A REPORT

ON

PLANTIX LIKE APPLICATION USING DEEP LEARNING AND NEURAL NETWORKS

BY

MEHUL JAIN 2019A3PS1315H

AT

Edutech Learning Solutions Pvt. Ltd. ,Vadodara

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(JUNE, 2021)

A REPORT

ON

PLANTIX LIKE APPLICATION

USING DEEP LEARNING AND NEURAL NETWORKS

BY

MEHUL JAIN 2019A3PS1315H ELECTRICAL AND ELECTRONICS ENGINEERING

Prepared in partial fulfillment of the
Practice School-I Course Nos.
BITS C221/BITS C231/BITS C241

AT

Edutech Learning Solutions Pvt. Ltd. ,Vadodara

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(JUNE, 2021)

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my Practice School 1 faculty, Dr. Raghunath Reddy for their able guidance and support in completing Part 1 of my project.

I would also like to extend my gratitude to the Practice School station coordinator, Ketan Patel and mentor, Bhavin Darji who provided us with all the resources without which it would not have been possible.

I would also like to thank Birla Institute of Technology and Science, Pilani to provide me with such an valuable opportunity which will undoubtedly provide me lots of experience.

Date:	**Mehul Jain**
25th June 2021	**2019A3PS1315H**

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
(RAJASTHAN)
Practice School Division

Station: Edutech Learning Solutions Pvt. Ltd.

Centre: Paranjape Building, Opp. Gas project office, Dandia Bazar, Vadodara- 390001
Duration: 31st May 2021 to 25th June 2021 (Part 1) Date of Start: 31st May 2021
Date of Submission: 25th June 2021
Title of the Project: PLANTIX LIKE APPLICATION USING DEEP LEARNING AND NEURAL NETWORKS
ID No./Name(s)/
Discipline(s)/of
the student(s): Mehul Jain | 2019A3PS1315H | Electrical and Electronics Engineering

Name(s) and
designation(s)
of the
expert(s): Bhavin Darji | Embedded Engineer

Name(s) of
the PS
Faculty: Dr. Raghunath Reddy

Key Words: Deep Learning, Neural Networks, Image Processing, etc.
Project Areas: Information and Technology, Electrical and Electronics
Abstract: Predict the leaves and the health of leaves as predicted by a Plantix like application.

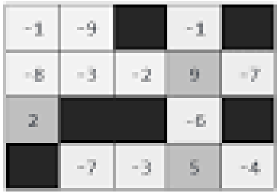
Signature(s) of Student(s) Signature of PS Faculty
Date Date

TABLE OF CONTENTS

INTRODUCTION

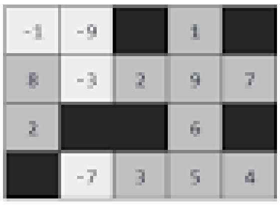
Digital Image Processing is the method of extracting an unstructured data structure like images to structured data like pixelated matrices of images, text extracted from images or probability of a possibility of a classification. It is done in various programming languages like Python, C++, Java, MATLAB, Octave, etc.

Before moving forward, let us look into what an image is for a computer. The measuring unit for images is pixels. The height, width, size and many more physical quantities related to images are measured in pixels. An image of size 5 x 4 is given below:



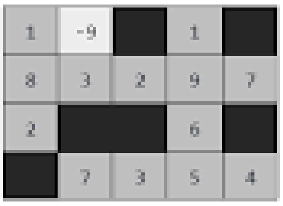
-1	-9		-1	
-8	-3	-2	9	-7
2			-6	
	-7	-3	5	-4

Input Matrix



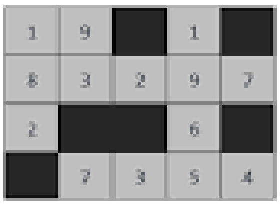
-1	-9		1	
8	-3	2	9	7
2			6	
	-7	3	5	4

After end of Pass 1



1	-9		1	
8	3	2	9	7
2			6	
	7	3	5	4

After end of Pass 2



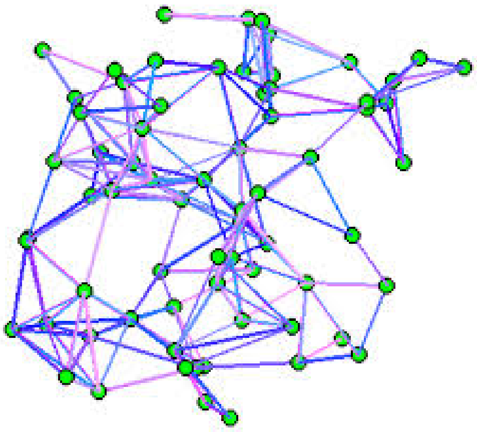
1	9		1	
8	3	2	9	7
2			6	
	7	3	5	4

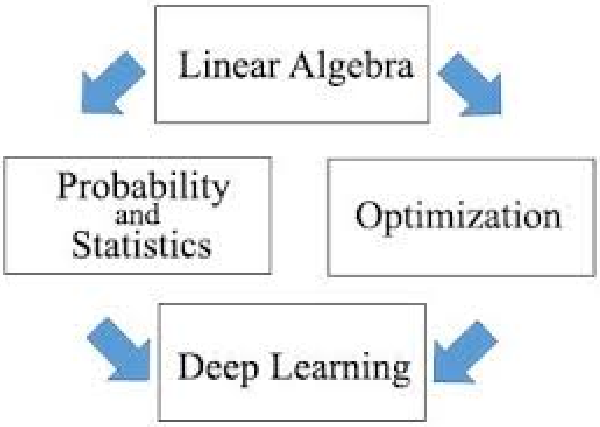
After end of Pass 3

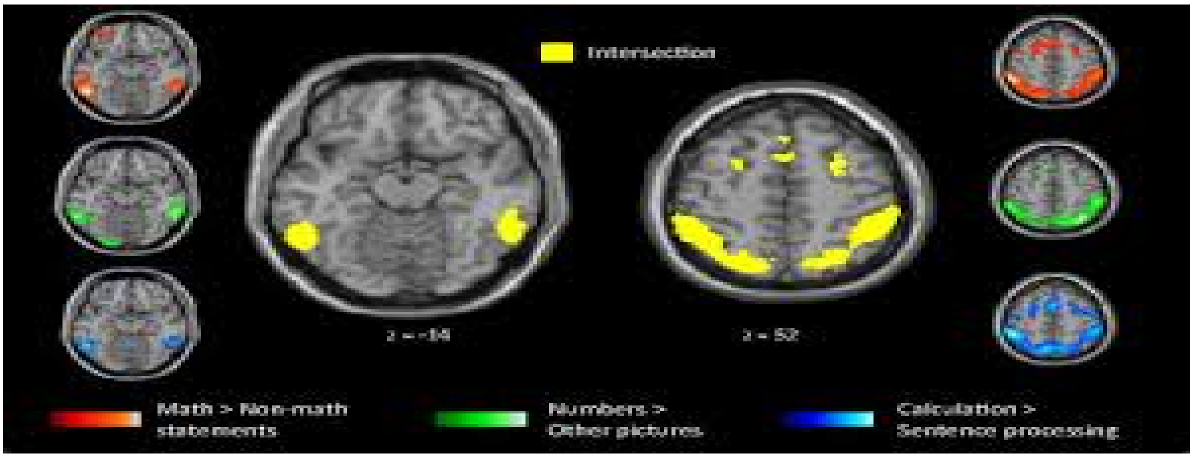
The matrix is of width 5 pixels and height of 4 pixels. The dimension or the size of the image is given as (5, 4). As seen above, different operations are performed on the image matrix to give a different preprocessed image. Each pixel is basically a vector. The value of the vector is given by Red, Green, Blue, Alpha or RGBA values. The direction of the vector is given by its column and row indices or coordinates.

MATHEMATICS BEHIND IMAGE PROCESSING

Different theories of Mathematics are used in Image Processing. It includes statistical theories of probability, set theory, histograms, whisker plots, distribution of data, vectors, modification of matrices, vector and scalar multiplication, fourier transforms and many more. Each output requires a different approach. The best approach is generally found using the calculus theories along different dimensions of space.





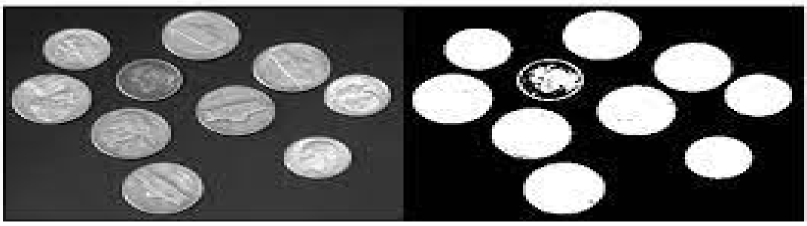


As seen above, different graph theories, Linear Algebra and Calculus are used.

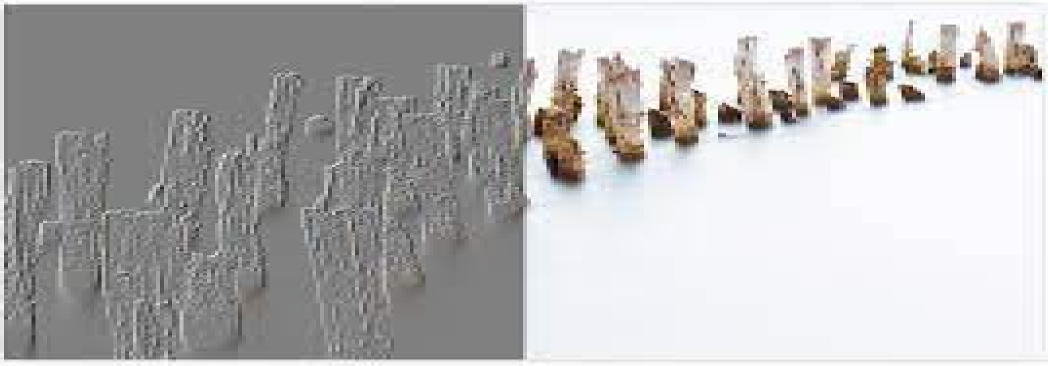
PREPROCESSING OF IMAGES

Preprocessing of images is the very first step of analysing an image. Preprocessing of images involves many different functions which include Gray Scaling, Embossing, Binarizing and Enhancing images. This is done by various different methods which include dot product, element wise operations, transverse operations and many more.

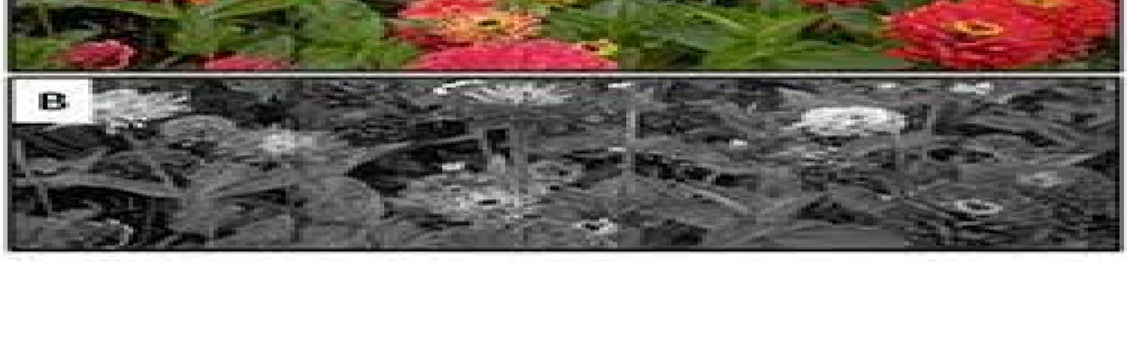
Given below is a binarized image where the threshold value is 134.



Given below is an embossed image which is used for edge detection.



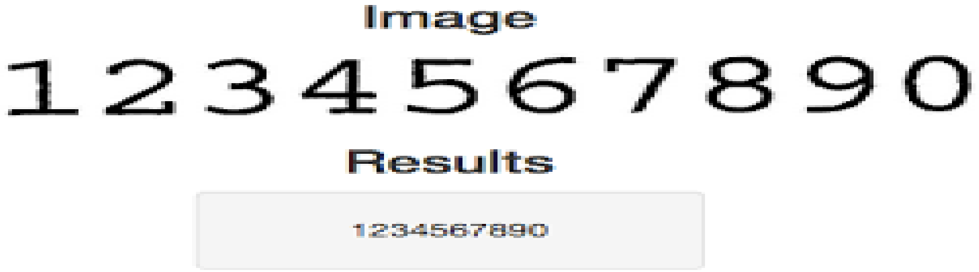
Given below is a gray scale image which is the average RGB value of each pixel.



PROCESSING OF IMAGES

Processing of images is the next step after preprocessing images. The processing of preprocessed images involves the use of libraries like OpenCV, Tesseract, Kraken, etc. The library used needs to have correct formatted and preprocessed images.

Given below is a text extraction library, Tesseract which needs Binarized images.



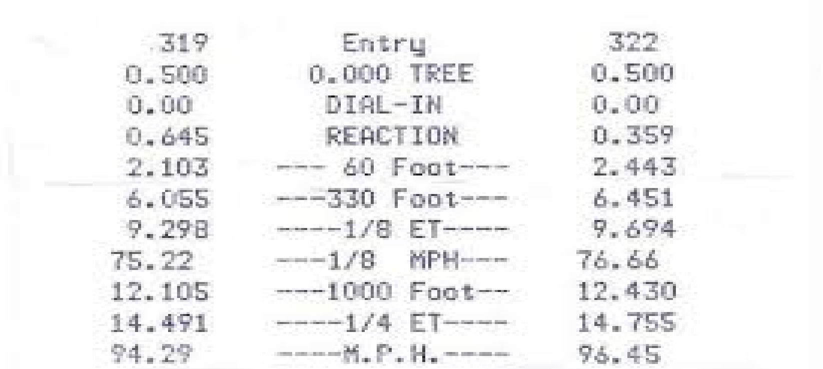
Image

1 2 3 4 5 6 7 8 9 0

Results

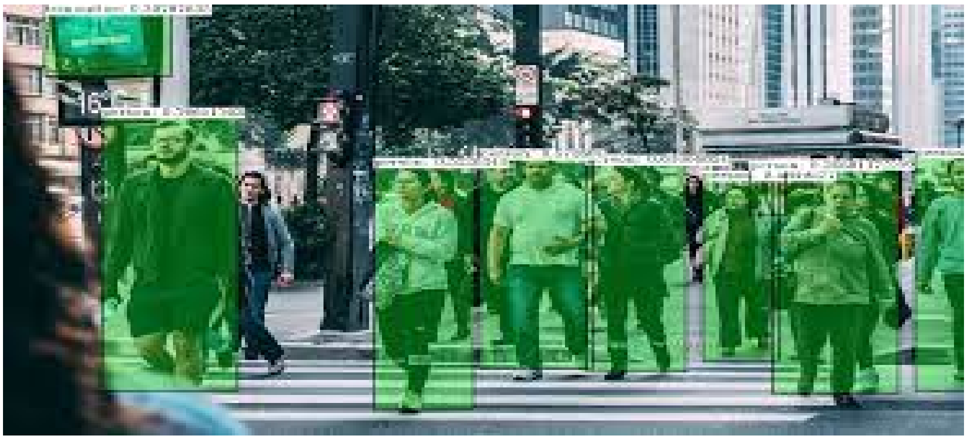
1234567890

Given below is a column wise text extractor on images like



```
  319          Entry          322
 0.500      0.000 TREE       0.500
 0.00        DIAL-IN         0.00
 0.645       REACTION        0.359
 2.103    --- 60 Foot---     2.443
 6.055    ---330 Foot---     6.451
 9.298    ----1/8 ET----     9.694
75.22     ---1/8  MPH---    76.66
12.105    ---1000 Foot--    12.430
14.491    ----1/4 ET----    14.755
94.29     ----M.P.H.----    96.45
```

The Object Detection library, OpenCV which requires RGB images as below.



OPENCV AND ITS OUTCOMES

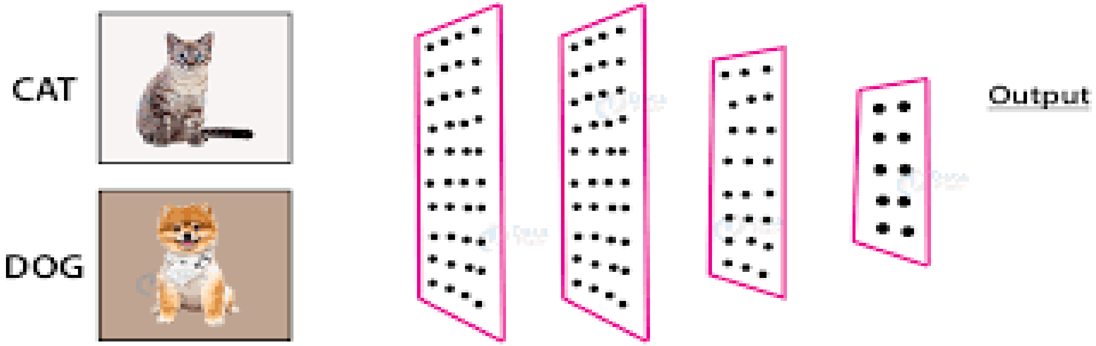
OpenCV is used to identify non structured data from an image data and to convert into structured data. It is used in computer vision projects and is a pre-trained library which makes object detection easier and faster. It is written in C++. It uses edge-detection algorithms to classify images as negative or positive. The method of using OpenCV is:

1) Collect all negative files and positive files.

2) Train the datasets separately.

3) Generate a haar cascade file in XML format containing the algorithm used to classify images.

Below is a positive and negative image for a cat vs non cat image classification.



PROJECT OUTPUTS OF OPENCV

OpenCV for Plantix like application is used for the following objectives with codes:

1) Gray Scale images: Each pixel is taken as the average of RGB values of the corresponding pixels.

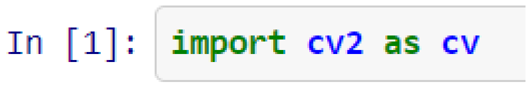
2) Binarizing images: Each pixel's gray scale value is compared with a threshold value and assigned a 0 or 255 accordingly.

3) Applying OpenCV on the Binarized images.

OpenCV is applied on images based on the algorithms generated by the XML file. The XML files for a few are open source and a few can be generated from the Cascade-Trainer-GUI application. The training of 50 positive images and 50 negative images takes around 5 minutes.

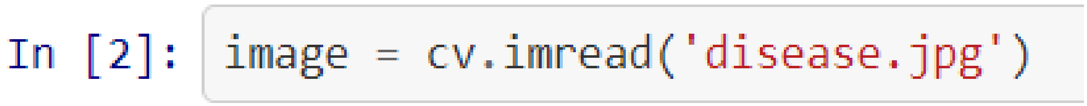
To use OpenCV library:

1) Import OpenCV library as cv.



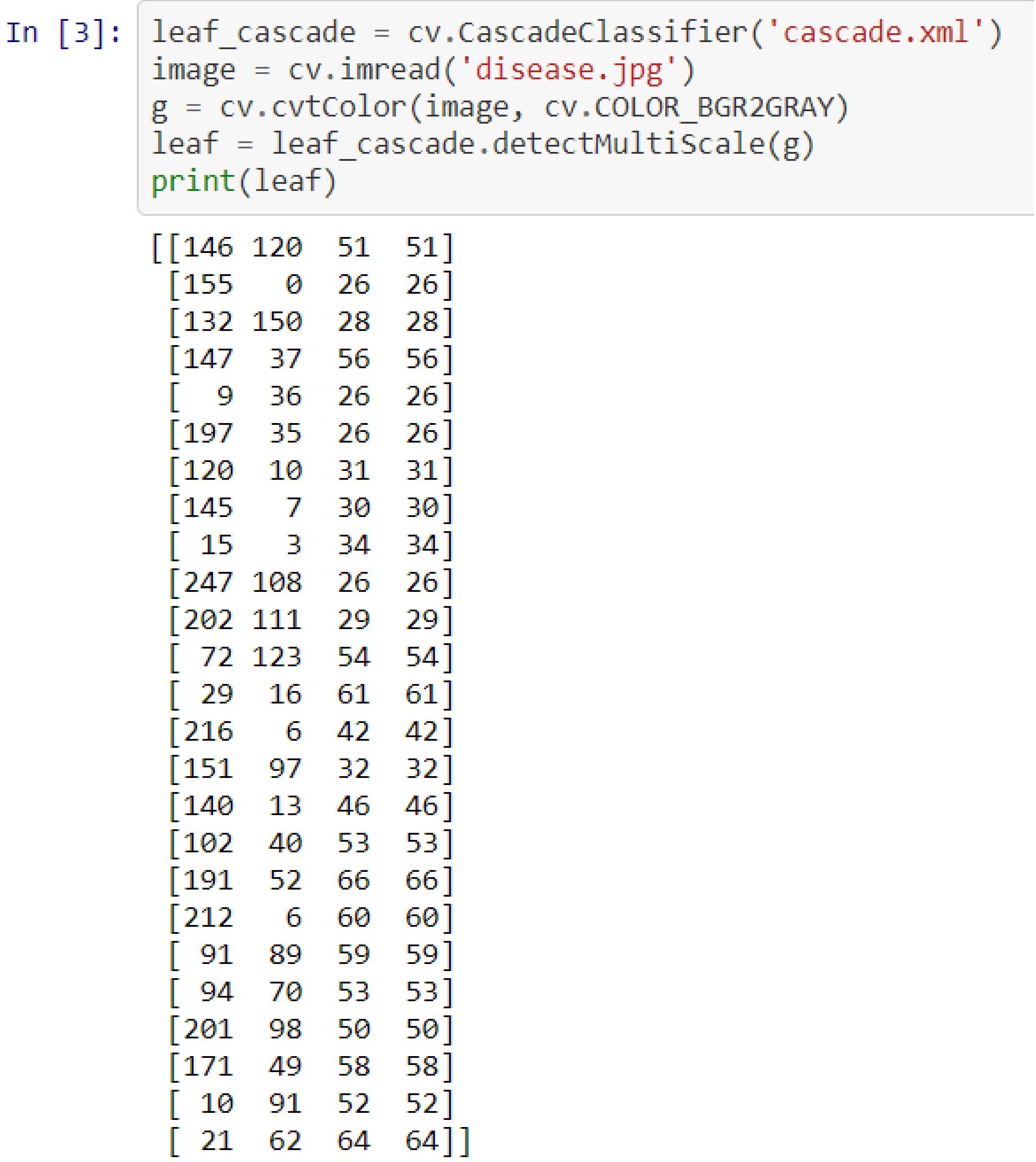
```
In [1]: import cv2 as cv
```

2) Use the OpenCV library to read the image.



```
In [2]: image = cv.imread('disease.jpg')
```

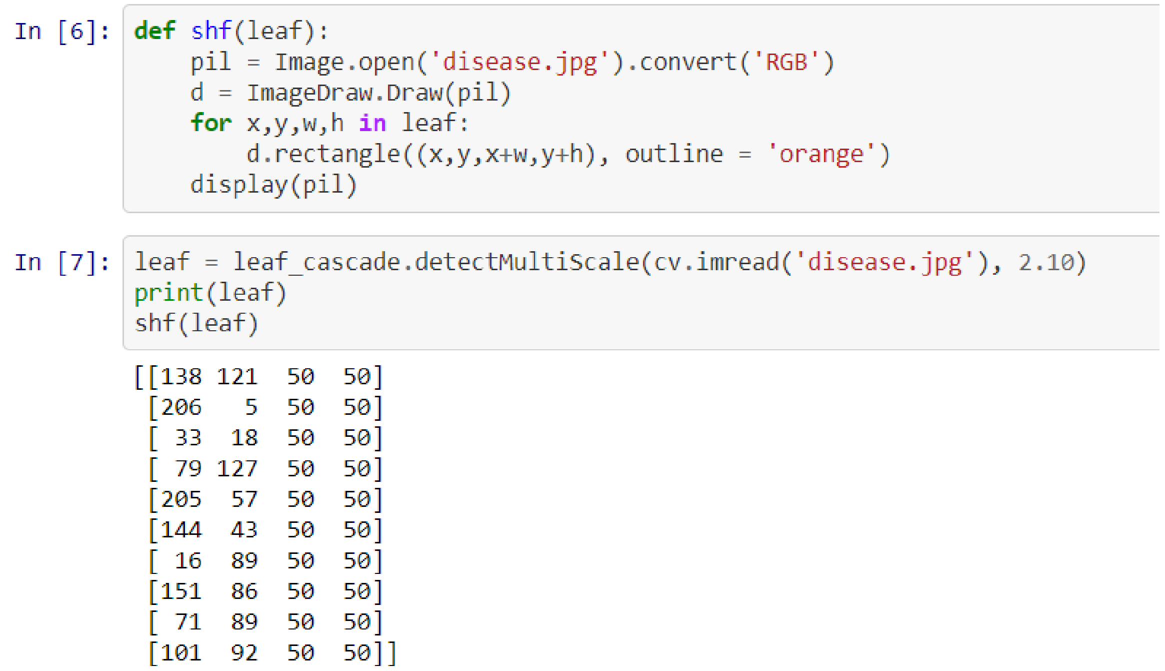
3) Import the haar cascade XML file.



```
In [3]: leaf_cascade = cv.CascadeClassifier('cascade.xml')
        image = cv.imread('disease.jpg')
        g = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
        leaf = leaf_cascade.detectMultiScale(g)
        print(leaf)
```

```
[[146 120  51  51]
 [155   0  26  26]
 [132 150  28  28]
 [147  37  56  56]
 [  9  36  26  26]
 [197  35  26  26]
 [120  10  31  31]
 [145   7  30  30]
 [ 15   3  34  34]
 [247 108  26  26]
 [202 111  29  29]
 [ 72 123  54  54]
 [ 29  16  61  61]
 [216   6  42  42]
 [151  97  32  32]
 [140  13  46  46]
 [102  40  53  53]
 [191  52  66  66]
 [212   6  60  60]
 [ 91  89  59  59]
 [ 94  70  53  53]
 [201  98  50  50]
 [171  49  58  58]
 [ 10  91  52  52]
 [ 21  62  64  64]]
```

4) Apply the haar cascade algorithm on the negative image opened.

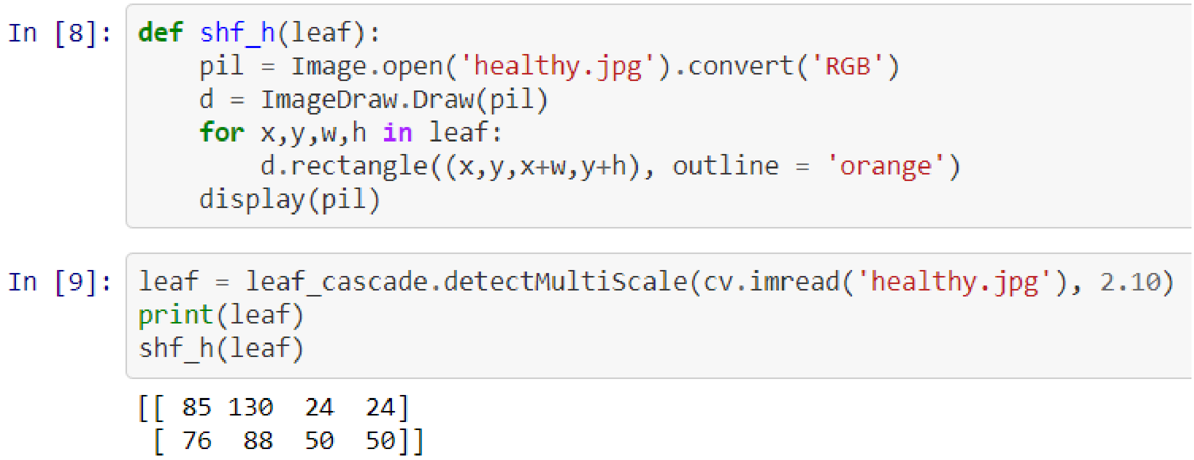


```
In [6]: def shf(leaf):
            pil = Image.open('disease.jpg').convert('RGB')
            d = ImageDraw.Draw(pil)
            for x,y,w,h in leaf:
                d.rectangle((x,y,x+w,y+h), outline = 'orange')
            display(pil)
```

```
In [7]: leaf = leaf_cascade.detectMultiScale(cv.imread('disease.jpg'), 2.10)
        print(leaf)
        shf(leaf)
```

```
[[138 121  50  50]
 [206   5  50  50]
 [ 33  18  50  50]
 [ 79 127  50  50]
 [205  57  50  50]
 [144  43  50  50]
 [ 16  89  50  50]
 [151  86  50  50]
 [ 71  89  50  50]
 [101  92  50  50]]
```

5) Apply the haar cascade algorithm on the positive image opened.

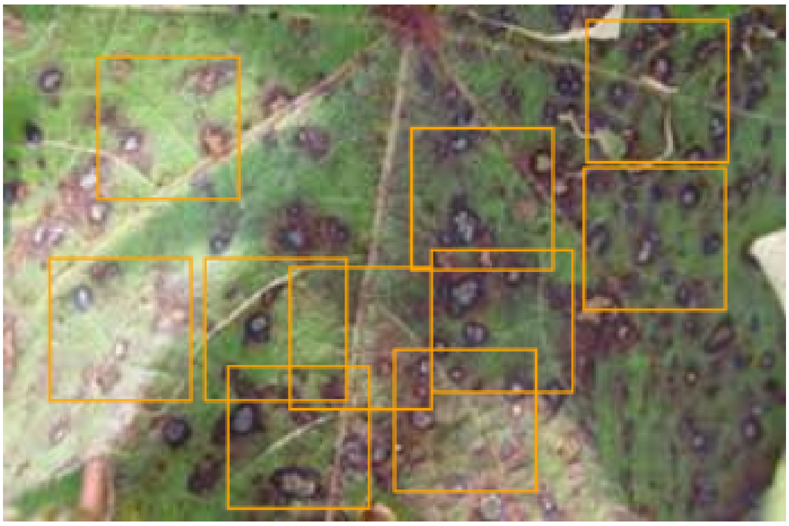


```
In [8]: def shf_h(leaf):
            pil = Image.open('healthy.jpg').convert('RGB')
            d = ImageDraw.Draw(pil)
            for x,y,w,h in leaf:
                d.rectangle((x,y,x+w,y+h), outline = 'orange')
            display(pil)
```

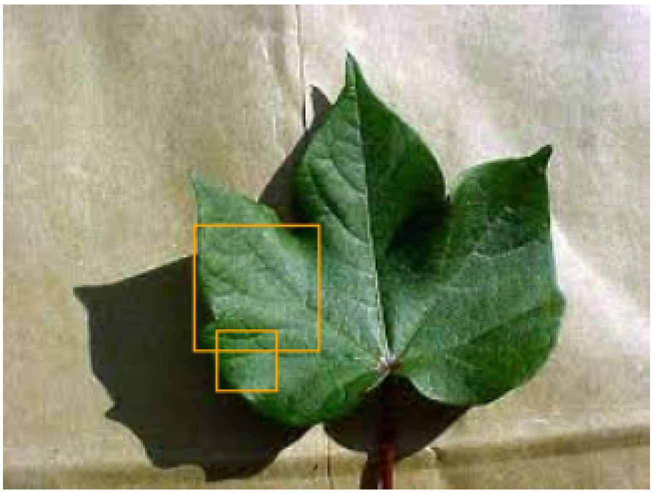
```
In [9]: leaf = leaf_cascade.detectMultiScale(cv.imread('healthy.jpg'), 2.10)
        print(leaf)
        shf_h(leaf)
```

```
[[ 85 130  24  24]
 [ 76  88  50  50]]
```

6) Draw the rectangles for a positive image.



7) Draw the rectangles for a negative image.



8) Observing the outcomes, we can classify if the prediction and outcome was correct or not.

The dataset used was:

1) Training Set: 70% of the total number of images.

2) Testing Set: 30% of the total number of images.

The next step is to get the accuracy higher. Use the neural networks which are deeper than the algorithm used by OpenCV. Along with finding the disease spots, finding the probability of a particular disease in the leaves.

DISADVANTAGES OF OPENCV

OpenCV generates algorithms depending on the positive and negative images which is based on binary classification. The haar cascade XML file takes a lot of time to be generated.

Apart from binary classification, there is multiclass classification in which OpenCV needs to be applied by only dividing the problem statement into many binary classification problems which means that it would be quadratic time complexity which is not suitable for fast applications.

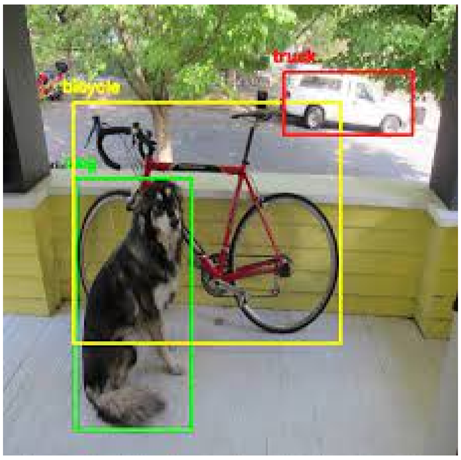
The haar cascade XML file for images of our own does not give much accuracy. The accuracy for the Plantix like application was around 70~80 percent.

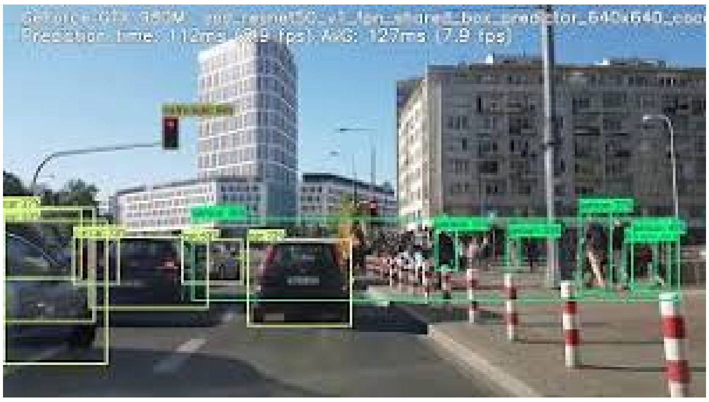
In the below image, the low confidence images are higher in number as compared to fully detected objects.

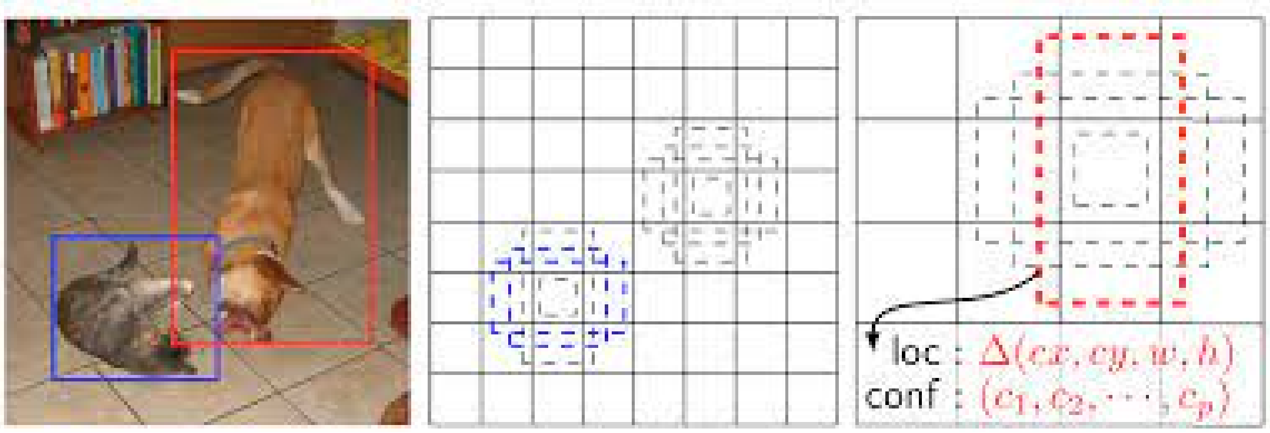
CONCLUSION AND WHAT TO USE NEXT?

Apart from OpenCV with haar cascade generated algorithms, use OpenCV with object detection models like YOLO algorithms and others like ResNet, Conv2D algorithms and many more.

These algorithms use OpenCV or OpenCV like algorithms but are much faster. Not only is the speed of the algorithm good but also about the fact that it detects multiple classes for multiclass classification. These algorithms are highly accurate and some are also preferred over OpenCV alone because it integrates with practical applications more easily.




Geforce GTX 960M ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco
Prediction time: 112ms (8.9 fps) AVG: 127ms (7.9 fps)


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

APPENDIX AND REFERENCES

1) Github Repository used: https://github.com/mehul14062001/Practice-School-1

2) Cascade-Trainer-GUI used: https://amin-ahmadi.com/cascade-trainer-gui/

3) Google Images

4) Resources provided by Bhavin Darji Sir.

5) Terms used in OpenCV: https://docs.opencv.org/master/d5/d0b/classcv_1_1aruco_1_1Dictionary.html

GLOSSARY

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.

- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometric image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.

- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.

- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.

- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.

- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.