Plant Disease Detection Using Different Algorithms
(Major Project)

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Saksham Varma (181485)

Under the supervision of

Dr. Vipul Sharma

to

ACKNOWLEDGE

Firstly, We would like to express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully

We are grateful and wish our profound indebtedness to Dr Vivek Sehgal, Professor & HOD of the Department of CSE & IT, Jaypee University of Information Technology, Waknaghat.

Deep Knowledge & keen interest of my supervisor in the field of “Deep Learning and AI” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win.

In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Regards,

Saksham Varma 181485
IT | JUIT

Table of content :

1. Chapter 01 Introduction
   - 1.2 Problem Statement

2. Chapter 02 Literature Survey
3. Chapter 03 System Development

   Image Acquisition
   Image Pre Processing
   Feature Extraction
   -3.1 Development
   -3.2 Frontend
   -3.3 Dataset

4. Chapter 04 Performance Analysis

   4.1Accury on Machine Learning
   4.2 Accuracy on Deep Learning

5. Chapter 05 Conclusion

   References
   Appendices

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Plant Detection Using Different Algorithms"** is in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my work carried out over a period from January 2022 to June 2021 under the supervision of Dr Vivek Kumar Sehgal Professor and Head of Department CSE.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Saksham Varma 181485)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Dr. Vipul Sharma
Designation: Assistant Professor (Grade-II)
Department Name: Computer Science & Engineering and Information Technology
Dated: 17-05-2022

List of Abbreviations:

AI - Artificial intelligence

ML - Machine Learning

GPVS - General Purpose Vision System

GPU - Graphical processing unit

List of Figures

Fig 1:Basic Block diagram of Image Processing

Fig 2: Front End for the web application

Fig 3:Classification of the figures

Fig 4:Classification of the figures

Fig 5:Different spot diseases found in vegetation

Fig 6:Resnet50

Fig 7:VGG16

Fig 8:VGG19

Fig 9:Inception V3

Fig 10:DenseNet121

Fig 11:Total Classification of dataset

Fig 12:Final accuracy

Chapter 1: Introduction

This research paper describes digital image processing-based techniques for detecting and classifying leaf diseases in a variety of agricultural plants. This helps in the development of various disease management measures that benefit agriculture. Automatic disease diagnosis and analysis is based on that particular symptom, and cost concentration is very beneficial to farmers. Early detection of illness is a serious concern in agricultural research. Plant diseases are caused by organisms such as fungi, bacteria, and viruses, so it is important to improve the appropriate approach in a particular area. All of these studies are aimed at detecting and classifying plant lesions as soon as possible.

This computerised reasoning condition to a great extent relies upon the capacities of AI for empowering innovative arrangements with intellectual characteristics. The essential objective of computerised reasoning spotlights on decreasing human mistakes while guaranteeing quicker tasks. Hence, you can plainly see how both AI and blockchain expect to make processes quicker. Utilising the two of them together certainly presents some intriguing possibilities for growing the utilizations of blockchain across different areas.

Machine Learning has as of late empowered enormous advances in artificial intelligence however these outcomes can be exceptionally unified. The huge datasets required are for the most part restrictive; forecasts are frequently sold on a for each inquiry premise; and distributed models can immediately become out of date without work to procure more information and keep up with them To address centralization in AI, systems to share machine learning models on a public blockchain while keeping the models allowed to use for deduction have been proposed. One model is Decentralised and Collaborative. For instance, Machine learning on Blockchain from Microsoft Research. The work centres around the depiction of a few potential impetus systems to urge members to add information to prepare a model.

Chapter 1.2: Problem Statement

Agriculture is a significant part of the Indian economy. The Indian agriculture sector employs about half of the country's workers. India is known as the world's largest democracy. Pulses, rice, wheat, spices, and spice products are produced in significant quantities. Farmer's economic development depends on the quality of the items they produce, which is dependent on plant growth and development. As a result, in the sphere of agriculture, disease detection in plants plays an important role. a supporting role Plants are particularly susceptible to illnesses that disrupt their growth, which can lead to death. This, in turn, has an impact on the farmer's ecology. Use this method to detect a plant disease at an early stage. It is advantageous to use an automatic illness detection technique. Plant diseases manifest themselves in various areas of the plant, such as the leaves. It takes a long time to manually detect plant illness using leaf photos. As a result, computer approaches must be developed to automate the process of disease identification and categorization using leaf photos.

Plant disease detection is still a work in progress research topic, despite the challenges described in the problem statement. Over the years, various ways have been offered. Using pathogen vectors, a strategy of detecting and distinguishing plant diseases can be achieved in traditional systems. Algorithms for machines This approach has been used to diagnose sugar beet illnesses, with classification accuracy ranging from 65 percent to 90 percent depending on the kind and stage of the disease. Plant disease classification was performed with Kmeans as a clustering algorithm, again employing a leaf-based approach and using ANN as an automatic detection tool. ANN consists of ten hidden layers. With the example of a healthy leaf, the number of outings is 6, which is the number of classes represented by five disorders.

Chapter 2: Literature Survey

Image processing techniques have been used in a variety of fields in recent years, including automation, medicine, and so on. Image processing systems necessitate the use of a camera, a computer, and the necessary software. Image acquisition, pre-processing, segmentation, feature extraction, and classification are all steps in plant disease detection.

On reviewing various research papers. We observed that some of the models mentioned in the research paper were similar as we can see from [1] that the approach was similar to our model but with indifference of efficiency which made the model to calculate.

Many segments showed us that even using the similar models the efficiency couldn't go up the scale to be defined as unique. To compare to that literature we have generated an easy to approach model with classic UI gestures to emphasise our finding and make it easier to work with.

Table: Consisting about various findings from different research report

No.	Reference Titles	Algorithm/Techniques	Significance	Measuring
1	Plant infection detection using image processing	Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks	data augmentation and image annotation technologies	
2	Plant Detection Using Various Algorithms	disease detection model that uses deep-CNNs is proposed by introducing the GoogLeNet Inception structure and Rainbow concatenation.	Detecting accuracy and precission	
3	An open access repository of images on plant health to enable the development of mobile disease diagnostics	crucial role is played by the image processing in detection of plant disease	K-means clustering algorithm and classification technique such as Artificial neural network (feed forward back propagation)	Accuracy and precision
4.				

Chapter 3: System Development

- Deep learning model :

  Using the score we add the model to the blockchain and reward the maker. Here we store the model , layers and hyper parameters used to train the model. For every better score the transaction takes place .

  Here one can use a pre-trained model such as Resnet and customise it to get better accuracy.

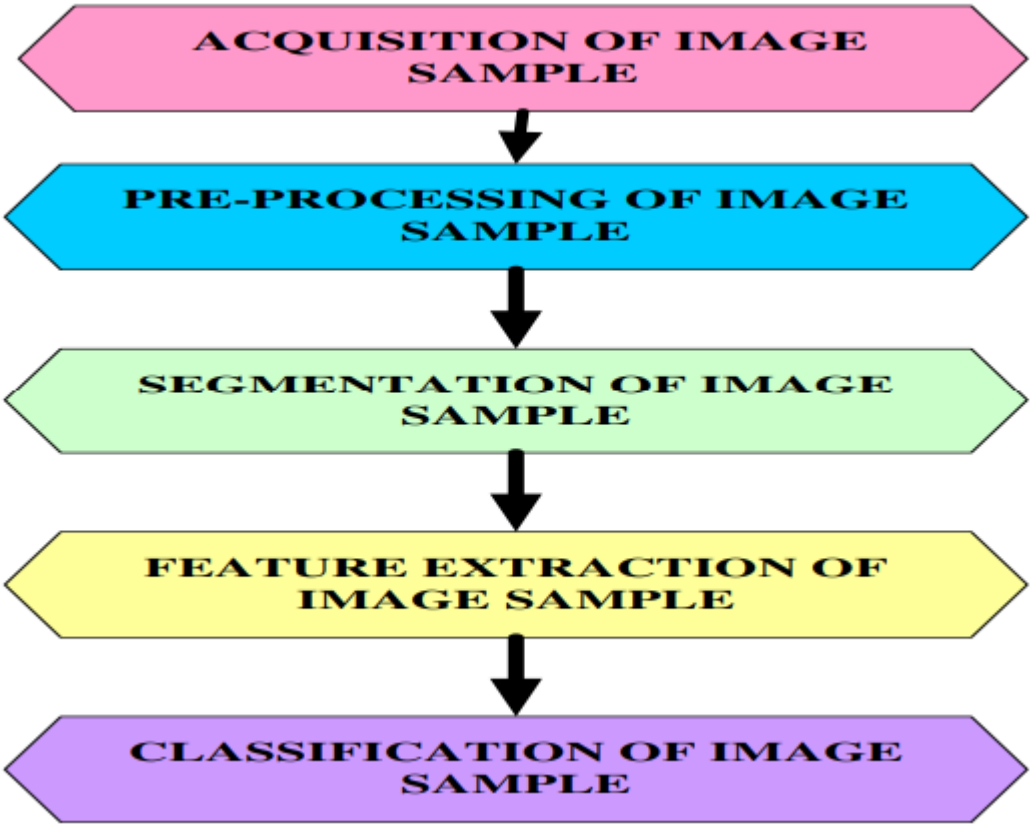


Fig. Basic Block diagram of Image Processing

- **Image Acquisition**

The first step in **acquiring images** is to take **pictures** of the leaves with a smartphone or digital camera. The route is used to load the saved **leaf** images from the database. The **images** of the leaf samples are shown in Figure

- **Image Pre-processing**

  Pre-processing improves image quality by removing unwanted distortions. Here, the image is cropped to the region of interest (ROI), image smoothing and contrast enhancement are performed. Figure 3 shows the photo after editing.

- **Image Segmentation**

  Image segmentation is a technique for separating an image into sub-images. To partition and reassemble the image, we employ the Kmean segmentation technique, which involves colour estimation. We do not consider green leaves because they are common. For feature extraction, we chose a cluster image that shows an affected area. The fractional images of leaves seen below.

- **Feature Extraction**

  Feature extraction refers to the part of the image that is interesting and from which the required information is retrieved. The area of interest (ROI) will be smaller than it was in the original photograph. One of the most effective texture analysis approaches is the grayscale co-occurrence matrix (GLCM). It estimates the image's characteristics using quadratic statistical approaches. GLCM calculates pixels in the image that have a specified intensity where the grey value appears. The total number of occurrences of a pixel of a specific intensity in the spatial domain will be the

outcome. The amount of grey levels will determine the size of the GLCM.

- **Image Classification**

  Fungi, bacteria, and viruses cause diseased leaves. The insect can also harm leaves, resulting in leaf spot disease. Depending on the stage and organism involved, infected leaf portions will vary in size and colour. Spots will be seen in a variety of colours, including yellow, brown, beige, and black. The disease was classified using textural information from the GLCM. The disease is divided into three categories: anthracnose, Cercospora leaf spot, and Bacterial blight.

We used following pre-trained models for simulation and one CNN model acting as Mother Block :

- Resnet50

  ResNet-50 is nothing but a convolutional Neural Network model . ResNet-50 is 50 layers deep. In this you can load a pre trained version of the network which is trained on more than a million images from the ImageNet .(a dataset that has 100,000+ images across 200 different classes)

- VGG16

  It is a type of convolutional neural network model. This model achieves around 92.7% top-5 test accuracy in ImageNet. VGG-16 is 16 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet.

- VGG19

  It is also a convolutional neural network model that is 19 layers deep (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer).

- Resnet34

  It is a convolutional neural network model. Resnet34 is a 34 layer convolutional neural network. Resnet34 can be utilised as a state of the art image classification model. This is a model that has been pre-trained on the ImageNet dataset.

- DenseNet121

  It is a convolutional Neural Network model which connects each layer to every other layer in a feed-forward fashion. DenseNets have various compelling advantages like they can alleviate the vanishing-gradient problem they can also strengthen feature propagation and not only this they also encourage feature reuse and last but not least substantially reduce the number of parameter

3.1 Development

For the application we made a web application using the deep learning model we created and saving the model in the model section in our web application.

With regular auto updation in a server we can always get a new best model in regular intervals from the blockchain .

The web application is made using HTML, CSS , Javascript for frontend and Flask for server side.

I have used fastai which is built on top of Pytorch. Dataset consists of 38 disease classes from PlantVillage dataset and 1 background class from Stanford's open dataset of background images DAGS. 80% of the dataset is used for training and 20% for validation.

3.2 Frontend:

For different users to make it easier for them to work easily and utilise the model more efficiently. The front end was made in a simpler UI. The simple UI will help in conversing with algorithms to opt the desired results.

FRONTEND	
HTML	For basic layout and functions
CSS	For styling and improving UI
JAVASCRIPT	For dynamic styling and recalling attributes

The frontend uses basic knowledge of HTML , Javascript and CSS

- We have created button and applied an animation for the button
- We created and applied icon for the page
- We applied CSS and the interface of the form.
- Created the basic interface of the page using HTML

3.3 Backend:

BACKEND	
JAVASCRIPT	Create forms and linking the frontend to backend
FLASK	Library used for creating the UI

The backend was created using basic knowledge of Javascript and Flask

- We created the synchronisation between Flask (which used the model to provide output) , Javascript (which supported the interface between frontend and backend).
- Created form's backend using javascript and simple error popup system.
- Provided the incoming of image and display of result after going through the model which was trained previously.
- Upload of the image as file.

The following image is a snippet of our front end code which shows the front end of the title page which will then lead to further processing. This was developed using HTML and Javascript and styled using CSS.

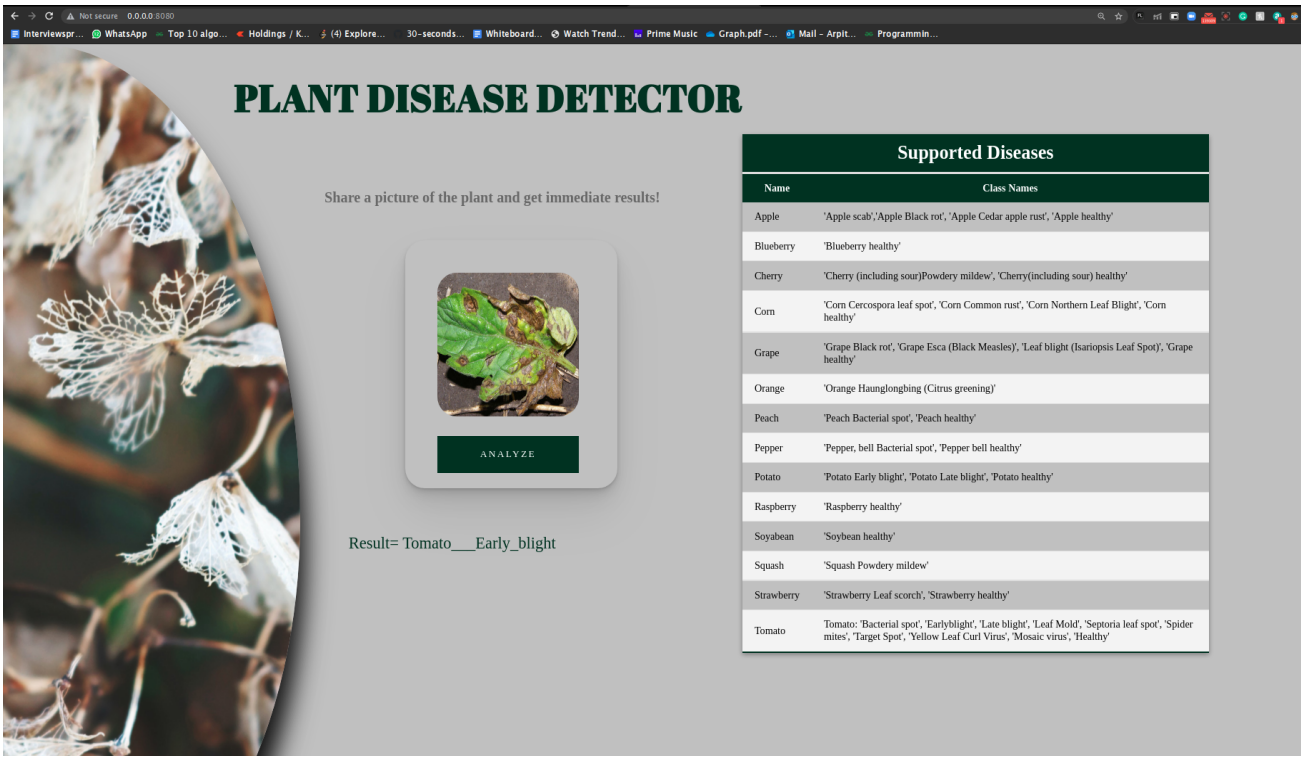


Fig2. Depicting the Front end

3.4 Dataset

Mobile phones or digital cameras are used to take images of infected leaves of different plants. Image processing techniques are applied on those images to get useful features for analysing. The various steps involved are shown in the figures.

The data used in the model for developing our Machine Learning model is a Dataset by SP Mohanty for plant diseases [4] was used . It is a public dataset of 54306 images for 14 crop species and for 26 diseases . The dataset consists of colour ,grayscale and segmented images of the data . The image is resized to 256×256 pixels consisting of only one leaf.

The data records contain 54,309 images. The images span 14 crop species: Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Bell Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, Tomato. In containers images of 17 fungal diseases, 4 bacterial diseases, 2 mould (oomycete) diseases, 2 viral diseases, and 1 disease caused by a mite. 12 crop species also have images of healthy leaves that are not visibly affected by a disease. Table 1 summarises the dataset. The data records are available through the website www.plantvillage.org. A file mapping each the URL of each image to the classification has been deposited at.

Table.: List of crops and their disease status currently in the PlantVillage

Name	Fungi	Bacteria	Mould	Virus	Mite	Healthy
Apple (3172)	Gymnosporangium juniperi virginianae (276) Venturia inaequalis (630) Botryosphaeria obtusa (621)					(1645)
Blueberry						(1502)
Cherry (1996)	Podosphaera spp (1052)					(854)
Corn (3852)	Cercospora zeae maydis (513) Puccinia sorghi (1192) Exserohilum turcicum (985)					(1162)
Grape (4063)	Guignardia bidwellii (1180) Phaeomoniella-- spp. (1384) Pseudocercospor a vitis (1076)					(423)
Orange (5507)		Candidatus Liberibacter (5507)				

Peach (2657)		Xanthomonas campestris (2291)				(360)
Bell Pepper (2475)		Xanthomonas campestris (997)				(1478)
Potato (2152)	Alternaria solani (1000)		Phytophthora Infestans (1000)			(152)
Raspberry (371)						(371)
Soybean (5090)						(5090)
Squash (1835)	Erysiphe cichoracearum / Sphaerotheca fuliginea (1835)					
Strawberry (1565)	Diplocarpon earliana (1109)					(456)
Tomato (18,162)	Alternaria solani (1000) Septoria lycopersici (1771) Corynespora cassiicola (1404) Fulvia fulva (952)	Xanthomonas campestris pv. Vesicatoria (2127)	Phytophthora Infestans (1910)	Tomato Yellow Leaf Curl Virus (5357) Tomato Mosaic Virus (373)	Tetranychus urticae (1676)	(1592)

The data is classified into following categories :

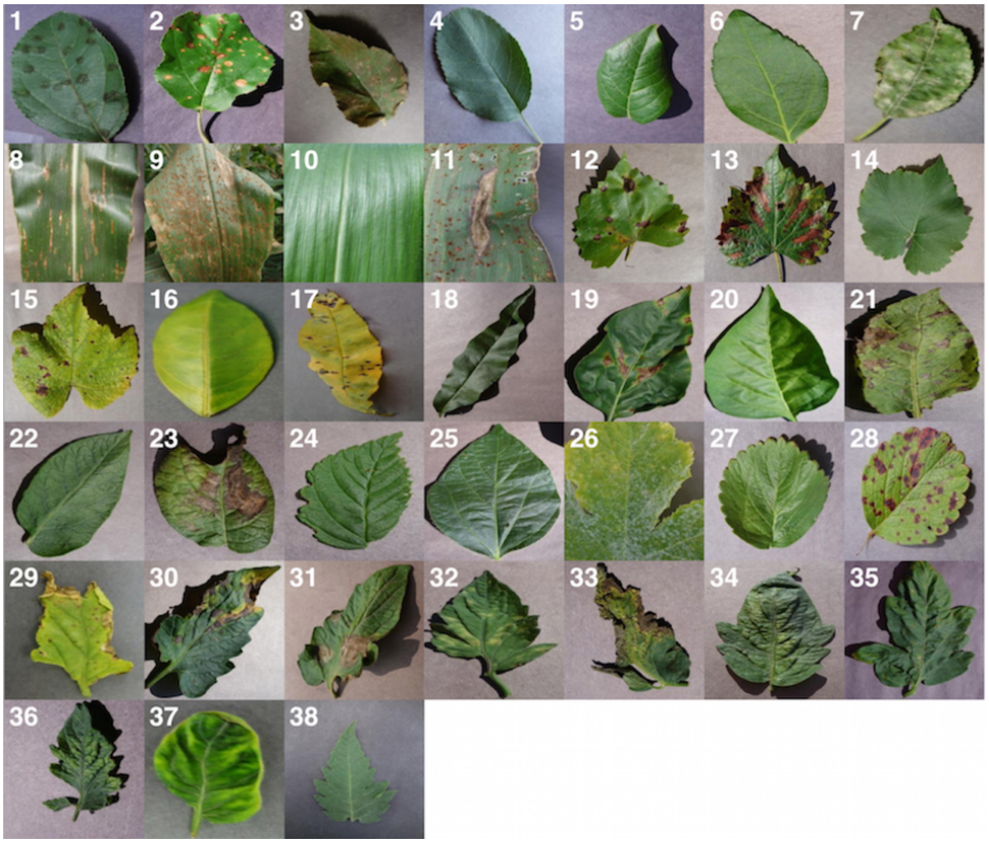


Fig3. Classification of the figures

Which can be sorted according to plant:

Name	Class Names
Apple	'Apple scab','Apple Black rot', 'Apple Cedar apple rust', 'Apple healthy'
Blueberry	'Blueberry healthy'
Cherry	'Cherry (including sour)Powdery mildew', 'Cherry(including sour) healthy'
Corn	'Corn Cercospora leaf spot', 'Corn Common rust', 'Corn Northern Leaf Blight', 'Corn healthy'
Grape	'Grape Black rot', 'Grape Esca (Black Measles)', 'Leaf blight (Isariopsis Leaf Spot)', 'Grape healthy'
Orange	'Orange Haunglongbing (Citrus greening)'
Peach	'Peach Bacterial spot', 'Peach healthy'
Pepper	'Pepper, bell Bacterial spot', 'Pepper bell healthy'
Potato	'Potato Early blight', 'Potato Late blight', 'Potato healthy'
Raspberry	'Raspberry healthy'
Soyabean	'Soybean healthy'
Squash	'Squash Powdery mildew'
Strawberry	'Strawberry Leaf scorch', 'Strawberry healthy'
Tomato	Tomato: 'Bacterial spot', 'Earlyblight', 'Late blight', 'Leaf Mold', 'Septoria leaf spot', 'Spider mites', 'Target Spot', 'Yellow Leaf Curl Virus', 'Mosaic virus', 'Healthy'

Fig5. Different spot diseases found in vegetation

Chapter 4: PERFORMANCE ANALYSIS

4.1 Accuracy For Machine learning :

- Decision Tree Classifier : 92.98%
- Logistic Regression Classifier : 95.9%
- Random Forest Classifier : 96.49%
- Gaussian Naive Bayesian: 92.98%
- Support Vector Classifier : 97.08%
- Nearest Neighbours Classifier: 95.9%

4.2 Accuracy For Deep learning:

- Resnet50 :

  Accuracy = 89.16%

  F1 Score = 0.8845819234848022

  Confusion Matrix =

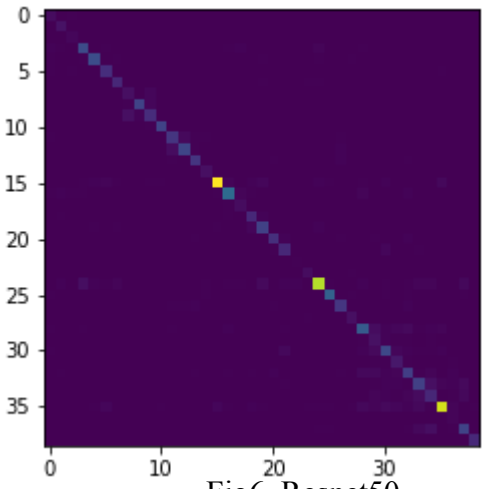


Fig6. Resnet50

- VGG16 :

  Accuracy = 92.61%

  F1 Score = 0.9922231435775757

  Confusion Matrix =

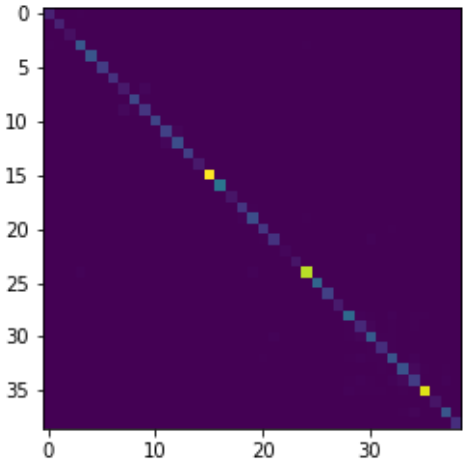


Fig7 VGG16

- VGG19 :

  Accuracy = 92.05%

  F1 Score = 0.9936198592185974

  Confusion matrix =

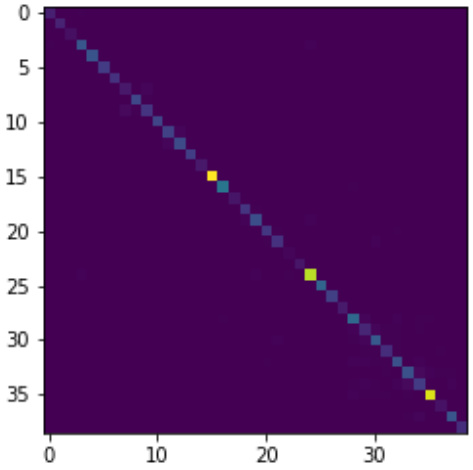


Fig8 VGG19

- InceptionV3 :

  Accuracy = 91.62%

  F1 Score =0.9164

  Confusion matrix=

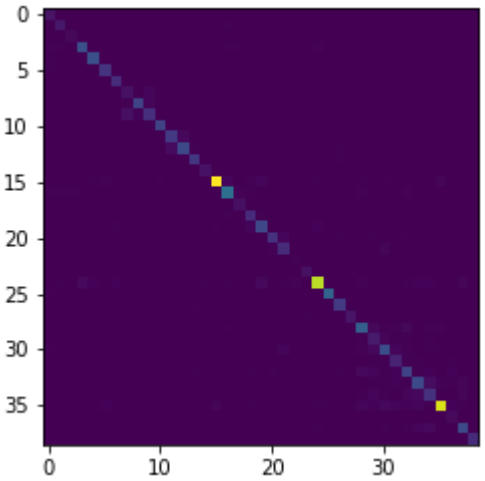


Fig9. Inception V3

- DenseNet121 :

  Accuracy = 91.15%

  F1 Score = 0.9151003360748291

  Confusion Matrix =

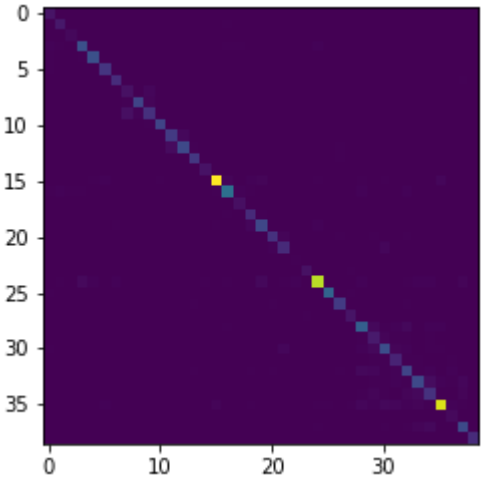


Fig10. DenseNet121

Adding model in blockchain:
Randomly adding best model to blockchain:

- Custom model using LeNet algorithm:

  Accuracy = 94%
  F1 Score =0.94

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 252, 252, 6)	456
average_pooling2d_6 (AveragePooling2D)	(None, 126, 126, 6)	0
conv2d_7 (Conv2D)	(None, 122, 122, 20)	3020
average_pooling2d_7 (AveragePooling2D)	(None, 61, 61, 20)	0
flatten_3 (Flatten)	(None, 74420)	0
dropout_6 (Dropout)	(None, 74420)	0
dense_9 (Dense)	(None, 120)	8930520
dropout_7 (Dropout)	(None, 120)	0
dense_10 (Dense)	(None, 84)	10164
dense_11 (Dense)	(None, 38)	3230

Total params: 8,947,390
Trainable params: 8,947,390
Non-trainable params: 0

Fig11.Total Classification of dataset

	precision	recall	f1-score	support
0	0.86	0.99	0.92	97
1	0.96	0.89	0.92	123
2	1.00	1.00	1.00	213
3	0.95	0.90	0.92	197
4	0.91	0.97	0.94	220
5	0.99	0.98	0.98	161
6	0.91	0.91	0.91	67
7	1.00	0.95	0.97	893
8	0.96	0.97	0.96	382
9	0.86	0.91	0.89	70
10	0.85	0.96	0.90	139
11	0.90	0.93	0.91	233
12	0.72	1.00	0.84	28
13	0.96	0.84	0.90	174
14	0.85	0.88	0.86	161
15	0.84	0.95	0.89	22
16	0.91	1.00	0.95	40
17	0.99	0.96	0.97	913
18	0.98	0.99	0.98	267
19	0.89	0.99	0.94	168
20	0.96	0.95	0.96	85
21	0.94	0.95	0.94	294
22	0.85	0.84	0.85	148
23	0.88	0.95	0.91	265
24	0.92	0.86	0.89	311
25	0.81	0.94	0.87	125
26	0.88	0.94	0.91	256
27	0.93	0.89	0.91	315
28	0.90	0.86	0.88	222
29	0.98	0.96	0.97	922
30	0.78	0.93	0.85	55
31	0.92	0.93	0.93	223
32	0.88	0.99	0.93	214
33	0.96	0.92	0.94	190
34	0.95	0.92	0.94	137
35	0.95	0.89	0.92	89
36	1.00	0.99	1.00	200
37	0.94	0.94	0.94	132
accuracy			0.94	8751
macro avg	0.91	0.94	0.92	8751
weighted avg	0.94	0.94	0.94	8751

Fig12. Final accuracy

Chapter 5: CONCLUSION

This documentation provides a clear statement that the main goal of the proposed work is to identify the disease. To identify leaf diseases, various segmentation techniques were employed. The subgrouping and classification of foliar diseases was developed using image thresholds, K-means clustering, and neural networks (NN). Different algorithms have been tested on the various effects of plant diseases. With experimental results significantly supporting accurate results in less than computation time, this is the neural network that provides the best accuracy results when compared to other networks. The model was able to confirm the disease's identity. The diagnosis is straightforward. Diseases were found in some cases in key plots maintained by experimental research stations to identify disease presence in a given area. The specialist made the diagnosis in these cases once more. We keep all images in PlantVillage's database using expert diagnostics. The database only contains professionally defined sheets.

Our model made use of various algorithms which eventually gave us satisfactory results through its efficiency. This efficiency was achieved by testing and training our model which was done through the database we found. The model made use of several different models such as Resnet and VGGetc. These models gave us an outstanding efficiency of 94%. This will help many agriculturist and scientist in their research work as it will be free to use and will be made even simpler in coming time by using various algorithms.

Reference

[1]Zhou, Jianlong; Chen, Fang (2018). *[Human–Computer Interaction Series] Human and Machine Learning || Deep Learning for Plant Diseases: Detection and Saliency Map Visualisation. , 10.1007/978-3-319-90403-0(Chapter 6), 93–117.* doi:10.1007/978-3-319-90403-0_6

[2] Mohanty, Sharada P.; Hughes, David P.; Salathé, Marcel (2016). Using Deep Learning for Image-Based Plant Disease Detection. Frontiers in Plant Science, 7(), 1419–. doi:10.3389/fpls.2016.01419

[4] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." Frontiers in plant science 7 (2016): 1419.

[5]Miller, Sally A.; Beed, Fen D.; Harmon, Carrie Lapaire (2009). *Plant Disease Diagnostic Capabilities and Networks. Annual Review of Phytopathology, 47(1), 15–38.* doi:10.1146/annurev-phyto-080508-081743

Appendix

Referred data :
Plant disease detection by SP Mohanty .
Breast cancer detection

Referred chart:

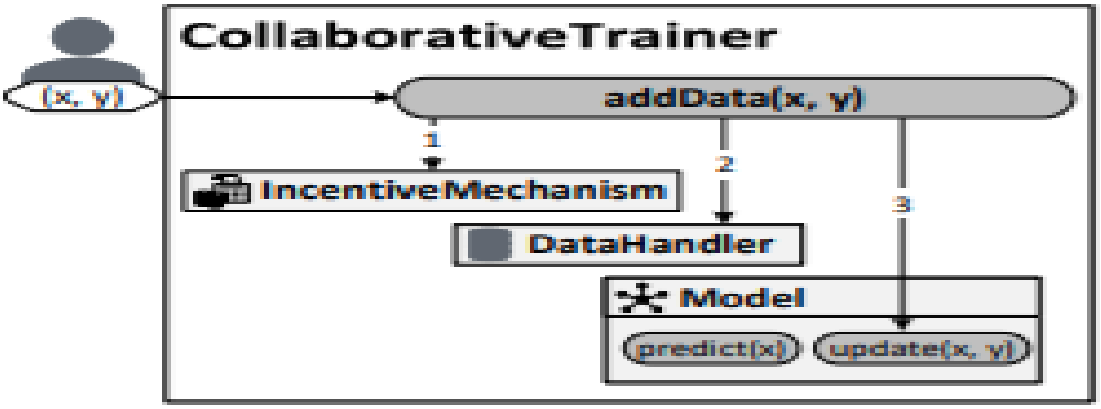


Fig.9

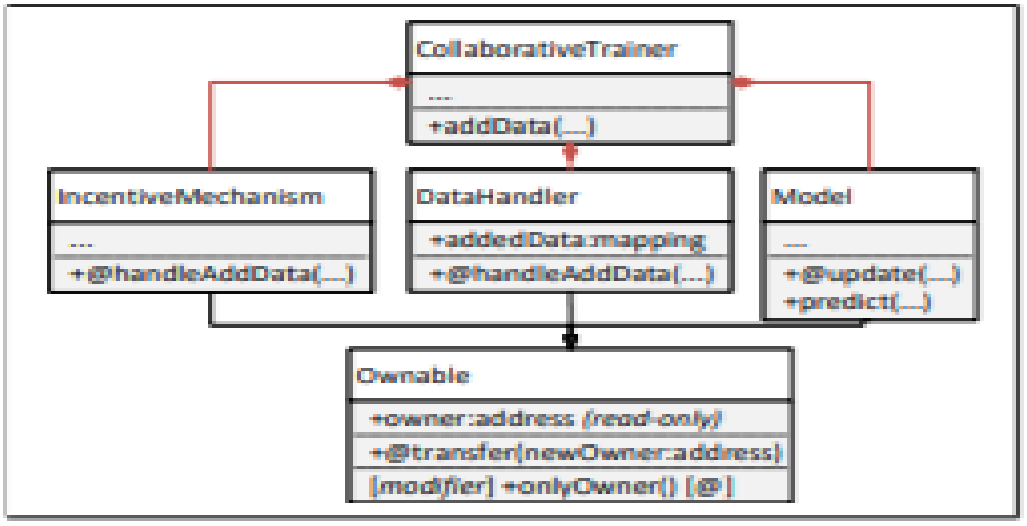


Fig. 10

CODE SNIPPETS

index.html

```
                <tbody>
                    <tr>
                        <td>Apple</td>
                    <td>'Apple scab','Apple Black rot', 'Apple Cedar apple rust', 'Apple healthy'</td>
                    </tr>
                    <tr>
                        <td>Blueberry</td>
                    <td>'Blueberry healthy'</td>
                    </tr>
                    <tr>
                        <td>Cherry</td>
                    <td>'Cherry (including sour)Powdery mildew', 'Cherry(including sour) healthy'</td>
                    </tr>
                    <tr>
                        <td>Corn</td>
                    <td>'Corn Cercospora leaf spot', 'Corn Common rust', 'Corn Northern Leaf Blight', 'Corn healthy'</td>
                    </tr>
                    <tr>
                        <td>Grape</td>
                    <td>'Grape Black rot', 'Grape Esca (Black Measles)', 'Leaf blight (Isariopsis Leaf Spot)', 'Grape healthy'</td>
                    </tr>
                    <tr>
                        <td>Orange</td>
                    <td>'Orange Haunglongbing (Citrus greening)'</td>
                    </tr>
                    <tr>
                        <td>Peach</td>
                    <td>'Peach Bacterial spot', 'Peach healthy'</td>
                    </tr>
                    <tr>
                        <td>Pepper</td>
                    <td>'Pepper, bell Bacterial spot', 'Pepper bell healthy'</td>
                    </tr>
                    <tr>
                        <td>Potato</td>
                    <td>'Potato Early blight', 'Potato Late blight', 'Potato healthy'</td>
                    </tr>
                    <tr>
                        <td>Raspberry</td>
                    <td>'Raspberry healthy'</td>
                    </tr>
                    <tr>
                        <td>Soyabean</td>
                    <td>'Soybean healthy'</td>
                    </tr>
                    <tr>
                        <td>Squash</td>
                    <td>'Squash Powdery mildew'</td>
                    </tr>
                    <tr>
                        <td>Strawberry</td>
                    <td>'Strawberry Leaf scorch', 'Strawberry healthy'</td>
                    </tr>
                    <tr>
                        <td>Tomato</td>
```

Line 126, Column 8 — Tab Size: 4 — HTML

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Plant Disease Detector</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://fonts.googleapis.com/css?family=Merriweather&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Abril+Fatface&display=swap" rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.5.0.min.js"
        integrity="sha256-xNzN2a4ltkB44Mc/Jz3pT4iU1cmeR0FkXs4pru/JxaQ=" crossorigin="anonymous"></script>
    <!-- STYLE CSS -->
    <link rel="stylesheet" href="../static/style.css">
    <link rel="icon" href="../static/leaf.png" type="image/x-icon">
</head>

<body>

                    <path
                        d="M115.0,115.0 C114.9,115.1 118.7,116.5 119.8,115.4 L133.7,101.6 C136.9,99.2 139.9,98.4 142.2,98.6 C133.8,88.0 127.5,74.4 143.8,58.0 C148.5,53.4 154.0,51.2 159.7,51.0 C160.3,49.4 163.2,43.6
                        171.4,40.1 C171.4,40.1 176.1,42.5 178.8,56.2 C183.1,58.6 187.2,61.8 190.9,65.4 C194.5,69.0 197.7,73.2 200.1,77.6 C213.8,80.2 216.3,84.9 216.3,84.9 C212.7,93.1 206.9,96.0 205.4,96.6 C205.1,102.4
                        203.0,107.8 198.3,112.5 C181.9,128.9 168.3,122.5 157.7,114.1 C157.9,116.9 156.7,120.9 152.7,124.9 L141.0,136.5 C139.8,137.7 141.6,141.9 141.8,141.8 Z"
                        fill="currentColor" class="octo-body"></path>
                </svg>
            </a>
    </div>
    <div class="container"><img src="../static/bgd.jpg" id="right"></div>
    <!-- <div class="wrapper"> -->
    <div class="heading">
        PLANT DISEASE DETECTOR</div>
    <div class="text">
        <p><b>Share a picture of the plant and get immediate results!</b></p>
    </div>
    <!-- </div> -->
    <div class="inner">
        <form id="analysis-form" enctype="multipart/form-data">
            <div class="imgbox"><img src="../static/img.png" class="img" id="selected_image">
                <div class="middle">
                    <div class="add"><span>SELECT IMAGE</span></div>
                    <input name="Select File" id="imagefile" type="file" />
                </div>
            </div>
            <button id="analyze-button" class="analyze-button">
                <span>Analyze</span>
            </button>
        </form>
        <div id="result"><b>Result : </b></div>
    </div>
    <div >
        <img src="../static/left.png" class="left">
    </div>

    <div>
        <table class="styled-table">
            <thead>
```

126, Column 8 — Tab Size: 4 — HTML

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

In [2]:
```
from fastai import *
from fastai.vision import *
from fastai.metrics import error_rate, accuracy
import gdown
```

In [3]:
```
PATH_IMG = Path('PlantVillage/')
```

In [5]:
```
bs = 64 # batch_size
```

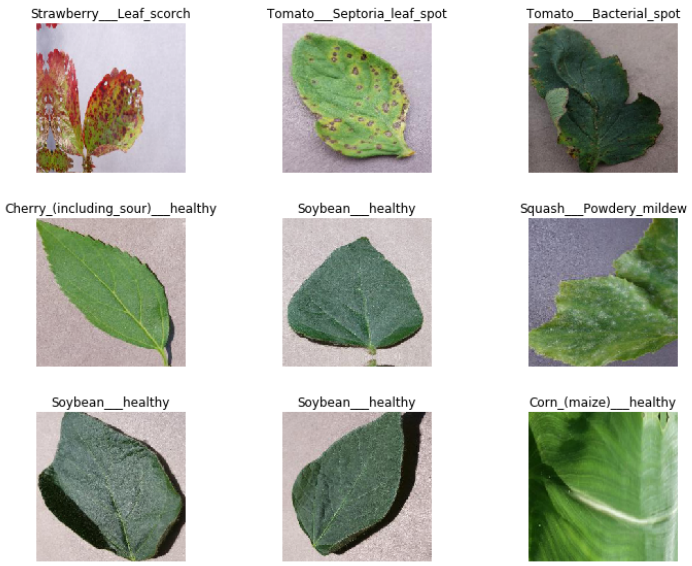
In [25]:
```
img_data = ImageDataBunch.from_folder(path=PATH_IMG, train='train', valid='val', ds_tfms=get_transforms(), size=224, bs=bs)
```

In [26]:
```
img_data.normalize(imagenet_stats)
```

Out[26]:
```
ImageDataBunch;

Train: LabelList (44016 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_
Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus
Path: PlantVillage;

Valid: LabelList (11004 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_
Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus
Path: PlantVillage;

Test: None
```

In [27]:
```
img_data.show_batch(rows=3, figsize=(10,8))
```

In [27]:
```
img_data.show_batch(rows=3, figsize=(10,8))
```



In [28]:
```
img_data.c
```

Out[28]:
```
39
```

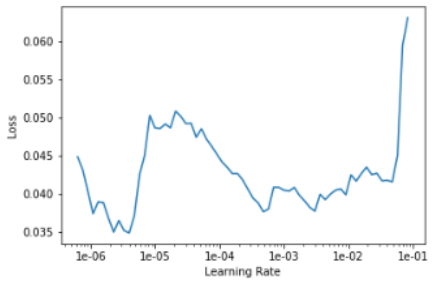
In [29]:
```
img_data.classes
```

Out[29]:
```
['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',
 'background']
```

In [38]:
```
model.lr_find()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [39]:
```
model.recorder.plot()
```



In [41]:
```
model.unfreeze()
model.fit_one_cycle(3, max_lr=slice(1e-03, 1e-02))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.347306	0.843974	0.800527	0.199473	04:36
1	0.131069	0.062115	0.978644	0.021356	04:37
2	0.038473	0.020997	0.993457	0.006543	04:37

In [43]:
```
model.fit_one_cycle(5, max_lr=slice(1e-03, 1e-02))
```

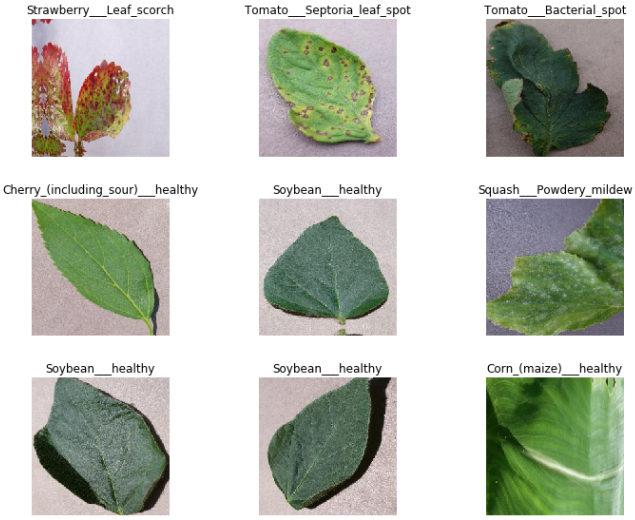
epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.124815	0.142209	0.957197	0.042803	04:37
1	0.144097	0.150233	0.959924	0.040076	04:37
2	0.093149	0.058316	0.981461	0.018539	04:36
3	0.041538	0.019202	0.995093	0.004907	04:36
4	0.018872	0.013773	0.996365	0.003635	04:36

In [44]:
```
model.save('train_lr_8_cycles')
```

In [45]:
```
model.freeze()
model.lr_find()
model.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [27]:
```
img_data.show_batch(rows=3, figsize=(10,8))
```



In [28]:
```
img_data.c
```

Out[28]:
```
39
```

In [0]:
```
!pip install -q pandas
# http://pytorch.org/
from os.path import exists
from wheel.pep425tags import get_abbr_impl, get_impl_ver, get_abi_tag

!pip install -q Pillow==4.3.0
!pip install -q PIL
!pip install -q image
import PIL

#%reload_ext autoreload        <------------- comment out
#%autoreload 0                <------------- comment out
%matplotlib inline

!pip install --no-cache-dir -I pillow

def register_extension(id, extension):
    PIL.Image.EXTENSION[extension.lower()] = id.upper()
PIL.Image.register_extension = register_extension
def register_extensions(id, extensions):
    for extension in extensions:
        register_extension(id, extension)
PIL.Image.register_extensions = register_extensions

from PIL import Image
def register_extension(id, extension): Image.EXTENSION[extension.lower()] = id.upper()
Image.register_extension = register_extension
def register_extensions(id, extensions):
  for extension in extensions: register_extension(id, extension)
Image.register_extensions = register_extensions
```

```
     |████████████████████████████████| 5.8MB 2.5MB/s
     |████████████████████████████████| 112kB 42.7MB/s
  Building wheel for olefile (setup.py) ... done
ERROR: Could not find a version that satisfies the requirement PIL (from versions: none)
ERROR: No matching distribution found for PIL
Collecting pillow
  Downloading https://files.pythonhosted.org/packages/ba/90/8a24e6220cfcf6a3a0162535d5b926e774117e384ff921908e07e4c92bda/Pillow-7.1.1-cp36-cp36m-manylinux1_x86_64.whl (2.1MB)
     |████████████████████████████████| 2.1MB 2.8MB/s
```

In [0]:

```
!apt-get install -y -qq software-properties-common python-software-properties module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}

!mkdir -p drive
!google-drive-ocamlfuse drive

import os
#os.listdir('/content/drive/AMD/CNV')
#trainDir = '/content/drive/My Drive/AMD/CNV'

from google.colab import drive
drive.mount('/content/drive')
data = "/content/drive/My Drive/Plant Disease/PlantVillage"
```

In [0]:

```
import torch
import helper
from torch import nn
from torch import optim
import torch.nn.functional as F
from torchvision import datasets, transforms, models
import torchvision.models as models

from torchvision import datasets ,transforms


#Changning the transform of the data-
transform_train = transforms.Compose([transforms.RandomHorizontalFlip(),
                                      transforms.RandomResizedCrop(224),
                                      # transforms.CenterCrop(224),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
                                      ])

transform_test = transforms.Compose([transforms.RandomHorizontalFlip(),
                                      transforms.RandomResizedCrop(224),
                                      # transforms.CenterCrop(224),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
                                      ])

# choose the training and test datasets
train_data = datasets.ImageFolder(data+"/train", transform=transform_train)
test_data = datasets.ImageFolder(data+"/val", transform = transform_test)
#n_classes = test_data.shape[1]
n_classes = len(test_data.classes)
print(n_classes)

batch_size = 16

dataloader_train = torch.utils.data.DataLoader(train_data, batch_size, shuffle=True, num_workers=2)
dataloader_test = torch.utils.data.DataLoader(test_data, batch_size, num_workers=2)
```

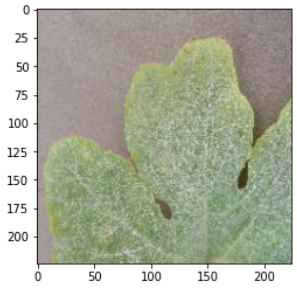
In [0]:

```
import matplotlib.pyplot as plt
import numpy as np

norm_mean = [0.485, 0.456, 0.406]
norm_std = [0.229, 0.224, 0.225]

def imshow_numpy(image, ax=None, title=None):
    if ax is None:
        fig, ax = plt.subplots()

    ax.grid(False)

    # PyTorch tensors assume the color channel is the first dimension
    # but matplotlib assumes is the third dimension
    image = image.transpose((1, 2, 0))

    # Undo preprocessing
    mean = np.array(norm_mean)
    std = np.array(norm_std)
    image = std * image + mean

    # Image needs to be clipped between 0 and 1 or it looks like noise when displayed
    image = np.clip(image, 0, 1)

    ax.imshow(image)

    return ax
```

In [0]:

```
images, labels = next(iter(dataloader_train))
imshow_numpy(images[0].numpy())
print(images.shape)
```

```
torch.Size([16, 3, 224, 224])
```



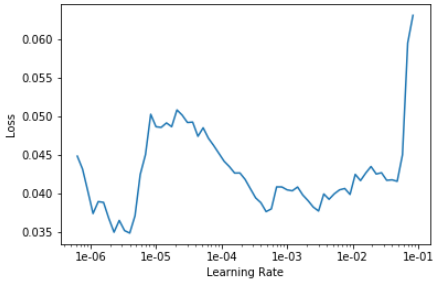
```
      (denselayer11): _DenseLayer(
        (norm1): BatchNorm2d(832, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(832, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer12): _DenseLayer(
        (norm1): BatchNorm2d(864, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(864, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer13): _DenseLayer(
        (norm1): BatchNorm2d(896, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(896, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer14): _DenseLayer(
        (norm1): BatchNorm2d(928, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(928, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer15): _DenseLayer(
        (norm1): BatchNorm2d(960, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(960, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
      (denselayer16): _DenseLayer(
        (norm1): BatchNorm2d(992, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu1): ReLU(inplace=True)
        (conv1): Conv2d(992, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (norm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu2): ReLU(inplace=True)
        (conv2): Conv2d(128, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      )
    )
    (norm5): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (classifier): Linear(in_features=1024, out_features=1000, bias=True)
)
```

```
Epoch: 1        Training Loss: 2.070873         Validation Loss: 0.951872
Validation loss decreased (inf --> 0.951872).  Saving model ...
tensor([[ 13.,   3.,   5.,  ...,   0.,   2.,   0.],
        [  1.,   9.,   1.,  ...,   0.,   0.,   0.],
        [  0.,   0.,   0.,  ...,   0.,   0.,   0.],
        ...,
        [  0.,   0.,   0.,  ...,   0.,   0.,   0.],
        [ 12.,  11.,   0.,  ...,  37., 532.,   3.],
        [  5.,   1.,   0.,  ...,   0.,   3., 363.]])
Class 0
TP 13.0, TN 54628.0, FP 24.0, FN 623.0
Recall = 0.020440252497792244
Precision = 0.3513513505458832
F1 Score = 0.038632988929748535
Class 1
TP 9.0, TN 54654.0, FP 9.0, FN 616.0
Recall = 0.014399999752640724
Precision = 0.5
F1 Score = 0.02799377776682377
Class 2
TP 0.0, TN 55007.0, FP 0.0, FN 281.0
Recall = 0.0
Precision = nan
F1 Score = nan
Class 3
TP 712.0, TN 53104.0, FP 538.0, FN 934.0
Recall = 0.43256378173828125
Precision = 0.569599986076355
F1 Score = 0.4917127192020416
Class 4
TP 773.0, TN 53284.0, FP 502.0, FN 729.0
Recall = 0.5146471261978149
Precision = 0.6062744855880737
F1 Score = 0.5567158460617065
Class 5
TP 300.0, TN 54085.0, FP 149.0, FN 754.0
Recall = 0.2846299707889557
Precision = 0.6681514382362366
F1 Score = 0.39920157194137573
Class 6
TP 247.0, TN 54276.0, FP 157.0, FN 608.0
Recall = 0.2888889014720917
Precision = 0.6113861203193665
F1 Score = 0.3923749029636383
Class 7
TP 15.0, TN 54744.0, FP 18.0, FN 511.0
Recall = 0.028517110273241997
Precision = 0.4545454680919647
F1 Score = 0.05366726592183113
Class 8
TP 819.0, TN 53644.0, FP 429.0, FN 396.0
Recall = 0.6740740537643433
```

In [38]:
```
model.lr_find()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

In [39]:
```
model.recorder.plot()
```



In [41]:
```
model.unfreeze()
model.fit_one_cycle(3, max_lr=slice(1e-03, 1e-02))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.347306	0.843974	0.800527	0.199473	04:36
1	0.131069	0.062115	0.978644	0.021356	04:37
2	0.038473	0.020997	0.993457	0.006543	04:37

In [43]:
```
model.fit_one_cycle(5, max_lr=slice(1e-03, 1e-02))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.124815	0.142209	0.957197	0.042803	04:37
1	0.144097	0.150233	0.959924	0.040076	04:37
2	0.093149	0.058316	0.981461	0.018539	04:36
3	0.041538	0.019202	0.995093	0.004907	04:36

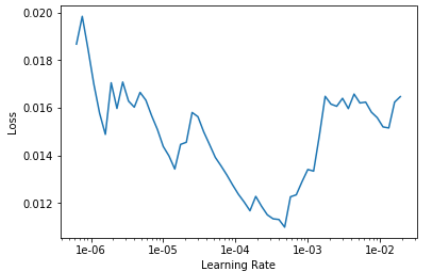
In [43]:
```
model.fit_one_cycle(5, max_lr=slice(1e-03, 1e-02))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.124815	0.142209	0.957197	0.042803	04:37
1	0.144097	0.150233	0.959924	0.040076	04:37
2	0.093149	0.058316	0.981461	0.018539	04:36
3	0.041538	0.019202	0.995093	0.004907	04:36
4	0.018872	0.013773	0.996365	0.003635	04:36

In [44]:
```
model.save('train_lr_8_cycles')
```

In [45]:
```
model.freeze()
model.lr_find()
model.recorder.plot()
```

```
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```



In [47]:
```
lr = 1e-3/2
model.fit_one_cycle(2, slice(lr))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.018528	0.014205	0.996274	0.003726	03:25
1	0.016184	0.013566	0.996547	0.003453	03:26

In [48]:
```
model.fit_one_cycle(3, slice(lr))
```

In [48]:
```
model.fit_one_cycle(3, slice(lr))
```

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.020793	0.013504	0.996092	0.003908	03:26
1	0.021284	0.013225	0.996547	0.003453	03:27
2	0.011413	0.013450	0.996547	0.003453	03:27

In [49]:
```
model.save('train_final5_cycles')
```

In [50]:
```
model.load('train_final5_cycles')
```

Out[50]:
```
Learner(data=ImageDataBunch;

Train: LabelList (44016 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_
Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus
Path: PlantVillage;

Valid: LabelList (11004 items)
x: ImageList
Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224),Image (3, 224, 224)
y: CategoryList
Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus,Tomato___Tomato_Yellow_Leaf_Curl_
Virus,Tomato___Tomato_Yellow_Leaf_Curl_Virus
Path: PlantVillage;
```

PLAGUE REPORT

Saksham_Report

ORIGINALITY REPORT

30%	27%	18%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	pdfs.semanticscholar.org Internet Source	7%
2	www.ir.juit.ac.in:8080 Internet Source	3%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	3%
4	www.semanticscholar.org Internet Source	2%
5	www.coursehero.com Internet Source	2%
6	annals-csis.org Internet Source	2%
7	ijarcce.com Internet Source	1%
8	Justin D. Harris. "Chapter 10 Analysis of Models for Decentralized and Collaborative AI on Blockchain", Springer Science and Business Media LLC, 2020 Publication	1%

9	studenten365.com Internet Source	1 %
10	arxiv.org Internet Source	1 %
11	www.irjmets.com Internet Source	1 %
12	Rahul Keru Patil, Suhas Shivlal Patil. "Cognitive Intelligence of Internet of Things in Smart Agriculture Applications", 2020 IEEE Pune Section International Conference (PuneCon), 2020 Publication	1 %
13	drago1234.github.io Internet Source	1 %
14	www.ijraset.com Internet Source	1 %
15	"Intelligent Learning for Computer Vision", Springer Science and Business Media LLC, 2021 Publication	1 %
16	ijsrm.in Internet Source	<1 %
17	reetvolas.com Internet Source	<1 %

18	Singh, Vijai, Varsha, and A K Misra. "Detection of unhealthy region of plant leaves using image processing and genetic algorithm", 2015 International Conference on Advances in Computer Engineering and Applications, 2015. Publication	<1 %
19	www.ijert.org Internet Source	<1 %
20	Halil Durmus, Ece Olcay Gunes, Murvet Kirci. "Disease detection on the leaves of the tomato plants by using deep learning", 2017 6th International Conference on Agro-Geoinformatics, 2017 Publication	<1 %
21	"Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018)", Springer Science and Business Media LLC, 2020 Publication	<1 %
22	R Anand, S Veni, J Aravinth. "An application of image processing techniques for detection of diseases on brinjal leaves using k-means clustering method", 2016 International Conference on Recent Trends in Information Technology (ICRTIT), 2016 Publication	<1 %