PLANT IDENTIFICATION

A Project Report Of Capstone Project – 2

Submitted by

YASH SHARMA

(16SCSE101420)

in partial fulfillment for the award of the degree

B.TECH

IN

COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

UNDER THE SUPERVISION OF

DR. R. RAJKUMAR ,ASST. PROFESSOR

MAY 2020

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**PLANT IDENTIFICATION USING LEAVES"** is the bonafide work of "**YASH SHARMA(16SCSE101420)**" who carried out the project work under my supervision.

SIGNATURE OF HEAD	**SIGNATURE OF SUPERVISOR**
DR.MUNISH SHABARWAL	DR. SANJEEV KUMAR PIPAL,
Phd(Management),Phd(CS)	M.Tech,Ph.D.,
Professor & Dean,	**Professor**
School of Computing Science & Engineering.	**School of Computing Science & Engineering.**

TABLE OF CONTENTS

ABSTRACT

To present the Plant identification software with some functionalities and modules . and to identify some common plants and provide information about them. and saving results for searched plants and also for improvement of application by saving and providing unknown plants and flowers. Out of all available organs of plant, leaf is selected to obtain the features of plant. Plant identification is the process of matching a specimen plant to a known taxon. It uses various methods, most commonly single-access keys or multi-access keys.

We are identifying plants by using image classifier which uses machine learning which is implemented in this using tensorflow js . tensorflow provides database for machine learning .

By which we class identify different species of plants, leaves, flowers , barks this can be done by clicking a photo with your camera and uploading image in software and get identify the plant in just few seconds.

1. INTRODUCTION

Now, learning more about plants is just a photo click away. The Plant Identification app is a journal guide for all types of plants, flowers, mushrooms, trees. A user just has to click a photo and an instant report about that particular plant is shown through the mobile app.

The photo is stored safely in the app, and a user can go through that photo and its related details whenever he wants to recall the plant. The most important in any plant identifier app development is the database that is inbuilt in the app for quick information.

The unique algorithm of Plant identification makes it a very reliable and quick resolver for any kind of plant or flower on the globe. To make sure that plant app has such a powerful database to provide your customers with a vivid variety of nature's knowledge.

In these plant identifiers application, a user can click a photo and get desired information. He can also organically search for the plant species name by "search bar". You can create an app for plants with a "photo drag" feature. A user who is in a hurry to reach somewhere pauses for a bit to take a photo of an unknown plant. He can put the photo afterward in the plant identification app. Instead of clicking a photo opening the app at that very moment, your app development will create this exception.

1.1 OBJECTIVE

To increase the awareness of distinct plants, flowers, trees this is a simple practical learning based activity which provides details about plants by clicking real plant photo and collecting its information.

It also helps to identify unknown plants and let you know about them just by clicking their pictures.

1. To improve yields.

2. Protecting Crops.

3. Much more reliable source of food and medicines.

4. Better Plantation and caring of Plants.

5. Identification of poisonous plants.

1.2 SCOPE

It can help to know about undiscover plants . if no data is matched and can provide more accurate predictions for plants by collecting real images and adding them to databases from users.

1.3 METHODOLOGIES USED:

1. Machine learning is a data analytics technique that teaches computers to do what comes naturally to humans and animals: learn from experience. Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. The algorithms adaptively improve their performance as the number of samples available for learning increases. Deep learning is a specialized form of machine learning.

2. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

3. TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

It is used fpr image classification and using neural network for identification .

2. LITERATURE REVIEW

Plant identification is the process of matching a specimen plant to a known taxon. It uses various methods, most commonly single-access keys or multi-access keys.

The ability to know, or identify, plants allows us to assess many important rangeland or pasture variables that are critical to proper management: range condition, proper stocking rates, forage production, wildlife habitat quality, and rangeland trend, either upward or downward.

Natural resource managers, especially those interested in grazing and wildlife management, must be able to evaluate the presence or absence of many plant species in order to assess these variables.

3. PROBLEM STATEMENT

There are various Plant identification softwares available in this world developed and researched by various other developers. The variety of Plant identification softwares provide user with the plenty of features and characteristics. As it goes the saying that you cannot have happiness without sorrows, applies here. Every software has a drawback which pulls it back from the race. The pros and cons of the various Plant identification softwares are given in details here. And to overcome these problems we are devising the versatile Plant identification software system.

Could use better access to identify plants, discover plants, to get information about plants to know about distinct species of plants,

Many images are not clear which creates problem to identify plant.

Plants may not match to any species in the database or matches to more than 2 or more plants. Same types of leaves are also problem.

- Many plants having similar type of leaves are more pragmatic to identify.

Environmental damages to plants due to various causes.

- Plant form or shape
- Plant size
- Where the plant is growing
- Site characteristics: Is the plant growing in wet or dry conditions, or in a sunny or shady area?

- What are the color and sizes of any seeds or fruit? What is the fall color of plant.
- Bark characteristics: Is the bark smooth, or does it have a rough or flaky texture.

4. PROPOSED MODEL

- Purposed model we add the following functionalities and tools to make the current version more better than the previous one by fulfilling the changes mentioned in scope of plant identification and other tools management.
- This model can also identify flowers, leaf, barks and fruits
- It shows the regions where the particular type of plant can be found.
- Where the plants typically grow .
- caring tips for plants.
- Ordering of plants.
- Better AI system for identification in low vision and in more generalised way.
- To improve yields by farming good quality of crops. It identify disease in plants and leaves.

4.2 RESEARCH AREA

This paper presents a computer based automatic plant identification system. Out of all available organs of plant, leaf is selected to obtain the features of plant. Five geometrical parameters are calculated using digital image processing techniques. On the basis of these geometrical parameters six basic morphological features are extracted.

Vein feature as a derived feature is extracted based on leaf structure. At the first stage leaf images are obtained using digital scanner. Then above mentioned morphological features are extracted which act as input to the classification stage. Recognition accuracy of the proposed algorithm is tested. Accuracy of this algorithm is tested on two different databases and compared. False acceptance ratio and false rejection ratio for both databases is calculated. Total 12 kinds of plants are classified using this algorithm. Dataset consists of 92 images of total 12 plants. This method implements effective algorithm used for plant identification and classification as it is independent of leaf maturity. Proposed method is easy to implement and fast in execution. This research paper uses Euclidean classifier and statistical approach for identifying plants.

Following are the features which are resulted by the research conducted for plant identification.

- Feature extraction.
- Plant identification.
- vein feature.

- false acceptance rate.
- false rejection rate.

5. EXISTING SYSTEM

In Existing models of plant identification there are so many applications which are based on different api , different architecture different methods and technologies . there can be many versions of applications on same technology like neural networks, big data , deep learning and can be implemented using javascript ,python ,java ,tensorflow .

It is possible by using machine learning algorithms and classification techniques such as image classification , object detection ,Artificial intelligence and many other ways.

So the existing models are so many but none of them gives us full accuracy and this is not Possible unless we have data of every plant species of this universe . so every method try To get to the maximum accuracy with the available data.

In market there are so many apps available which identify plants , flower , leaves by images Within few seconds and provides every related info. Just Take a photo and upload it, let us identify it with our 'magic' and view the results in seconds. Make sure you have clear photos to be allowed to insert multiple photos of your plant to get the highest possible accuracy. using cutting edge methods of machine learning (AKA artificial intelligence) and train customized deep convolutional neural networks to ensure the best possible results.

6. IMPLEMENTATION

1. Simply getting text results using python.

Plant.py file

```
import base64
import requests

# encode image to base64
with open("107.jpg", "rb") as file:
    images = [base64.b64encode(file.read()).decode("ascii")]

your_api_key = "SAWFOXTtXiRbopdb7K580"
json_data = {
    "images": images,
    "modifiers": ["similar_images"],
    "plant_details": ["common_names", "url", "wiki_description", "taxonomy"]
}

response = requests.post(
    "https://api.plant.identification ",
    json=json_data,
    headers={
        "Content-Type": "application/json",
        "Api-Key": "ldiuJBTtXiRbopdb7K580"
}).json()

for suggestion in response["suggestions"]:
    print(suggestion["plant_name"])    # Taraxacum officinale
    print(suggestion["plant_details"]["common_names"])    # ["Dandelion"]
    print(suggestion["plant_details"]["url"])    #
https://en.wikipedia.org/wiki/Taraxacum_officinale
```

2. For using image classification using tensorflow libraries.
   And training data.

```
import numpy as np
import os
import sys
import tarfile
from six.moves.urllib.request import urlretrieve
from six.moves import cPickle as pickle
from PIL import Image
import math
import random
import re
import scipy.io
import PIL
from numpy import *
from pylab import *
from PIL import Image
from collections import defaultdict
import tensorflow as tf
import matplotlib.pyplot as plt

# Load data
DROPOUT = 0.5
LEARNING_RATE  = 0.1
VALIDATION_SIZE = 0
TRAINING_ITERATIONS = 50000
WEIGHT_DECAY = 0.00005

net_data = load("bvlc_alexnet.npy").item()

out_pool_size = [8, 6, 4]
hidden_dim = 0
for item in out_pool_size:
  hidden_dim = hidden_dim + item * item

data_folder = './102flowers'
labels = scipy.io.loadmat('imagelabels.mat')
setid = scipy.io.loadmat('setid.mat')

labels = labels['labels'][0] - 1
trnid = np.array(setid['tstid'][0]) - 1
tstid = np.array(setid['trnid'][0]) - 1
valid = np.array(setid['valid'][0]) - 1

num_classes = 102
data_dir = list()
for img in os.listdir(data_folder):
  data_dir.append(os.path.join(data_folder, img))

data_dir.sort()

# ------------------------------------------------------------------------
```

```
# Ultils
def print_activations(t):
  print(t.op.name, ' ', t.get_shape().as_list())

def dense_to_one_hot(labels_dense, num_classes):
  num_labels = labels_dense.shape[0]
  index_offset = np.arange(num_labels) * num_classes
  labels_one_hot = np.zeros((num_labels, num_classes))
  labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
  return labels_one_hot

def read_images_from_disk(input_queue):
  label = input_queue[1]
  file_contents = tf.read_file(input_queue[0])
  example = tf.image.decode_jpeg(file_contents, channels=3)
  # example = tf.cast(example, tf.float32 )
  return example, label

def weight_variable(shape, name):
  initial = tf.truncated_normal(shape, stddev=0.01, name=name)
  return tf.Variable(initial)

def bias_variable(shape, name):
  initial = tf.constant(0.1, shape=shape, name=name)
  return tf.Variable(initial)


def conv(input, kernel, biases, k_h, k_w, c_o, s_h, s_w, padding = "VALID", group = 1):
  '''From https://github.com/ethereon/caffe-tensorflow
  '''
  c_i = input.get_shape()[-1]
  assert c_i % group == 0
  assert c_o % group == 0
  convolve = lambda i, k: tf.nn.conv2d(i, k, [1, s_h, s_w, 1], padding=padding)

  if group == 1:
    conv = convolve(input, kernel)
  else:
    input_groups = tf.split(axis=3, num_or_size_splits=group, value=input)
    kernel_groups = tf.split(axis=3, num_or_size_splits=group, value=kernel)
    output_groups = [convolve(i, k) for i, k in zip(input_groups, kernel_groups)]
    conv = tf.concat(axis=3, values=output_groups)
  return tf.reshape(tf.nn.bias_add(conv, biases), [-1] + conv.get_shape().as_list()[1:])

def conv2d(x, W, stride_h, stride_w, padding='SAME'):
  return tf.nn.conv2d(x, W, strides=[1, stride_h, stride_w, 1], padding=padding)

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

```
def max_pool_3x3(x):
  return tf.nn.max_pool(x, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1], padding='SAME')

def max_pool_4x4(x):
  return tf.nn.max_pool(x, ksize=[1, 4, 4, 1], strides=[1, 4, 4, 1], padding='SAME')

# Spatial Pyramid Pooling block
# https://arxiv.org/abs/1406.4729
def spatial_pyramid_pool(previous_conv, num_sample, previous_conv_size, out_pool_size):
  """
  previous_conv: a tensor vector of previous convolution layer
  num_sample: an int number of image in the batch
  previous_conv_size: an int vector [height, width] of the matrix features size of previous
convolution layer
  out_pool_size: a int vector of expected output size of max pooling layer

  returns: a tensor vector with shape [1 x n] is the concentration of multi-level pooling
  """
  for i in range(len(out_pool_size)):
    h_strd = h_size = math.ceil(float(previous_conv_size[0]) / out_pool_size[i])
    w_strd = w_size = math.ceil(float(previous_conv_size[1]) / out_pool_size[i])
    pad_h = int(out_pool_size[i] * h_size - previous_conv_size[0])
    pad_w = int(out_pool_size[i] * w_size - previous_conv_size[1])
    new_previous_conv = tf.pad(previous_conv, tf.constant([[0, 0], [0, pad_h], [0, pad_w], [0,
0]]))
    max_pool = tf.nn.max_pool(new_previous_conv,
                ksize=[1,h_size, h_size, 1],
                strides=[1,h_strd, w_strd,1],
                padding='SAME')
    if (i == 0):
      spp = tf.reshape(max_pool, [num_sample, -1])
    else:
      spp = tf.concat(axis=1, values=[spp, tf.reshape(max_pool, [num_sample, -1])])

  return spp

# --------------------------------------------------------------------------
# Modeling
size_cluster = defaultdict(list)
for tid in trnid:
  img = Image.open(data_dir[tid])
  key = (img.size[0] - img.size[0] % 10, img.size[1] - img.size[1] % 10)
  size_cluster[key].append(tid)

size_cluster_keys = size_cluster.keys()

train_accuracies = []
train_cost = []
validation_accuracies = []
x_range = []
```

```
batch_size = 20
print('Training ...')

# Training block
# 1. Combime all iamges have the same size to a batch.
# 2. Then, train parameters in a batch
# 3. Transfer trained parameters to another batch
it = 0
while it < TRAINING_ITERATIONS:
  graph = tf.Graph()
  with graph.as_default():
    y_train = labels[size_cluster[size_cluster_keys[it%len(size_cluster_keys)]]]
    if len(y_train) < 50:
      batch_size = len(y_train)

    y_train = dense_to_one_hot(y_train, num_classes)
    x_train = [data_dir[i] for i in size_cluster[size_cluster_keys[it%len(size_cluster_keys)]]]

    input_queue_train = tf.train.slice_input_producer([x_train, y_train],
                        num_epochs=None,
                        shuffle=True)

    x_train, y_train = read_images_from_disk(input_queue_train)

    print(size_cluster_keys[it%len(size_cluster_keys)])
    x_train = tf.image.resize_images(x_train,
                  [size_cluster_keys[it%len(size_cluster_keys)][1]/2,
                  size_cluster_keys[it%len(size_cluster_keys)][0]/2],
                  method=1, align_corners=False)

    x_train, y_train = tf.train.batch([x_train, y_train], batch_size = batch_size)

    x = tf.placeholder('float', shape = x_train.get_shape())
    y_ = tf.placeholder('float', shape = [None, num_classes])

    conv1W = tf.Variable(net_data["conv1"][0])
    conv1b = tf.Variable(net_data["conv1"][1])
    conv2W = tf.Variable(net_data["conv2"][0])
    conv2b = tf.Variable(net_data["conv2"][1])
    conv3W = tf.Variable(net_data["conv3"][0])
    conv3b = tf.Variable(net_data["conv3"][1])
    conv4W = tf.Variable(net_data["conv4"][0])
    conv4b = tf.Variable(net_data["conv4"][1])
    conv5W = tf.Variable(net_data["conv5"][0])
    conv5b = tf.Variable(net_data["conv5"][1])
    fc6W = weight_variable([hidden_dim * 256, 4096], 'fc6W')
    fc6b = tf.Variable(net_data["fc6"][1])
    fc7W = tf.Variable(net_data["fc7"][0])
    fc7b = tf.Variable(net_data["fc7"][1])
    fc8W = weight_variable([4096, num_classes], 'W_fc8')
```

```
    fc8b = bias_variable([num_classes], 'b_fc8')
    keep_prob = tf.placeholder('float')


    def model(x):
      # conv1
      conv1 = tf.nn.relu(conv(x, conv1W, conv1b, 11, 11, 96, 4, 4, padding="SAME",
group=1))
      # lrn1
      # lrn(2, 2e-05, 0.75, name='norm1')
      lrn1 = tf.nn.local_response_normalization(conv1,
                          depth_radius=5,
                          alpha=0.0001,
                          beta=0.75,
                          bias=1.0)
      # maxpool1
      maxpool1 = tf.nn.max_pool(lrn1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
padding='VALID')
      # conv2
      conv2 = tf.nn.relu(conv(maxpool1, conv2W, conv2b, 5, 5, 256, 1, 1, padding="SAME",
group=2))
      # lrn2
      # lrn(2, 2e-05, 0.75, name='norm2')
      lrn2 = tf.nn.local_response_normalization(conv2,
                          depth_radius=5,
                          alpha=0.0001,
                          beta=0.75,
                          bias=1.0)
      # maxpool2
      maxpool2 = tf.nn.max_pool(lrn2, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
padding='VALID')
      # conv3
      conv3 = tf.nn.relu(conv(maxpool2, conv3W, conv3b, 3, 3, 384, 1, 1, padding="SAME",
group=1))
      # conv4
      conv4 = tf.nn.relu(conv(conv3, conv4W, conv4b, 3, 3, 384, 1, 1, padding="SAME",
group=2))
      # conv5
      conv5 = tf.nn.relu(conv(conv4, conv5W, conv5b, 3, 3, 256, 1, 1, padding="SAME",
group=2))
      print int(conv5.get_shape()[0]), int(conv5.get_shape()[1]), int(conv5.get_shape()[2])
      maxpool5 = spatial_pyramid_pool(conv5,
                    int(conv5.get_shape()[0]),
                    [int(conv5.get_shape()[1]), int(conv5.get_shape()[2])],
                    out_pool_size)
      # fc6
      fc6 = tf.nn.relu_layer(tf.reshape(maxpool5, [-1, int(prod(maxpool5.get_shape()[1:]))]),
fc6W, fc6b)
      fc6_drop = tf.nn.dropout(fc6, keep_prob)
      # fc7
```

```
      fc7 = tf.nn.relu_layer(fc6_drop, fc7W, fc7b)
      fc7_drop = tf.nn.dropout(fc7, keep_prob)
      # fc8
      fc8 = tf.nn.xw_plus_b(fc7_drop, fc8W, fc8b)
      return fc8

    logits = model(x)
    cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits,
labels=y_))
    regularizers = tf.nn.l2_loss(conv1W) + tf.nn.l2_loss(conv1b) + \
            tf.nn.l2_loss(conv2W) + tf.nn.l2_loss(conv2b) + \
            tf.nn.l2_loss(conv3W) + tf.nn.l2_loss(conv3b) + \
            tf.nn.l2_loss(conv4W) + tf.nn.l2_loss(conv4b) + \
            tf.nn.l2_loss(conv5W) + tf.nn.l2_loss(conv5b) + \
            tf.nn.l2_loss(fc6W) + tf.nn.l2_loss(fc6b) + \
            tf.nn.l2_loss(fc7W) + tf.nn.l2_loss(fc7b) + \
            tf.nn.l2_loss(fc8W) + tf.nn.l2_loss(fc8b)

    loss = tf.reduce_mean(cross_entropy + WEIGHT_DECAY * regularizers)

    cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits,
labels=y_))
    # optimisation loss function
    global_step = tf.Variable(0)
    learning_rate = tf.train.exponential_decay(LEARNING_RATE, global_step, 1000, 0.9,
staircase=True)
    train_step = tf.train.AdagradOptimizer(learning_rate).minimize(loss)

    # evaluation
    correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
    predict = tf.argmax(logits, 1)
    saver = tf.train.Saver({v.op.name: v for v in [conv1W, conv1b,
                         conv2W, conv2b,
                         conv3W, conv3b,
                         conv4W, conv4b,
                         conv5W, conv5b,
                         fc6W, fc6b,
                         fc7W, fc7b,
                         fc8W, fc8b]})


  with tf.Session(graph=graph) as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(coord=coord)
    if os.path.exists('./alex_model_spp.ckpt'):
      saver.restore(sess, './alex_model_spp.ckpt')
```

```
    cnt_tmp = 0
    xtrain, ytrain = sess.run([x_train, y_train])
    for i in range(10):
      it = it + 1
      _, train_accuracy, cost = sess.run([train_step, accuracy, cross_entropy],
                   feed_dict = {x: xtrain,
                          y_: ytrain,
                          keep_prob: 1.0})

      print('training_accuracy => %.4f, cost value => %.4f for step %d'
          %(train_accuracy, cost, it))

      if (train_accuracy > 0.95):
        cnt_tmp = cnt_tmp + 1

      if (cnt_tmp > 10):
        break

      train_accuracies.append(train_accuracy)
      x_range.append(it)
      train_cost.append(cost)

    saver.save(sess, './alex_model_spp.ckpt')
    coord.request_stop()
    coord.join(threads)
  sess.close()
  del sess

# Plot accuracy and loss curve
plt.plot(x_range, train_cost,'-b')
plt.ylabel('spp_cost')
plt.xlabel('step')
plt.savefig('spp_cost.png')
plt.close()
plt.plot(x_range, train_accuracies,'-b')
plt.ylabel('spp_accuracies')
plt.ylim(ymax = 1.1)
plt.xlabel('step')
plt.savefig('spp_accuracy.png')


# ---------------------------------------------------------------------------
# Testing block
# 1. Gather all images have the same size into a batch
# 2. Feed to Alexnet_SPP to predict the expected labels
it = 0
result = list()
f = open('result_spp.txt', 'w')
while it < len(tstid):
  if (it % 10 == 0):
```

```
    print(it)
  graph = tf.Graph()
  with graph.as_default():
    # with tf.device('/cpu:0'):
    img = Image.open(data_dir[tstid[it]])
    filename_queue = tf.train.string_input_producer([data_dir[tstid[it]]])
    reader = tf.WholeFileReader()
    key, value = reader.read(filename_queue)
    my_img = tf.image.decode_jpeg(value, channels = 3)
    # my_img = tf.cast(my_img, tf.float32)
    my_img = tf.image.resize_images(my_img,
                  [img.size[1] / 2,
                  img.size[0] / 2],
                  method = 1,
                  align_corners = False)

    my_img = tf.expand_dims(my_img, 0)

    x = tf.placeholder('float', shape=my_img.get_shape())
    print(my_img.get_shape())
    conv1W = tf.Variable(net_data["conv1"][0])
    conv1b = tf.Variable(net_data["conv1"][1])
    conv2W = tf.Variable(net_data["conv2"][0])
    conv2b = tf.Variable(net_data["conv2"][1])
    conv3W = tf.Variable(net_data["conv3"][0])
    conv3b = tf.Variable(net_data["conv3"][1])
    conv4W = tf.Variable(net_data["conv4"][0])
    conv4b = tf.Variable(net_data["conv4"][1])
    conv5W = tf.Variable(net_data["conv5"][0])
    conv5b = tf.Variable(net_data["conv5"][1])
    fc6W = weight_variable([hidden_dim * 256, 4096], 'fc6W')
    fc6b = tf.Variable(net_data["fc6"][1])
    fc7W = tf.Variable(net_data["fc7"][0])
    fc7b = tf.Variable(net_data["fc7"][1])
    fc8W = weight_variable([4096, num_classes], 'W_fc8')
    fc8b = bias_variable([num_classes], 'b_fc8')
    keep_prob = tf.placeholder('float')


    def model(x):
      # conv1
      conv1 = tf.nn.relu(conv(x, conv1W, conv1b, 11, 11, 96, 4, 4, padding="SAME",
group=1))
      # lrn1
      # lrn(2, 2e-05, 0.75, name='norm1')
      lrn1 = tf.nn.local_response_normalization(conv1,
                          depth_radius=5,
                          alpha=0.0001,
                          beta=0.75,
                          bias=1.0)
```

```
    # maxpool1
    maxpool1 = tf.nn.max_pool(lrn1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
padding='VALID')
    # conv2
    conv2 = tf.nn.relu(conv(maxpool1, conv2W, conv2b, 5, 5, 256, 1, 1, padding="SAME",
group=2))
    # lrn2
    # lrn(2, 2e-05, 0.75, name='norm2')
    lrn2 = tf.nn.local_response_normalization(conv2,
                      depth_radius=5,
                      alpha=0.0001,
                      beta=0.75,
                      bias=1.0)
    # maxpool2
    maxpool2 = tf.nn.max_pool(lrn2, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],
padding='VALID')
    # conv3
    conv3 = tf.nn.relu(conv(maxpool2, conv3W, conv3b, 3, 3, 384, 1, 1, padding="SAME",
group=1))
    # conv4
    conv4 = tf.nn.relu(conv(conv3, conv4W, conv4b, 3, 3, 384, 1, 1, padding="SAME",
group=2))
    # conv5
    conv5 = tf.nn.relu(conv(conv4, conv5W, conv5b, 3, 3, 256, 1, 1, padding="SAME",
group=2))
    maxpool5 = spatial_pyramid_pool(conv5,
                int(conv5.get_shape()[0]),
                 [int(conv5.get_shape()[1]), int(conv5.get_shape()[2])],
                 out_pool_size)
    # fc6
    fc6 = tf.nn.relu_layer(tf.reshape(maxpool5, [-1, int(prod(maxpool5.get_shape()[1:]))]),
fc6W, fc6b)
    fc6_drop = tf.nn.dropout(fc6, keep_prob)
    # fc7
    fc7 = tf.nn.relu_layer(fc6_drop, fc7W, fc7b)
    fc7_drop = tf.nn.dropout(fc7, keep_prob)
    # fc8
    fc8 = tf.nn.xw_plus_b(fc7_drop, fc8W, fc8b)
    prob = tf.nn.softmax(fc8)
    return prob

  logits = model(x)
  predict = tf.argmax(logits, 1)
  saver = tf.train.Saver({v.op.name: v for v in [conv1W, conv1b,
                      conv2W, conv2b,
                      conv3W, conv3b,
                      conv4W, conv4b,
                      conv5W, conv5b,
                      fc6W, fc6b,
                      fc7W, fc7b,
```

```
                    fc8W, fc8b]})

  with tf.Session(graph=graph) as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(coord=coord)
    saver.restore(sess, './alex_model_spp.ckpt')
    image = sess.run(my_img)
    predict = predict.eval(feed_dict={x: image, keep_prob: 1.0})
    result.append(predict[0])
    f.write(data_dir[tstid[it]] + '\t' + str(predict[0]) + '\t' + str(labels[tstid[it]]))
    f.write('\n')
    coord.request_stop()
    coord.join(threads)
  sess.close()
  del sess
  it = it + 1

print('Test accuracy: %f' %(sum(np.array(result) ==
np.array(labels[tstid])).astype('float')/len(tstid)))
f.close()
```

3. For getting many results for more images same time.

```
import base64
import requests
from time import sleep


key = "K1AY53YkYJjsc8X8 --"


def encode_files(file_names):
    files_encoded = []
    for file_name in file_names:
        with open(file_name, "rb") as file:
            files_encoded.append(base64.b64encode(file.read()).decode("ascii"))
    return files_encoded


def identify_plant(file_names):
    images = encode_files(file_names)

    params = {
```

```
        "api_key": key,
        "images": images,
        "latitude": 49.1951239,
        "longitude": 16.6077111,
        "datetime": 1582830233,
        "modifiers": ["crops_fast", "similar_images"],
        }

    headers = {
        "Content-Type": "application/json"
        }

    response = requests.post("https://api.plant.identification ",
                             json=params,
                             headers=headers).json()

    return get_result(response["id"])


def get_result(identification_id):
    params = {
        "api_key": "G2AldiuJBTtXiRbopdb7K580",
        "plant_language": "en",
        "plant_details": ["common_names",
                          "url",
                          "name_authority,",
                          "wiki_description",
                          "taxonomy",
                          "synonyms"],
        }

    headers = {
        "Content-Type": "application/json"
        }

    endpoint = "https://api.plant.identificationresult/"

    while True:
        print("Waiting for suggestions...")
        sleep(5)
        response = requests.post(endpoint + str(identification_id),
                                 json=params,
                                 headers=headers).json()
        if response["suggestions"] is not None:
            return response


if __name__ == '__main__':
    print(identify_plant(["101.jpg", "107.jpg"]))
```

4. Html file for front end and interface .

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Plant identification app</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="keywords" content="plant, flower, identification, recognition, plant name,
app, application, id, identifier, plant identification, plant identifier, plant id">
    <meta name="description" content="Take a photo, upload it, let us identify it with our
'magic' and view the results instantly. For free!">
    <link href="C:\Users\RUDRA\Desktop\cas.css" rel="stylesheet" type="text/css">
    <style>
        #loading-page .loader-container {
            margin: 100px auto;
            text-align: center;
        }
        #loading-page .loader {
            border: 1px solid rgba(73, 255, 86, 0.23);
            border-top: 2px solid #1b5e20;
            border-radius: 50%;
            width: 50px;
            height: 50px;
```

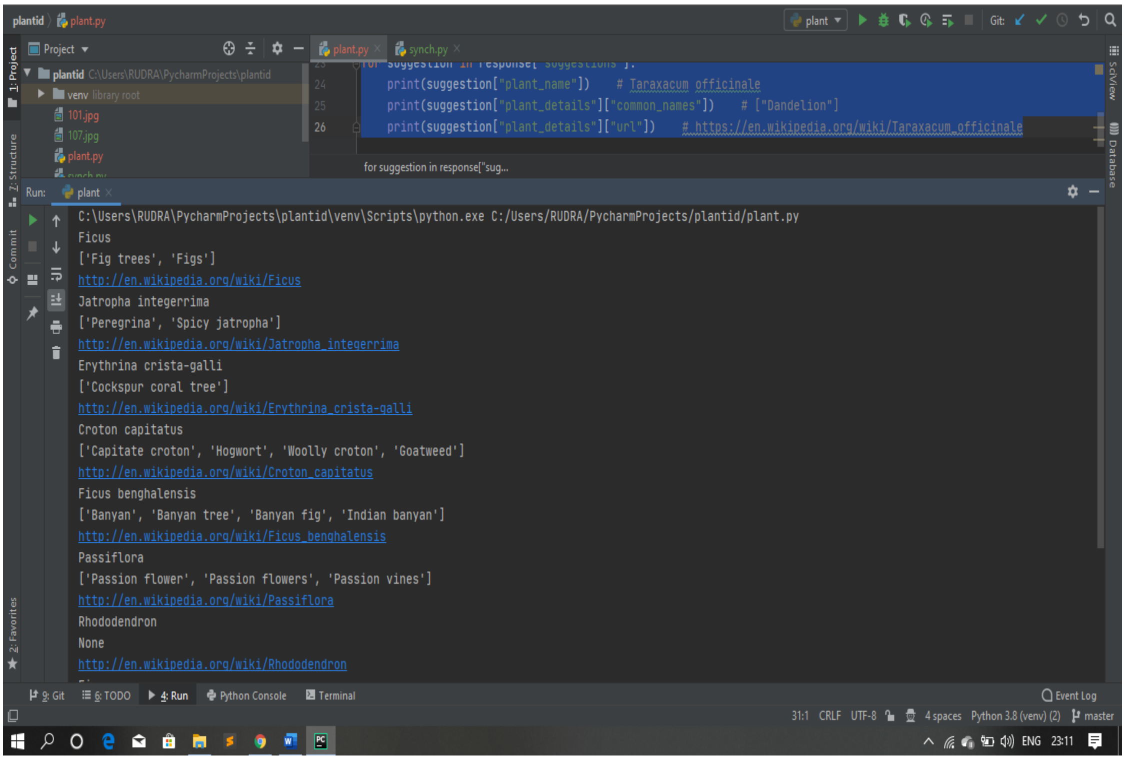
```
            margin: auto;

            animation: spin 1.4s linear infinite;

        }

        #loading-page .loader-container span {

            margin: 20px;

            display: block;

        }

        @keyframes spin {

            0% { transform: rotate(0deg); }

            100% { transform: rotate(360deg); }

        }

    </style>

    <meta property="og:image" content="assets/plantidcard.png"/>

    <!-- Google Tag Manager -->

    <script>(function(w,d,s,l,i){w[l]=w[l]||[];

            w[l].push({

            'gtm.start':new Date().getTime(),event:'gtm.js'});

            var

f=d.getElementsByTagName(s)[0],j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';

            j.async=true;j.src='https://www.googletagmanager.com/gtm.js?id='+i+dl;

            f.parentNode.insertBefore(j,f);

            })

            (window,document,'script','dataLayer','GTM-WJF2WZ4');

            </script>

    <!-- End Google Tag Manager -->
```

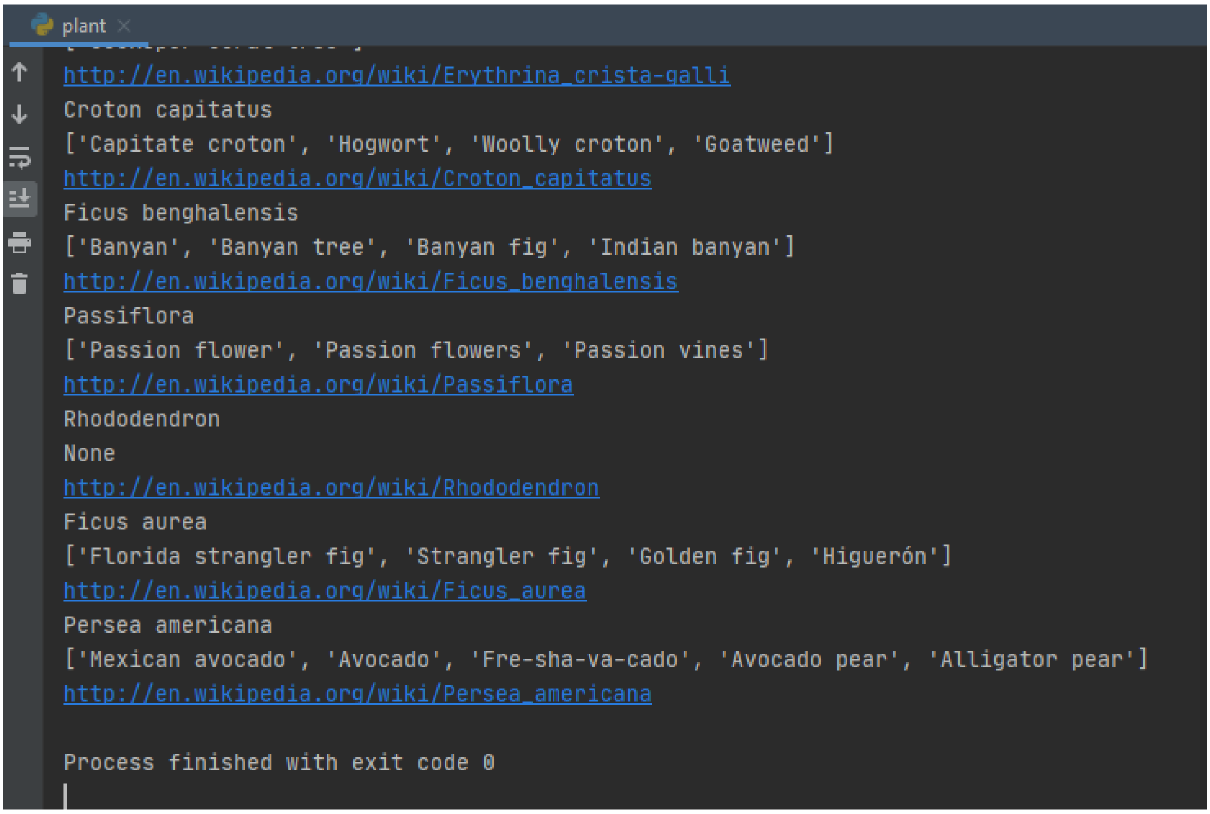
```
  <link rel="manifest" href="manifest.webmanifest">

  <meta name="theme-color" content="#1b5e20">

  <link rel="stylesheet" href="C:\Users\RUDRA\Desktop\stats.css"></head>

     <body>

   <!-- Google Tag Manager (noscript) -->

   <noscript>

   <iframe src="https://www.googletagmanager.com/ns.html?id=GTM-WJF2WZ4"

   height="0" width="0" style="display:none;visibility:hidden">

    </iframe></noscript>

   <!-- End Google Tag Manager (noscript) -->

   <app-root></app-root>


   <div id="loading-page" style="text-align: center">

       <div style="height: 200px"></div>

       <img src="/assets/logo.large.png" alt=""/>

       <div class="loader-container">

           <div class="loader"></div>

       </div>

   </div>

</body>

</html>
```

7. RESULT

7.1 OUTPUT

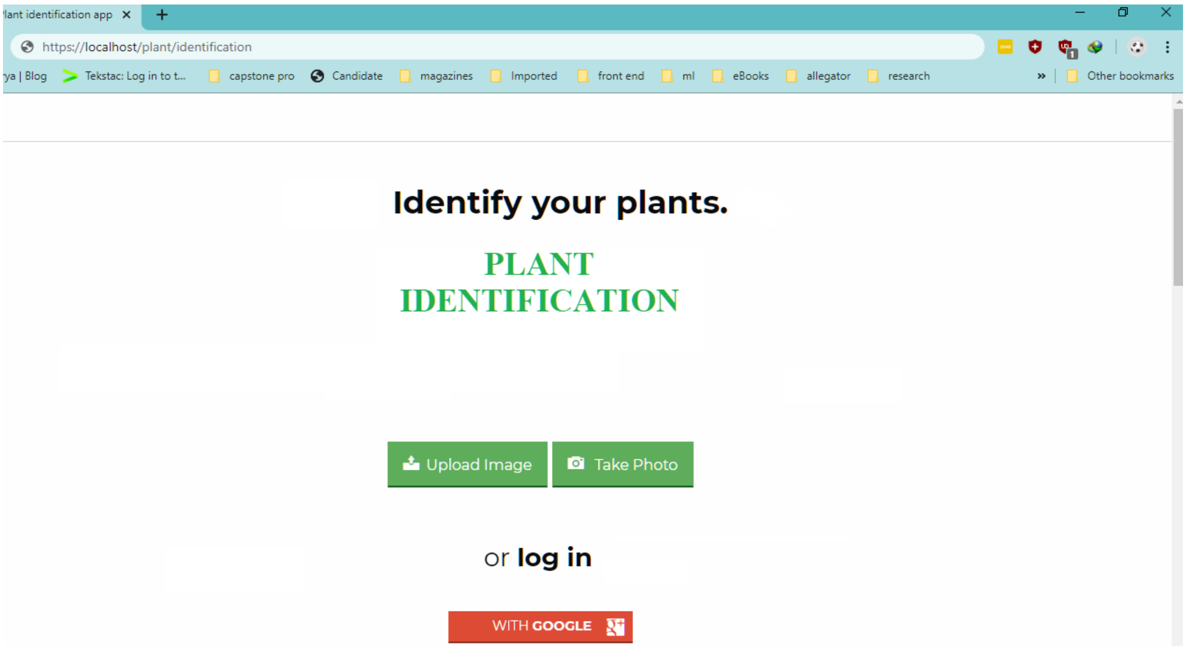
OUTPUT FOR SIMPLE PYTHON PROGRAM FOR ANALYSIS OF DATA FOR VERIFICATION OF PLANTS IMAGE.



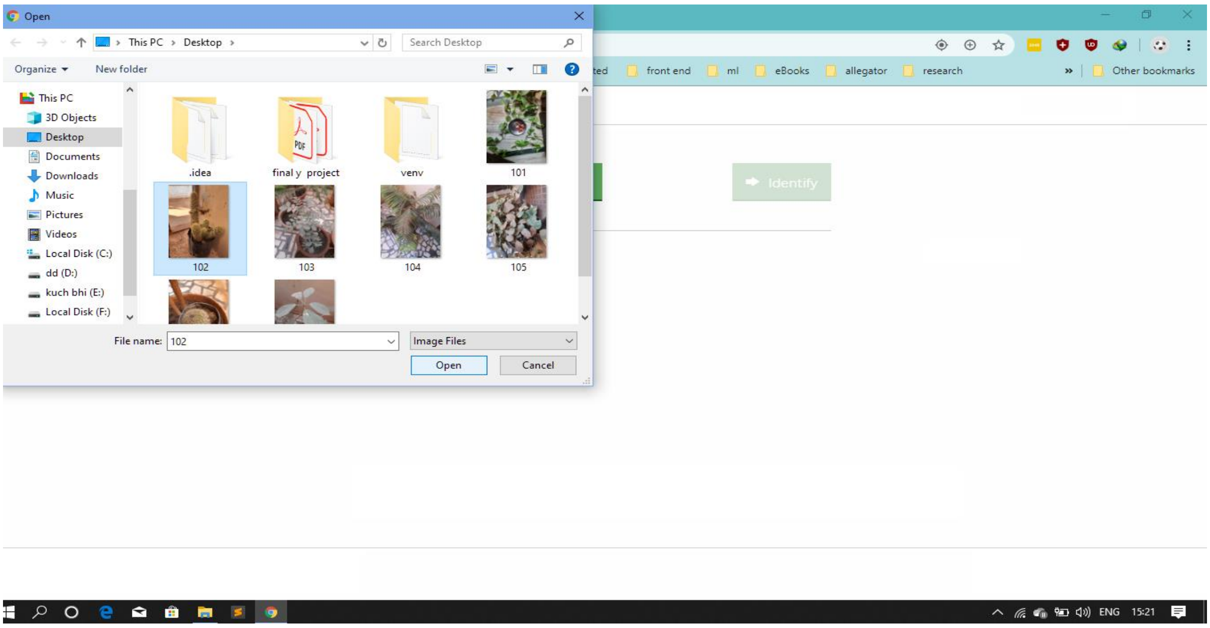


7.2 SCREENSHOTS

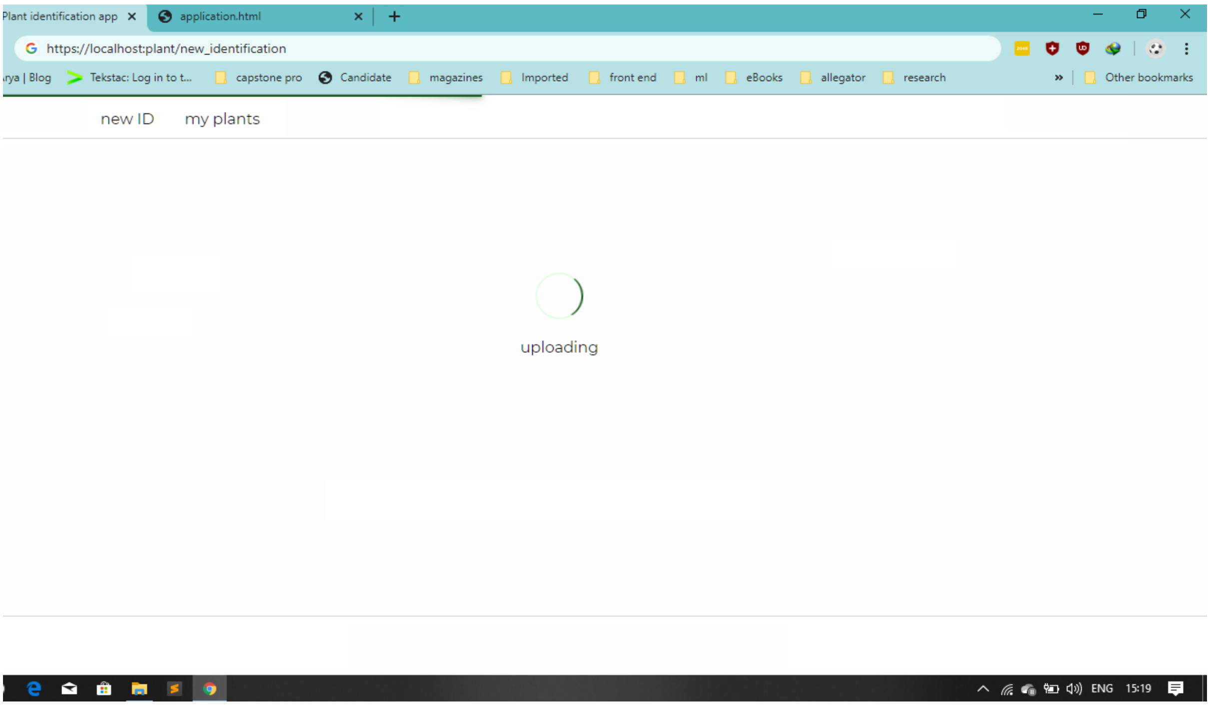
a. Main page:containing upload image and take photo button or login option



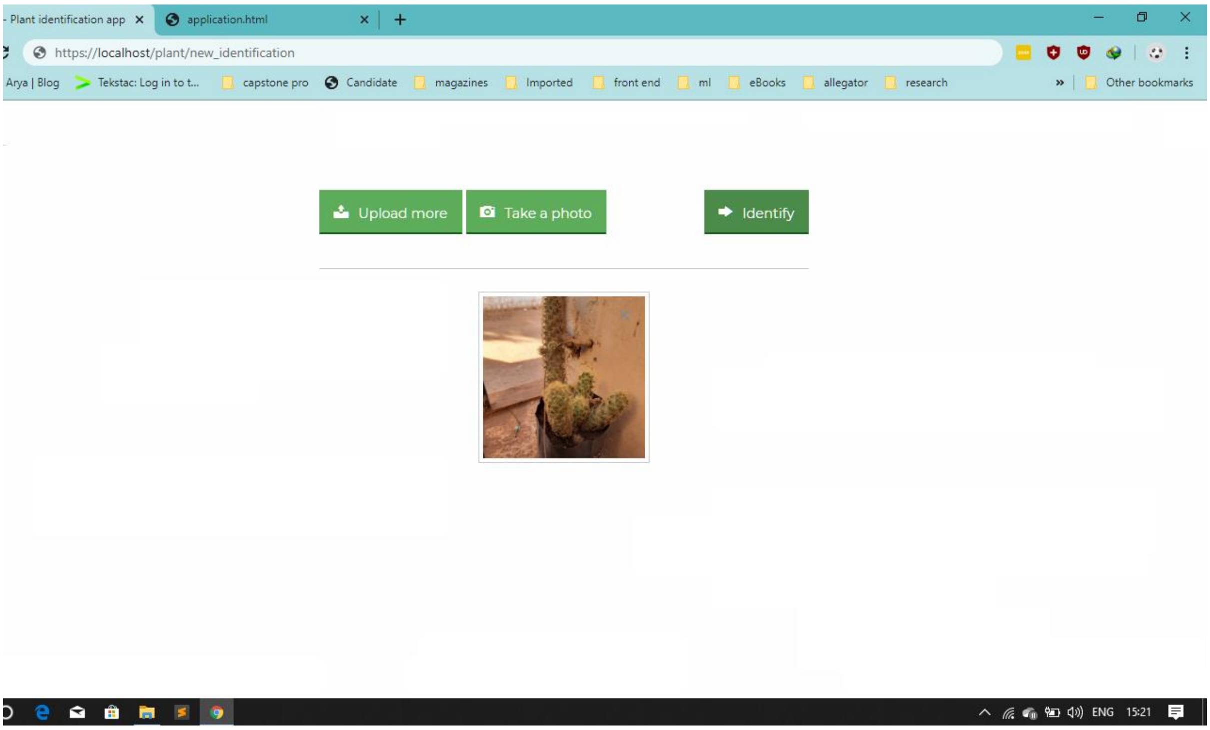
b. Clicking on upload button to browse the image.



c. Selecting the plant image and uploading it.

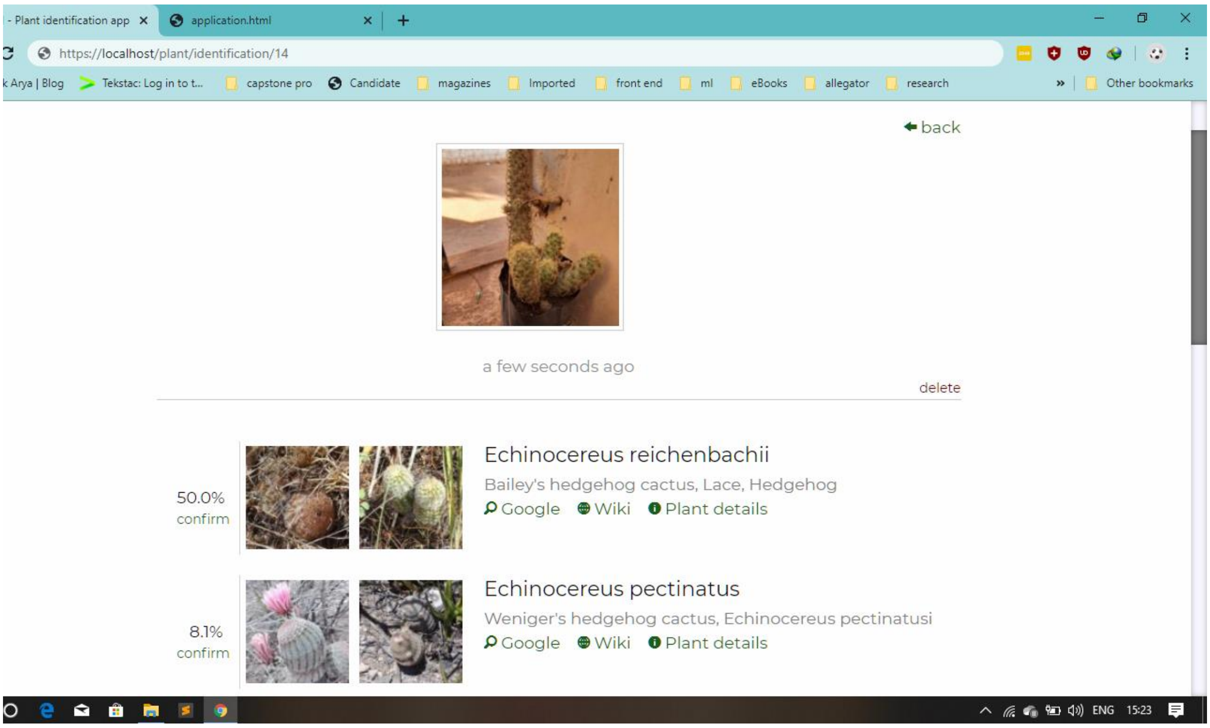


d. Showing the selected image to identify image.

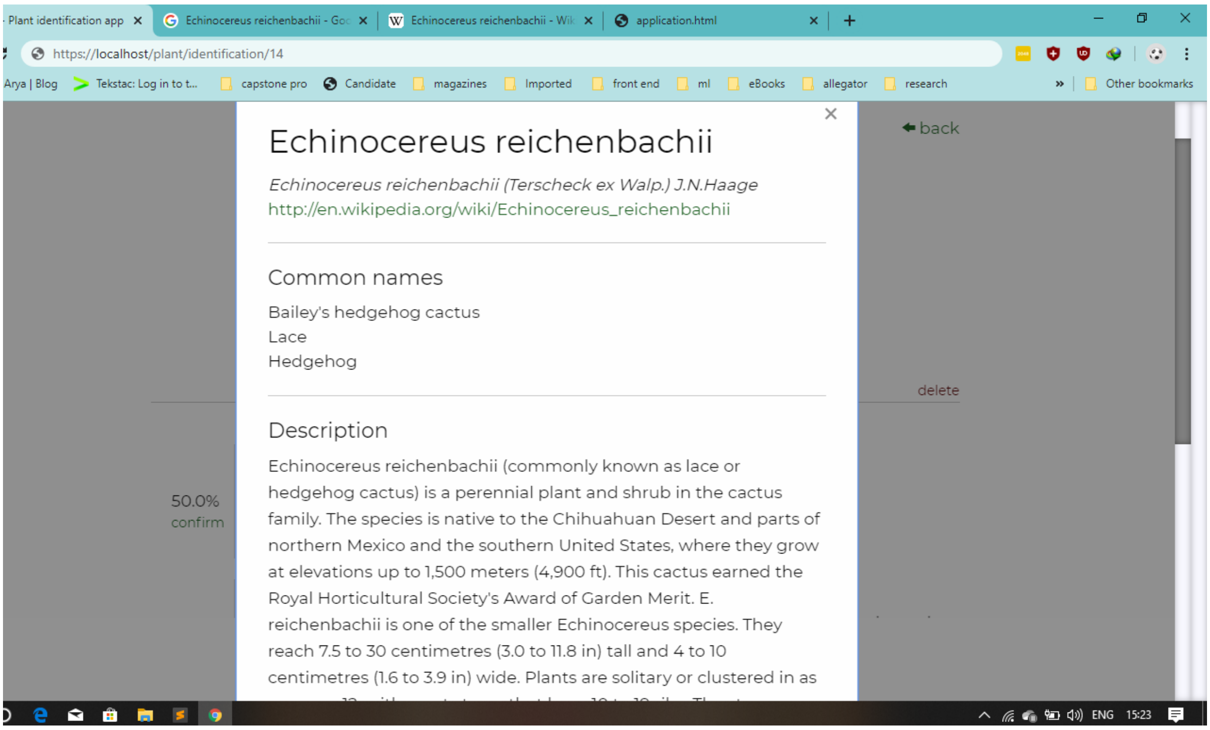


e. Giving the result of identify photo plant scientific name as

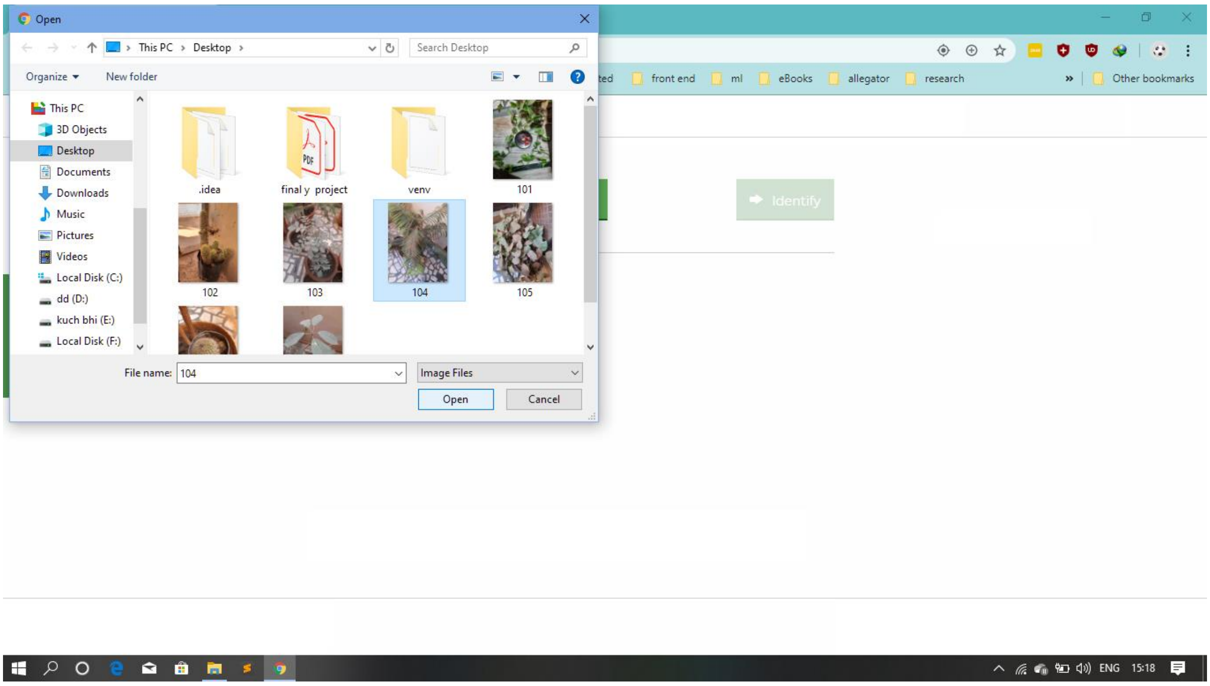
"Echinocereus reichenbachi"



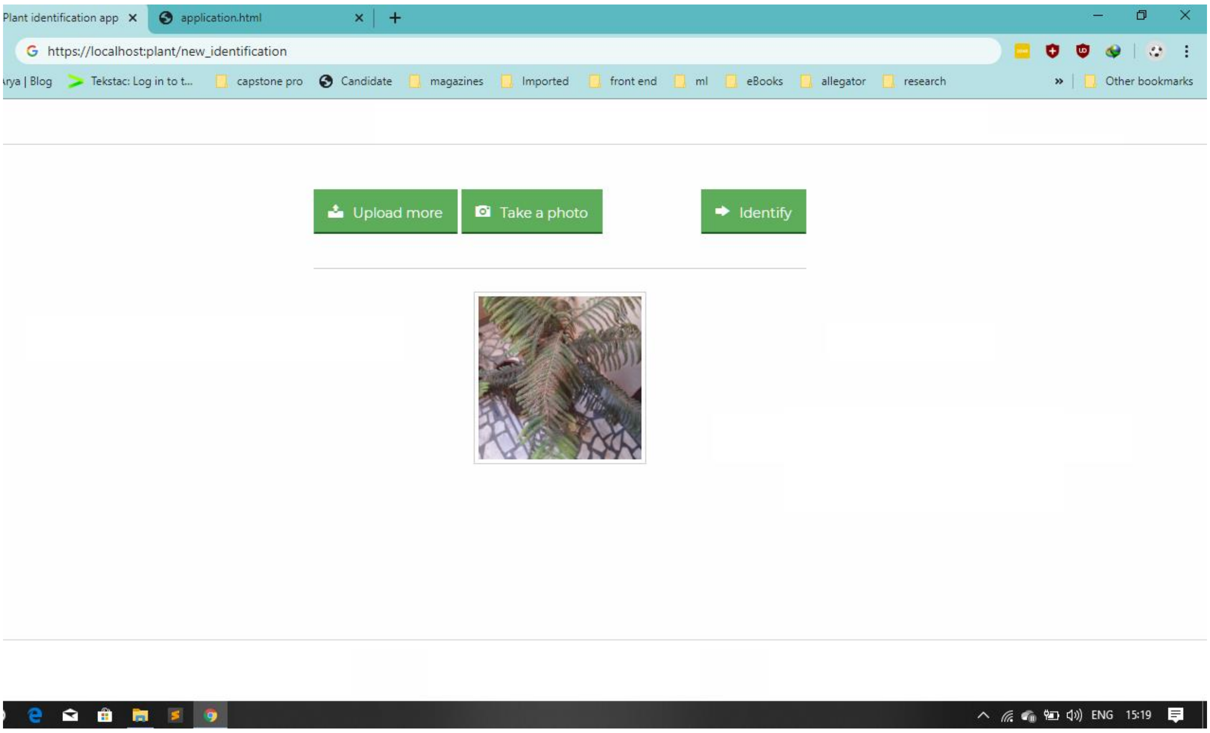
f. Onclicking plant details at shown result gives detailed info of that plant.



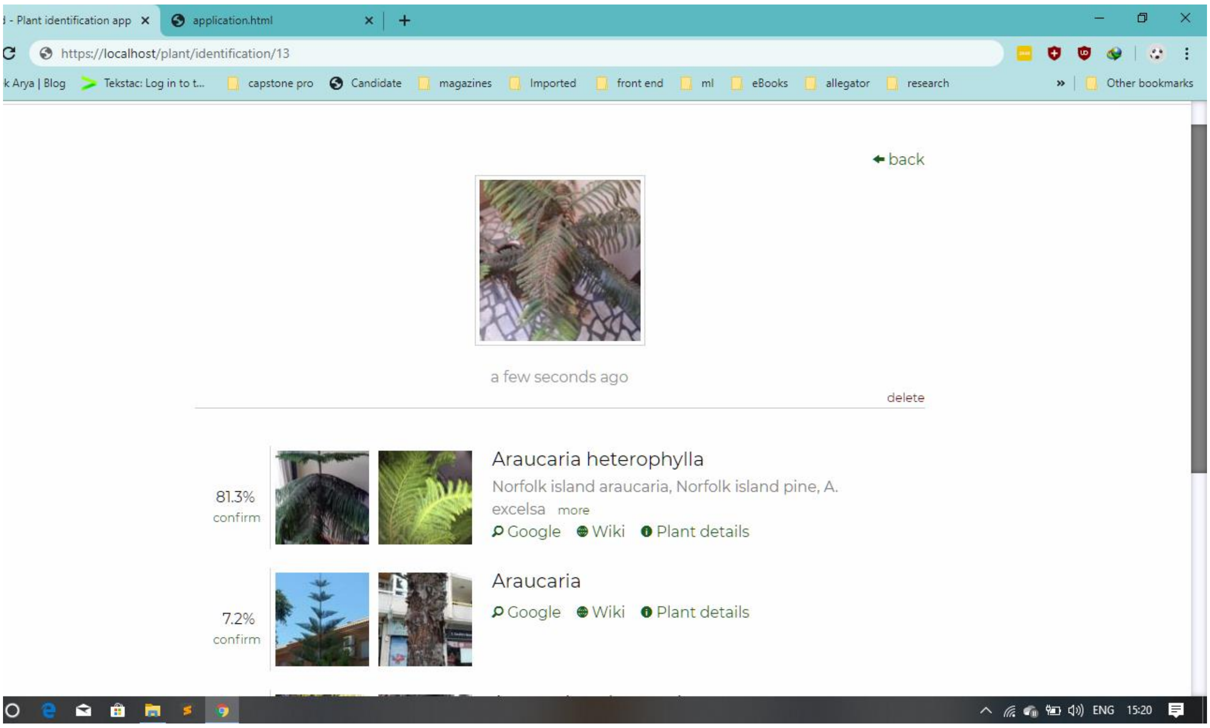
g. Trying the same for other plant image (2).



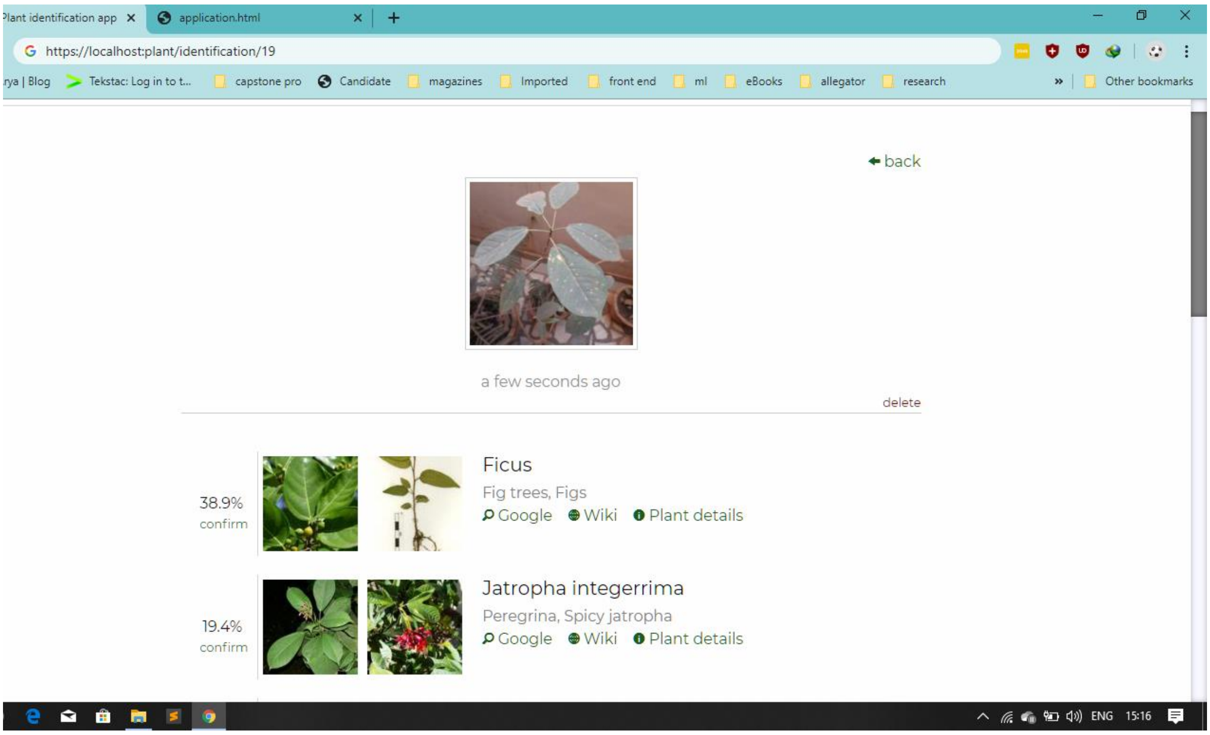
h. Showing the image for second plant and pressing identify button.



i. Showing the matched images in results.



j. Trying the same for a known plant which is available in locality and named as figs tree lets check data for this plant.



Its shows correct result but with less accuracy this may vary with picture quality and climate also.

But it recognize plants correctly.

8. FUTURE ENHANCEMENT

Further classification techniques and data training can lead to improve accuracy in blur images (poor quality images) and researches in this area can make possible such followings as Identification of poisonous plants, to improve yields, Protecting Crops, Better Plantation and caring of Plants. And can provide more details about plants such as Plant form or shape, Plant size and age, Where these plants grow such as Is the plant growing in wet or dry conditions, or in a sunny or shady area, What are the colour and sizes of any seeds or fruit? What is the colour of plant., Bark characteristics: Is the bark smooth, or does it have a rough or flaky texture. These can be possible to know just by clicking plants image in near future as many enhancements are going on this.

This is much helpful so that we can understand the importance of plants and nature.

Tensorflow team has make an app to identify infected plants which helped in identifying that crops and plants to get safe from it in Africa it happened with cassava leaves on which most farmers depend so that app helped them a lot so that they can have more reliable source of food and fruits from plants.

9. REFERENCES

1. Bhardwaj, M. kaur, “A review on plant recognition and classification techniques using leaf images,” International Journal of Engineering Trends and Technology, Volume 4.

2. International Journal of Innovative Research in Science, Engineering and Technology Vol. 4, Special Issue 6, May 2015.

3. https://www.tensorflow.org/about/case-studies.

4. https://identifythatplant.com/benefits-from-mastering-the-skill-of-plant-identification.

5. The Research on the Application of Plant Identification and Mobile Learning APP based on Expert System Cixiao Wang Graduate School of Education, Peking University, No.5 Yiheyuan Road Haidian District,Beijing, P.R. China