixel is either black or white, therefore only one bit is needed per pixel. Whereas Grey scale images use 8 bits per pixel.

For color images things get slightly difficult. In color images the number of bits at every dot is termed as the height of the image. It is also referred to as the bit plane. For bit planes consisting of x, 2x color are possible.

Different methods are available to store the color information of an image. One of the methods is an RGB image, also termed a true color image. For every pixel red, green and blue components are stored in three dimensional arrays.

1.1 Project Summary

For centuries and even in modern times, farmers usually detect crop diseases with their naked eye which makes it a tough decision for them on what type of fertilizers are they exactly supposed to use. It requires detailed knowledge about the types of diseases and a lot of experience to make sure that it's the same disease that they assumed it was. Some of the diseases look almost similar which often leaves them confused.



They all look the same, well in case if the farmer makes wrong predictions and uses the wrong fertilizers or more than the normal dose i.e, the threshold (every plant has some limit of fertilizers usage that is followed), it can mess up the whole plant (or) soil and cause enough damage to the fields. So, How to prevent this from happening? To prevent this situation we need better and perfect guidance for plant diseases and the ability to distinguish between two or more similar types of diseases in visuals, this is where our project comes in handy.

1.2 Purpose

There are different types of plant disease, but the majority of these diseases can be categorized into the three different categories which are bacterial disease, viral disease and fungal disease. The most ideal way to detect the disease is the classification followed by detection. Classification is done on the basis of shape and texture features.

Bacterial Disease

This is also known as bacterial leaf spot. Bacterial leaf spot is majorly detected in stone fruits such cherry, plum etc. In this disease black spots or dark spots occur on the different parts of leaves. Yellow halos are also a symptom of this disease. Spot size is of irregular nature. Bacterial spots occur on the different parts on the top and bottom start occurring and if these spots cluster together in any section of the leaf then this results in killing of that section by this disease. Wet and cool formation also contribute to the formation of the bacterial disease in the plant leaves. In these formations bacterial leaf spots can spread very quickly. Most bacterial leaf spots occur on the aged leaves but it can destroy the tissues of the new leaves too.

Viral Disease

Viral diseases are caused by viruses and as viruses are intracellular, so these diseases attack inside out. Viral diseases are sometimes very difficult to identify. Virus can affect any region of the plants such as leaves, roots, stem and others. Abnormal patterns are observed on the affected area; green and yellow coloration is seen in leaves affected with the virus. The life span of the plant or its parts affected with the viral disease is very less. It directly affects productivity and other factors. Wrinkles on the different parts of the leaves is also the primary symptom of these diseases. Every virus life span is very high as compared to the other types of the disease, because each virus if not properly cured gives rise to a new type of the virus so it is important for timely prevention of these diseases.

Fungal Disease

Fungal disease occurs because of the fungi or fungal organism. One of the properties of the fungi is that it spread with wind and the water. Gray green spots on the leaf of the plants are observed and if not properly cured they start spreading toward the outer region of the leaf. Wilting scabs are the primary symptoms of fungal disease. Fungal disease attacks on the plant leaves result in the yellowness of leaves at the end.

1.3 Scope of the Project

To make our model communicate with the App we will convert it into the TensorFlow lite version, tflite is made for mobile Versions. So, we can build or create a mobile app and make the app communicate with the model.

1.4 Technology and Literature Review

Objectives of First Paper: Rice Disease detection Pattern Recognition Techniques. Published in the 11th International Conference on Computer and Information Technology.

Gist of Paper: The point of this paper is to depict a product model framework for the discovery of malady in rice plants based on different pictures of the rice plants. Pictures of the tainted piece of the rice plant are taken utilizing a computerized camera. With the end goal to identify the abandoned piece of the plant different procedures like picture division, picture developing and so forth. By utilizing the neural system the tainted piece of the leaf is grouped. Picture preparing and delicate processing procedures are joined on the infected plant.

Procedures embraced in paper:
- Preparing & design examination strategies of images
- Binary cutoff methods
- Border layout calculation using eight-availability strategy
- Self-organizing map(SOM)

In this examination paper, the diseased part of the rice plant leaf is identified with the help of the self-organizing map. Testing is done using four different images of the crop. Infected region is extracted using neural networks pattern recognition techniques.

By utilizing effective example acknowledgment procedures, the framework will have the capacity to do the opportune finding of the field issue and the proposal will assist the ranchers with taking the suitable measure to build the nature of the harvest .It won't just decrease the improvement cost later on yet in addition spare the earth too.

Objective of Second Paper: Detection of plant leaf diseases using image segmentation and soft computing techniques. Published in Information Processing in agriculture.

Gist of Paper: This paper monitors crop growth using the image segmentation techniques. Noise filtering is done and features are extracted and then image is further classified to detect the diseased part.

Strategies/Methodology embraced in paper:
- Support Vector Machine (SVM)
- Artificial Neural Network (ANN)
- Dispersion method
- Self-sorting out element

Using image segmentation techniques and machine learning algorithms the information for ripening stages of crop and infected part recognition is made.

There were a few issues for doing extraction of vague shading pixels from the foundation of the picture.
- Neural arrangements don't permit better division of the grape leaf illness pixels.
- The framework will show programmed determination ability with extremely successful execution for the further agrarian item investigation/review framework improvement.

Objective of Third Paper: Remote Area Plant diseases detection using Image Processing. Published in IOSR Journal of Electronics and Communication Engineering.

Gist of Paper: Infected parts in the plant can be detected with help of color, and other changing properties by using classification algorithms.

Strategy/Methodology embraced in paper:

- Segmentation
- RGB
- Color transformation
- Image acquisition
- Classification

Different pixel information is extracted and Green leaves pixel and diseased leaf pixel are compared by finding the ratio of pixel corresponding to the healthy leaf to the pixel corresponding to the infected leaf. Background is removed and different regions of the images are formatted after the image acquisition.

- Using image segmentation to extract the image feature is best. But the important thing is the level of the results which are derived using how reliable.
- Results exhibited intriguing enhancement in the forecast framework. It is possibly a promising option in contrast to existing expectation models.
- Further these approaches are scalable and can be modified as per the requirements.

Objective of Fourth Paper: Image processing for smart farming: Detection of disease. Published in IEEE second International Conference.

Gist of Paper: The point of this paper is to separate the various diseases in the different parts and then apply the suitable algorithm and to design the approaches in order to detect the diseases using artificial neural networks. Two databases of the image are used, one for training and other for the testing.

Strategy/Methodology embraced in paper

- Image Segmentations
- ML calculations
- Support vector machines
- Artificial Neural Networks

All the disease of the apple and grapes are correctly identified using digital image processing and machine learning neural networks. Results get improved when contrasted with the discriminative models.

Machine Learning approaches are adaptable and furthermore can give secluded methodology to the information investigation particularly for the new area of 'plant pressure examination'. It will likewise help in the quality revelation process and in addition the presentation of novel determinations conventions for the complex aggressive attributes like biotic and abiotic stress and yield.

Objective of Fifth Paper: A digital image processing based algorithm which detects and recognizes plant diseases and various symptoms. Bio system Engineering Volume 102, Issue 1, January 2009, pp. 9-21.

Gist of Paper: The idea proposed here is to process and analyze the colored images to identify the affected area and various visible symptoms of the diseases.

Strategy/Methodology embraced in paper

- Acquisition and Preprocessing
- Pixel contrast enhancement
- Segmentation
- Classification

The proposed methodology is applied to twenty different kinds of images and the result is made on the basis of white and black colors in the resulting image. Black color is used to represent the symptom of disease and for the unaffected region. The results are also compared with the binary images in which one represented the disease region and zero represented the not diseased part. Some variation was there in both the results.

Objective of Sixth Paper: Plant disease detection using image processing. 2015 international conference on computing communication control and automation.

Gist of Paper: the paper provides different types of image processing techniques which can directly be implemented in MATLAB for preprocessing of the image and also brief about the image segmentation and image classification. For the feature extraction the paper provides algorithmic methods which can easily calculate shape and color oriented features.

Strategies/Methodologies embraced in paper:

- Image preprocessing
- Image enhancement
- SVM classification
- Semantic networks
- K means clustering
- Neural network based classifier

The feature extraction and the image segmentation algorithm used in this paper are efficient with very high accuracy. Different diseases are identified with very high precision rate and accuracy. Clustering algorithm approach is also very much efficient fast, the clustering algorithm segments the image in the different clusters in a very short span of time. For a small dataset of images the algorithm used in this paper is very effective. But if the dataset size gets larger then there may come some sort of distortion in the accuracy of the above approach.

Objective of Seventh Paper: Smart Farming: Pomegranate Disease Detection Using Image Processing" Second International Symposium on Computer Vision and the Internet 2015.

Gist of Paper: The paper provides the image processing techniques and the algorithm which help the farmers to successfully identify the disease in pomegranate. Image acquisition and image processing of the input image is done using the filter commands in the MATLAB by pixel values of the input image gets more clarified and the disease which is present in the pomegranate is successfully detected. The approach also provides users two different options that are with intent search and without intent search.

Strategies/Methodologies adopted

- Image preprocessing
- Feature extraction
- Morphology
- Color Coherence Vector (CCV)
- Clustering
- Training and classification
- Intent search

Remote area farmers can also identify the disease in the pomegranate crop as this algorithm gives users options with and without intent. It lets the user upload the image in the system for further processing. This approach is very effective. Also experimental readings show that the algorithm has eighty two percent accuracy considering the average of both cases. In almost every case pomegranate disease is identified. Web base approach is also very fast and there is no distortion occurring in communication of images in transition of images in this approach. Pixel values remain maintained and cleared.

1.5 Hardware and Software Requirements

- A computer running either Windows, macOS or the Linux operating system
- A stable internet connection
- An internet based IDE for python like Google Collab or platform like Kaggle or any preferred IDE in Anaconda Navigator.

System Analysis and Design

2.1 Study of Current System

We analyzed 70,295 images of plant leaves, which have a spread of 38 class labels assigned to them. Each class label is a crop-disease pair, and we make an attempt to predict the crop-disease pair given just the image of the plant leaf. Figure (1) shows one example each from every crop-disease pair from the train dataset. In all the approaches described in this report, we resize the images to 256 × 256 pixels, and we perform both the model optimization and predictions on these downscaled images.

FIGURE 1

Figure 1. Example of leaf images from the PlantVillage dataset, representing every crop-disease pair used. **(1)** Apple Scab, *Venturia inaequalis* **(2)** Apple Black Rot, *Botryosphaeria obtusa* **(3)** Apple Cedar Rust, *Gymnosporangium juniperi-virginianae* **(4)** Apple healthy **(5)** Blueberry healthy **(6)** Cherry healthy **(7)** Cherry Powdery Mildew, *Podoshaera clandestine* **(8)** Corn Gray Leaf Spot, *Cercospora zeae-maydis* **(9)** Corn Common Rust, *Puccinia sorghi* **(10)** Corn healthy **(11)** Corn Northern Leaf Blight, *Exserohilum turcicum* **(12)** Grape Black Rot, *Guignardia bidwellii*, **(13)** Grape Black Measles (Esca), *Phaeomoniella aleophilum, Phaeomoniella chlamydospora* **(14)** Grape Healthy **(15)** Grape Leaf Blight, *Pseudocercospora vitis* **(16)** Orange Huanglongbing (Citrus Greening), *Candidatus Liberibacter spp.* **(17)** Peach Bacterial Spot, *Xanthomonas campestris* **(18)** Peach healthy **(19)** Bell Pepper Bacterial Spot, *Xanthomonas campestris* **(20)** Bell Pepper healthy **(21)** Potato Early Blight, *Alternaria solani* **(22)** Potato healthy **(23)** Potato Late Blight, *Phytophthora infestans* **(24)** Raspberry healthy **(25)** Soybean healthy **(26)** Squash Powdery Mildew, *Erysiphe cichoracearum* **(27)** Strawberry Healthy **(28)** Strawberry Leaf Scorch, *Diplocarpon earlianum* **(29)** Tomato Bacterial Spot, *Xanthomonas campestris pv. vesicatoria* **(30)** Tomato Early Blight, *Alternaria solani* **(31)** Tomato Late Blight, *Phytophthora infestans* **(32)** Tomato Leaf Mold, *Passalora fulva* **(33)** Tomato Septoria Leaf Spot, *Septoria lycopersici* **(34)** Tomato Two Spotted Spider Mite, *Tetranychus urticae* **(35)** Tomato Target Spot, *Corynespora cassiicola* **(36)** Tomato Mosaic Virus **(37)** Tomato Yellow Leaf Curl Virus **(38)** Tomato healthy.

We evaluate the applicability of deep convolutional neural networks for the classification problem described above. We focus on a popular architecture named AlexNet, which was designed in the context of the "Large Scale Visual Recognition Challenge" (ILSVRC) for the ImageNet dataset.

The AlexNet architecture follows the same design pattern as the LeNet-5 architecture from the 1990s. The LeNet-5 architecture variants are usually a set of stacked convolution layers followed by one or more fully connected layers. The convolution layers optionally may have a normalization layer and a pooling layer right after them, and all the layers in the network usually have ReLu non-linear activation units associated with them. AlexNet consists of 5 convolution layers, followed by 3 fully connected layers, and finally ending with a softMax layer. The first two convolution layers (conv{1, 2}) are each followed by normalization and a pooling layer, and the last convolution layer (conv5) is followed by a single pooling layer. The final fully connected layer (fc8) has 38 outputs in our adapted version of AlexNet (equaling the total number of classes in our dataset), which feeds the softMax layer. The softMax layer finally exponentially normalizes the input that it gets from (fc8), thereby producing a distribution of values across the 38 classes that add up to 1. These values can be interpreted as the confidences of the network that a given input image is represented by the corresponding classes. All of the first 7 layers of AlexNet have a ReLu non-linearity activation unit associated with them, and the first two fully connected layers (fc{6, 7}) have a dropout layer associated with them, with a dropout ratio of 0.5.

2.2 Problem and Weakness of Current System

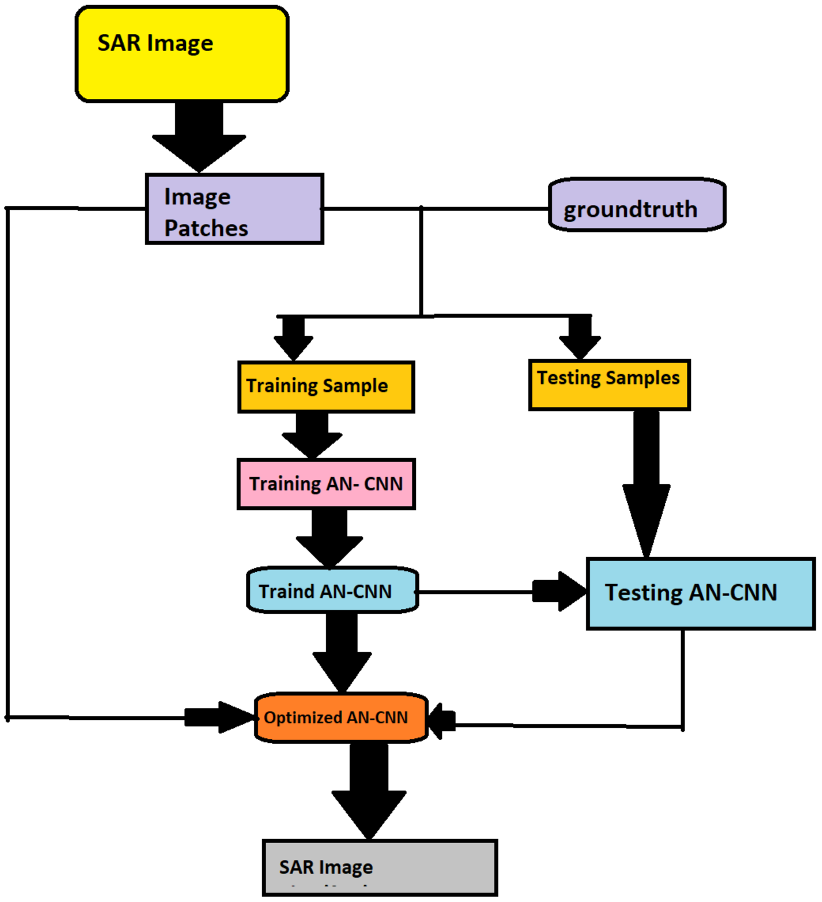
As of now, there is no graphical interface for our system. Right now we have to upload our input manually in our code which isn't very feasible and time effective, so our next aim here is to convert this into a mode that can interact with mobile devices now to accomplish that we will convert this model into Tensorflow Lite which is especially used to interact with mobile devices.

2.3 Requirement of New System

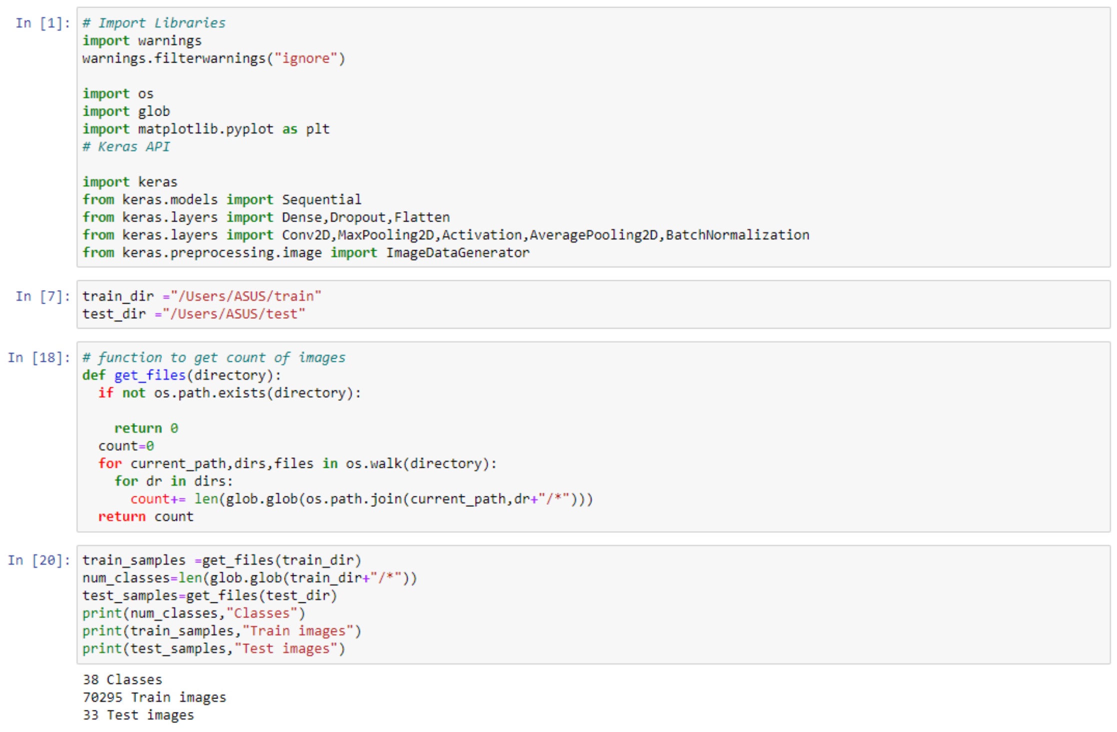
In the current dataset, we only have coloured images which is somewhat a limitation since there's also a possibility that our input from the user can also be a grayscale picture. Nowm, to overcome that we plan on expanding our dataset and add grayscale pictures of the current that we have in our possession.

2.4 System Design

The given flowchart basically explains the working of our system in brief.



Implementation and Screenshots

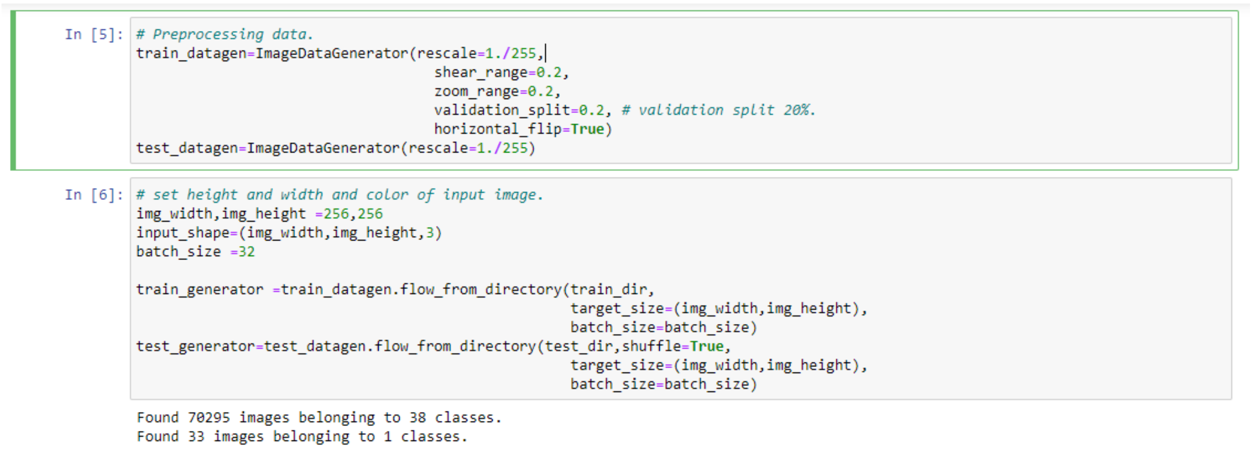


```
In [1]: # Import Libraries
        import warnings
        warnings.filterwarnings("ignore")

        import os
        import glob
        import matplotlib.pyplot as plt
        # Keras API

        import keras
        from keras.models import Sequential
        from keras.layers import Dense,Dropout,Flatten
        from keras.layers import Conv2D,MaxPooling2D,Activation,AveragePooling2D,BatchNormalization
        from keras.preprocessing.image import ImageDataGenerator
```

```
In [7]: train_dir ="/Users/ASUS/train"
        test_dir ="/Users/ASUS/test"
```

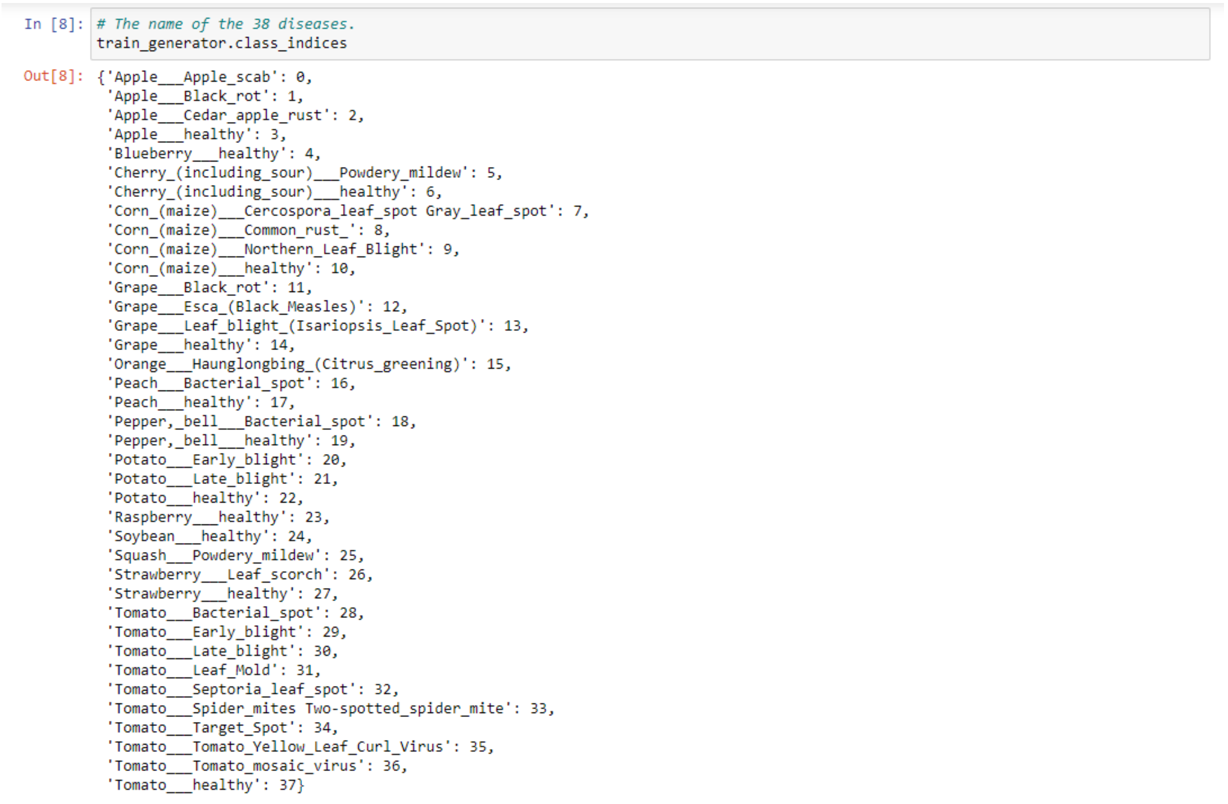
```
In [18]: # function to get count of images
         def get_files(directory):
           if not os.path.exists(directory):

             return 0
           count=0
           for current_path,dirs,files in os.walk(directory):
             for dr in dirs:
               count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
           return count
```

```
In [20]: train_samples =get_files(train_dir)
         num_classes=len(glob.glob(train_dir+"/*"))
         test_samples=get_files(test_dir)
         print(num_classes,"Classes")
         print(train_samples,"Train images")
         print(test_samples,"Test images")

         38 Classes
         70295 Train images
         33 Test images
```

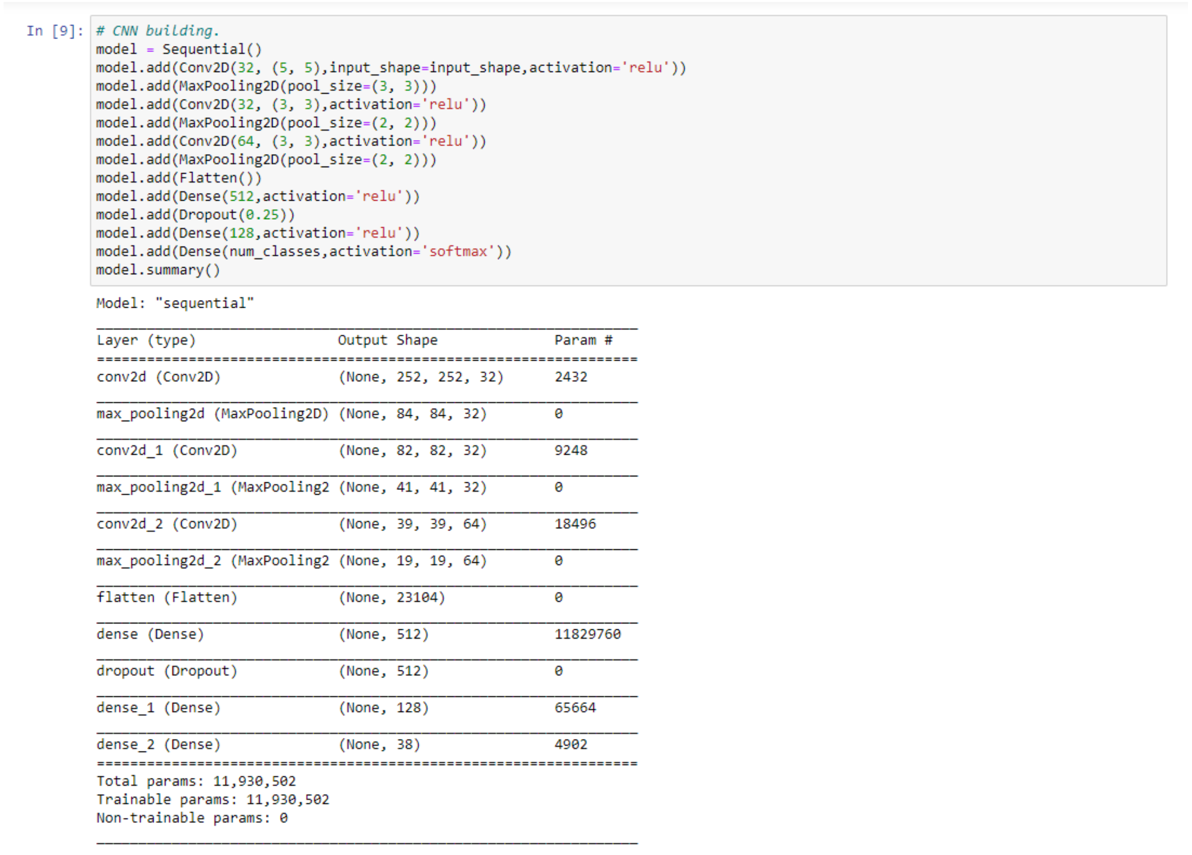


```
In [5]: # Preprocessing data.
        train_datagen=ImageDataGenerator(rescale=1./255,
                                         shear_range=0.2,
                                         zoom_range=0.2,
                                         validation_split=0.2, # validation split 20%.
                                         horizontal_flip=True)
        test_datagen=ImageDataGenerator(rescale=1./255)
```

```
In [6]: # set height and width and color of input image.
        img_width,img_height =256,256
        input_shape=(img_width,img_height,3)
        batch_size =32

        train_generator =train_datagen.flow_from_directory(train_dir,
                                                           target_size=(img_width,img_height),
                                                           batch_size=batch_size)
        test_generator=test_datagen.flow_from_directory(test_dir,shuffle=True,
                                                           target_size=(img_width,img_height),
                                                           batch_size=batch_size)

        Found 70295 images belonging to 38 classes.
        Found 33 images belonging to 1 classes.
```



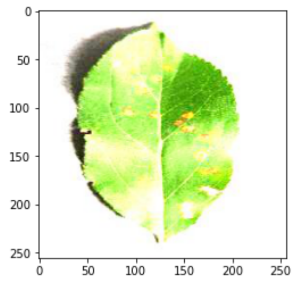
```
In [8]: # The name of the 38 diseases.
        train_generator.class_indices

Out[8]: {'Apple___Apple_scab': 0,
         'Apple___Black_rot': 1,
         'Apple___Cedar_apple_rust': 2,
         'Apple___healthy': 3,
         'Blueberry___healthy': 4,
         'Cherry_(including_sour)___Powdery_mildew': 5,
         'Cherry_(including_sour)___healthy': 6,
         'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 7,
         'Corn_(maize)___Common_rust_': 8,
         'Corn_(maize)___Northern_Leaf_Blight': 9,
         'Corn_(maize)___healthy': 10,
         'Grape___Black_rot': 11,
         'Grape___Esca_(Black_Measles)': 12,
         'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 13,
         'Grape___healthy': 14,
         'Orange___Haunglongbing_(Citrus_greening)': 15,
         'Peach___Bacterial_spot': 16,
         'Peach___healthy': 17,
         'Pepper,_bell___Bacterial_spot': 18,
         'Pepper,_bell___healthy': 19,
         'Potato___Early_blight': 20,
         'Potato___Late_blight': 21,
         'Potato___healthy': 22,
         'Raspberry___healthy': 23,
         'Soybean___healthy': 24,
         'Squash___Powdery_mildew': 25,
         'Strawberry___Leaf_scorch': 26,
         'Strawberry___healthy': 27,
         'Tomato___Bacterial_spot': 28,
         'Tomato___Early_blight': 29,
         'Tomato___Late_blight': 30,
         'Tomato___Leaf_Mold': 31,
         'Tomato___Septoria_leaf_spot': 32,
         'Tomato___Spider_mites Two-spotted_spider_mite': 33,
         'Tomato___Target_Spot': 34,
         'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 35,
         'Tomato___Tomato_mosaic_virus': 36,
         'Tomato___healthy': 37}
```



```
In [9]: # CNN building.
        model = Sequential()
        model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu'))
        model.add(MaxPooling2D(pool_size=(3, 3)))
        model.add(Conv2D(32, (3, 3),activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Conv2D(64, (3, 3),activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Flatten())
        model.add(Dense(512,activation='relu'))
        model.add(Dropout(0.25))
        model.add(Dense(128,activation='relu'))
        model.add(Dense(num_classes,activation='softmax'))
        model.summary()

        Model: "sequential"
        _________________________________________________________________
        Layer (type)                 Output Shape              Param #
        =================================================================
        conv2d (Conv2D)              (None, 252, 252, 32)      2432
        _________________________________________________________________
        max_pooling2d (MaxPooling2D) (None, 84, 84, 32)        0
        _________________________________________________________________
        conv2d_1 (Conv2D)            (None, 82, 82, 32)        9248
        _________________________________________________________________
        max_pooling2d_1 (MaxPooling2 (None, 41, 41, 32)        0
        _________________________________________________________________
        conv2d_2 (Conv2D)            (None, 39, 39, 64)        18496
        _________________________________________________________________
        max_pooling2d_2 (MaxPooling2 (None, 19, 19, 64)        0
        _________________________________________________________________
        flatten (Flatten)            (None, 23104)             0
        _________________________________________________________________
        dense (Dense)                (None, 512)               11829760
        _________________________________________________________________
        dropout (Dropout)            (None, 512)               0
        _________________________________________________________________
        dense_1 (Dense)              (None, 128)               65664
        _________________________________________________________________
        dense_2 (Dense)              (None, 38)                4902
        =================================================================
        Total params: 11,930,502
        Trainable params: 11,930,502
        Non-trainable params: 0
        _________________________________________________________________
```

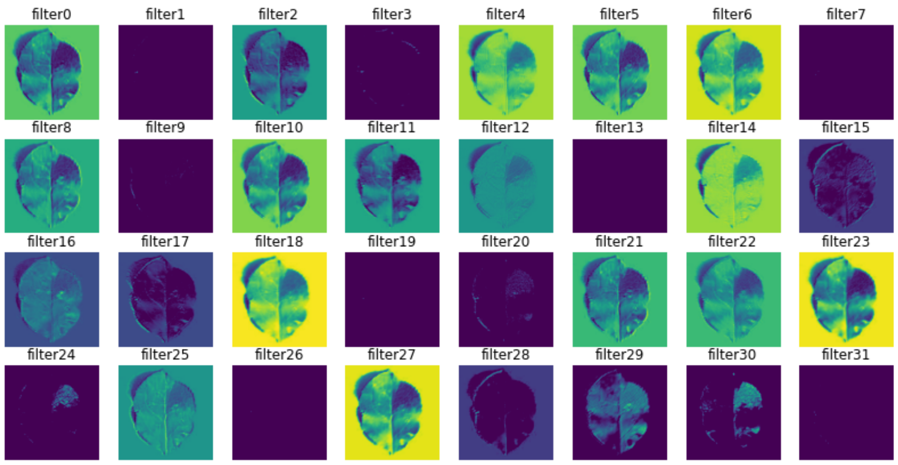
```
In [10]: model_layers = [ layer.name for layer in model.layers]
         print('layer name : ',model_layers)

layer name :  ['conv2d', 'max_pooling2d', 'conv2d_1', 'max_pooling2d_1', 'conv2d_2', 'max_pooling2d_2', 'flatten', 'dense', 'dr
opout', 'dense_1', 'dense_2']
```

```
In [11]: # Take one image to visualize it's changes after every layer
         from keras.preprocessing import image
         import numpy as np
         img1 = image.load_img('/Users/ASUS/test/test/AppleCedarRust1.JPG')
         plt.imshow(img1);

         #preprocess image
         img1 = image.load_img('/Users/ASUS/test/test/AppleCedarRust1.JPG', target_size=(256, 256))
         img = image.img_to_array(img1)
         img = img/255
         img = np.expand_dims(img, axis=0)
```

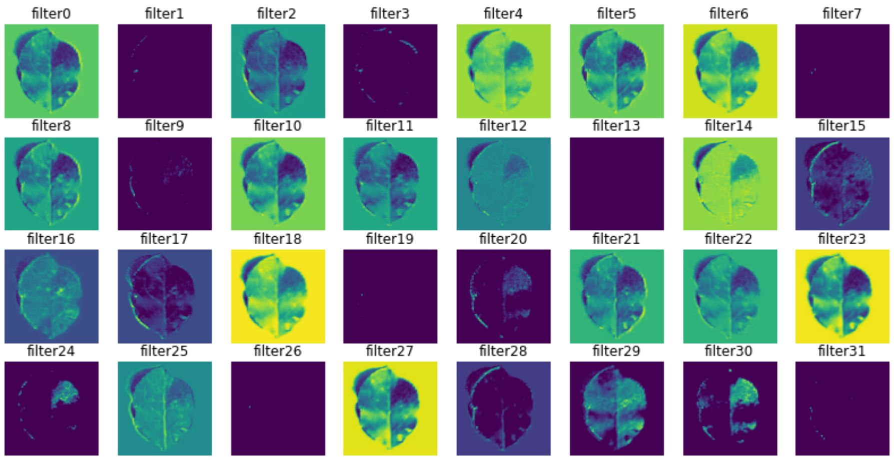


```
In [12]: # Visualizing output after every layer.
         from keras.models import Model
         conv2d_output = Model(inputs=model.input, outputs=model.get_layer('conv2d').output)
         max_pooling2d_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d').output)
         conv2d_1_output = Model(inputs=model.input,outputs=model.get_layer('conv2d_1').output)
         max_pooling2d_1_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d_1').output)
         conv2d_2_output = Model(inputs=model.input,outputs=model.get_layer('conv2d_2').output)
         max_pooling2d_2_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d_2').output)
         flatten_output = Model(inputs=model.input,outputs=model.get_layer('flatten').output)
         conv2d_features = conv2d_output.predict(img)
         max_pooling2d_features = max_pooling2d_output.predict(img)
         conv2d_1_features = conv2d_1_output.predict(img)
         max_pooling2d_1_features = max_pooling2d_1_output.predict(img)
         conv2d_2_features = conv2d_2_output.predict(img)
         max_pooling2d_2_features = max_pooling2d_2_output.predict(img)
         flatten_features = flatten_output.predict(img)

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x00000202EF
BAB820> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating
@tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. Fo
r (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option tha
t relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/funct
ion#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x00000202EF
C89C10> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating
@tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. Fo
r (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option tha
t relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/funct
ion#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:7 out of the last 7 calls to <function Model.make_predict_function.<locals>.predict_function at 0x00000202EF
C89AF0> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating
@tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. Fo
r (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option tha
t relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/funct
ion#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

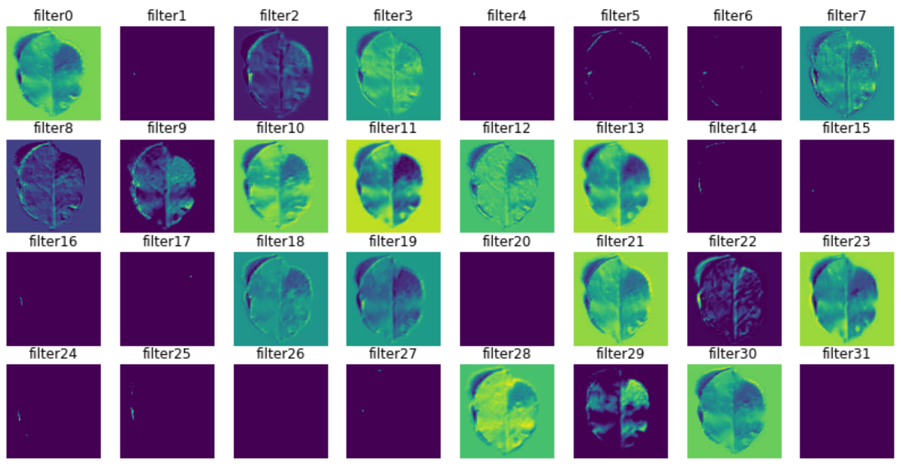
```
In [13]: import matplotlib.image as mpimg

         fig=plt.figure(figsize=(14,7))
         columns = 8
         rows = 4
         for i in range(columns*rows):
             #img = mpimg.imread()
             fig.add_subplot(rows, columns, i+1)
             plt.axis('off')
             plt.title('filter'+str(i))
             plt.imshow(conv2d_features[0, :, :, i], cmap='viridis') # Visualizing in color mode.
         plt.show()
```



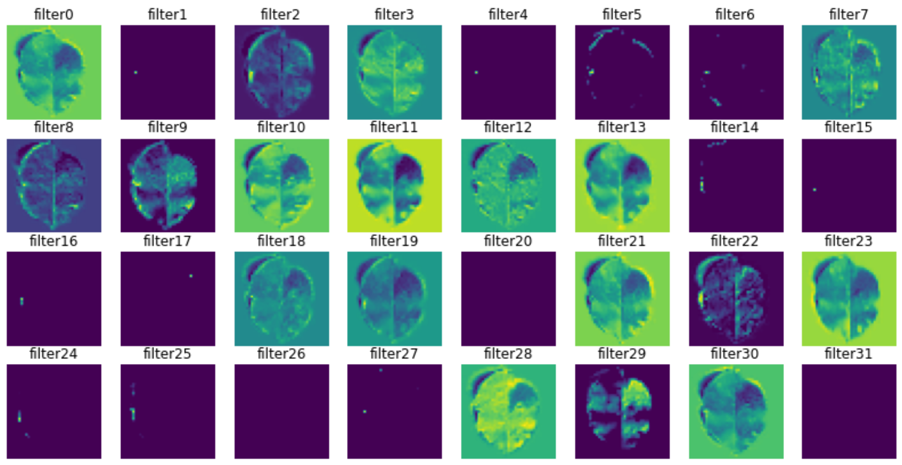
```
In [14]: import matplotlib.image as mpimg

         fig=plt.figure(figsize=(14,7))
         columns = 8
         rows = 4
         for i in range(columns*rows):
             #img = mpimg.imread()
             fig.add_subplot(rows, columns, i+1)
             plt.axis('off')
             plt.title('filter'+str(i))
             plt.imshow(max_pooling2d_features[0, :, :, i], cmap='viridis')
         plt.show()
```



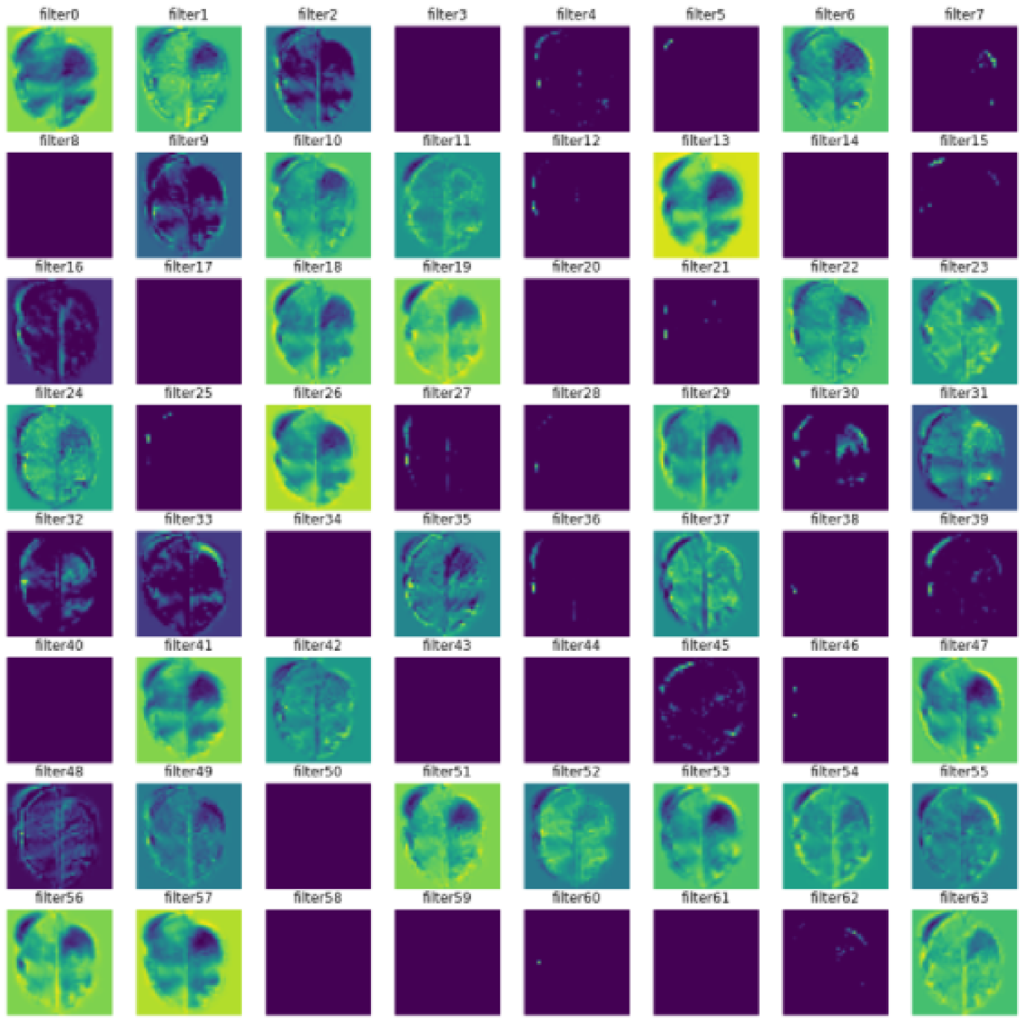
In [16]:
```python
import matplotlib.image as mpimg

fig=plt.figure(figsize=(14,7))
columns = 8
rows = 4
for i in range(columns*rows):
    #img = mpimg.imread()
    fig.add_subplot(rows, columns, i+1)
    plt.axis('off')
    plt.title('filter'+str(i))
    plt.imshow(conv2d_1_features[0, :, :, i], cmap='viridis')
plt.show()
```



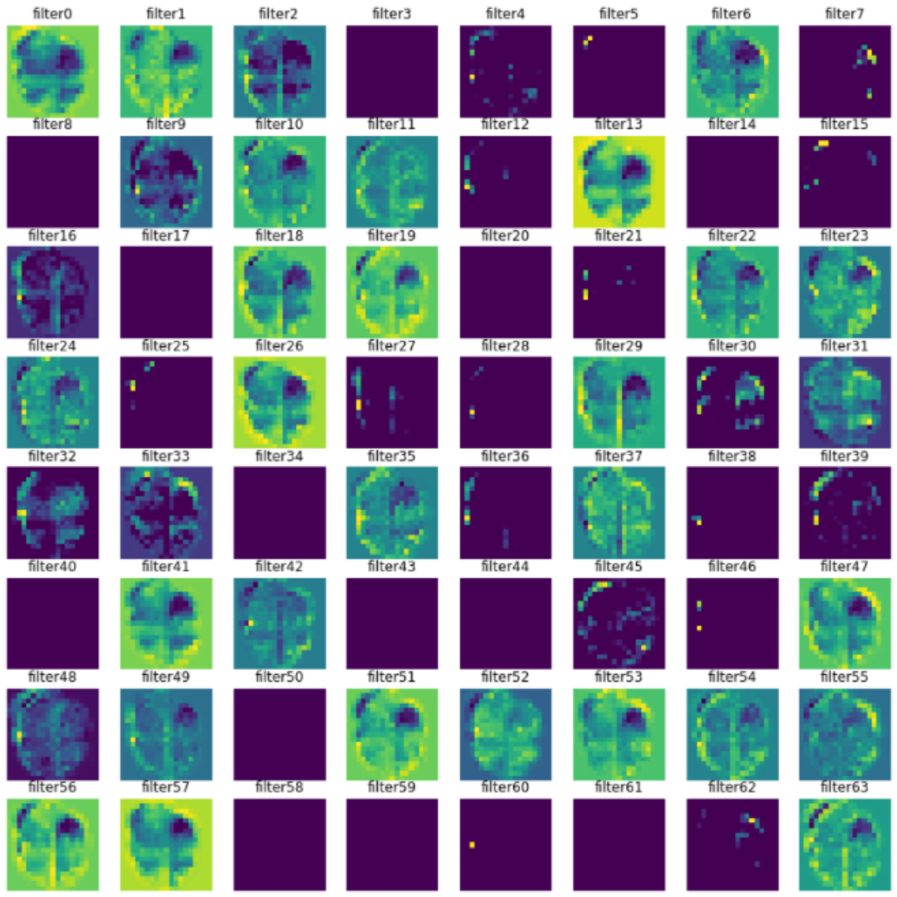
In [17]:
```python
# we can also visualize in color mode.
import matplotlib.image as mpimg

fig=plt.figure(figsize=(14,7))
columns = 8
rows = 4
for i in range(columns*rows):
    #img = mpimg.imread()
    fig.add_subplot(rows, columns, i+1)
    plt.axis('off')
    plt.title('filter'+str(i))
    plt.imshow(max_pooling2d_1_features[0, :, :, i], cmap='viridis')
plt.show()
```



```
In [19]: import matplotlib.image as mpimg

         fig=plt.figure(figsize=(16,16))
         columns =8
         rows = 8
         for i in range(columns*rows):
             #img = mpimg.imread()
             fig.add_subplot(rows, columns, i+1)
             plt.axis('off')
             plt.title('filter'+str(i))
             plt.imshow(conv2d_2_features[0, :, :, i], cmap='viridis')
         plt.show()
```



```
In [21]: import matplotlib.image as mpimg

         fig=plt.figure(figsize=(14,14))
         columns = 8
         rows = 8
         for i in range(columns*rows):
             #img = mpimg.imread()
             fig.add_subplot(rows, columns, i+1)
             plt.axis('off')
             plt.title('filter'+str(i))
             plt.imshow(max_pooling2d_2_features[0, :, :, i],cmap='viridis')
         plt.show()
```



```
In [22]: # validation data.
         validation_generator = train_datagen.flow_from_directory(
             train_dir, # same directory as training data
             target_size=(img_height, img_width),
             batch_size=batch_size)

         Found 70295 images belonging to 38 classes.
```

```
In [23]: # Model building to get trained with parameters.
         opt=keras.optimizers.Adam(lr=0.001)
         model.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
         train=model.fit_generator(train_generator,
                                   epochs=5,
                                   steps_per_epoch=train_generator.samples // batch_size,
                                   validation_data=validation_generator,
                                   verbose=1)

         Epoch 1/5
         2196/2196 [==============================] - 8065s 4s/step - loss: 2.1357 - accuracy: 0.3907 - val_loss: 0.4942 - val_accuracy:
         0.8404
         Epoch 2/5
         2196/2196 [==============================] - 10074s 5s/step - loss: 0.5838 - accuracy: 0.8150 - val_loss: 0.2978 - val_accurac
         y: 0.9016
         Epoch 3/5
         2196/2196 [==============================] - 8310s 4s/step - loss: 0.3965 - accuracy: 0.8712 - val_loss: 0.2390 - val_accuracy:
         0.9226
         Epoch 4/5
         2196/2196 [==============================] - 21198s 10s/step - loss: 0.3197 - accuracy: 0.8962 - val_loss: 0.1724 - val_accurac
         y: 0.9426
         Epoch 5/5
         2196/2196 [==============================] - 6605s 3s/step - loss: 0.2712 - accuracy: 0.9131 - val_loss: 0.1488 - val_accuracy:
         0.9505
```

```
In [24]: # Save entire model with optimizer, architecture, weights and training configuration.
         from keras.models import load_model
         model.save('crop.h5')
```

```
In [25]: # Save model weights.
         from keras.models import load_model
         model.save_weights('crop_weights.h5')
```

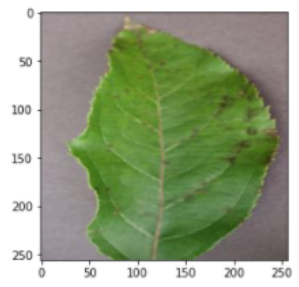
```
In [26]: # Get classes of model trained on
         classes = train_generator.class_indices
         classes
```

```
Out[26]: {'Apple___Apple_scab': 0,
          'Apple___Black_rot': 1,
          'Apple___Cedar_apple_rust': 2,
          'Apple___healthy': 3,
          'Blueberry___healthy': 4,
          'Cherry_(including_sour)___Powdery_mildew': 5,
          'Cherry_(including_sour)___healthy': 6,
          'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot': 7,
          'Corn_(maize)___Common_rust_': 8,
          'Corn_(maize)___Northern_Leaf_Blight': 9,
          'Corn_(maize)___healthy': 10,
          'Grape___Black_rot': 11,
          'Grape___Esca_(Black_Measles)': 12,
          'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)': 13,
          'Grape___healthy': 14,
          'Orange___Haunglongbing_(Citrus_greening)': 15,
          'Peach___Bacterial_spot': 16,
          'Peach___healthy': 17,
          'Pepper,_bell___Bacterial_spot': 18,
          'Pepper,_bell___healthy': 19,
          'Potato___Early_blight': 20,
          'Potato___Late_blight': 21,
          'Potato___healthy': 22,
          'Raspberry___healthy': 23,
          'Soybean___healthy': 24,
          'Squash___Powdery_mildew': 25,
          'Strawberry___Leaf_scorch': 26,
          'Strawberry___healthy': 27,
          'Tomato___Bacterial_spot': 28,
          'Tomato___Early_blight': 29,
          'Tomato___Late_blight': 30,
          'Tomato___Leaf_Mold': 31,
          'Tomato___Septoria_leaf_spot': 32,
          'Tomato___Spider_mites Two-spotted_spider_mite': 33,
          'Tomato___Target_Spot': 34,
          'Tomato___Tomato_Yellow_Leaf_Curl_Virus': 35,
          'Tomato___Tomato_mosaic_virus': 36,
          'Tomato___healthy': 37}
```

```
In [28]: # Loading model and predict.
         from keras.models import load_model
         model=load_model('crop.h5')

         Classes = ["Apple___Apple_scab",
          "Apple___Black_rot",
          "Apple___Cedar_apple_rust",
          "Apple___healthy",
          "Blueberry___healthy",
          "Cherry_(including_sour)___Powdery_mildew",
          "Cherry_(including_sour)___healthy",
          "Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot",
          "Corn_(maize)___Common_rust_",
          "Corn_(maize)___Northern_Leaf_Blight",
          "Corn_(maize)___healthy",
          "Grape___Black_rot",
          "Grape___Esca_(Black_Measles)",
          "Grape___Leaf_blight_(Isariopsis_Leaf_Spot)",
          "Grape___healthy",
          "Orange___Haunglongbing_(Citrus_greening)",
          "Peach___Bacterial_spot",
          "Peach___healthy",
          "Pepper,_bell___Bacterial_spot",
          "Pepper,_bell___healthy",
          "Potato___Early_blight",
          "Potato___Late_blight",
          "Potato___healthy",
          "Raspberry___healthy",
          "Soybean___healthy",
          "Squash___Powdery_mildew",
          "Strawberry___Leaf_scorch",
          "Strawberry___healthy",
          "Tomato___Bacterial_spot",
          "Tomato___Early_blight",
          "Tomato___Late_blight",
          "Tomato___Leaf_Mold",
          "Tomato___Septoria_leaf_spot",
          "Tomato___Spider_mites Two-spotted_spider_mite",
          "Tomato___Target_Spot",
          "Tomato___Tomato_Yellow_Leaf_Curl_Virus",
          "Tomato___Tomato_mosaic_virus",
          "Tomato___healthy"]
```

```
In [31]: import numpy as np
         import matplotlib.pyplot as plt

         # Pre-Processing test data same as train data.
         img_width=256
         img_height=256
         #model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

         from keras.preprocessing import image

         def prepare(img_path):
             img = image.load_img(img_path, target_size=(256, 256))
             x = image.img_to_array(img)
             x = x/255
             return np.expand_dims(x, axis=0)


         result = model.predict_classes([prepare('/Users/ASUS/test/test/AppleScab1.JPG')])
         disease=image.load_img('/Users/ASUS/test/test/AppleScab1.JPG')
         plt.imshow(disease)
         print (Classes[int(result)])
```

```
Apple___Apple_scab
```



Limitation and Future Enhancement

On a broader perspective we plan on making this project an android application that can be used by people who actually need it. Right now, at this level this project only works on a small dataset where you have to manually enter your input, this is a limitation and this is also our doorway to future enhancements. We plan on developing an android application for which this model will play a big and important role.

Conclusion

Plant diseases are the main nutritional threat, and these threats must be overcome before leading to further loss of the entire field. However, farmers often cannot distinguish similar symptoms of different diseases that may be misleading or cause excessive fertilizer. A hierarchical convolutional neural network (CNN) called a deep learning algorithm is used to reduce this loss. This can be done through this mobile app "not all farmers, but some people use them".

References

[1] Evolutionary Artificial Neural Networks in Neutron Spectrology, Jose Mauel Ortiz-Rodriguez, Ma. Del Rosario Martinez-Blanco and Hector Rene Vega-Carrillo.

[2]S.Bashir,N.Sharma,Remote Area Plant disease detection using Image Processing, IOSR Journal of Electronics and Communication Engineering , Volume 2, Issue 6 ,pp.31-34,2012.

[3]J.Behmann,A.K.Mahlein,T.Rumpf,C.Romer,L.Plumer, A review of advanced machine learning methods for the detection of biotic stress in precision crop protection, Springer Media New York, Precision Agriculture,Volume 16, Issue 3, pp. 239–260, 2015.

[4]J.G.A.Barbedo,Digital image processing techniques for detecting, quantifying and classifying plant diseases, ArnalBarbedoSpringerPlus , pp.1- 12,2013.

[5] F.Qin, D.Liu, B.Sun, L.Ruan, Z.Ma, H.Wang,Identification of Alfalfa Leaf Diseases Using ImageRecognitionTechnology,Plos Journals, pp.1-15, 2016.

[6]A.Meunkaewjinda, P.Kumsawat and K.Attakitmongcol,Grape leaf disease detection from color imagery using hybrid intelligent system, 5th International Conference on Electrical Engineering/Electronics,Computer,Telecommunications and Information Technology, Volume: 1,pp. 513 - 516,2008.

[7]E.Omrani, B.Khoshnevisan, S.Shamshirband, H.Saboohi, N.B.Anuar, M.H.N.M.Nasir, 'Potential of radial basis function-based support vector regression for apple disease detection', Department of Biosystem Engineering, pp.2-19,2014.

[8]A.Singh,B.G.Subramanian,A.K.Singh,S.Sarkar, Machine Learning for High-Throughput Stress Phenotyping in Plants,Trends in Plant Science, Volume 21, Issue 2, pp. 110-124, 2016.

[9]V .Singh,ı segmentationı 49,2017.ı [10]A.Camargo, J.S.Smith, An image-processing based algorithm to automatically identify plant disease visual symptoms,Biosystems

Engineering ,Volume 102, Issue 1, pp. 9-21, 2009.

[10] Leaf Disease detection of the cotton plants using image processing techniques, Pranita P. Gulve, Sharayu S. Tambe, published in 2015
[11] An Investigation Into Machine Learning Regression Techniques for the Leaf Rust Disease Detection Using Hyperspectral Measurement, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9, 2016.

[12]Chen CH, Huang WT, Tan TH, Chang CC, Chang YJ, Using K nearest neighbor classification to detect abnormal Lung Sound, Sensors (base).

[13]M.Jhuria, A.Kumar, R.Borse, Image processing forSmart Farming : Detection of disease and food grading, Institute of Electrical and Electronics Engineering, 2013.ı [14] N. Bhardwaj, G. Kaur, P.K.Singh, A Systematic Review on Image Enhancement Techniques, Sensors and Image Processing, Advances in Intelligent Systems and Computing (AISC).

[15] G. Kaur, N. Bhardwaj, P.K.Singh, An Analytic Review on Image Enhancement Techniques Based on Soft Computing Approach, Sensors and Image Processing, Advances in Intelligent Systems and Computing(AISC), Springer, pp. 255-265, 2018.

[16] K. Vasudeva, P.K. Singh,Y. Singh, A Methodical Review on Issues of Medical Image Management System with Watermarking Approach, Indian Journal of Science and Technology, Vol 9(32), DOI: 10.17485/ijst/2016/v9i32/100188, ISSN: 0974-5645, 2016.

[17] D. Agarwal, A. Gupta, P. K.Singh, A systematic review on Artificial Bee Colony Optimization Technique, International Journal of Control Theory and Application, Vol. 9(11), pp. 5487-5500, 2016.

[18] A. Sharma, P.K. Singh, P. Khurana, Analytical Review on Object Segmentation and Recognition, published in proceedings of 6thInternational Conference - Cloud System and Big Data Engineering (Confluence), 14th - 15thJanuary, 2016, Noida, India, IEEE, pp. 524 – 530,2016.

[19] R. Bhardwaj, P .K. Singh, Analytical Review on Human Activity Recognition in Video, published in proceedings of 6th International Conference - Cloud System and Big Data Engineering (Confluence), 14th-15thJanuary, 2016, Noida, India, IEEE, pp. 531 – 536,2016.

[20]S.Arivazhagan,R.N.Shebiah, S.Ananthi,S.V.Varthini ,Using texture detection features, recognizing unhealthy region of plant leaves and classification of plant leaf disease , AgricEngInt: CIGR Journal, Vol. 15, No.1,pp.211-217,2013.