**PROJECT REPORT**

(Project Term January-May 2022)

**Leaf Detection**

Submitted by

**Abhishek K Singh 11907787**

**INT 247**

**(B. Tech CSE)**

Under the Guidance of

**Dr. Sagar Pande**

# **School of Computer Science and Engineering**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**



**CERTIFICATE**

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. He has completed this Project under my guidance and supervision. The present work is the result of his original investigation, effort, and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

**Name of the Mentor:** Dr.Sagar Pande

**School of Computer Science and Engineering,**

Lovely Professional University,

Phagwara, Punjab.

**DECLARATION**

I hereby declare that the project work entitled Leaf Detection is an authentic record of my own work carried out as requirements of Project for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara, under the guidance of Sagar Pande, during January to March 2022. All the information furnished in this project report is based on my own intensive work and is genuine.

**Name of Student:** Abhishek K Singh

**Registration Number:** 11907787

# **ACKNOWLEDGEMENT**

I take this opportunity to express my deep gratitude and most sincere thanks to my teachers, parents, and friends for giving most valuable suggestion, helpful guidance, and encouragement in the execution of this project work.

I would like to thank my course mentor for guiding me in making the Project work successful.

**INDEX**

Title Page………………………………………………………………………………………………(1)

Certificate…...………………………………………………………………………………………….(2)

Declaration……..…….……………………………………………………………….………………..(3)

Acknowledgement…………………...…………………………………………........…………………(4)

Index……………………...…………………………………………….………………………………(5)

**1. ABSTRACT 6**

**2. OBJECTIVES 7**

**3. INTRODUCTION 8**

## **3.1. LEAF DETECTION 8**

## **3.1.1. DATASETS USED 9**

## **3.1.2. FRAMEWORK FOR LEAF DETECTION 10**

**3.1.3. ADVANTAGES 10**

**3.1.4. DISADVANTAGES 10**

**3.1.5. APPLICATIONS 11**

**3.1.6. CHALLENGES 11**

**3.1.7. WHY USE THE LEAF FOR DETECTION? 11**

**3.2. PROBLEM 12**

**3.2.1. DATA FLOW DIAGRAM (DFD) 12**

**3.2.2. METHODOLOGY 12**

**4. SOURCE CODE 17**

**5. RESULT 40**

**6. FUTURE SCOPE 43**

**6. CONCLUSION 44**

**7. REFERENCES 45**

**ABSTRACT**

Leaf detection with a computer algorithm is a very challenging task. With the advancement in computer vision, and machine learning, it is possible to detect leaves from images. The leaf detection system is based on convolutional neural network (CNN). Features of leaves are the vital identifiers for leaves because it corresponds to the variety of plants and trees.

There have been many attempts to make an automatic leaf detection tool as it has applications in many fields such as monitoring growth in plants, disease detection in plants, driving assist systems. The parts of a leaf consist of midrib, tip (or axis), margin, leaf base, veins, venules, etc. leaf detection is growing rapidly as a sub-field of image processing. Some of the possible applications are monitoring growth in plants and trees, disease detection in plants, etc.

The result observed is as: final training, validation and testing is done successfully with the addition of the images with detection of leaf successfully done.

**OBJECTIVES**

* The objective of the leaf detection is detecting the leaves in the images of plants and trees. It is very useful and can be used as a base for many real-time applications. It can be used as a part of many useful applications like monitoring growth patterns in plants and trees, keeping a check on any signs of disease in plants, crop research.
* We can easily identify the leaves in a plant or tree without any effort, but automatic detection of leaves is important due to its real-time applications.
* Pretrained CNN based features represent the most discriminative features and hence allows for a better performance.
* Features from convolutional layer of the pretrained model are obtained and then fed as input to the classifier. For feature extraction, pre-trained convolutional neural networks model is used.

**INTRODUCTION**

Leaf detection is one of the challenging aspect in the field of image analysis. Leaf detection has been a topic of active research, proposing solutions to several practical problems. Leaf detection is probably the method that is used to identify and differentiate plants and trees mainly from their leaves. However, the recognition process used by the human brain for identifying leaves is very challenging. In this paper, a Genetic Algorithm (GA) based approach is proposed for leaf detection. The proposed algorithm recognizes an unknown image by comparing it with the known training images stored in the database and provides image with detected leaves in the image of plants or trees.

Leaf detection is a task of pattern recognition that is specifically performed on leaves. In other words, it can be described as classifying a leaf, either known or unknown by comparing a leaf with stored known leaf images in the database. It is also desirable to have a system that has the ability of learning to recognize unknown leaves. People have a good ability to recognize and distinguish between objects but recognizing objects automatically by computer is very difficult. The main goal of leaf detection technology is to match a given leaf image against the stored database of images. Leaf detection technique uses several other disciplines such as image processing, computer vision, pattern recognition, neural networks and psychology. With the current increase in population, the demand for food is increasng day by day and thus there is a need to produce good quality food without any wastage or loss and thus, it is very important to monitor the growth patterns of plants and check for any disease in the plants during the crop production phase.

**3.1. LEAF DETECTION**

A leaf detection system is developed to demonstrate the application of the proposed method. The system follows the leaf-based approach***,*** and it consists of two stages, namely, training and recognition stages. Training stage computes the representational bases for images in the domain of interest (that is reference images) and converts them into training image representations. The training image representations of each image are stored in the library. Using the representational bases recognition stage translates the testing image into probe image representation. Testing image is then matched with reference images which are stored in the library to identify the leaf image.

Leaf detection is a method used for the detection and recognition of leaves in plants and trees with the incorporation of technological capabilities. It includes the services that can recognize basic features of leaves (midrib, tip, margin, veins, venules, and leaf base). It is a technology which uses markers to detect features in leaves of plants and trees.

Leaf detection system is a computer-based technology and therefore, it uses algorithms to instantaneously detect leaves, code features of leaves, and recognize the parts of leaves. It does this by analyzing leaves in images and video through computer powered cameras embedded in laptops, mobile phones, and digital signage systems, or cameras that are mounted onto computer screens. Leaf analysis through computer powered cameras generally follow three steps:

* Leaf detection: Locating leaves in the scene, in an image or video footage.
* Detection of features of leaves: Extracting information about features of leaves from detected leaves.
* Classification of parts of leaf: Analyzing the features of leaves and changes in the appearance of features of leaves and classifying this information into interpretative categories.

**3.1.1. DATASETS USED**

I used the Leaf Detection dataset, which is the standard for evaluating the leaf detection technologies, consists of more than 5,346 leaf images of more than 1000 subjects. The dataset used in my experiment consists of 1140 leaf images. The images are of size 1024 x 1024, and some are 416 x 416. Three sets consisting of galleries and probes were used for evaluating the performances of the algorithms. The training set is the set of known leaf images, which are used for training. The images in the probe set are the unknown leaf images to be recognized. Database – <https://www.kaggle.com/datasets/alexo98/leaf-detection?select=train.csv> (Leaf Detection)

* The data consists of images of leaves. The leaves are centred and occupy nearly the same amount of space in each image.

The content of the dataset is images of plants, trees, or even individual leaves. Each image has bounding boxes annotated around each leaf, but only those that can be seen easily (or are clearly visible).

The images are downloaded from Google images and Bing images and annotated with labelImg. The latter images are from a dataset where each leaf is annotated based on category of leaf. Every leaf is annotated as “leaf”, and so, there is no separate column for this in .csv file.

### **3.1.2. FRAMEWORK FOR LEAF DETECTION**

### Leaf detection is a technique that takes the image of a leaf (query image) and compares it with the previously recorded images in the database. This is done by comparing the invariant features obtained from the techniques that capture the representative variability of the leaves or the structure, the shape, and the leaf attributes like distance between the leaf veins, outlines of the leaves, etc. Leaf detection has the benefit of being a passive, non-intrusive system to detect leaves in images in a natural and friendly way. The main benefit of this technique is that the leaf images can be taken from a distance as might be required in identifying the presence of the leaves in plants or trees and can also be helpful in leaf disease detection as it detects the leaves.

**3.1.3. ADVANTAGES**

* The advantages of leaf detection are that it is beneficial for businesses as they can process images, and videos in real-time monitoring video feeds or automatic video analytics, thus saving the costs and making life better for farmers and crop observers as it helps in monitoring leaves for any disease detection in the leaves.
* For businesses, since leaf detection software delivers raw images and helps in focusing the leaves in the images, it can provide valuable information about the leaf disease.
* It is important because of its ability to look for any signs of disease in leaves.
* Leaf detection analysis is a practical means of going beyond the typical survey approach. It is a way of proctoring the progress of growth of plants, trees, and crops by capturing images of leaves, all while getting feedback. When feedback is taken in this format, it becomes genuinely non-intrusive when it comes to user experience and easy.

**3.1.4. DISADVANTAGES**

* The disadvantages of leaf detection are that most leaf detection coding schemes rely on the traditional systems used to classify only the clear images of leaves and are very labor-intensive if done by trained human coders rather than software.
* Programs of research that use leaf detection to examine leaf growth require extensive cross-method validations to connect configurations of leaves with growth labels in leaves.
* Measuring changes in leaves is difficult. The changes in the leaves are often fleeting, hidden, and conflicted and sometimes may not be clear.

**3.1.5. APPLICATIONS**

It has become one of the active research areas especially in recent years as it has a variety of wide applications in the areas:

* Farming
* Food research
* Crop growth monitoring
* Early disease detection in plants

Leaf detection-

* Food production
* Research - Growth pattern in plants

**3.1.6. CHALLENGES**

The challenges associated with leaf detection can be attributed to the following factors:

* Presence or absence of structural components: Features of leaves such as midrib, vein, venules, and insects usually on leaves may or may not be present and there is a great deal of variability among these components including shape, color, and size.
* Pose: The images of a leaf vary due to the relative camera-face pose (frontal, tilted, profile, upside down).
* Leaf appearance: The appearance of leaves is directly affected by the clarity of the images of leaves.
* Occlusion: Leaves may be partially occluded by other objects. For an example, in an image with a group of plants, some leaves may partially occlude other leaves (leaf detection).
* Image orientation: Leaf images directly vary for different rotations about the camera’s optical axis.
* Imaging conditions: When the image is formed, factors such as lighting, and camera characteristics affect the appearance of a leaf.
* Age: Images taken after some time may not match with the images in database.

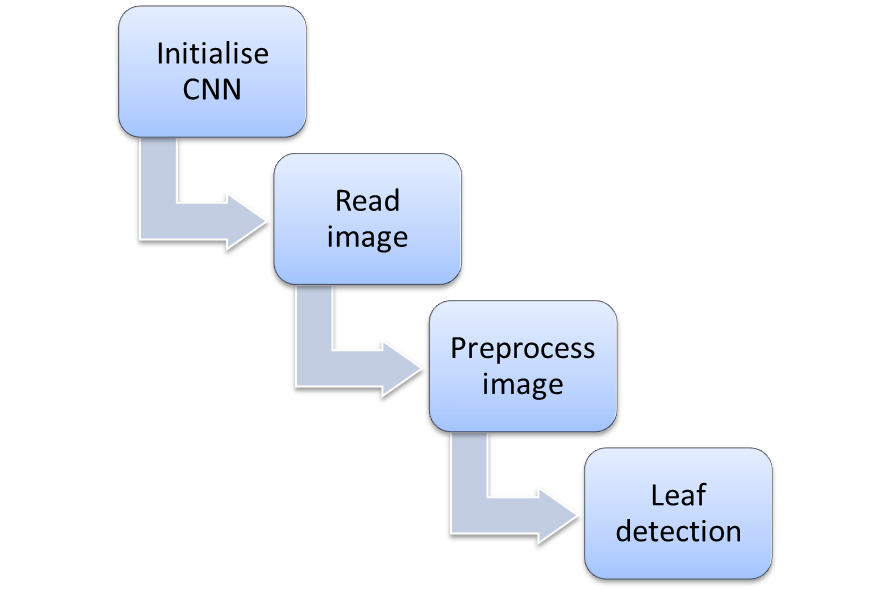
**3.1.7. WHY USE THE LEAF FOR DETECTION?**

Leaf detection-based techniques have emerged as the most promising option for recognizing plants and trees, instead of having the whole plant or tree images for detection of disease as leaves are the most important part and it helps in identifying any signs disease through the change in its looks. and so forth, these methods examine a plant or tree’s physiological characteristics to determine calculations of presence of any disease. Additionally, a plant or tree’s biological traits such as images of leaves cannot be misplaced, stolen, or forged. Leaf detection-based technologies include identification based on physiological characteristics (such as midrib, margin, tip, vein, venules, leaf blade, shape, color, and size).

**3.2. PROBLEM**

The recognition of plant or trees through leaf detection technique relies on detecting the individual features such as the midrib, margin, tip, vein, venules, leaf blade and defining a leaf model by the position, size, and relationships among these features. Such approaches to leaf detection are difficult to extend to multiple views and often been quite fragile, requiring a good initial guess to guide them. Research related to human strategies in the field of leaf detection has shown that individual features and their immediate relationships comprise an insufficient representation to account for the performance of leaf detection.

**3.2.1. DATA FLOW DIAGRAM (DFD)**



**3.2.2. METHODOLOGY**

Convolutional neural network (CNN) is the most popular way of analyzing images. CNN is different from a multi-layer perceptron (MLP) as they have hidden layers, called convolutional layers. It is a type of neural network model which allows to extract higher representations for image content. It takes the image’s raw pixel data, trains the model, then extracts the features automatically for better classification.

CNN are also known as shift invariant or space invariant artificial neural networks, based on the shared weight architecture of the convolutional kernels or filters that slide along input features and provide translation equivariant responses known as feature maps. CNNs are regularized versions of multilayer perceptron. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters or kernels through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

Convolutional networks are a specialized type of neural networks that use convolution in place of general matrix multiplication in at least one of their layers. A convolutional neural network consists of an input layer, hidden layers, and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions.

Convolutional neural networks usually require a large amount of training data to avoid overfitting. A common technique is to train the network on a larger dataset from a related domain. Once the network parameters have converged, an additional training step is performed using the in-domain data to fine-tune the network weights, this is known as transfer learning. Furthermore, this technique allows convolutional network architectures to successfully be applied to problems with tiny training sets.

Application of CNN:

* Image recognition
* Video analysis
* Natural language processing
* Anomaly detection

Architecture:

* Initialize the CNN (Convolutional Neural Network) – Shift invariant or space invariant artificial neural networks (analyze visual imagery).
* Read the image
* Detect leaves in the image
* Pre-process all the leaves
* Run a forward pass on all the leaves
* Get predicted leaf detection model

Parts of a leaf

* Apex (or tip)
* Margin
* Midrib

Leaf blade

* Vein
* Venules
* Leaf base

The Genetic Algorithm (GA) is a stochastic search method based on the mechanics of natural selection and genetics analogous to natural evolution. Central to the idea of GA is a population of individuals, each representing a possible solution to the given problem. Each, known as chromosome (usually represented by a bit string consisting of 0s and 1s), is assigned to a fitness value based on how good their solution to the problem is. The individuals then evolve through successive iterations called generations. During one generation, highly fit individuals are given the opportunity to mate with other individuals in the population. Since the least fit individual in the population are less likely to get selected for mating (or reproduction), they disappear from future generations. As a result, the population of individuals converges to an optimal solution to the problem. GAs is robust and can deal successfully with a wide range of problem areas, including those which are difficult for other methods to solve.

To apply GA for leaf detection, a template of the leaf image obtained from averaging the gradation level of pixels of several similar looking leaf images of several plants or trees is constructed. The template leaf image is then moved through the whole image to find the location where the most suitable match exists. This process applies GA for the optimization of five parameters such as, center position of the template image, scaling of the template, rotation of the template and matching rate between the input image and the template image.

Selection operator is a process in which chromosomes are selected into a mating pool according to their fitness function. Good chromosomes that contribute their gene-inherited knowledge to breed for the next generation are chosen. Here we use conventional elitist selection scheme to select an elitist chromosome with the highest fitness value, which is copied directly into the new population of next generation. The other chromosomes are selected by a roulette-wheel selection process, where the selection probability of each individual is proportional to its fitness value.

This operator randomly chooses a crossover point where two parent chromosomes break and then exchanges the chromosome parts after that point. As a result, two offspring are generated by combining the partial features of two chromosomes. If a pair of chromosomes does not cross over, then the chromosome cloning takes place, and the offspring are created as exact copies of each parent. Here we have studied single point cross-over, two-point cross-over and uniform cross-over operators. The cutting points are selected randomly within the chromosome for exchanging the contents.

Mutation, which is rare in nature, represents a change in the gene and aids us in avoiding loss of genetic diversity. Its role is to provide a guarantee that the search algorithm is not trapped on a local optimum.

This operator alters a randomly selected gene of chromosome with a very low probability, PM. For each chromosome, generate a random value between [0,1]. If the random value is less than PM, then choose a bit at a random location to flip its value from 0 to 1, or 1 to 0.

The fundamental steps employed for the genetic algorithm are as follows:

**Step 1:** Initialization: Generate randomly a population of chromosomes of size N: x1, x2,.....,xN. Assign the crossover probability Pc and the mutation probability PM.

**Step 2:** Evaluation: Evaluate the fitness function to measure the performance or fitness for individual chromosome in the population. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.

**Step 3:** Selection: Select a pair of chromosomes for mating (or reproduction). Use the roulette wheel selection procedure, where each chromosome is given a slice of a circular roulette wheel. The area of the slice within the wheel is equal to the chromosome fitness ratio. Obviously, the highly fit chromosomes occupy the largest areas, where the chromosomes with least fit have much smaller segments in the wheel. To select a chromosome for mating, a random number is generated in the interval [0,100] and the chromosome whose segment spans the random number is selected.

**Step 4:** Cross-over: Produce two off-springs from two parent chromosomes. With the cross-over probability Pc, exchange parts of the two selected chromosomes and create two offspring.

**Step 5:** Mutation: Apply the conventional mutation operation to the population with a mutation rate PM. With this mutation probability, randomly change the gene values in the two offspring chromosomes.

**Step 6:** Termination test: If a predefined termination condition is satisfied, go to Step 7, else go to Step 2.

**Step 7:** Preservation: Keep the best chromosome.

**Step 8:** End.

On processing the genetic operation, the leaf area is detected on the image. The exact locations of the features of leaves are then searched. The leaf parts are localized in this experiment. These are the midrib, tip, leaf base, margin, veins, venules, etc. Features of leaves are extracted from the leaf profile depending on their geometrical arrangement on the leaf skeleton.

**SOURCE CODE**

import pandas as pd

import numpy as np

import cv2

import os

import re

from PIL import Image

import time

import albumentations as A

from albumentations.pytorch.transforms import ToTensor

from albumentations.pytorch import transforms

import torch

import torchvision

from torchvision.models.detection.faster\_rcnn import FastRCNNPredictor

from torchvision.models.detection import FasterRCNN

from torchvision.models.detection.rpn import AnchorGenerator

from torch.utils.data import DataLoader, Dataset

from torch.utils.data.sampler import SequentialSampler

from matplotlib import pyplot as plt

DIR\_INPUT = 'Downloads/leaf detection dataset'

DIR\_TRAIN = f'{DIR\_INPUT}/train'

DIR\_TEST = f'{DIR\_INPUT}/test'

# Loading the device now

device = torch.device('cuda') if torch.cuda.is\_available() else torch.device('cpu')

# Reading and parsing the CSV

train\_df = pd.read\_csv(os.path.join(DIR\_INPUT,"train.csv"))

train\_df

Out-

|  | **image\_id** | **width** | **height** |  | **bbox** |  |  | |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | LEAF\_0009.jpg | 1024 | 1024 |  | [473, 273, 289, 335] | | |
| **1** | LEAF\_0009.jpg | 1024 | 1024 |  | [588, 516, 272, 318] | | |
| **2** | LEAF\_0009.jpg | 1024 | 1024 |  | [510, 780, 218, 244] | | |
| **3** | LEAF\_0009.jpg | 1024 | 1024 |  | [766, 822, 246, 201] | | |
| **4** | LEAF\_0009.jpg | 1024 | 1024 |  | [1, 813, 240, 211] | | |
| **...** | ... | ... | ... |  | ... | | |
| **5341** | LEAF\_1112.jpg | 416 | 416 |  | [152, 43, 139, 345] | | |
| **5342** | LEAF\_1112.jpg | 416 | 416 |  | [320, 80, 79, 261] | | |
| **5343** | LEAF\_1113.jpg | 416 | 416 |  | [116, 34, 180, 313] | | |
| **5344** | LEAF\_1114.jpg | 416 | 416 |  | [158, 11, 117, 397] | | |
| **5345** | LEAF\_1115.jpg | 416 | 416 |  | [96, 84, 295, 225] | | |

5346 rows × 4 columns

train\_df.ndim

Out- 2

train\_df.columns

Out- Index(['image\_id', 'width', 'height', 'bbox'], dtype='object')

train\_df.dtypes

Out-

image\_id object

width int64

height int64

bbox object

dtype: object

train\_df.describe()

Out-

|  | **width** | **height** |
| --- | --- | --- |
| **count** | 5346.000000 | 5346.000000 |
| **mean** | 614.458661 | 614.458661 |
| **std** | 285.117926 | 285.117926 |
| **min** | 416.000000 | 416.000000 |
| **25%** | 416.000000 | 416.000000 |
| **50%** | 416.000000 | 416.000000 |
| **75%** | 1024.000000 | 1024.000000 |
| **max** | 1024.000000 | 1024.000000 |

train\_df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5346 entries, 0 to 5345

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 image\_id 5346 non-null object

1 width 5346 non-null int64

2 height 5346 non-null int64

3 bbox 5346 non-null object

dtypes: int64(2), object(2)

memory usage: 167.2+ KB

train\_df.isnull()

Out-

|  | **image\_id** | **width** | **height** | **bbox** |
| --- | --- | --- | --- | --- |
| **0** | False | False | False | False |
| **1** | False | False | False | False |
| **2** | False | False | False | False |
| **3** | False | False | False | False |
| **4** | False | False | False | False |
| **...** | ... | ... | ... | ... |
| **5341** | False | False | False | False |
| **5342** | False | False | False | False |
| **5343** | False | False | False | False |
| **5344** | False | False | False | False |
| **5345** | False | False | False | False |

5346 rows × 4 columns

train\_df['x'] = -1

train\_df['y'] = -1

train\_df['w'] = -1

train\_df['h'] = -1

def expand\_bbox(x):

r = np.array(re.findall("([0-9]+[.]?[0-9]\*)", x))

if len(r) == 0:

r = [-1, -1, -1, -1]

return r

train\_df[['x', 'y', 'w', 'h']] = np.stack(train\_df['bbox'].apply(lambda x: expand\_bbox(x)))

train\_df.drop(columns=['bbox'], inplace=True)

train\_df['x'] = train\_df['x'].astype(np.float)

train\_df['y'] = train\_df['y'].astype(np.float)

train\_df['w'] = train\_df['w'].astype(np.float)

train\_df['h'] = train\_df['h'].astype(np.float)

image\_ids = train\_df['image\_id'].unique()

valid\_ids = image\_ids[-4:]

valid\_ids = np.append(valid\_ids,image\_ids[:4])

train\_ids = image\_ids[4:-4]

valid\_df = train\_df[train\_df['image\_id'].isin(valid\_ids)]

train\_df = train\_df[train\_df['image\_id'].isin(train\_ids)]

valid\_df.shape, train\_df.shape

Out- ((195, 7), (5151, 7))

valid\_df

Out-

|  | **image\_id** | **width** | **height** | **x** | **y** | **w** | **h** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | LEAF\_0009.jpg | 1024 | 1024 | 473.0 | 273.0 | 289.0 | 335.0 |
| **1** | LEAF\_0009.jpg | 1024 | 1024 | 588.0 | 516.0 | 272.0 | 318.0 |
| **2** | LEAF\_0009.jpg | 1024 | 1024 | 510.0 | 780.0 | 218.0 | 244.0 |
| **3** | LEAF\_0009.jpg | 1024 | 1024 | 766.0 | 822.0 | 246.0 | 201.0 |
| **4** | LEAF\_0009.jpg | 1024 | 1024 | 1.0 | 813.0 | 240.0 | 211.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **5341** | LEAF\_1112.jpg | 416 | 416 | 152.0 | 43.0 | 139.0 | 345.0 |
| **5342** | LEAF\_1112.jpg | 416 | 416 | 320.0 | 80.0 | 79.0 | 261.0 |
| **5343** | LEAF\_1113.jpg | 416 | 416 | 116.0 | 34.0 | 180.0 | 313.0 |
| **5344** | LEAF\_1114.jpg | 416 | 416 | 158.0 | 11.0 | 117.0 | 397.0 |
| **5345** | LEAF\_1115.jpg | 416 | 416 | 96.0 | 84.0 | 295.0 | 225.0 |

195 rows × 7 columns

valid\_df.columns

Out-

Index(['image\_id', 'width', 'height', 'x', 'y', 'w', 'h'], dtype='object')

valid\_df.ndim

Out- 2

valid\_df.dtypes

Out-

image\_id object

width int64

height int64

x float64

y float64

w float64

h float64

dtype: object

valid\_df.describe()

Out-

|  | **width** | **height** | **x** | **y** | **w** | **h** |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 | 195.000000 |
| **mean** | 1005.292308 | 1005.292308 | 409.994872 | 416.558974 | 68.097436 | 92.476923 |
| **std** | 105.266925 | 105.266925 | 259.382425 | 277.906070 | 68.033995 | 92.451566 |
| **min** | 416.000000 | 416.000000 | 1.000000 | 1.000000 | 13.000000 | 16.000000 |
| **25%** | 1024.000000 | 1024.000000 | 180.500000 | 172.000000 | 29.000000 | 40.000000 |
| **50%** | 1024.000000 | 1024.000000 | 391.000000 | 398.000000 | 41.000000 | 59.000000 |
| **75%** | 1024.000000 | 1024.000000 | 606.000000 | 645.500000 | 77.500000 | 89.000000 |
| **max** | 1024.000000 | 1024.000000 | 994.000000 | 995.000000 | 439.000000 | 612.000000 |

valid\_df.info()

Out-

<class 'pandas.core.frame.DataFrame'>

Int64Index: 195 entries, 0 to 5345

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 image\_id 195 non-null object

1 width 195 non-null int64

2 height 195 non-null int64

3 x 195 non-null float64

4 y 195 non-null float64

5 w 195 non-null float64

6 h 195 non-null float64

dtypes: float64(4), int64(2), object(1)

memory usage: 12.2+ KB

valid\_df.isnull()

Out-

|  | **image\_id** | **width** | **height** | **x** | **y** | **w** | **h** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **5341** | False | False | False | False | False | False | False |
| **5342** | False | False | False | False | False | False | False |
| **5343** | False | False | False | False | False | False | False |
| **5344** | False | False | False | False | False | False | False |
| **5345** | False | False | False | False | False | False | False |

195 rows × 7 columns

class LeafDataset(Dataset):

def \_\_init\_\_(self, dataframe, image\_dir, transforms=None):

super().\_\_init\_\_()

self.image\_ids = dataframe['image\_id'].unique()

self.df = dataframe

self.image\_dir = image\_dir

self.transforms = transforms

def \_\_getitem\_\_(self, index: int):

image\_id = self.image\_ids[index]

records = self.df[self.df['image\_id'] == image\_id]

image = cv2.imread(f'{self.image\_dir}/{image\_id}', cv2.IMREAD\_COLOR)

# image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB).astype(np.float32)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY).astype(np.float32)

image = np.reshape(image,image.shape+(1,))

image /= 255.0

boxes = records[['x', 'y', 'w', 'h']].values

boxes[:, 2] = boxes[:, 0] + boxes[:, 2]

boxes[:, 3] = boxes[:, 1] + boxes[:, 3]

area = (boxes[:, 3] - boxes[:, 1]) \* (boxes[:, 2] - boxes[:, 0])

area = torch.as\_tensor(area, dtype=torch.float32)

# there is only one class

labels = torch.ones((records.shape[0],), dtype=torch.int64)

# suppose all instances are not crowd

iscrowd = torch.zeros((records.shape[0],), dtype=torch.int64)

target = {}

target['boxes'] = boxes

target['labels'] = labels

target['image\_id'] = torch.tensor([index])

target['area'] = area

target['iscrowd'] = iscrowd

if self.transforms:

sample = {

'image': image,

'bboxes': target['boxes'],

'labels': labels

}

sample = self.transforms(\*\*sample)

image = sample['image']

target['boxes'] = torch.stack(tuple(map(torch.tensor, zip(\*sample['bboxes'])))).permute(1, 0)

return image, target, image\_id

def \_\_len\_\_(self) -> int:

return self.image\_ids.shape[0]

# This Albumentation for now it is empty.

def transform():

return A.Compose([

# A.Cutout(num\_holes=10,max\_h\_size=15,max\_w\_size=15,p=1),

# A.OneOf([

# A.RandomSunFlare(src\_radius=200,num\_flare\_circles\_lower=6,num\_flare\_circles\_upper=8,p=1),

# A.RandomRain(slant\_lower=-10,slant\_upper=10,drop\_length=20,drop\_width=1,p=1),

# A.RandomFog(fog\_coef\_lower=0.05, fog\_coef\_upper=0.1, alpha\_coef=0.08, p=1),

# ], p=1),

# A.OneOf([

# A.RandomBrightnessContrast(brightness\_limit=0.2, contrast\_limit=0.2, brightness\_by\_max=True, p=1),

# A.RandomGamma(gamma\_limit=(80,165),p=1),

# ], p=1),

ToTensor(),

], bbox\_params={'format': 'pascal\_voc', 'label\_fields': ['labels']})

def collate\_fn(batch):

return tuple(zip(\*batch))

train\_dataset = LeafDataset(train\_df, DIR\_TRAIN, transform())

valid\_dataset = LeafDataset(valid\_df, DIR\_TRAIN, transform())

train\_data\_loader = DataLoader(

train\_dataset,

batch\_size=16,

shuffle=False,

num\_workers=4,

collate\_fn=collate\_fn

)

valid\_data\_loader = DataLoader(

valid\_dataset,

batch\_size=1,

shuffle=False,

num\_workers=4,

collate\_fn=collate\_fn

)

class Averager:

def \_\_init\_\_(self):

self.current\_total = 0.0

self.iterations = 0.0

def send(self, value):

self.current\_total += value

self.iterations += 1

@property

def value(self):

if self.iterations == 0:

return 0

else:

return 1.0 \* self.current\_total / self.iterations

def reset(self):

self.current\_total = 0.0

self.iterations = 0.0

# HELPER FUNCTIONS FOR VIZUALISING / PREDICTING

def get\_boxes(tensor,index,score=0.5):

if index >= len(tensor) or index<0:

return 0

temp\_boxes = []

for i in range(len(tensor[index]['boxes'])):

if tensor[index]['scores'][i] > score:

temp\_boxes.append(tensor[index]['boxes'][i].cpu().detach().numpy().astype(np.int32))

return temp\_boxes

def get\_sample\_image(itr):

images, targets, image\_ids = next(it)

images = list(image.to(device) for image in images)

targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

boxes = targets[0]['boxes'].cpu().numpy().astype(np.int32)

sample = images[0].permute(1,2,0).cpu().numpy()

sample = np.reshape(sample,(sample.shape[1],sample.shape[1]))

fig, ax = plt.subplots(1, 1, figsize=(16, 8))

for box in boxes:

cv2.rectangle(sample,

(box[0], box[1]),

(box[2], box[3]),

0, 2)

ax.set\_axis\_off()

ax.imshow(sample,cmap='gray')

def get\_validation\_image(itr):

images, targets, image\_ids = next(itr)

images = list(img.to(device) for img in images)

targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

boxes = targets[0]['boxes'].cpu().numpy().astype(np.int32)

sample = images[0].permute(1,2,0).cpu().numpy()

model.eval()

outputs = model(images)

outputs = [{k: v.to(device) for k, v in t.items()} for t in outputs]

boxes = get\_boxes(outputs,0)

# boxes = outputs[1]['boxes'].cpu().detach().numpy().astype(np.int32)

sample = images[0].permute(1,2,0).cpu().numpy()

boxes = get\_boxes(outputs,0)

sample = images[0].permute(1,2,0).cpu().numpy()

sample = np.reshape(sample,(sample.shape[1],sample.shape[1]))

fig, ax = plt.subplots(1, 1, figsize=(16, 8))

for box in boxes:

cv2.rectangle(sample,

(box[0], box[1]),

(box[2], box[3]),

0, 2)

ax.set\_axis\_off()

ax.imshow(sample,cmap='gray')

def load\_test\_dataset():

data\_path = DIR\_TEST

test\_dataset = torchvision.datasets.ImageFolder(

root=data\_path,

transform=torchvision.transforms.Compose([

torchvision.transforms.Grayscale(num\_output\_channels=1),

torchvision.transforms.ToTensor(),]

))

test\_loader = torch.utils.data.DataLoader(

test\_dataset,

batch\_size=1,

num\_workers=1,

shuffle=False

)

return test\_loader

def get\_test\_image(itr,score = 0.5):

image, targets= next(itr)

sample = image

image = image.to(device)

model.eval()

outputs = model(image)

outputs = [{k: v.to(device) for k, v in t.items()} for t in outputs]

boxes = get\_boxes(outputs,0,score)

fig, ax = plt.subplots(1, 1, figsize=(16, 8))

print(sample.shape)

img = sample[0].permute(1,2,0).cpu().numpy()

print(img.shape)

img = np.array(img)

img = np.reshape(img,(img.shape[1],img.shape[1]))

print(img.shape)

for box in boxes:

x,y,w,h = box

cv2.rectangle(np.float32(img),

(int(box[0]), int(box[1])),

(int(box[2]), int(box[3])),

0, 2)

ax.set\_axis\_off()

ax.imshow(img,cmap='gray')

Sample of training data augmented

it = iter(train\_data\_loader)

get\_sample\_image(it)

A black and white photo of a flower

Description automatically generated with low confidence

get\_sample\_image(it)

A group of leaves

Description automatically generated with low confidence

get\_sample\_image(it)

A picture containing plant, tree

Description automatically generated

get\_sample\_image(it)

A picture containing text

Description automatically generated

get\_sample\_image(it)

A close-up of some leaves

Description automatically generated with low confidence

get\_sample\_image(it)

A close-up of a leaf

Description automatically generated with low confidence

get\_sample\_image(it)

A close-up of a plant

Description automatically generated with low confidence

get\_sample\_image(it)

A close-up of a leaf

Description automatically generated with low confidence

**Loading ResNet50 trained on COCO**

model = torchvision.models.detection.fasterrcnn\_resnet50\_fpn(pretrained=True)

num\_classes = 2

# 1 class (leaf) + background

# get number of input features for the classifier

in\_features = model.roi\_heads.box\_predictor.cls\_score.in\_features

# replace the pre-trained head with a new one

model.roi\_heads.box\_predictor = FastRCNNPredictor(in\_features, num\_classes)

model.to(device)

print("Model loaded")

Out- Model loaded

# Training

params = [p for p in model.parameters() if p.requires\_grad]

optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight\_decay=0.0005)

# lr\_scheduler = torch.optim.lr\_scheduler.StepLR(optimizer, step\_size=3, gamma=0.1)

lr\_scheduler = None

num\_epochs = 15

loss\_hist = Averager()

itr = 1

previous\_epoch = 1000

es\_rate = 0

es\_threshold = 2 # How many epochs without improvement to early stop

for epoch in range(num\_epochs):

loss\_hist.reset()

min\_loss = 1000

for images, targets, image\_ids in train\_data\_loader:

images = list(image.to(device) for image in images)

targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

loss\_dict = model(images, targets)

losses = sum(loss for loss in loss\_dict.values())

loss\_value = losses.item()

loss\_hist.send(loss\_value)

optimizer.zero\_grad()

losses.backward()

optimizer.step()

if itr % 50 == 0:

print(f"Iteration #{itr} loss: {loss\_value}")

itr += 1

# update the learning rate

if lr\_scheduler is not None:

lr\_scheduler.step()

min\_loss = loss\_hist.value

if min\_loss < previous\_epoch:

previous\_epoch = min\_loss

es\_rate = 0

else:

if es\_rate < es\_threshold:

es\_rate += 1

elif es\_rate >= es\_threshold:

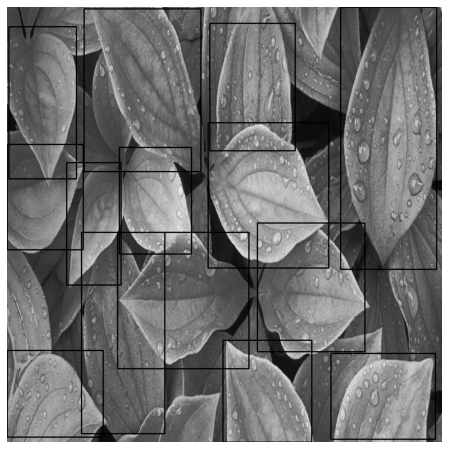
break

print(f"Epoch #{epoch} loss: {loss\_hist.value}")

# Validation (On data from Training)

it = iter(valid\_data\_loader)

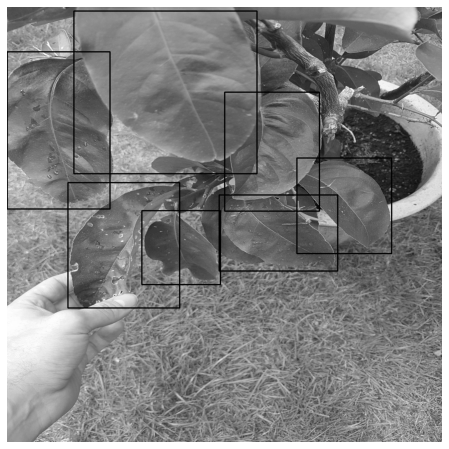
get\_validation\_image(it)



get\_validation\_image(it)



get\_validation\_image(it)



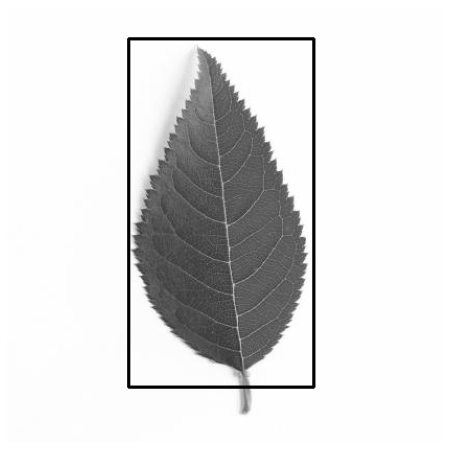
get\_validation\_image(it)



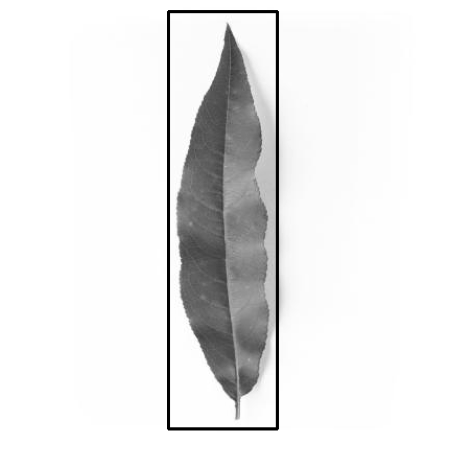
get\_validation\_image(it)



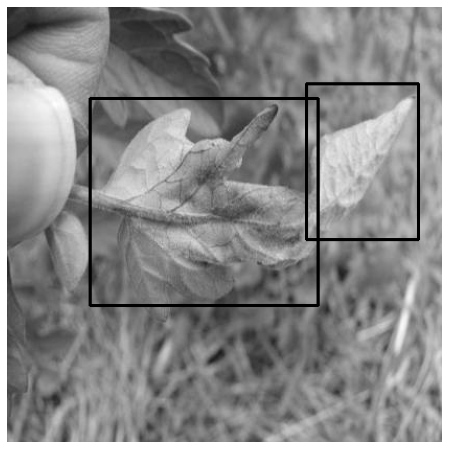
get\_validation\_image(it)



get\_validation\_image(it)



get\_validation\_image(it)



# Testing

image\_list = os.listdir(DIR\_TEST+"/leaf")

print(image\_list)

Out-

['TEST\_007.jpg', 'TEST\_001.jpg', 'TEST\_004.jpg', 'TEST\_005.jpg', 'TEST\_006.jpg', 'TEST\_002.jpg', 'TEST\_003.jpg']

it = iter(load\_test\_dataset())

start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

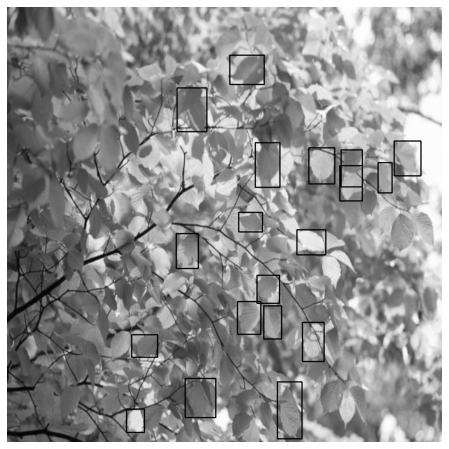
Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.16729450225830078



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

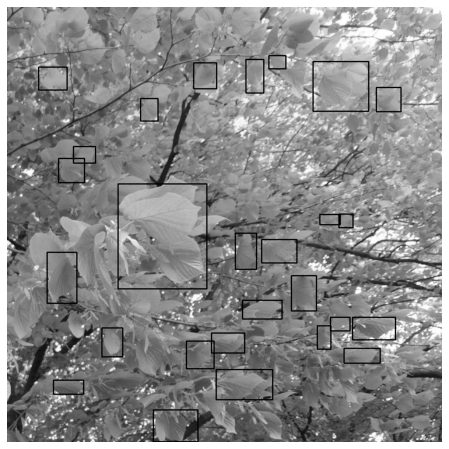
Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.10052013397216797



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

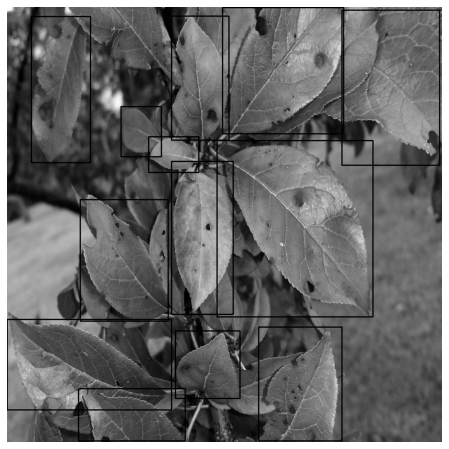
Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.09701108932495117



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

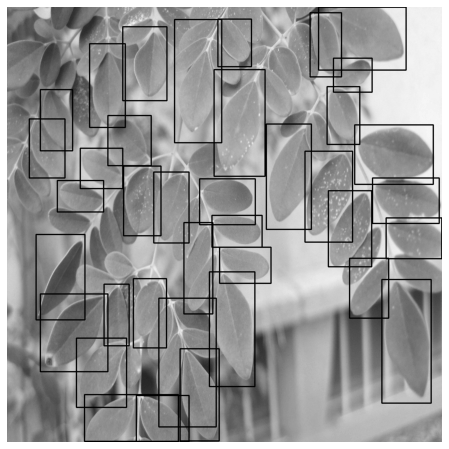
Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.09490728378295898



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.10325217247009277



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

0.09249734878540039



start = time.time()

get\_test\_image(it,0.5)

print(time.time()-start)

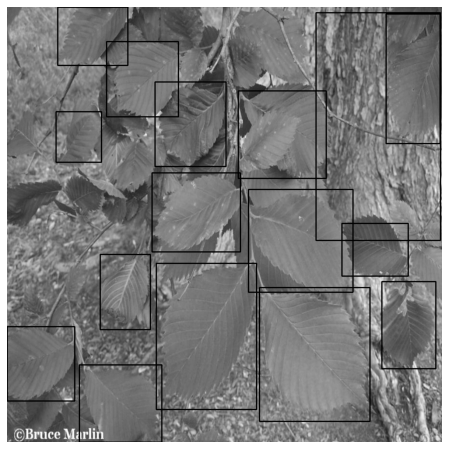
Out-

torch.Size([1, 1, 1024, 1024])

(1024, 1024, 1)

(1024, 1024)

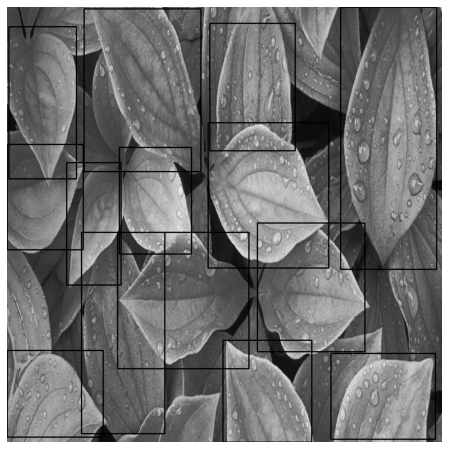
0.08995962142944336

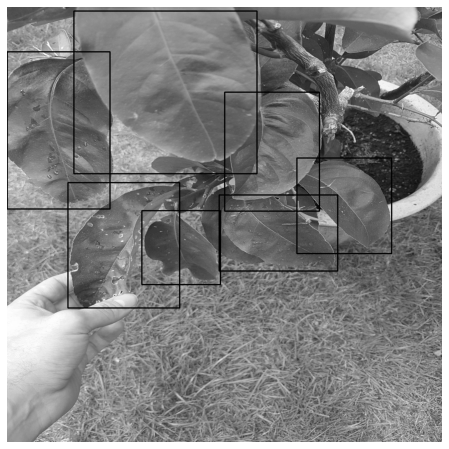


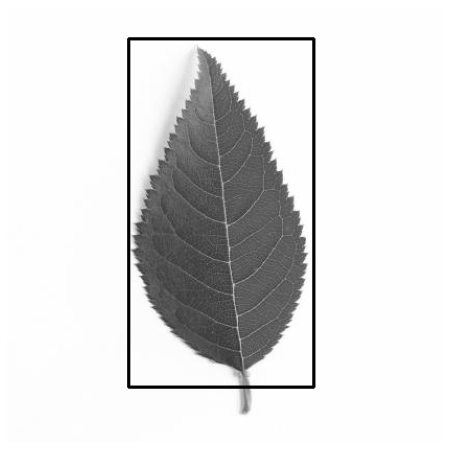
torch.save(model, 'leaves\_fasterrcnn\_model.pth')

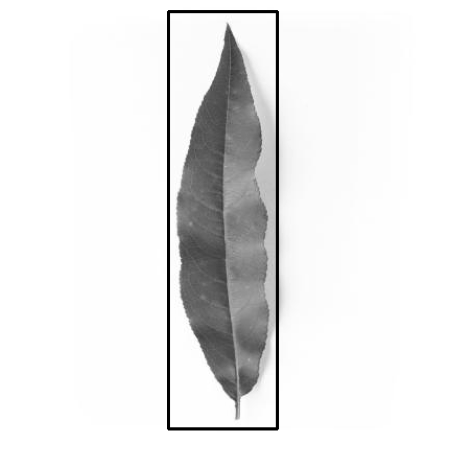
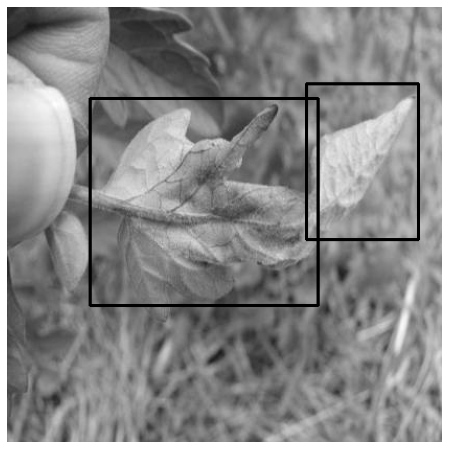
**RESULT**

**Validation Results**

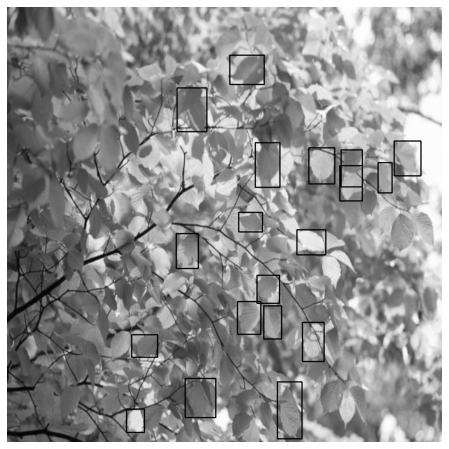
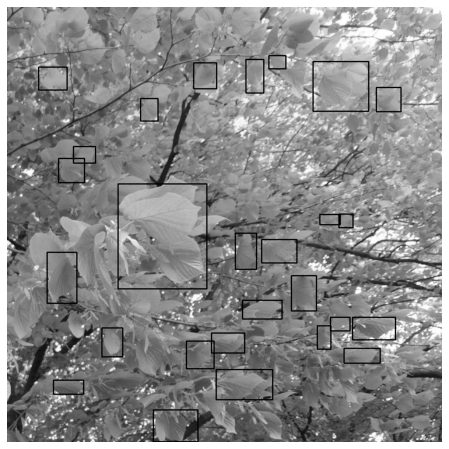
** **

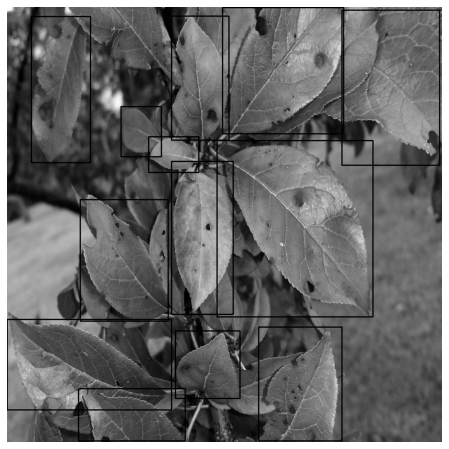
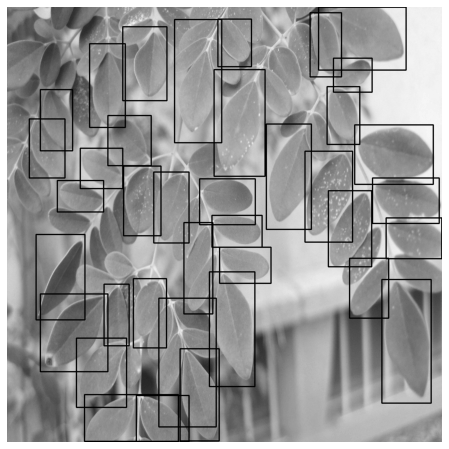
** **

** **

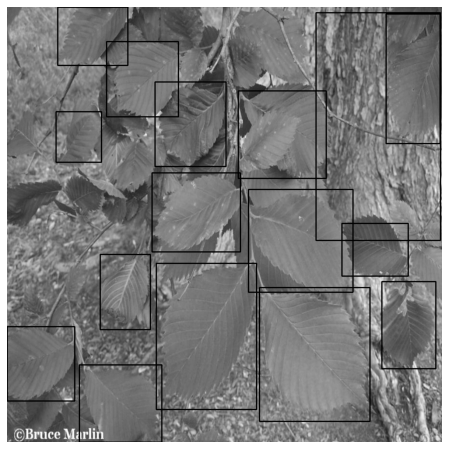
** **

**Testing Results**

** **

** **

** **

****

**CONCLUSION**

Leaf detection is a way of plant identification and disease detection in plants that uses the advantages of CNN and supervised learning. The main advantage is that the algorithm works with different orientations. It also could be the starting step for many of the leaf disease detection systems as it detects the leaf.

Detection of leaves is a problem that appears time and again as it is important to monitor growth patterns of plants and trees. This task, which seems effortless for humans, does not lend itself easily to computational approaches. Though human beings accomplish these tasks countless times a day, they are still very challenging for machine vision. Most of the researchers attach this kind of problem with leaf localization and feature selection with frontal view of leaves and without leaf features and normal lighting conditions although the variation between the images of the same leaf is too large due to the changes in leaves, shape, color, size, leaf position variation, lighting conditions, etc. In this study, leaf detection has been implemented using genetic algorithm to search for the leaf of a particular plant or tree in an image. The effectiveness of the leaf detection algorithm has been tested both in simple and complex backgrounds for different types of leaf and non-leaf images. The algorithm can detect the leaves in the images with different backgrounds and lighting conditions. Our next approach is to extend the algorithm for overlapping leaves in images and to detect leaf diseases and develop a gaze estimation algorithm that will be able to detect the different features in a leaf image and estimate any signs of disease in the plant.

**FUTURE SCOPE**

Leaf detection is a technique that is highly used to detect the difference in plants and detection of any signs of disease in the plants with the help of its amazing image processing technologies. It allows us to monitor and detect the growth pattern of plants through the changes in leaves.

There is a growth in the industry of leaf detection market due to the rising applications in various industry sectors. There are multiple growth factors available in the market that are growing the need of the leaf detection system for crop and food production market, market competition, wide use of technology in smartphones, rising plant differentiation and identification applications of IoT technology. The other factors of the growing market advantages of the leaf detection system rate are the increasing populations and in turn an increase in demand of food which can be fulfilled by good quality food production through continuously monitoring the plants.

# **REFERENCES**

1. <https://www.kaggle.com/alexo98/leaf-detection> - Kaggle (Leaf Detection)
2. <https://en.wikipedia.>org/wiki/Convolutional\_neural\_network - Wikipedia (Convolutional Neural Network)
3. <https://www.stackoverflow.com> – Stack Overflow (for query resolution)