A
Project Report
On
"Real-Time Leaf Detection using Jetson TX-2 in Deep Learning"

(IT345 –Software Group Project)

Prepared by

Riya Dhameliya (16IT021)

Under the Supervision of

Dr. Amit Thakkar

Submitted to

Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
in Information Technology (IT)
for 5th semester B.Tech

Submitted at

Accredited with Grade A by NAAC
Accredited with Grade A by KCG

DEPARTMENT OF INFORMATION TECHNOLOGY
(NBA Accredited)
Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist: Anand, Pin: 388421.
June, 2018

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Accredited with Grade A by NAAC

Accredited with Grade A by KCG

CERTIFICATE

This is to certify that the report entitled **"Real-Time Leaf detection using Jetson TX-2 in Deep Learning"** is a bonafied work carried out by **Riya Dhameliya(16it021)** under the guidance and supervision of **Dr. Amit Thakkar** for **5th** Semester of Bachelor of Technology in **Information Technology** at Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate herself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of,

Dr. Amit Thakkar
Associate Professor
Department of Information Technology
CHARUSAT,Changa,Gujarat

Prof. Pinal Patel
Assistant Professor
Department of Information Technology
CHARUSAT,Changa,Gujarat

Dr.Parth Shah
Head Of Department
Department of Information Technology
CHARUSAT,Changa,Gujarat

Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Ta. Petlad, Dist. Anand, Pin:388421. Gujarat

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude to **Dr. Amit Thakkar**, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the staff of Information Technology, who gave the permission to use all required equipment and the necessary materials to complete the task of "JETSON-TX2". A special thanks goes to the Head of Department **Prof. Parth Shah** for providing us an opportunity of Software Group Project.

I am obliged to all my faculties for their valuable guidance and co-operation in achieving the goal. I have to appreciate the guidance given by other supervisor as well as the panels especially in my project presentation that has improved my skills thanks to their comment and advices.

ABSTRACT

This project is for identifying plant species using automatic visual recognition. The system – called Leafsnap – identifies tree species from photographs of their leaves. Key to this system are computer vision components for discarding non-leaf images, segmenting the leaf from an untextured background, extracting features representing the curvature of the leaf's contour over multiple scales, and identifying the species from a dataset of the 184 trees in the Northeastern United States. Our system obtains state-of-the-art performance on the real-world images from the new Leafsnap Dataset – the largest of its kind. Throughout the paper, we document many of the practical steps needed to produce a computer vision system such as ours, which currently has nearly a million users.

CONTENT

Chapter 1: Introduction

1.1 DataSet:

Leafsnap is the first in a series of electronic field guides being developed by researchers from Columbia University, the University of Maryland, and the Smithsonian Institution. This free mobile app uses visual recognition software to help identify tree species from photographs of their leaves. Leafsnap contains beautiful high-resolution images of leaves, flowers, fruit, petiole, seeds, and bark. Leafsnap currently includes the trees of the Northeast and will soon grow to include the trees of the entire continental United States.

1.2 Scope:

- To recognize the leaf based on different spieces of 185 classes.
- To find plant spices by applying deep learning neural Network on Leaf.

1.3 Objective:

- Learn how CNN works and different models which we can use for image colorization.
- Comparison of models.
- Also we can colorize antique old images, black-white camera images into colorful images.

Chapter 2: Software and Hardware requirement

2.1 User characteristics:

The Leafsnap dataset, which consists of images of leaves taken from two different sources, as well as their automatically-generated segmentations:

- 1.23147 Lab images, consisting of high-quality images taken of pressed leaves, from the Smithsonian collection. These images appear in controlled backlit and front-lit versions, with several samples per species.
- 2.7719 Field images, consisting of "typical" images taken by mobile devices (iPhones mostly) in outdoor environments. These images contain varying amounts of blur, noise, illumination patterns, shadows, etc.
- The dataset currently covers all 185 tree species from the Northeastern United States.

2.2 Tools and Technology used:

Hardware:

- Computer/Laptop running i3 or high
- RAM 4 GB or high
- GPU NVIDIA GeForce 940 or high (>=900 FLOPS)
- JETSON-TX2

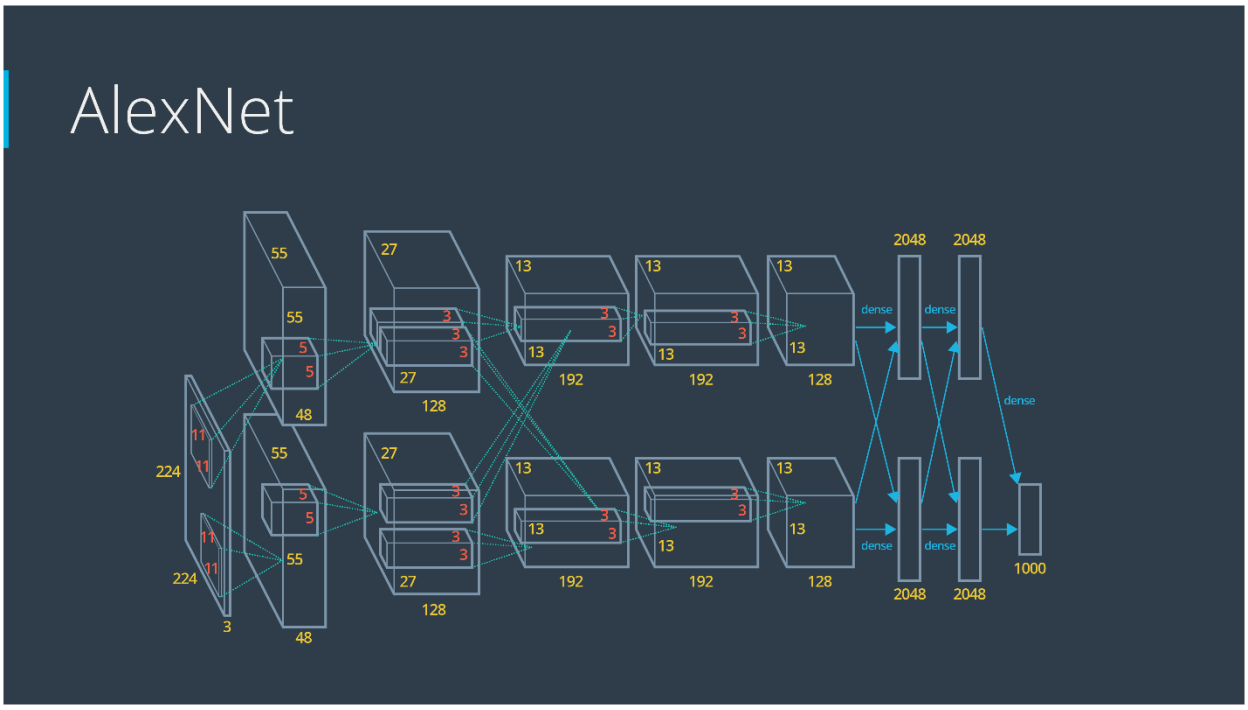
Software:

- Python Interpreter & Package Manager (pip)
- Python libraries: Tensorflow, SciPy

2.3 Model comparison:

We have to compare all the models using same dataset and noticeable parameters are complexity in time and accuracy.

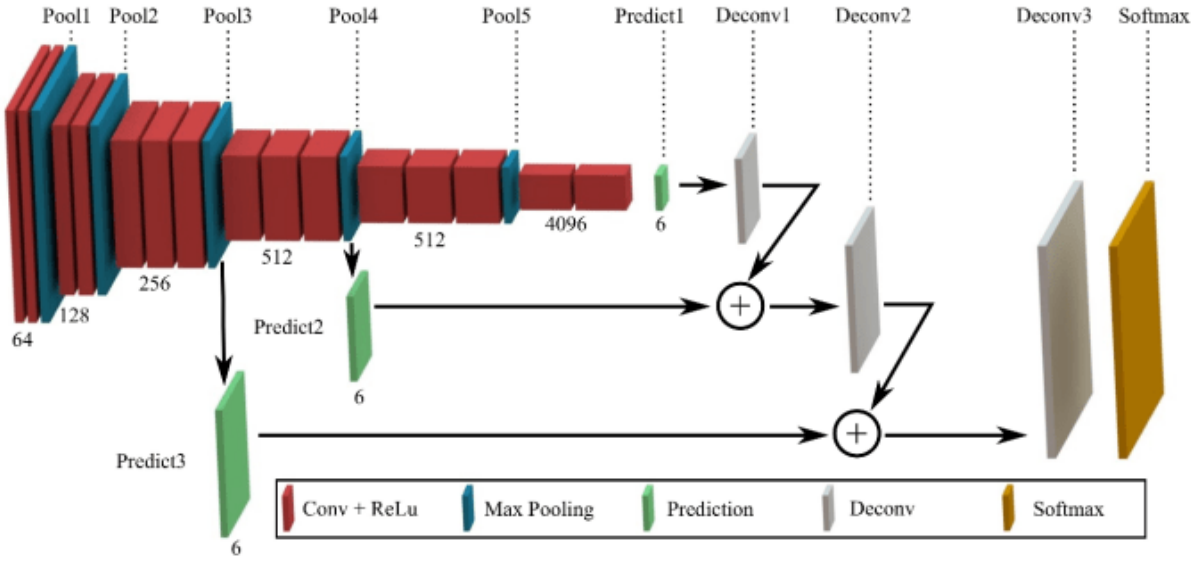
Basic structure of AlexNet:



The advantage of the ReLu over sigmoid is that it trains much faster than the latter because the derivative of sigmoid becomes very small in the saturating region and therefore the updates to the weights almost vanish. This is called vanishing gradient problem.

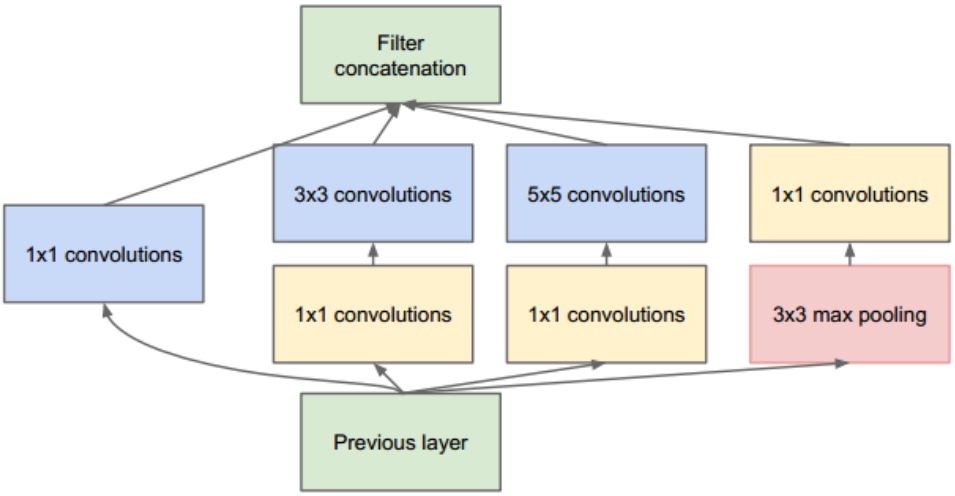
Another problem that this architecture solved was reducing the **over**-fitting by using a Dropout layer after every FC layer.

Basic structure of VGG-16:



It makes the improvement over AlexNet by replacing large kernel-sized filters(11 and 5 in the first and second convolutional layer, respectively) with multiple 3X3 kernel-sized filters one after another. because multiple non-linear layers increases the depth of the network which enables it to learn more complex features, and that too at a lower cost. It achieves the top-5 accuracy of 92.3 % on ImageNet.

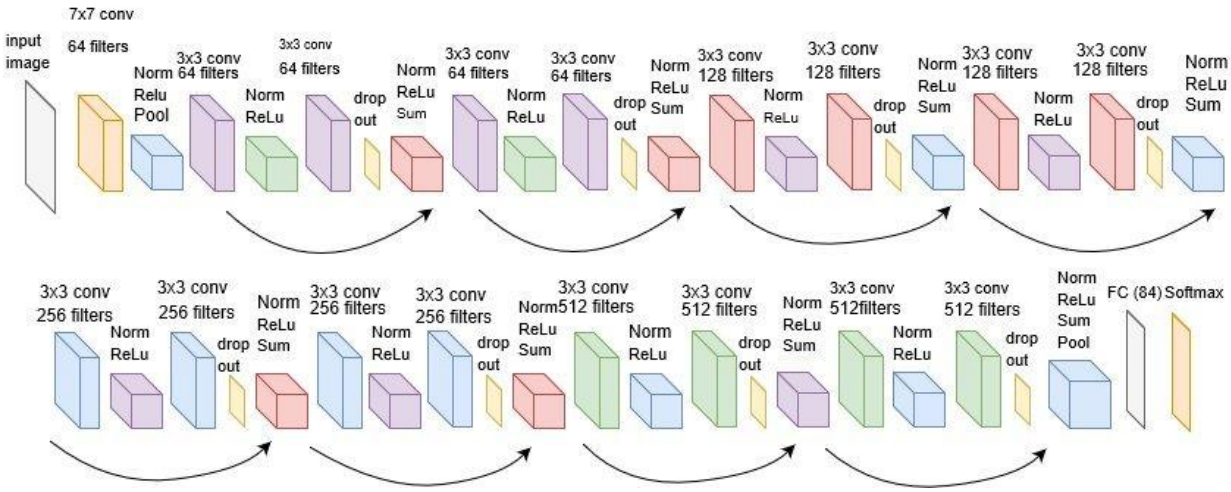
Basic structure of Inception/GoogleNet:



VGG achieves a phenomenal accuracy on ImageNet dataset, its deployment on even the most modest sized GPUs is a problem because of huge computational requirements, both in terms of memory and time. It becomes inefficient due to large width of convolutional layers.

It uses convolutions of different sizes to capture details at varied scales (5X5, 3X3, 1X1). Another change that GoogLeNet made, was to replace the fully-connected layers at the end with a simple global average pooling which averages out the channel values across the 2D feature map, after the last convolutional layer. This drastically reduces the total number of parameters. This can be understood from AlexNet, where FC layers contain approx. 90% of parameters. Use of a large network width and depth allows GoogLeNet to remove the FC layers without affecting the accuracy. It achieves 93.3% top-5 accuracy on ImageNet and is much faster than VGG.

Basic structure of ResNet:

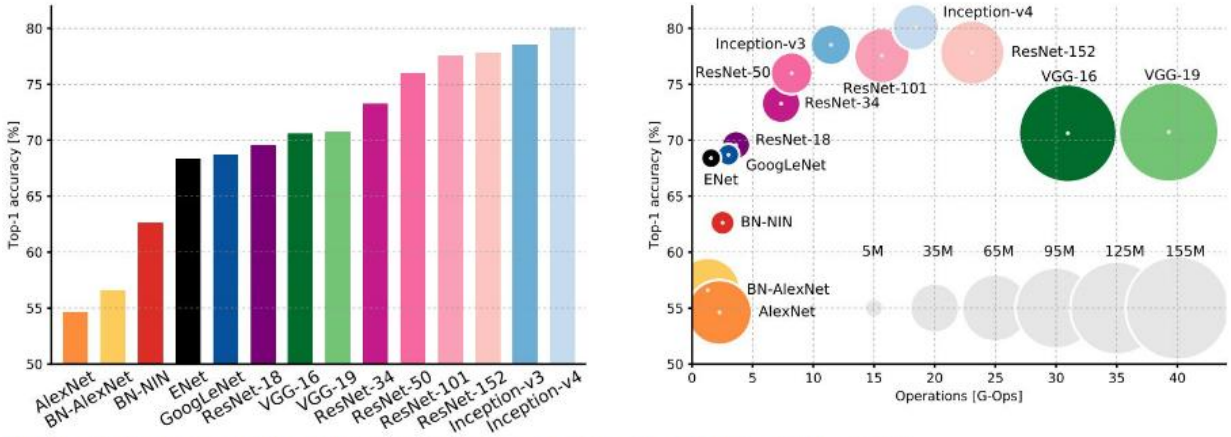


Similar to GoogLeNet, it uses a global average pooling followed by the classification layer.Through the changes mentioned, ResNets were learned with network depth of as large as 152. It achieves better accuracy than VGGNet and GoogLeNet while being computationally more efficient than VGGNet. ResNet-152 achieves 95.51 top-5 accuracies.

The architecture is similar to the VGGNet consisting mostly of 3X3 filters. From the VGGNet, shortcut connection as described above is inserted to form a residual network. This can be seen in the figure which shows a small snippet of earlier layer synthesis from VGG-19.

Some diagram based on comparisons:

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

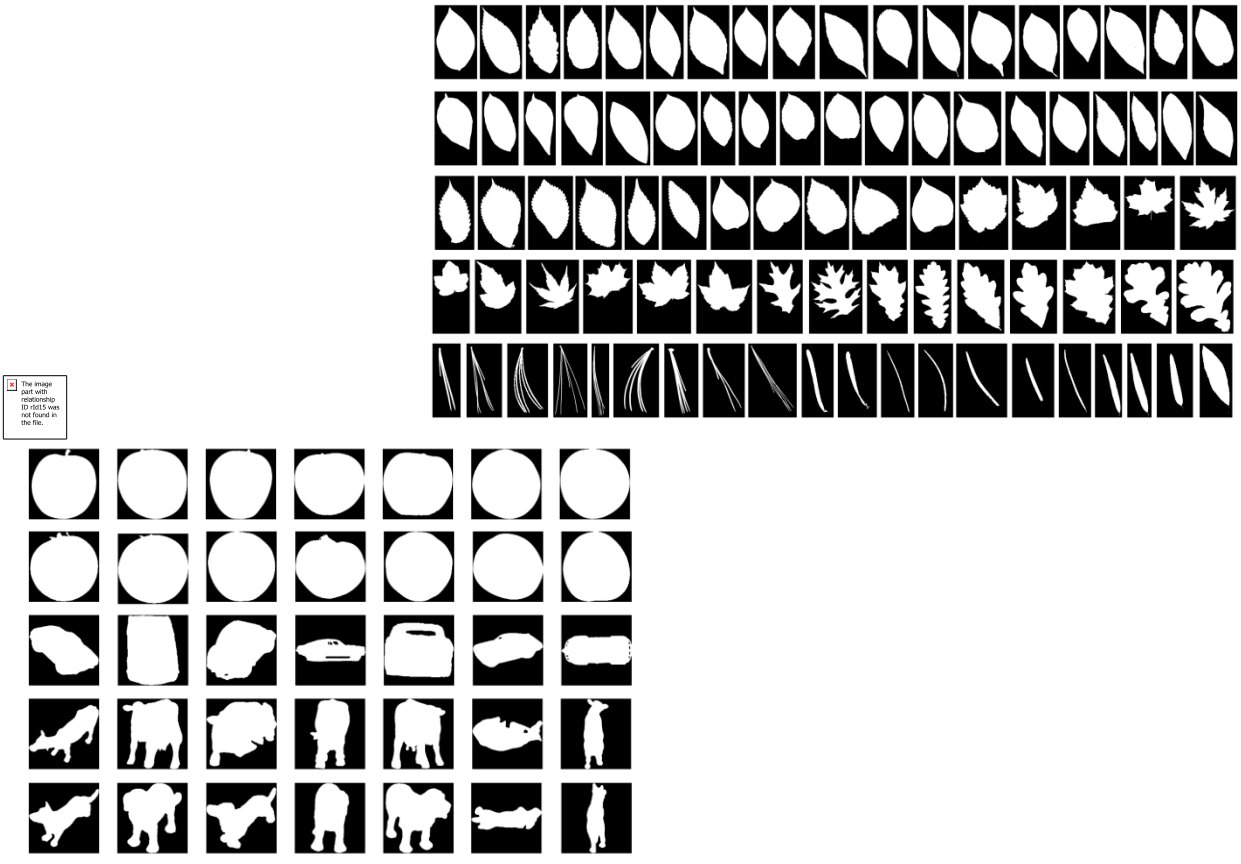
2.4 JETSON-TX2

Jetson TX2 is the fastest, most power-efficient embedded AI computing device. The latest addition to the industry-leading Jetson embedded platform, this 7.5-watt supercomputer on a module brings true AI computing at the edge. It's built around an NVIDIA Pascal™-family GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth. It features a variety of standard hardware interfaces that make it easy to integrate it into a wide range of products and form factors.

Chapter 3:Classification and Recognition

3.1 Leaf/Non-Leaf Classification

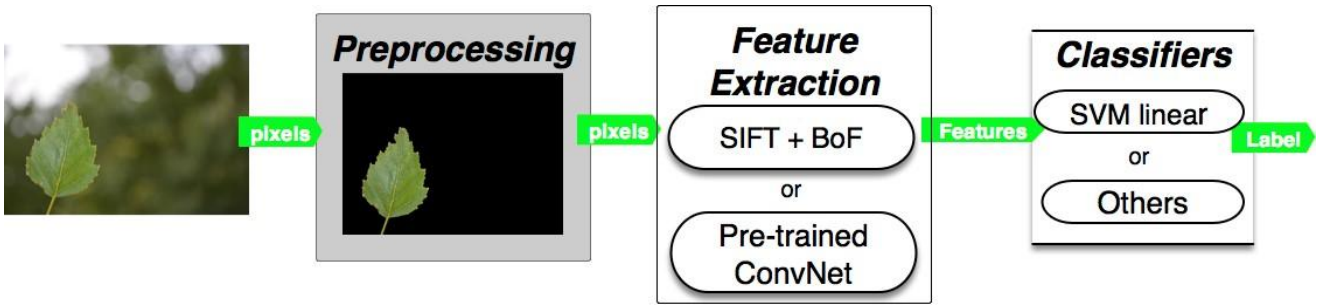
Untrained users initially try to take photos of leaves in-situ, with multiple leaves present amid clutter, often with severe lighting and blur artifacts, resulting in images that we cannot handle (usually due to segmentation failures). In addition, many users also take photos of objects that are not leaves. We address both of these issues by rst running a binary leaf/non-leaf classifier on all input images. If this classifier detects that an input image is not valid { of a single leaf, placed on a light, untextured background with no other clutter we inform the user of this fact and instruct them on how to take an appropriate image. We found this simple procedure very helpful for training users without the need for long tutorials or help pages, which often go unread. It also greatly reduces the computational load on our server, as images that fail this classi cation (about 37:1%) are discarded from further processing.



3.2 Color-Based Segmentation

The system uses the distinctive shapes of leaves as the sole recognition cue. Other features such as the color of the leaf, its venation pattern, or images of the owers are not suitable for various reasons they are either too highly variable across different leaves of the same species, undetectable due to the poor imaging capabilities of most mobile phone cameras, or only present at limited times of year. Reliable leaf segmentation is thus crucial in order to obtain shape descriptions that are sufficiently accurate for recognition. The requirement of photograph leaves is against a light, untextured background; however, even with this requirement, segmentation is challenging due to shadows, blur, ne-scale structures on leaves (such as serrations and thin stems), and specular reflections.

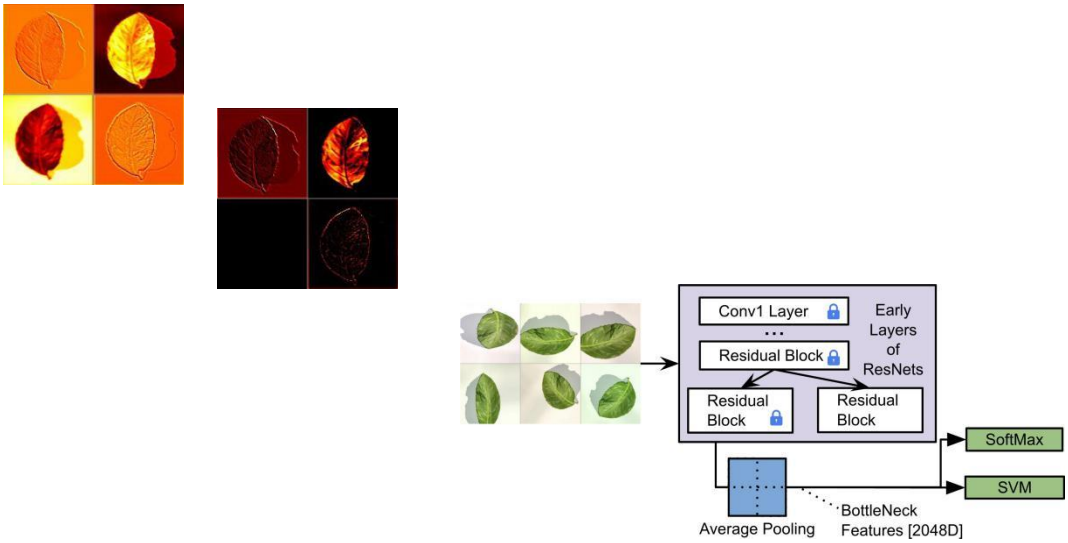
The images are segmented by estimating foreground and background color distributions and using these to independently classify each pixel. This initial segmentation, solved using Expectation-Maximization, is then post-processed to remove false positive regions and the leaf stem. The color-based segmentation has several advantages compared to other approaches. Leaves vary greatly in shape. Some species of leaves are compound (consisting of small leaflets); others are found grouped into clusters (e.g., pine needles). This gives rise to complex segmentation boundaries that are difficult to handle for edge-based methods, or region-based methods that bias towards compact shapes. The color-based pixelwise approach works much better. In addition, by not making absolute assumptions about the color distributions, our clustering approach is able to adapt to major sources of color variability such as lighting changes and natural variations in leaf color (i.e., although many leaves are a deep green, others have yellowish or brownish hues).



3.3 Curvature-Based Shape Features

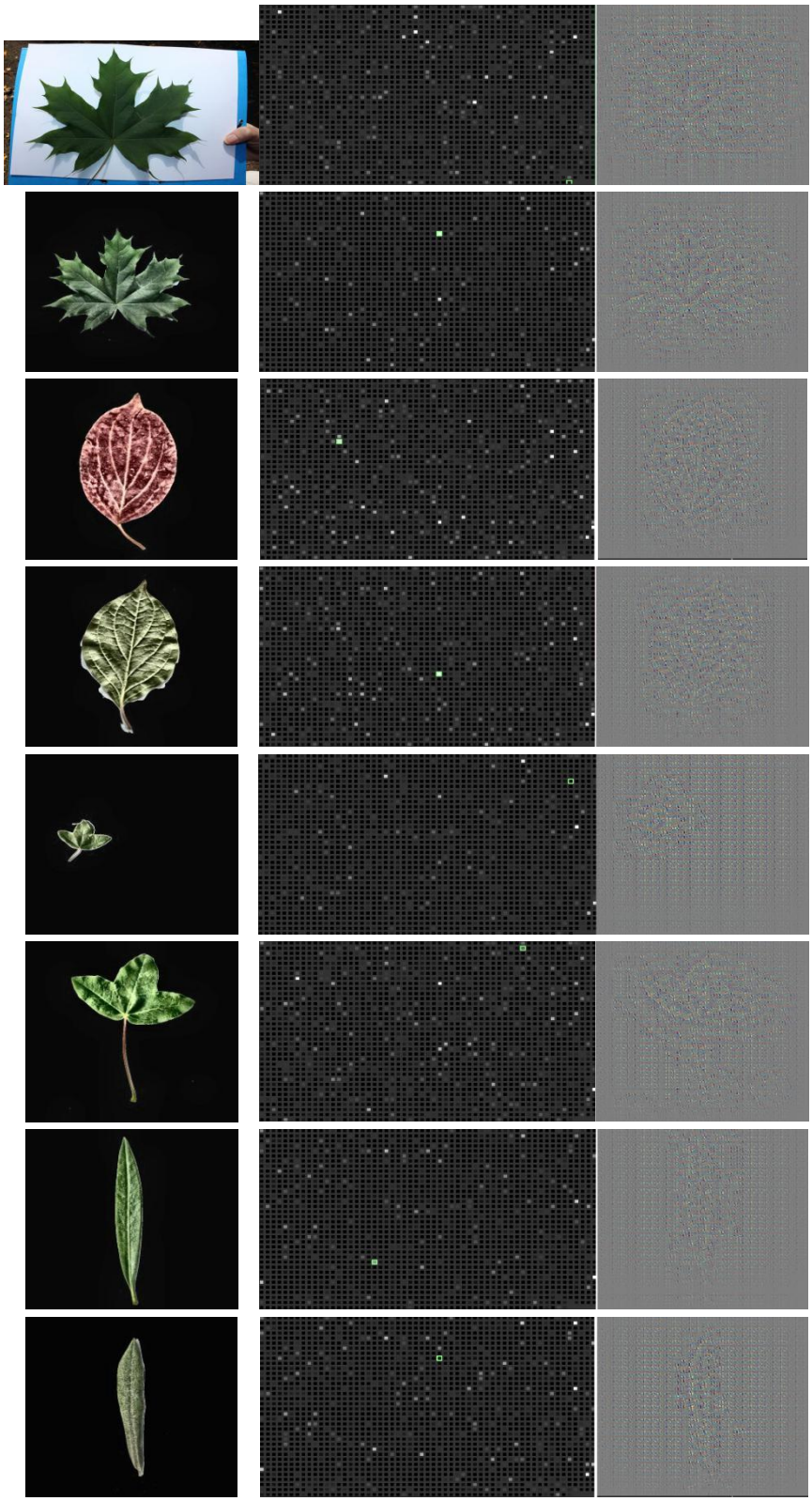
Leaf shape can be effectively represented using multiscale curvature measures. The below figure shows a diagrammatic example with segmented images of four different leaf species, and histograms of curvature along the boundary for each shape, computed at two different scales { coarse (large radius) on top, and near (small radius) on bottom. The pair of images in each row share the same general shape, but the images on the left have a smooth boundary, and the ones on the right have a serrated boundary. Thus, the histograms of coarse-scale values differ for each row, while the histograms of neffscale values differ for each column. By using both histograms together, we can distinguish these shapes from each other.

With the help of visualization, intuitively one can see that each filter is seizing different features. The output from later layers is more abstract and global than the earlier layers. We can also see that some of the filters are completely dark, means for that particular filter, it doesn't response to certain feature of the image after rectification.



Transfer Learning with ResNet

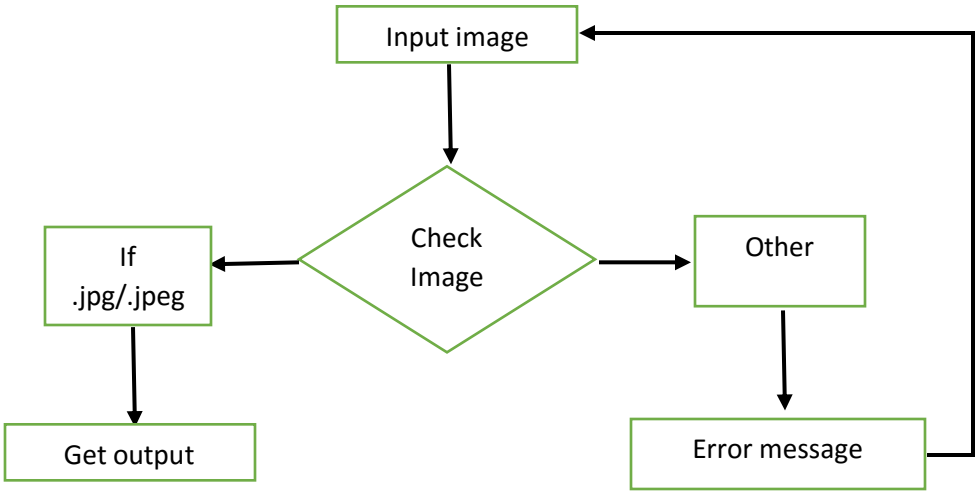
Thus, one can assume that the pre-trained weight is applicable to our problem, at least the early layers do generalize well to our feature space.



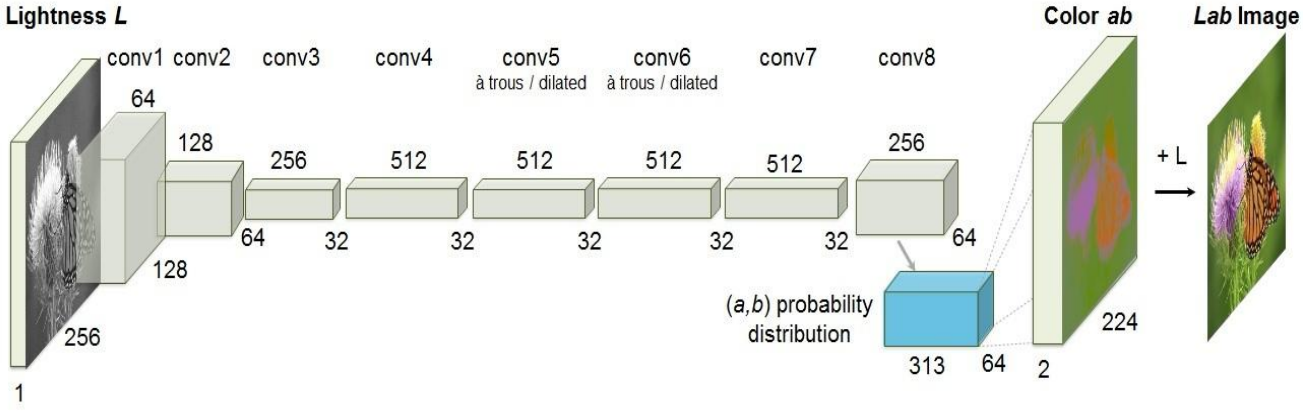
Chapter 4: System Design

4.1 Project Flow:

Flow Chart I (User Interface):



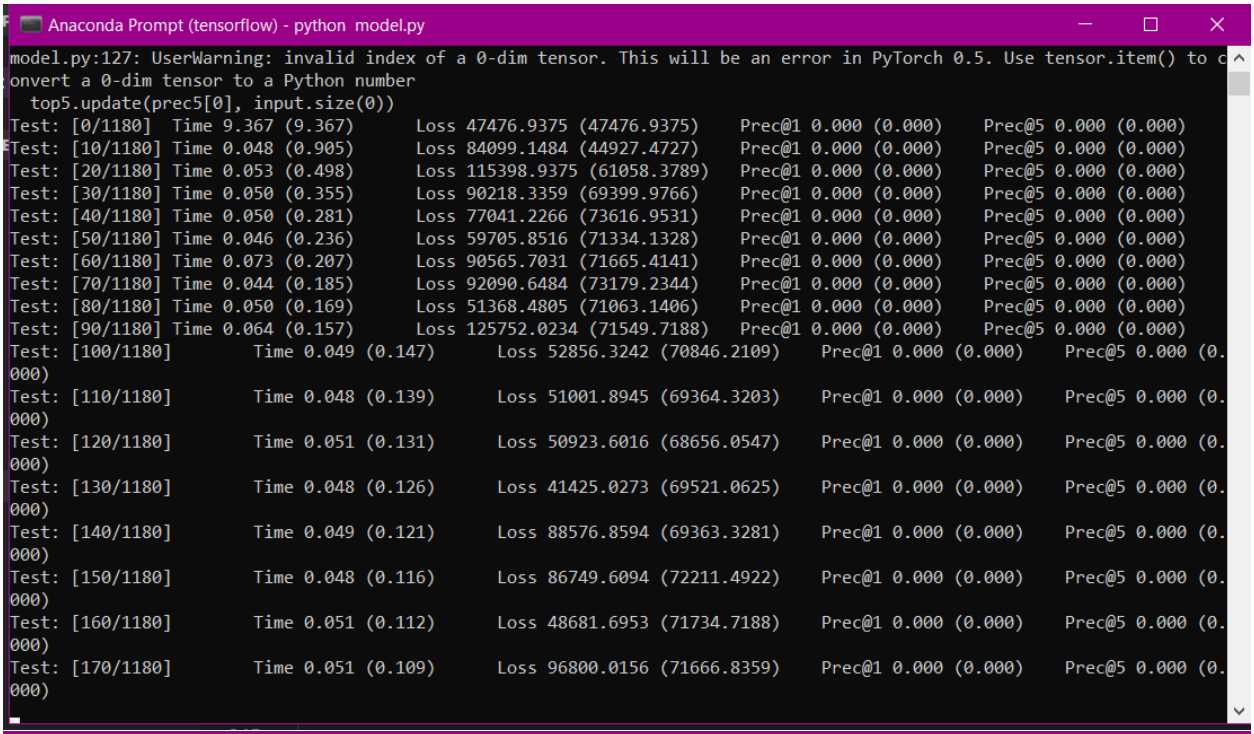
Flow Chart II (Model):

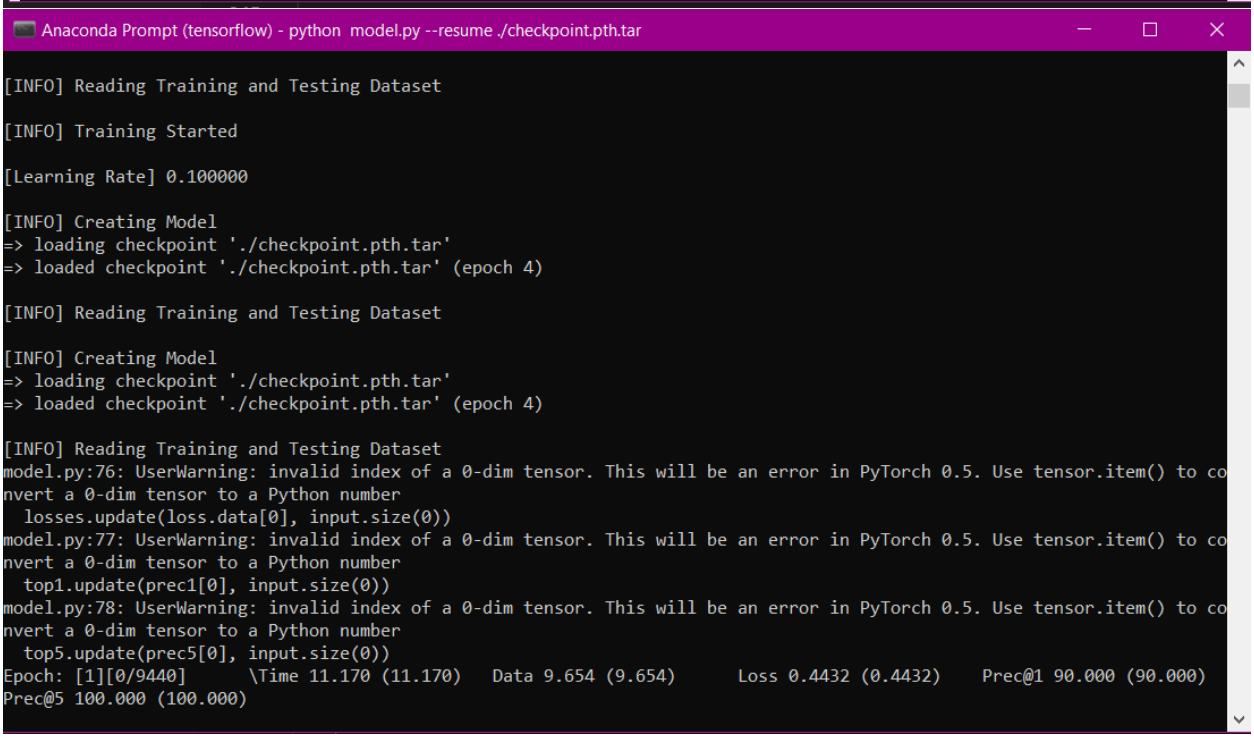


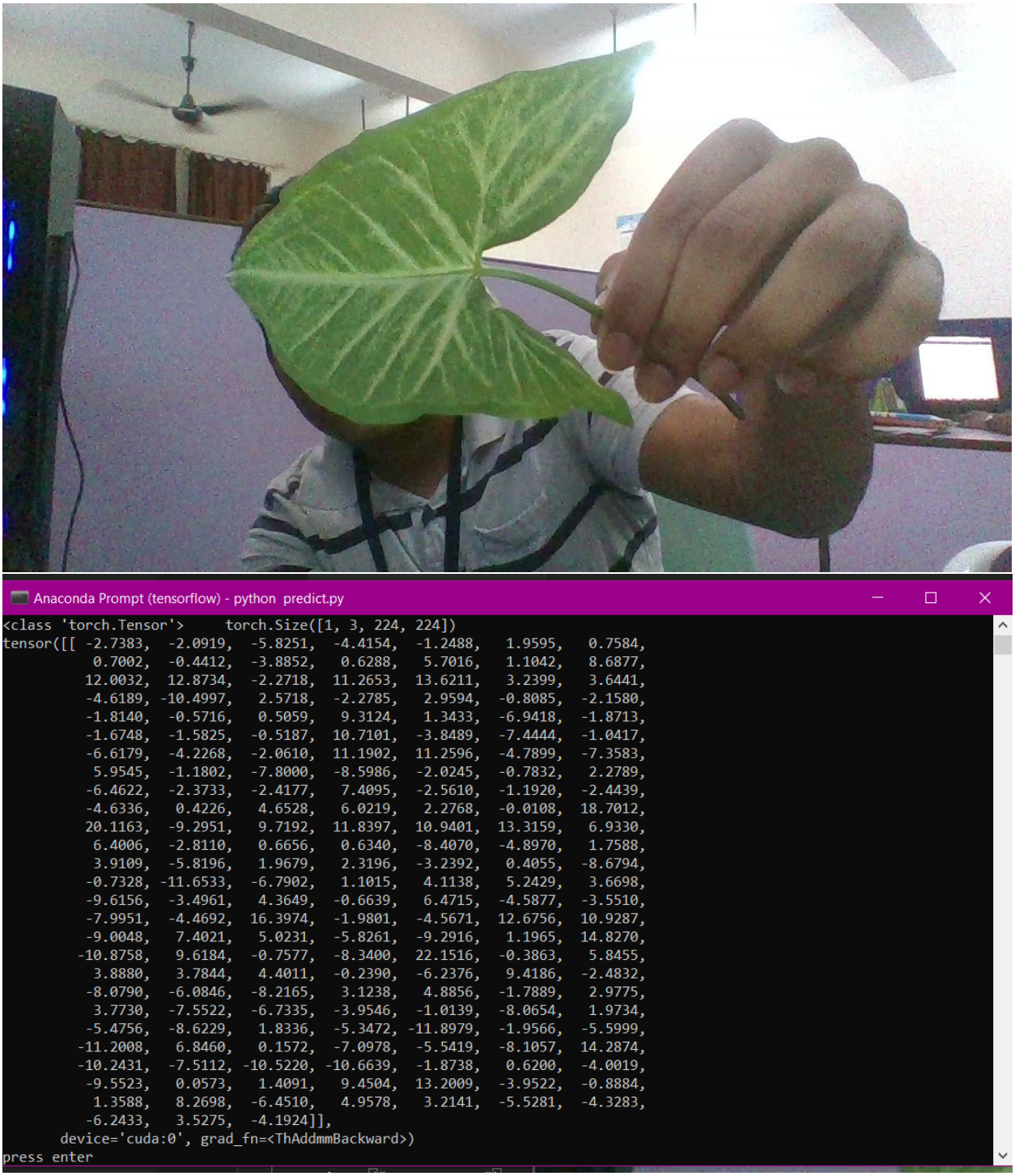
4.2 Major functionality:

- Provides visible interface for an AI means we are doing work on images so we can see working of model how accurate it is easily and change according to it.
- User can enter image to website from his computer and can get output on website.

4.3 Training and Testing Data






Anaconda Prompt (tensorflow) - python predict.py
<class 'torch.Tensor'> torch.Size([1, 3, 224, 224])
tensor([[-2.7383, -2.0919, -5.8251, -4.4154, -1.2488, 1.9595, 0.7584,
0.7002, -0.4412, -3.8852, 0.6288, 5.7016, 1.1042, 8.6877,
12.0032, 12.8734, -2.2718, 11.2653, 13.6211, 3.2399, 3.6441,
-4.6189, -10.4997, 2.5718, -2.2785, 2.9594, -0.8085, -2.1580,
-1.8140, -0.5716, 0.5059, 9.3124, 1.3433, -6.9418, -1.8713,
-1.6748, -1.5825, -0.5187, 10.7101, -3.8489, -7.4444, -1.0417,
-6.6179, -4.2268, -2.0610, 11.1902, 11.2596, -4.7899, -7.3583,
5.9545, -1.1802, -7.8000, -8.5986, -2.0245, -0.7832, 2.2789,
-6.4622, -2.3733, -2.4177, 7.4095, -2.5610, -1.1920, -2.4439,
-4.6336, 0.4226, 4.6528, 6.0219, 2.2768, -0.0108, 18.7012,
20.1163, -9.2951, 9.7192, 11.8397, 10.9401, 13.3159, 6.9330,
6.4006, -2.8110, 0.6656, 0.6340, -8.4070, -4.8970, 1.7588,
3.9109, -5.8196, 1.9679, 2.3196, -3.2392, 0.4055, -8.6794,
-0.7328, -11.6533, -6.7902, 1.1015, 4.1138, 5.2429, 3.6698,
-9.6156, -3.4961, 4.3649, -0.6639, 6.4715, -4.5877, -3.5510,
-7.9951, -4.4692, 16.3974, -1.9801, -4.5671, 12.6756, 10.9287,
-9.0048, 7.4021, 5.0231, -5.8261, -9.2916, 1.1965, 14.8270,
-10.8758, 9.6184, -0.7577, -8.3400, 22.1516, -0.3863, 5.8455,
3.8880, 3.7844, 4.4011, -0.2390, -6.2376, 9.4186, -2.4832,
-8.0790, -6.0846, -8.2165, 3.1238, 4.8856, -1.7889, 2.9775,
3.7730, -7.5522, -6.7335, -3.9546, -1.0139, -8.0654, 1.9734,
-5.4756, -8.6229, 1.8336, -5.3472, -11.8979, -1.9566, -5.5999,
-11.2008, 6.8460, 0.1572, -7.0978, -5.5419, -8.1057, 14.2874,
-10.2431, -7.5112, -10.5220, -10.6639, -1.8738, 0.6200, -4.0019,
-9.5523, 0.0573, 1.4091, 9.4504, 13.2009, -3.9522, -0.8884,
1.3588, 8.2698, -6.4510, 4.9578, 3.2141, -5.5281, -4.3283,
-6.2433, 3.5275, -4.1924]],
device='cuda:0', grad_fn=<ThAddmmBackward>)
press enter

4.4 Coding standards:

```
//CODE
import argparse
import cv2
import json
import numpy as np
import os
import pandas as pd
import scipy.misc
import shutil
import time
import torch
import torch.backends.cudnn as cudnn
import torch.nn as nn
import torch.nn.functional as F
import torch.nn.parallel
import torch.optim as optim
import torchvision
import torchvision.models as models
import utils

from PIL import Image
from averagemeter import *
from models import *
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from torch.autograd import Variable
from torch.utils.data import sampler
from torchvision import datasets
from torchvision import transforms

model = models.resnet101(pretrained=True)
model.fc = nn.Linear(2048, 185)
model = torch.nn.DataParallel(model).cuda()
checkpoint = torch.load('checkpoint.pth.tar')
model.load_state_dict(checkpoint['state_dict'])
min_img_size = 224
transform_pipeline =
transforms.Compose([transforms.Resize(min_img_size),
                                        transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225])])
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
traindir = os.path.join('dataset', 'train')
data_train = datasets.ImageFolder(traindir, transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
```

```
    normalize]))
classes=data_train.classes
model.eval()
stream=cv2.VideoCapture(0)
while True:
    frame1=stream.read()
    frame=transform_pipeline(frame1)
    output = model(frame)
    cv2.imshow('frame',frame1)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    print(output)
```

Chapter 5: Limitations and Future Enhancement

Limitation:

- To get more accurate output we want more powerful GPU and more RAM.
- More training time required for accurate models.
- After all training is completed then also if some awkward image we give as an input we may get different output.
- To implement more accurate model we want more data.

Future Enhancement:

Plant leaf Recognition plays a very important role in Agricultural fields where any of the annonymous leafs can be detected.This can be further extended for detecting the disease from which the plant is affected,so that the precautions can be taken by people from preventing the damage to be caused.

Chapter 6: Conclusion

We explore deep ConvNet with leaf classification problem. CNN codes and simple linear SVM gives the best results for both clean and noisy datasets. Further investigation gives some insights on impacts of background clutter, color/scale variations on CNN codes.Anyways these are the certain activities to be taken care of

1)Acquire more data and fine-tune ConvNets to fight over- fitting problem
2)Engage advanced techniques for image augmentation
3)Explore state-of-art method to detect and locate leaf from background. This should allow ConvNet to focus on the discriminating features.

References

- http://flask.pocoo.org/docs/0.12/
- arXiv:1603.08511
- http://caffe.berkeleyvision.org/
- https://www.tensorflow.org/api_docs/python/
- https://www.scipy.org/
- https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/
- https://keras.io/
- http://www.image-net.org/
- https://github.com/google/inception
- https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624