PLANT SPECIES IDENTIFICATION

Hemalatha. N

Step 1: prototype selection

Abstract

All living organisms on earth rely on the process of photosynthesis for food energy and oxygen. Humans depend almost entirely on plants for clean air and a liveable climate as well as for food, medicines, materials, and well beings. The ability of identifying plants and their requirements has always been an essential skill for who manage plant growth and health. It requires experts and difficult for non-experts to remember the specific botanic terms. The idea of automatic identification of plant species is approaching reality. AI (Artificial Intelligence) plays a major role in developing a solution. However, the advancement in the field of machine learning and deep learning helps in this matter. Deep learning model (CNN)can be used to extract the features from different leaf and for classification of plant species.

1.Problem Statement

Identification of plant species is usually done by Botanists and plant Ecologist. Identification of plant species is complex and difficult for non-experts like Farmers, Travelers, Foresters, Nature lovers, etc., Since there are lakhs of species in the world,

- Plant identification by conventional methods is difficult, time consuming and frustrating for many.
- Due to high similarity between some plant species, it is difficult to differentiate them easily.
- Many plants species cannot be identified which is endangered and non-endangered plant species in a proper way to reduce risk of extinction.
- Identifying the plants correctly is out of reach of an ordinary person as it requires specialized knowledge.
- Botanists do not have knowledge of all the existing plants in the world

To achieve these, automation of plant species identification is necessary. Hence, there is a need to develop an automated or computerized system to identify and classify the plants species. AI (Artificial Intelligence) plays a major role in developing a solution for the problem.

2.Market/customer/business need assessment

plant species knowledge is necessary for various purpose such as identifying a new or rare species, balancing of the ecosystem, medicinal purpose, agricultural industry etc. Image based methods are considered a promising approach for species identification.

In the era of the digital world, smartphones and digital cameras are available to everyone and are found in abundance. Instead of relying on the botanists or ecologist we can get a solution by developing plant species identification app which helps to identify harmless flowers and plants species. These helps plant lovers to identify plants and tree quickly without wasting time.

By developing the automation plant identifier, the application empowers the app users to recognize plants and know their names just by clicking the pictures on their smartphones or cameras. Also, these app helps to identify plants which are beneficial and capable of identifying a large number of cultivated plants in forests and garden.

3.Target specifications and Characterization

The target customers for this product or service are travellers, nature lovers, farmers, botanists, zoologist, medicinal industry, etc. Rather than relying on eyes and experience and botanists just by using this application which uses AI to identify species scientific name, by clicking single photo of the plant helps users with all attributes related to the plants like origin, name, common name, and other information. This application also helps in education field for botanists and biological students to get knowledge about species.

- ✓ Resolves Curiosity of Customers
- ✓ Appropriate educational tool for biological class
- ✓ Customer experience made easy

4.External Search (online information sources/references/links)

https://www.researchgate.net/publication/334255925_Plant_Species_Identification_based_on_Plant_Leaf_Using_Computer_Vision_and_Machine_Learning_Technique

https://ieeexplore.ieee.org/abstract/document/9036796

Information source

https://externlabs.com/blogs/plant-identifier-app-development

https://www.spaceotechnologies.com/blog/plant-identification-app-development

https://www.masterclass.com/articles/how-to-identify-a-plant

Data Description

The dataset used in this experiment is the Swedish Leaf Dataset.

Dataset Link: https://www.cvl.isy.liu.se/en/research/datasets/swedish-leaf/

Swedish Dataset, which is a database of 15 different plant species with a total of 1125 leaf images.

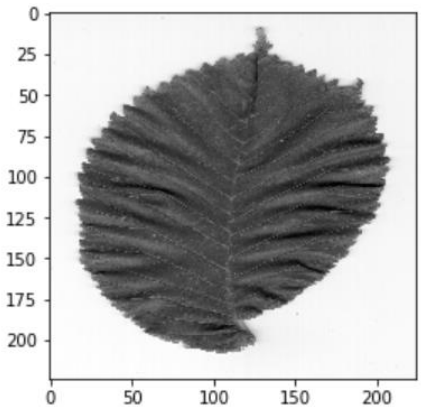
Importing the basic Libraries for data preprocessing

Classification for Swedish leaf Dataset

In [10]:
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import os
import cv2
from tqdm import tqdm

DATADIR = "c:/Internship_2023/"

CATEGORIES = ["leaf1", "leaf2","leaf3", "leaf4","leaf5", "leaf6","leaf7", "leaf8","leaf9", "leaf10","leaf11","leaf12","leaf13","l
```

Converting RGB images to grayscale image and resize

In [12]:
```
IMG_SIZE=224


new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))

plt.imshow(new_array, cmap='gray')
plt.show()
```



Building the CNN model and training the model

Basic_CNN

In [1]:
```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import pickle
import time

NAME="Basic_CNN_2_Augumented"

pickle_in = open("X_Augumented_Grayscale","rb")
X = pickle.load(pickle_in)

pickle_in = open("y_Augumented_Grayscale","rb")
y = pickle.load(pickle_in)

X = X/255.0

model = Sequential()
model.add(Conv2D(100, (3, 3), input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(100, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(15))
model.add(Activation('softmax'))

tensorboard = TensorBoard(log_dir="logs\\{}".format(NAME))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(X, y, epochs=10, validation_split=0.2,callbacks=[tensorboard])
```

Output:

```
Train on 6120 samples, validate on 1530 samples
Epoch 1/10
6120/6120 [==============================] - 76s 12ms/sample - loss: 1.2855 - acc: 0.6083 - val_loss: 0.6437 - val_acc: 0.8105
Epoch 2/10
6120/6120 [==============================] - 52s 8ms/sample - loss: 0.3387 - acc: 0.8907 - val_loss: 0.5205 - val_acc: 0.8373
Epoch 3/10
6120/6120 [==============================] - 53s 9ms/sample - loss: 0.1495 - acc: 0.9572 - val_loss: 0.4175 - val_acc: 0.8778
Epoch 4/10
6120/6120 [==============================] - 52s 9ms/sample - loss: 0.0890 - acc: 0.9717 - val_loss: 0.4570 - val_acc: 0.8830
Epoch 5/10
6120/6120 [==============================] - 51s 8ms/sample - loss: 0.0449 - acc: 0.9871 - val_loss: 0.5365 - val_acc: 0.8575
Epoch 6/10
6120/6120 [==============================] - 52s 9ms/sample - loss: 0.0171 - acc: 0.9956 - val_loss: 0.4547 - val_acc: 0.9013
Epoch 7/10
6120/6120 [==============================] - 51s 8ms/sample - loss: 0.0056 - acc: 0.9985 - val_loss: 0.6290 - val_acc: 0.8699
Epoch 8/10
6120/6120 [==============================] - 51s 8ms/sample - loss: 0.0085 - acc: 0.9979 - val_loss: 0.5039 - val_acc: 0.8987
Epoch 9/10
6120/6120 [==============================] - 51s 8ms/sample - loss: 0.0012 - acc: 1.0000 - val_loss: 0.4850 - val_acc: 0.8987
Epoch 10/10
6120/6120 [==============================] - 51s 8ms/sample - loss: 7.5762e-04 - acc: 1.0000 - val_loss: 0.5001 - val_acc: 0.90
46
```

Optimizing CNN model

Optimized CNN

In [2]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
# more info on callbakcs: https://keras.io/callbacks/ model saver is cool too.
from tensorflow.keras.callbacks import TensorBoard
import pickle
import time

NAME = "Final_Module_test_3_Augumented"

pickle_in = open("X_Augumented_Grayscale","rb")
X = pickle.load(pickle_in)

pickle_in = open("y_Augumented_Grayscale","rb")
y = pickle.load(pickle_in)

X = X/255.0

model = Sequential()

model.add(Conv2D(100, (5, 5), padding="same",strides=(2,2),activation="relu",input_shape=X.shape[1:]))
# model.add(Conv2D(228, (3, 3), input_shape=X.shape[1:]))
# model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(5, 5),strides=(5,5)))

model.add(Conv2D(250, (5, 5), padding="same",strides=(2,2),activation="relu"))
# model.add(Conv2D(228, (3, 3), input_shape=X.shape[1:]))
# model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(5, 5),strides=(2,2)))

# model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
# model.add(Dense(240))
# model.add(Activation('relu'))

# model.add(Dense(1))
# model.add(Activation('softmax'))



# model.add(Conv2D(100, (1, 1), padding="same",strides=(1,1)))
# model.add(Activation('relu'))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(15))
model.add(Activation('softmax'))

tensorboard = TensorBoard(log_dir="logs\\{}".format(NAME))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'],
              )

model.fit(X, y,
          epochs=10,
          validation_split=0.2,
          callbacks=[tensorboard])

model.summary()
```

```
Epoch 6/10
6120/6120 [==============================] - 13s 2ms/sample - loss: 0.0750 - acc: 0.9740 - val_loss: 0.1804 - val_acc: 0.9346
Epoch 7/10
6120/6120 [==============================] - 13s 2ms/sample - loss: 0.0661 - acc: 0.9765 - val_loss: 0.1065 - val_acc: 0.9608
Epoch 8/10
6120/6120 [==============================] - 13s 2ms/sample - loss: 0.0415 - acc: 0.9859 - val_loss: 0.0823 - val_acc: 0.9699
Epoch 9/10
6120/6120 [==============================] - 13s 2ms/sample - loss: 0.0238 - acc: 0.9940 - val_loss: 0.0891 - val_acc: 0.9732
Epoch 10/10
6120/6120 [==============================] - 13s 2ms/sample - loss: 0.0198 - acc: 0.9943 - val_loss: 0.0749 - val_acc: 0.9752
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 112, 112, 100)     2600
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 100)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 250)       625250
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 250)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 100)         25100
_________________________________________________________________
activation_1 (Activation)    (None, 4, 4, 100)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 15)                24015
_________________________________________________________________
activation_2 (Activation)    (None, 15)                0
=================================================================
Total params: 676,965
Trainable params: 676,965
Non-trainable params: 0
_________________________________________________________________
```

Iterative method for CNN_ optimizer

ResNet50

In [ ]:
```
from tensorflow.keras.applications.resnet50 import ResNet50
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras import layers, Model
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, plot_confusion_matrix
from tensorflow.keras.callbacks import TensorBoard
```

In [ ]:
```
img_height, img_width = 224,224
base_model = ResNet50(input_shape = (img_width, img_height, 3), # Shape of our images
                      include_top = False, # Leave out the last fully connected layer
                      weights = 'imagenet')
```

Base model

In [ ]:
```
base_model.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
```

In [ ]:
```
from tensorflow.keras.optimizers import RMSprop

# Flatten the output layer to 1 dimension
x = layers.Flatten()(base_model.output)
# Add a fully connected layer with 1,024 hidden units and ReLU activation
x = layers.Dense(1024, activation='relu')(x)
# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)
# Add a final softmax layer for classification
x = layers.Dense  (nb_categories, activation='softmax')(x)

model = Model(base_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001),
              loss = 'categorical_crossentropy',
              metrics = ['acc'])
```

Splitting of data into train, test and validation set

In [ ]:
```
#Number of images to load at each iteration
batch_size = 32
# only rescaling
train_datagen =  ImageDataGenerator(
    rescale=1./255
)
test_datagen =  ImageDataGenerator(
    rescale=1./255
)
# these are generators for train/test data that will read pictures #found in the defined subfolders of 'data/'
print('Total number of images for "training":')
train_generator = train_datagen.flow_from_directory(
train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical")
print('Total number of images for "validation":')
val_generator = test_datagen.flow_from_directory(
val_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(
test_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical",
shuffle=False)
```

```
Total number of images for "training":
Found 5272 images belonging to 15 classes.
Total number of images for "validation":
Found 1326 images belonging to 15 classes.
Total number of images for "testing":
Found 1125 images belonging to 15 classes.
```

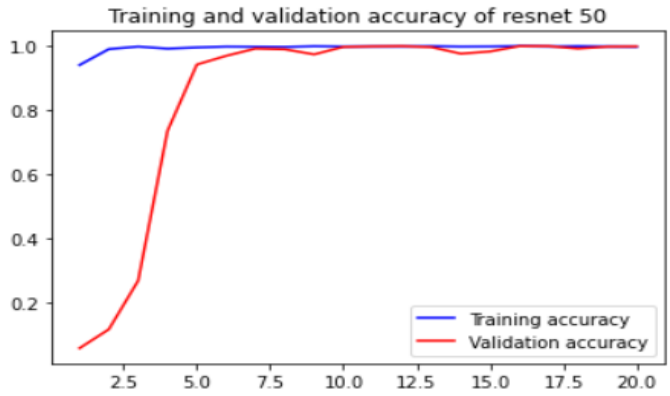
Fitting the ResNet50 model

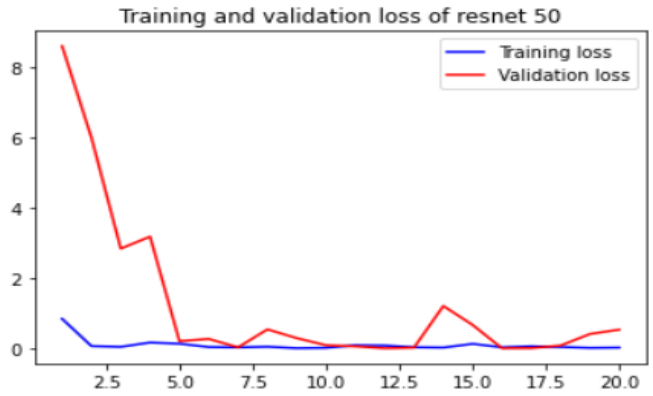
In [ ]:
```
history = model.fit(
            train_generator,
            validation_data = val_generator,
            epochs = 20,
            shuffle=True,
            verbose = 1,
            callbacks=[checkpoint,tensorboard])
```

Plot for analysing accuracy and train , validation loss

In [ ]:
```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1,len(acc)+1)
plt.figure()
plt.plot(epochs, acc, 'b', label = 'Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy of resnet 50')
plt.legend()
plt.savefig('Accuracy.jpg')
plt.figure()
plt.plot(epochs, loss, 'b', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss of resnet 50')
plt.legend()
plt.savefig('Loss.jpg')
```

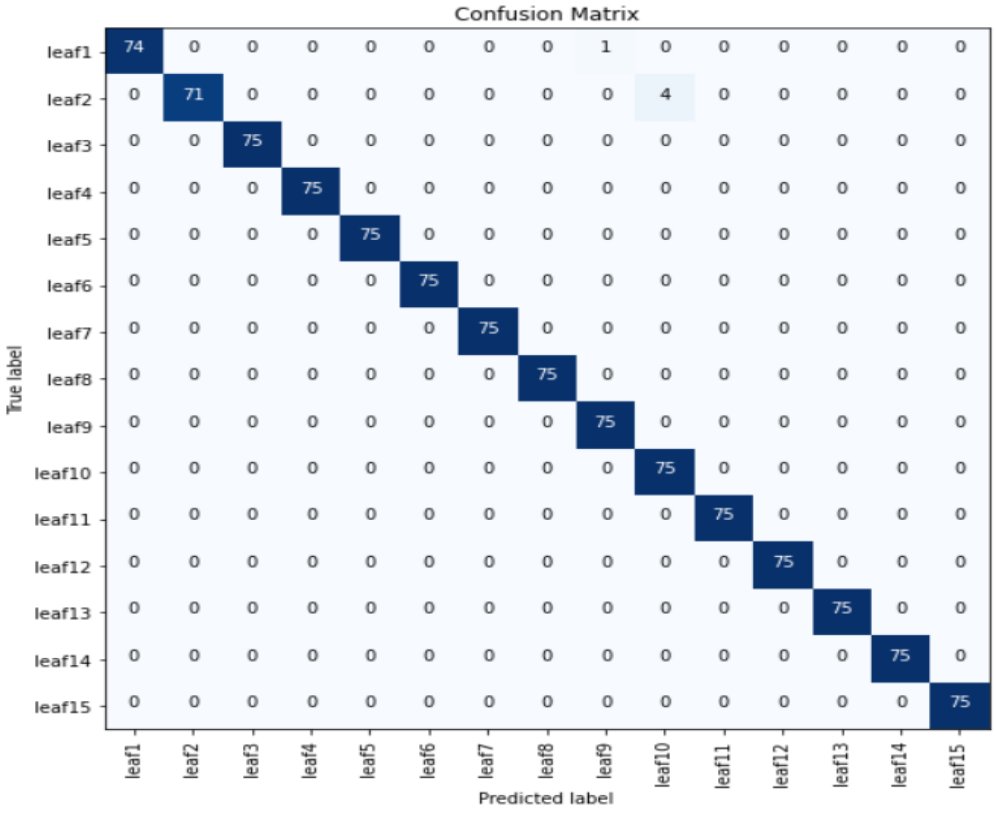
Output:





Confusion matrix

```
Confusion matrix, without normalization
```



Accuracy for test set data

```
In [ ]: accuracy = accuracy_score(test_generator.classes, y_pred)
        print("Accuracy in test set: %0.3f%% " % (accuracy * 100))

        Accuracy in test set: 99.556%
```

5.Bench marking

The existing products are Plant Net, Leaf Snap, I Naturalist, Plantsnap.

Plant Net: which helps in identifying plants species from the photographs.

Leaf Snap: plant care reminders and click free and unlimited snaps.

I Naturalist: helps in sharing the observations with community.

Plantsnap: Search the plants by the name and creates plant collections.

All the products mentioned below may or may not identify the plant species correctly, it takes time and they can just identify the species by their scientific names. If we click a picture In this applications they identify and give only the scientific names of the plant species which is difficult for non-experts like travellers, farmers etc to get information and knowledge about where the species origin, common names, etc.

For this there is a need to continuously search for implementation of better techniques which leads to better results or output or plant species identification.

Concept Generation

There are many ways to identify plants. One way is to look at the shape and size of the leaves. Another way is to look at the flowers and another way is to look at the fruit or seed. And finally by looking at the overall shape of plant.

We can leverage the power of AI to develop a system that can analyse image of different plants. The system will utilize deep learning algorithm (CNN) and computer vision techniques to identify patterns and uses high resolution images of leaves to help users identify the plant species.

Concept Development

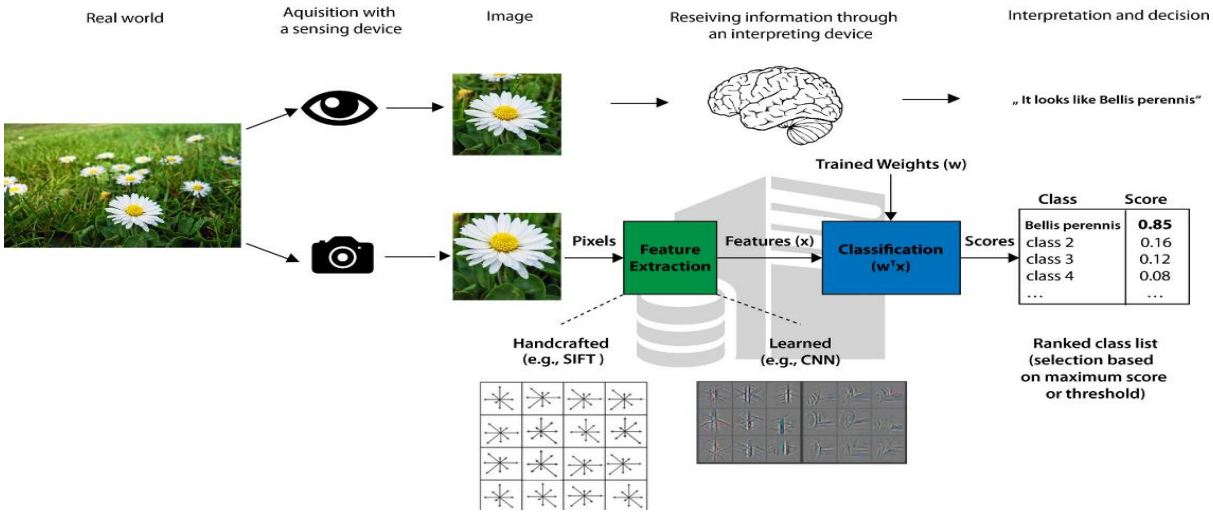
Key Features of plant species identification application

- Taking and uploading plant photos
- Plant identification by Fetching data
- Details about the plant species
- Geo tagging

Final Product Prototype (Abstract) with schematic diagram

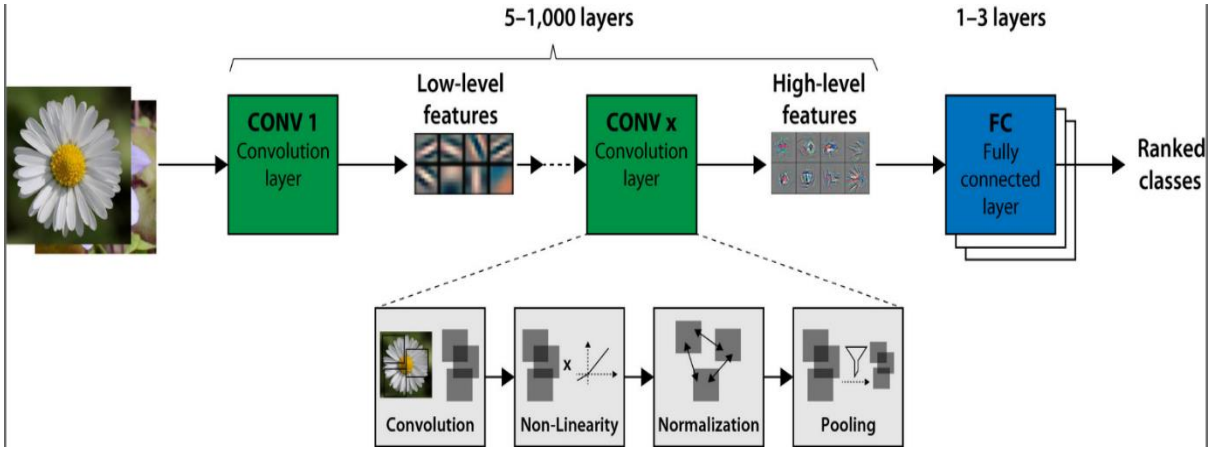
Accurate species identification is the basis for all aspects of research and is an essential component of workflow in biological research. Biologist and non-experts are asking for more efficient methods to meet the identification demand. Smart mobile devices, digital cameras as well as the mass digitisation of natural history collections led to an explosion of openly available image data in combination with modern AI techniques (machine learning and deep learning) offers tremendous opportunities for automated species identification.

Web page can be developed by using html , java, python or any frontend backend developer method.



How does algorithm works?

By using Convolutional Neural Network (CNN) the final product can be designed. CNN are comprised of one or more convolution layers followed by one or more fully connected layers as in a traditional multilayer neural network. The architecture of a CNN can be designed to take 2D structure of the input image. Local connections and weights followed by some form of pooling result in translation invariant features.



A) Feasibility

This project can be developed and deployed within a few years as SaaS(software as a service) for anyone to use.

B) Viability

As the smartphones users are more in India and the world, there will be curiosity for the peoples to get knowledge about the things around. These may help the farmers and nature lovers for identifying the unknown species. So it is viable to survive in the long-term future as well but improvements are necessary as new technologies emerge.

C) Monetization

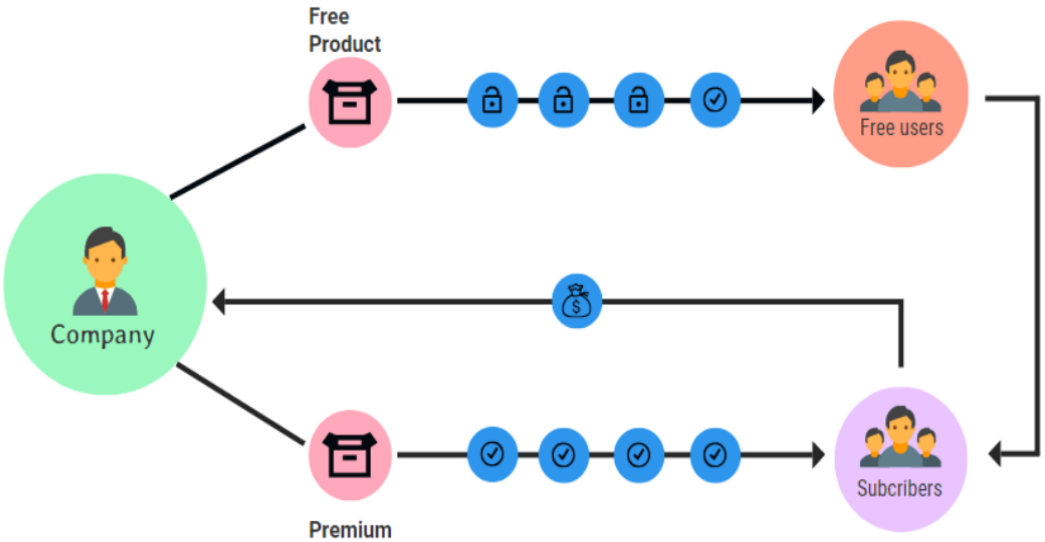
This service is directly monetizable as it can be directly released as a service on completion.

Step 2: Prototype development

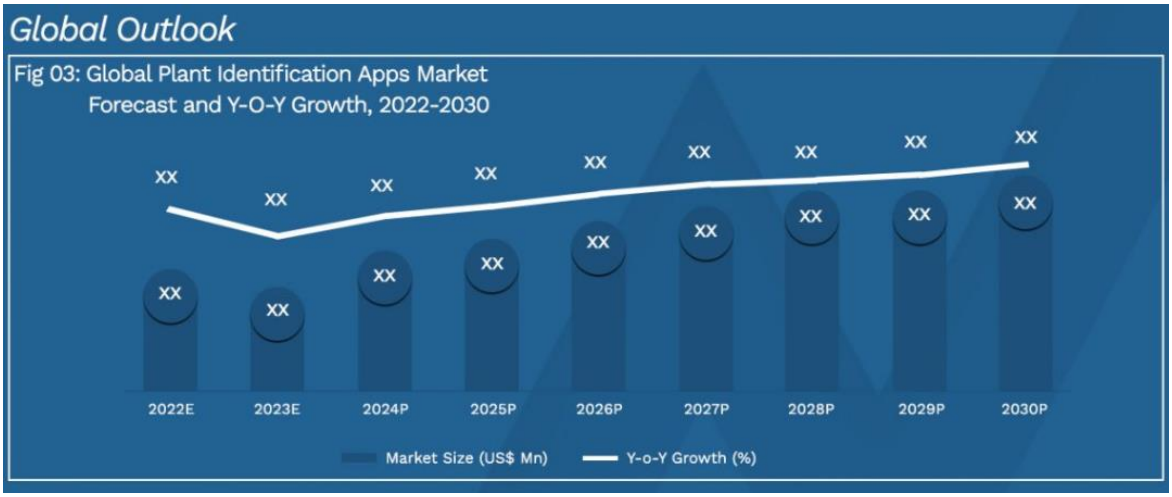
Github Link: https://github.com/Hemanagraj08/plants-species-classification-code

Step 3: Business Modelling

For this service, it is beneficial to use a Subscription Based Model, where initially some features will be provided for free to engage customers. A subscription-based model would charge users a fee to access the identification application. In a subscription model, customers are charged on a service based in which customers pay a weekly, monthly, or yearly. Subscription model can be used individuals who want to identify and get the knowledge of species.



Step 4: Financial Marketing



The above image shows the growth of the plant identification app in future. Identifying the plants correctly can be done as it requires specialized knowledge and only by the experts of botanical background. There are enormous plant species which is nearly 390,000 and each year new species are reported in the world, each plants are very different from another and may look similar but with different species. In manual identification botanists use specific defined characteristics of a plant as a key for identification which is helpful in identifying species. The identification keys involve features such as shape, texture, colour, vein structure and size of an unknow plant. Which is often time consuming and insufficient. Even botanists do not have the knowledge of all the existing plants in this world, they take a considerable amount of time to identify a plant species. Since the traditional identification methods are strenuous, there arises a need to automate the process of species identification which can serve the purpose of species recognition to some extent.

Conclusion

Identifying a plant is a useful skill that can make a difference during a critical survival situation in the wild or simply help to identify an unknown plant that sprouts up in garden. Building accurate knowledge of the identity, the geographic distribution and the evolution of living species is essential for a sustainable development of humanity as well as for biodiversity conservation. However, identification experts are drastically decreasing consequently the need for alternative and accurate identification methods applicable by non-experts is constantly increasing. Finding automatic methods for such identification is an important topic with high expectations.

Artificial intelligence systems will provide alternative tools for identification task. In a few years, we will rely on machine algorithms routinely to advance our core science while reducing the number of routine identifications performed by taxonomist allowing them to focus on experimental setup and data analytic. Furthermore, these technologies allow for a larger community observing our nature and contributing to develop an increasing interest of the society in their environment.