Code Implementation of a Medicinal Herb Classifier: An AI-Driven Approach with Tensor Flow and Python

Niranjana S

Research Associate, International Centre for Technological Innovations

ABSTRACT

There are lots of medicinal herbs present in Kerala. If identified and used properly, these herbs hold the potential to serve a multitude of purposes, ranging from providing effective home remedies for minor ailments to contributing to the prevention of lifestyle diseases. In this study, I propose a herb classifier using Convolutional Neural Networks (CNNs) in Python with the TensorFlow framework. As a part of this, I assessed the efficiency of Convolutional Neural Networks in distinguishing and categorizing the identified set of six herbs through training the model with a dataset. The performance of this model was evaluated through testing with new images. The proposed model achieved high accuracy, correctly predicting the name of 5 out of 6 new images tested.

Keywords: Convolutional Neural Networks (CNNs), Data, Dataset, Herbs, Model

INTRODUCTION

Herbs exhibit a diverse array of applications, contributing to health, well-being, and various other purposes. Their utilization spans across culinary endeavours, fragrance production, cosmetics, pest control, medicinal applications, and more. In the context of this study, a Convolutional Neural Network (CNN) model will be trained using images of six medicinal herbs commonly found in the surroundings of households in Kerala. These herbs include Cherukula (Aerva lanata), Kalluruki (Scoparia dulcis), Karuka (Cynodondactylon/Bermuda grass), Kizhukanelli (Phyllanthus niruri), Poovankurinila (Cyanthillium cinereum), and Thazhuthama (Boerhaaviadiffusa). Often, individuals overlook these beneficial herbs, mistakenly categorizing them as common weeds. This study aims to enhance the recognition and understanding of these medicinal herbs, promoting their identification amidst the surrounding vegetation typically found in Kerala's household fields.

Cherukula plant is anti-venomous and can be used as a medicine for snake bites. It is also useful in curing diarrhoea and bleeding disorders. This herb has internal healing properties and can repair the damaged internal mucous membranes Water[1]. Water boiled with the stem can be used as an effective eye wash for pink eyes. Kallurukki is used for the treatment of kidney stones. Likewise, all these herbs have many medicinal uses.

In the research led by Neda Alipour and her team, flower image classification using Convolutional Neural Network (CNN) yielded an impressive accuracy of 98.6%[2]. In a separate study conducted by Sanskruti Patel, employing machine learning techniques such as k-Nearest Neighbors (KNN), Random Forest, Support Vector Machine (SVM), and Multiple Kernel Learning + Support Vector Machine for flower image classification resulted in an accuracy approximately around 78%[3]. Given the superior accuracy demonstrated by CNN, it was chosen as the preferred method for herb classification.

Purpose:

The primary goal behind developing this herb classifier is to empower individuals to recognize the valuable herbs flourishing in the vicinity of their homes or fields. Beyond mere identification, the significance lies in the potential economic opportunities these herbs present. Given their medicinal properties, these herbs hold commercial value and can be sold in the market at a respectable price, offering a potential source of income for families. As modern healthcare trends lean towards antibiotics and homeopathy, the knowledge of traditional medicinal herbs is gradually fading among the younger generation. By aiding in the identification of these herbs, the classifier not only serves as a valuable resource for the present but also contributes to the preservation of traditional knowledge for the benefit of future generations.

METHODOLOGY

A. *Data Collection and Processing*

Images of the six herbs Cherukula, Kalluruki, Karuka, Kizhukanelli, Poovankurinila, and Thazhuthama were collected and stored in separate folders in data/herbs directory. A portion of the dataset was sourced from online repositories using Google searches, while additional data was personally captured using a camera. The size of the images collected was reduced to 224 x 224 pixels and converted to RGB colour space. Data was organized into a structured format, including image arrays and corresponding labels.

B. *Dataset Creation and Storage*

A dataset was created by categorizing images based on the identified herbs. The data was saved in the same directory of the code as a pickle file named 'herbdata.pickle'. This is for the easy data retrieval and reuse.

C. *Model Training*

The dataset was loaded and split into training and testing sets using the train_test_split function from sklearn.model_selection.A Convolutional Neural Network (CNN) model was constructed using Tensor Flow and Keras. The model architecture consists of convolutional layers, max-pooling layers, a flattening layer, dense layers, and a softmax output layer. The model was compiled with the Adam optimizer and sparse categorical crossentropy loss for multiclass classification. Training was performed using the training set with 30 epochs and a batch size of 14. The accuracy and Loss were analysed. Loss is the difference between the predicted value by the model and the true value. Accuracy is one of the metrics to measure the performance of the model[4]. The model should have low loss and better accuracy for better performance.

D. *Model Evaluation and Saving*

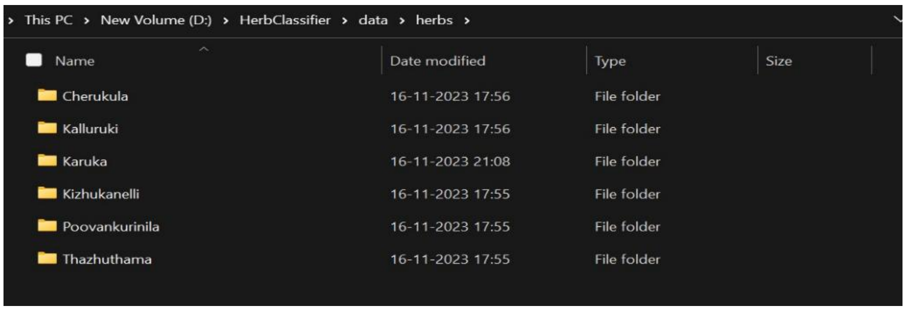
Using the testing set, the trained model was evaluated. Later, the out-of-range labels were filtered out to ensure model stability. The trained model was saved as 'myherbmodel.h5' for future use.

E. *Testing*

New images were captured by camera and then saved in a folder 'D:\Test' for the testing purpose. These images were loaded (getting feature and labels) using the load_data_from_folder function written. Then the pre-trained model ('myherbmodel.h5') was loaded for the purpose of prediction. Predictions were made on the new images using the model. Using Matplotlib, the image and predicted herb category was displayed for each image.

CODE IMPLEMENTATION

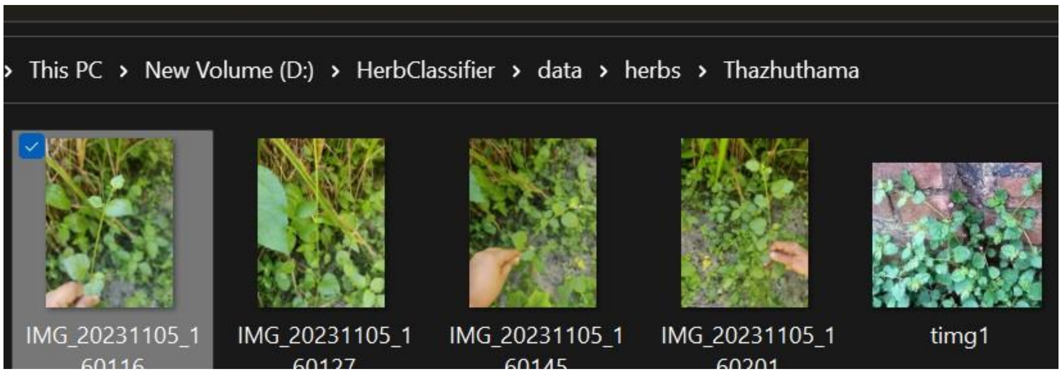
Collected data (images of the six herbs) saved in data/herbs directory.



This PC › New Volume (D:) › HerbClassifier › data › herbs ›

Name	Date modified	Type	Size
Cherukula	16-11-2023 17:56	File folder	
Kalluruki	16-11-2023 17:56	File folder	
Karuka	16-11-2023 21:08	File folder	
Kizhukanelli	16-11-2023 17:55	File folder	
Poovankurinila	16-11-2023 17:55	File folder	
Thazhuthama	16-11-2023 17:55	File folder	

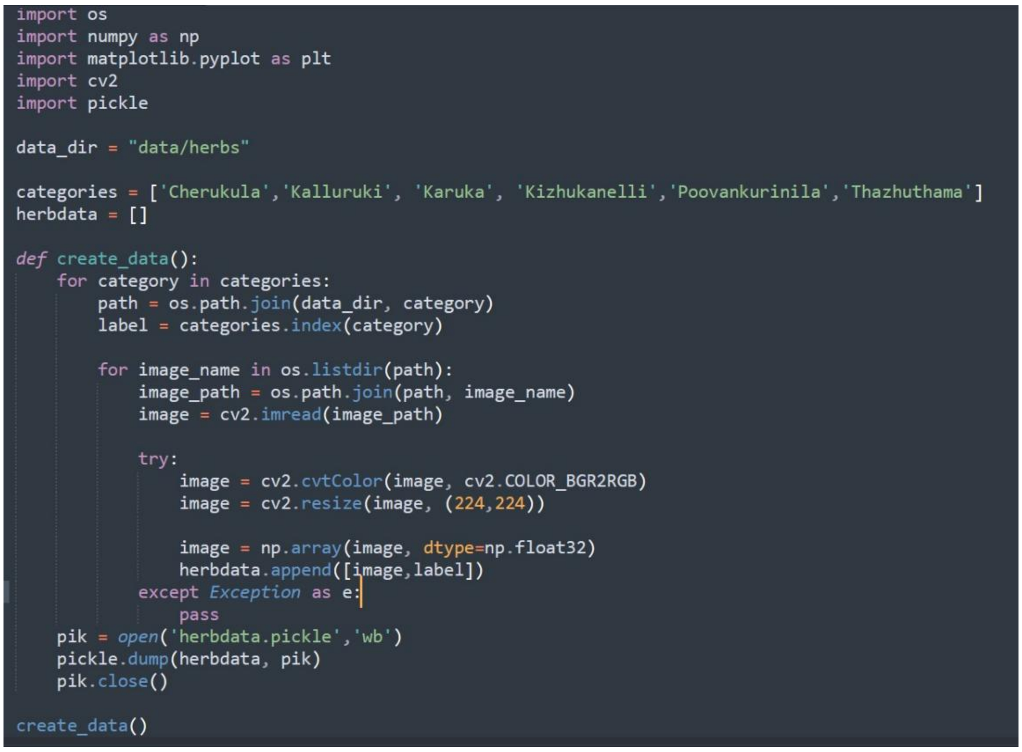
Image 1: Images of herbs stored in it's corresponding folders.



This PC › New Volume (D:) › HerbClassifier › data › herbs › Thazhuthama

Image 2: Image Samples of Thazhuthama

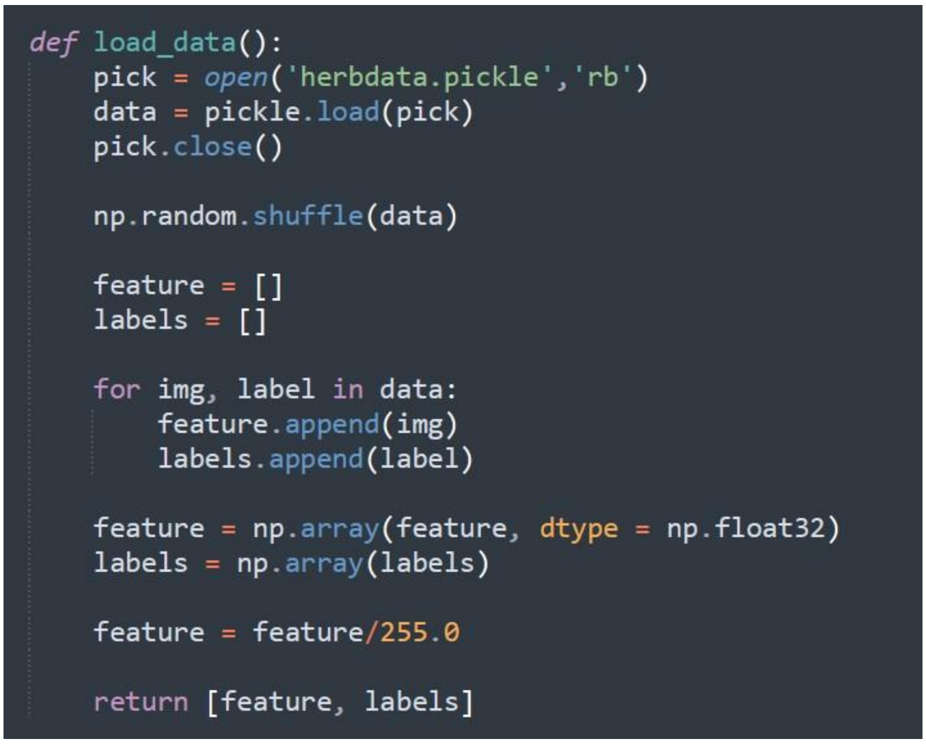
A function called create_data() is defined to process, create, and store the data. On calling this function, the images are read using OpenCV. Then the image is resized to 224 x 224 pixels and converted the color space of it to RGB. Then it is converted to a NumPy array of float32 type. This data is written into a binary file ‘herbdata.pickle’. This file will be saved in the code directory.



```
import os
import numpy as np
import matplotlib.pyplot as plt
import cv2
import pickle

data_dir = "data/herbs"

categories = ['Cherukula','Kalluruki', 'Karuka', 'Kizhukanelli','Poovankurinila','Thazhuthama']
herbdata = []

def create_data():
    for category in categories:
        path = os.path.join(data_dir, category)
        label = categories.index(category)

        for image_name in os.listdir(path):
            image_path = os.path.join(path, image_name)
            image = cv2.imread(image_path)

            try:
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                image = cv2.resize(image, (224,224))

                image = np.array(image, dtype=np.float32)
                herbdata.append([image,label])
            except Exception as e:
                pass
    pik = open('herbdata.pickle','wb')
    pickle.dump(herbdata, pik)
    pik.close()

create_data()
```

Image 3: process, create, and store the data.

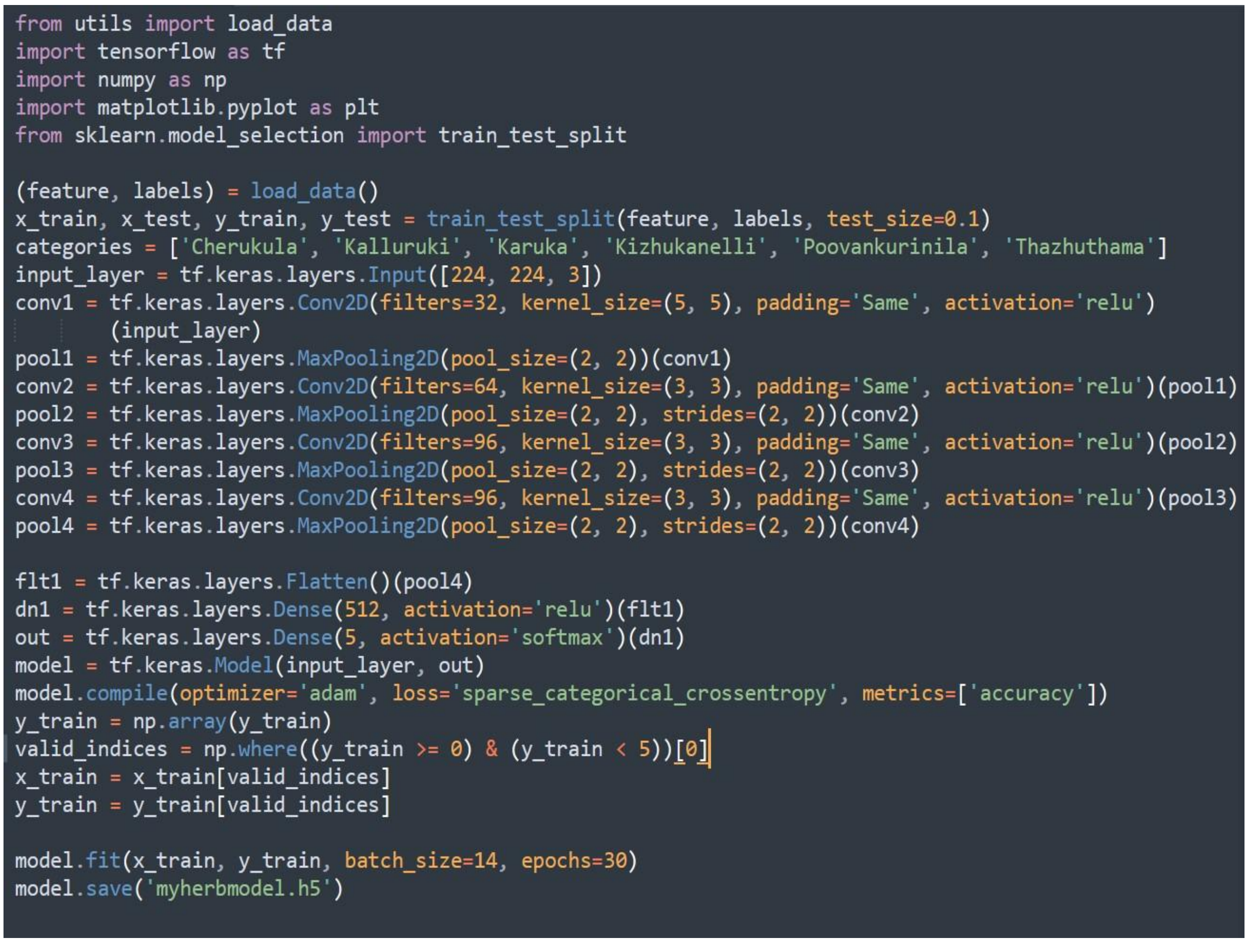
A function load_data() is defined to retrieve data from the dataset created, which in this case is herb.pickle file. Image data and labels are returned.



```
def load_data():
    pick = open('herbdata.pickle','rb')
    data = pickle.load(pick)
    pick.close()

    np.random.shuffle(data)

    feature = []
    labels = []

    for img, label in data:
        feature.append(img)
        labels.append(label)

    feature = np.array(feature, dtype = np.float32)
    labels = np.array(labels)

    feature = feature/255.0

    return [feature, labels]
```

Image 4: To retrieve data from the dataset.

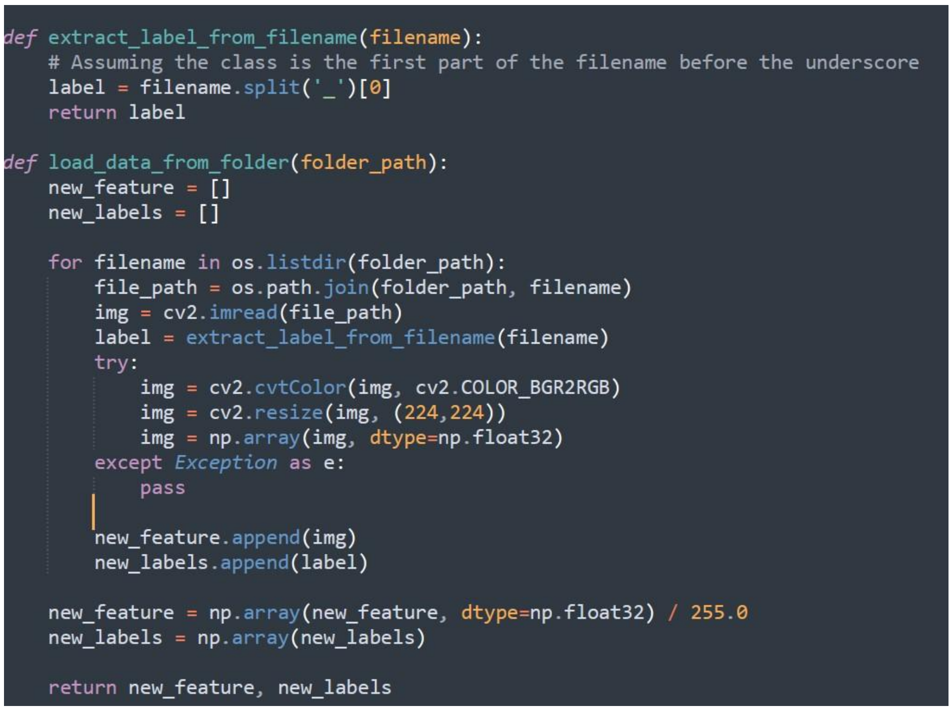
Model training: Train, evaluate, and save the model.



```
from utils import load_data
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

(feature, labels) = load_data()
x_train, x_test, y_train, y_test = train_test_split(feature, labels, test_size=0.1)
categories = ['Cherukula', 'Kalluruki', 'Karuka', 'Kizhukanelli', 'Poovankurinila', 'Thazhuthama']
input_layer = tf.keras.layers.Input([224, 224, 3])
conv1 = tf.keras.layers.Conv2D(filters=32, kernel_size=(5, 5), padding='Same', activation='relu')
        (input_layer)
pool1 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), padding='Same', activation='relu')(pool1)
pool2 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(conv2)
conv3 = tf.keras.layers.Conv2D(filters=96, kernel_size=(3, 3), padding='Same', activation='relu')(pool2)
pool3 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(conv3)
conv4 = tf.keras.layers.Conv2D(filters=96, kernel_size=(3, 3), padding='Same', activation='relu')(pool3)
pool4 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(conv4)

flt1 = tf.keras.layers.Flatten()(pool4)
dn1 = tf.keras.layers.Dense(512, activation='relu')(flt1)
out = tf.keras.layers.Dense(5, activation='softmax')(dn1)
model = tf.keras.Model(input_layer, out)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
y_train = np.array(y_train)
valid_indices = np.where((y_train >= 0) & (y_train < 5))[0]
x_train = x_train[valid_indices]
y_train = y_train[valid_indices]

model.fit(x_train, y_train, batch_size=14, epochs=30)
model.save('myherbmodel.h5')
```

Image 5: Training the model and saving it as myherbmodel.h5

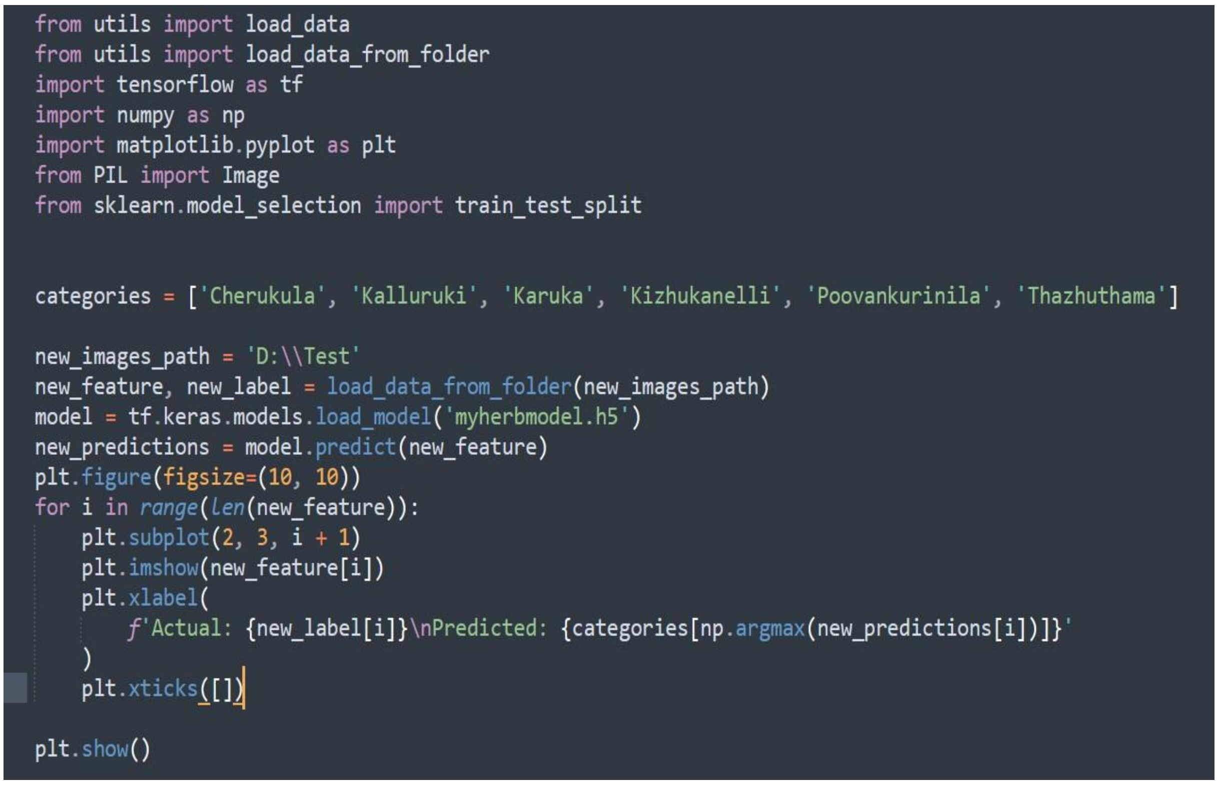
Testing with new images – new images of the herbs were captured and saved in 'D:\Test' folder. A function 'load_data_from_folder()' was defined to get image data from the given test directory.



```
def extract_label_from_filename(filename):
    # Assuming the class is the first part of the filename before the underscore
    label = filename.split('_')[0]
    return label

def load_data_from_folder(folder_path):
    new_feature = []
    new_labels = []

    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)
        img = cv2.imread(file_path)
        label = extract_label_from_filename(filename)
        try:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, (224,224))
            img = np.array(img, dtype=np.float32)
        except Exception as e:
            pass

        new_feature.append(img)
        new_labels.append(label)

    new_feature = np.array(new_feature, dtype=np.float32) / 255.0
    new_labels = np.array(new_labels)

    return new_feature, new_labels
```

Image 6: To load image data from the test folder.

Later, using Matplotlib, the images, their actual name and predicted name are shown.

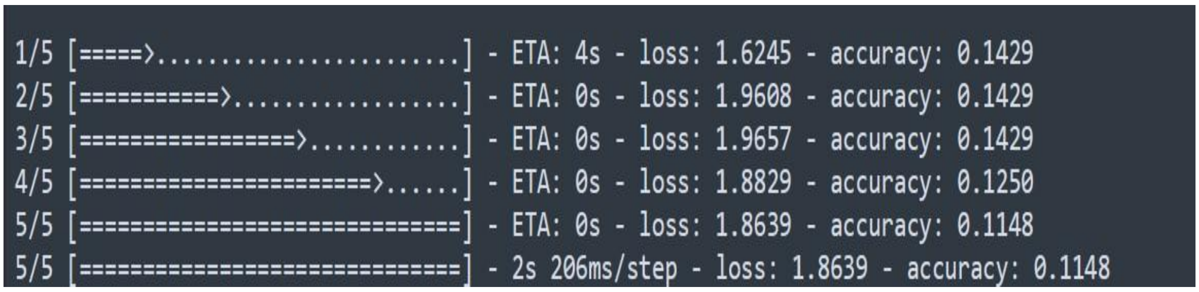


```python
from utils import load_data
from utils import load_data_from_folder
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from sklearn.model_selection import train_test_split


categories = ['Cherukula', 'Kalluruki', 'Karuka', 'Kizhukanelli', 'Poovankurinila', 'Thazhuthama']

new_images_path = 'D:\\Test'
new_feature, new_label = load_data_from_folder(new_images_path)
model = tf.keras.models.load_model('myherbmodel.h5')
new_predictions = model.predict(new_feature)
plt.figure(figsize=(10, 10))
for i in range(len(new_feature)):
    plt.subplot(2, 3, i + 1)
    plt.imshow(new_feature[i])
    plt.xlabel(
        f'Actual: {new_label[i]}\nPredicted: {categories[np.argmax(new_predictions[i])]}'
    )
    plt.xticks([])

plt.show()
```

Image 7: To predict the herb and show result.

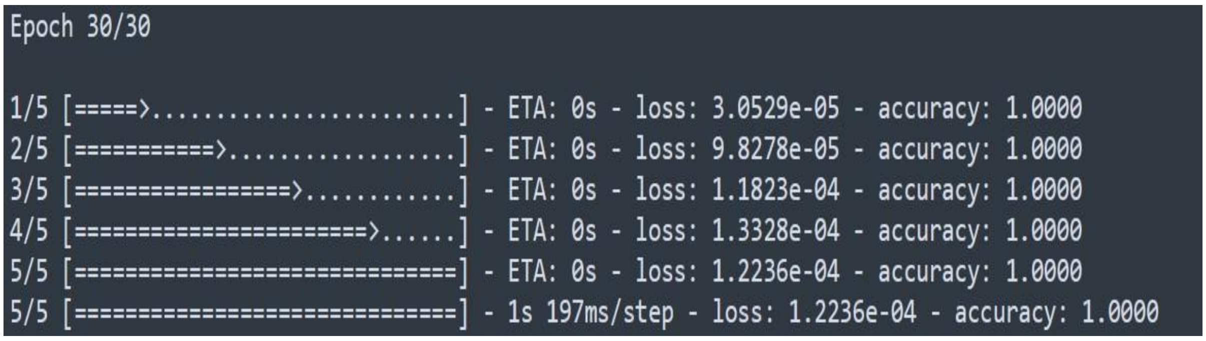
TEST RESULTS

Training Result – Loss and Accuracy values. After the model training an accuracy value of 1.000 and loss value of $1.223*10^{-4}$ was successfully achieved.



```
1/5 [=====>........................] - ETA: 4s - loss: 1.6245 - accuracy: 0.1429
2/5 [===========>..................] - ETA: 0s - loss: 1.9608 - accuracy: 0.1429
3/5 [=================>............] - ETA: 0s - loss: 1.9657 - accuracy: 0.1429
4/5 [=======================>......] - ETA: 0s - loss: 1.8829 - accuracy: 0.1250
5/5 [==============================] - ETA: 0s - loss: 1.8639 - accuracy: 0.1148
5/5 [==============================] - 2s 206ms/step - loss: 1.8639 - accuracy: 0.1148
```

Image 8: Loss and Accuracy values at Epoch 1/30



```
Epoch 30/30

1/5 [=====>........................] - ETA: 0s - loss: 3.0529e-05 - accuracy: 1.0000
2/5 [===========>..................] - ETA: 0s - loss: 9.8278e-05 - accuracy: 1.0000
3/5 [=================>............] - ETA: 0s - loss: 1.1823e-04 - accuracy: 1.0000
4/5 [=======================>......] - ETA: 0s - loss: 1.3328e-04 - accuracy: 1.0000
5/5 [==============================] - ETA: 0s - loss: 1.2236e-04 - accuracy: 1.0000
5/5 [==============================] - 1s 197ms/step - loss: 1.2236e-04 - accuracy: 1.0000
```

Image 9: Loss and Accuracy values at Epoch 30/30

Image Prediction Result: Five out of six images were predicted correctly. An accuracy of 83% is achieved.

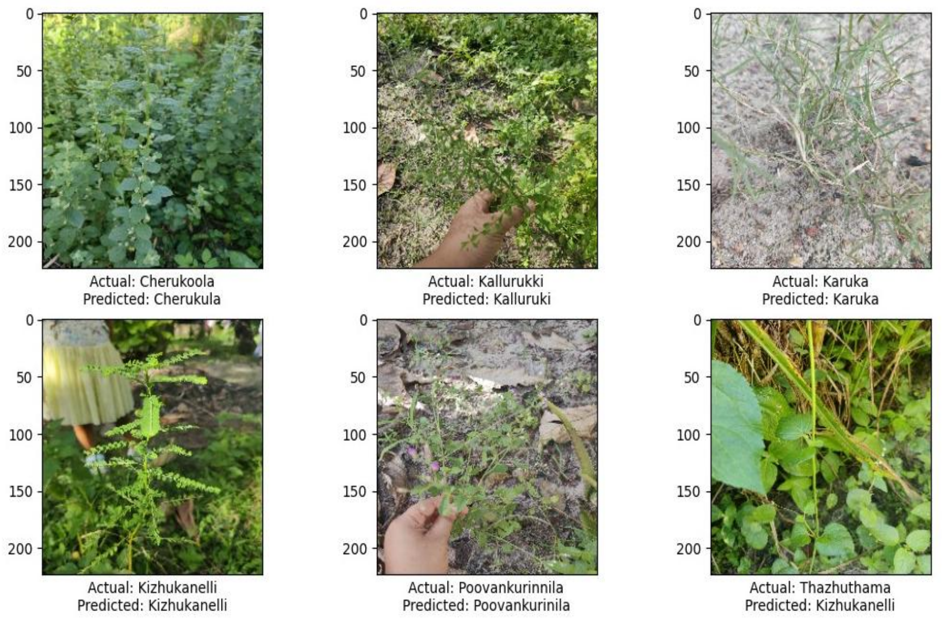


Image 10: Prediction result

CONCLUSION

After training the herb classifier model using Python and Tensor flow, the model achieved a remarkable accuracy of 1.000 with a very low loss of 1.23×10^{-4}. When tested with new captured images, model predicted five out of the six images correctly, resulting in a high accuracy. This high accuracy implies the ability to recognize the herb classes in the training set. Overall, the herb classifier showcases promising results and lays the foundation for further refinement and application in real-world scenarios.

FUTURE IMPLEMENTATIONS

- Increasing the number of training images has the potential to enhance the effectiveness of the herb classifier.
- Future enhancements may involve expanding the herb classes within the training set, broadening the classifier's capability to detect and predict a more extensive range of herb types.
- A mobile application which will detect the herbs among the weeds using image processing.

REFERENCES

[1]. Dashapushpam Part 7 – Cheroola (Ashmaribhedaka)-Aerva Lanata, 2021. Posts, Sakalya Ayurveda https://www.sakalya.com/post/dashapushpam-part-7-cheroola-ashmaribhedaka-aerva-lanata [visited on 20th November, 2023]
[2]. Alipour, N., Tarkhaneh, O., Awrangjeb, M. and Tian, H., 2021, May. Flower image classification using deep convolutional neural network. In 2021 7th International Conference on Web Research (ICWR) (pp. 1-4). IEEE.
[3]. Patel, I. and Patel, S., 2019. Flower identification and classification using computer vision and machine learning techniques. International Journal of Engineering and Advanced Technology (IJEAT), 8(6), pp.277-285.
[4]. Loss vs Accuracy, Harshit Kumar, 2018. Blog - Loss vs Accuracy, kharshit.github.io https://kharshit.github.io/blog/2018/12/07/loss-vs-accuracy [visited on 20th November, 2023]
[5]. Tutorial 10: Flower Classification with Deep Neural Network with Tensorflow and Python Programming, Md. Iqbaal Hossain, 2020. Youtube, https://www.youtube.com/watch?v=POO1gdUJ7yE [visited on 20th November, 2023].