```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import seaborn as sns
import pandas as pd
from skimage.filters import sobel
```

```
!pip3 install glob2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: glob2 in /usr/local/lib/python3.9/dist-packages (0.7)
```

```
pip install numpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (1.22.4)
```

```
import glob
```

```
pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.9/dist-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.9/dist-packages (from opencv-python) (1.22.4)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Resize images to
SIZE = 128
```

```
# Capture images and labels into arrays
# Start by creating empty lists
train_images = []
train_labels = []
```

```
for directory_path in glob.glob("/content/drive/MyDrive/Colab Notebooks/image_group/train/*"):
    label = directory_path.split(os.sep)[-1]
    print(label)
    for img_path in glob.glob(os.path.join(directory_path, "*.jpg")):
        print(img_path)
        img = cv2.imread(img_path, cv2.IMREAD_COLOR) #Reading color images
        if img is None:
            print('Wrong path:', path)
        else:
            img = cv2.resize(img, (SIZE, SIZE)) #Resize images
            #img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #Optional step. Change BGR to RGB
            train_images.append(img)
            train_labels.append(label)
```

```
spongebob
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/Spongebob Squarepants Photo_ spongebob.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants_ Cartoon _ Nick.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/Spongebob Squarepants_ Over 20 years of.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants Patrick Star.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants_ Cartoon _ Nick (1).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants Squidward.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants controversies.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants Patrick Star (1).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants _ Spongebob.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/When SpongeBob SquarePants Was Just a.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/Spongebob Squarepants reading book.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/Top 13 Krabby Patty Moments_ 🍔 _TBT.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/the SpongeBob Movie_ Sponge on.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob SquarePants.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob-SquarePants-Cartoon-Food-1.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/spongebob/SpongeBob Squarepants Story_ Cartoons.jpg
r2d2
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (8).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (3).jpg
```

```
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (1).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/2Q__ (1).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (5).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (6).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/2Q__ (2).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (4).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (2).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (9).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (10).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (16).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (17).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (14).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (11).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (13).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (15).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (19).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/2Q__ (3).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (20).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/9k_.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (21).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (18).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (24).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (22).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/images (23).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/9k_ (1).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/Hasbro Star Wars Smart App Enabled R2-D2 Remote_yyt.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/9k_ (2).jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/r2d2/Hasbro Star Wars Interactive R2d2 Stock Photo -_yyt.jpg
vader
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader18.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader23.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader11.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader5.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader8.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader19.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader7.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader28.jpg
/content/drive/MyDrive/Colab Notebooks/image_group/train/vader/vader6.jpg
```

```
train_images = np.array(train_images)
train_labels = np.array(train_labels)
np.shape(train_images)
```

```
(76, 128, 128, 3)
```

```
np.shape(train_labels)
```

```
(76,)
```

```
#Do exactly the same for test/validation images
# test
test_images = []
test_labels = []
for directory_path in glob.glob("/content/drive/MyDrive/Colab Notebooks/image_group/validation/*"):
    fruit_label = directory_path.split(os.sep)[-1]
    print(fruit_label)
    for img_path in glob.glob(os.path.join(directory_path, "*.jpg")):
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img, (SIZE, SIZE))
        #img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #Optional
        test_images.append(img)
        test_labels.append(fruit_label)
```

```
test_images = np.array(test_images)
test_labels = np.array(test_labels)
```

```
r2d2
spongebob
vader
```

```
np.shape(test_images)
```

```
(59, 128, 128, 3)
```

```
np.shape(test_labels)
```

```
(59,)
```

```
#Encode labels from text (folder names) to integers.
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(test_labels)
test_labels_encoded = le.transform(test_labels)
le.fit(train_labels)
train_labels_encoded = le.transform(train_labels)


#Split data into test and train datasets (already split but assigning to meaningful convention)
#If you only have one dataset then split here
x_train, y_train, x_test, y_test = train_images, train_labels_encoded, test_images, test_labels_encoded


# Normalize pixel values to between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0


# FEATURE EXTRACTOR function
# input shape is (n, x, y, c) - number of images, x, y, and channels
def feature_extractor(dataset):
    x_train = dataset
    image_dataset = pd.DataFrame()
    for image in range(x_train.shape[0]):  #iterate through each file
        #print(image)

        df = pd.DataFrame()  #Temporary data frame to capture information for each loop.
        #Reset dataframe to blank after each loop.

        input_img = x_train[image, :,:,:]
        img = input_img
    ################################################################
    #START ADDING DATA TO THE DATAFRAME
    #Add feature extractors, e.g. edge detection, smoothing, etc.

         # FEATURE 1 - Pixel values

        #Add pixel values to the data frame
        pixel_values = img.reshape(-1)
        df['Pixel_Value'] = pixel_values   #Pixel value itself as a feature
        #df['Image_Name'] = image   #Capture image name as we read multiple images

        # FEATURE 2 - Bunch of Gabor filter responses

                #Generate Gabor features
        num = 1  #To count numbers up in order to give Gabor features a lable in the data frame
        kernels = []
        for theta in range(2):   #Define number of thetas
            theta = theta / 4. * np.pi
            for sigma in (1, 3):  #Sigma with 1 and 3
                lamda = np.pi/4
                gamma = 0.5
                gabor_label = 'Gabor' + str(num)  #Label Gabor columns as Gabor1, Gabor2, etc.
    #               print(gabor_label)
                ksize=9
                kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta, lamda, gamma, 0, ktype=cv2.CV_32F)
                kernels.append(kernel)
                #Now filter the image and add values to a new column
                fimg = cv2.filter2D(img, cv2.CV_8UC3, kernel)
                filtered_img = fimg.reshape(-1)
                df[gabor_label] = filtered_img  #Labels columns as Gabor1, Gabor2, etc.
                print(gabor_label, ': theta=', theta, ': sigma=', sigma, ': lamda=', lamda, ': gamma=', gamma)
                num += 1  #Increment for gabor column label


        # FEATURE 3 Sobel
        edge_sobel = sobel(img)
        edge_sobel1 = edge_sobel.reshape(-1)
        df['Sobel'] = edge_sobel1

        #Add more filters as needed

        #Append features from current image to the dataset
        image_dataset = image_dataset.append(df)

    return image_dataset
```

```
#Extract features from training images
image_features = feature_extractor(x_train)
```

```
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
```

```
#Reshape to a vector for Random Forest / SVM training
n_features = image_features.shape[1]
image_features = np.expand_dims(image_features, axis=0)
X_for_RF = np.reshape(image_features, (x_train.shape[0], -1))  #Reshape to #images, features


#Define the classifier
from sklearn.ensemble import RandomForestClassifier
RF_model = RandomForestClassifier(n_estimators = 50, random_state = 42)


# Fit the model on training data
RF_model.fit(X_for_RF, y_train) #For sklearn no one hot encoding
```

```
▾ RandomForestClassifier
RandomForestClassifier(n_estimators=50, random_state=42)
```

```
#Predict on Test data
#Extract features from test data and reshape, just like training data
```

```
test_features = feature_extractor(x_test)
test_features = np.expand_dims(test_features, axis=0)
test_for_RF = np.reshape(test_features, (x_test.shape[0], -1))
```

```
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future v
  image_dataset = image_dataset.append(df)
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
```

```
#Predict on test
test_prediction = RF_model.predict(test_for_RF)
#Inverse le transform to get original label back.
test_prediction = le.inverse_transform(test_prediction)


#Print overall accuracy
from sklearn import metrics
print ("Accuracy = ", metrics.accuracy_score(test_labels, test_prediction))
```

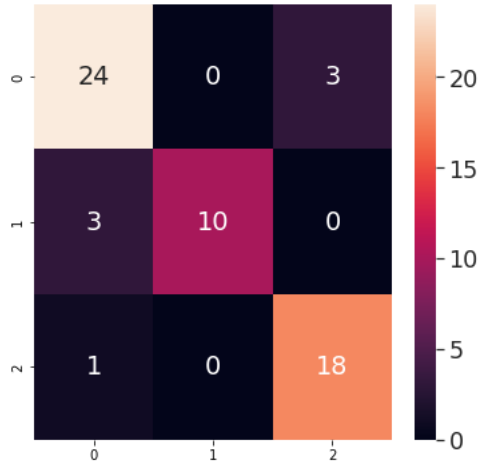
```
Accuracy =  0.8813559322033898
```

```
#Print confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test_labels, test_prediction)

fig, ax = plt.subplots(figsize=(6,6))         # Sample figsize in inches
```

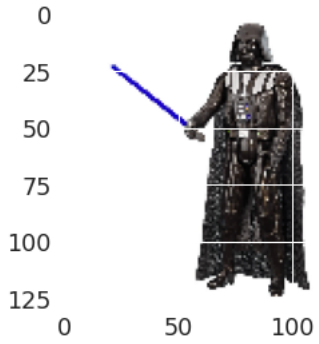
```
sns.set(font_scale=1.6)
sns.heatmap(cm, annot=True, ax=ax)
```

```
<Axes: >
```




```
#Check results on a few random images
import random
n=random.randint(0, x_test.shape[0]-1) #Select the index of image to be loaded for testing
img = x_test[n]
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7f42392397c0>
```




```
#Extract features and reshape to right dimensions
input_img = np.expand_dims(img, axis=0) #Expand dims so the input is (num images, x, y, c)
input_img_features=feature_extractor(input_img)
input_img_features = np.expand_dims(input_img_features, axis=0)
input_img_for_RF = np.reshape(input_img_features, (input_img.shape[0], -1))
```

```
Gabor1 : theta= 0.0 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor2 : theta= 0.0 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor3 : theta= 0.7853981633974483 : sigma= 1 : lamda= 0.7853981633974483 : gamma= 0.5
Gabor4 : theta= 0.7853981633974483 : sigma= 3 : lamda= 0.7853981633974483 : gamma= 0.5
<ipython-input-18-c685e83916bb>:56: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future ver
  image_dataset = image_dataset.append(df)
```

```
#Predict
img_prediction = RF_model.predict(input_img_for_RF)
img_prediction = le.inverse_transform([img_prediction])  #Reverse the label encoder to original name
print("The prediction for this image is: ", img_prediction)
print("The actual label for this image is: ", test_labels[n])
```

```
The prediction for this image is:  ['r2d2']
The actual label for this image is:  vader
/usr/local/lib/python3.9/dist-packages/sklearn/preprocessing/_label.py:155: DataConversionWarning: A column-vector y was passed when a 1
  y = column_or_1d(y, warn=True)
```

Below is new code based on your code above, please let me know what you think. Improvements to the code include replacing the Random Forest Classifier with a Convolutional Neural Network (CNN), which is more suitable for image classification tasks. The code now uses the TensorFlow and Keras libraries to create and train the CNN. Other changes are for readability and simplicity.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import os
import seaborn as sns
import pandas as pd
import glob
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
# Mount Google Drive (only necessary if using Google Colab)
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Image dimensions and path
SIZE = 128
train_path = "/content/drive/MyDrive/Colab Notebooks/image_group/train/*"
val_path = "/content/drive/MyDrive/Colab Notebooks/image_group/validation/*"

def load_images_labels(path):
    images = []
    labels = []

    for directory_path in glob.glob(path):
        label = directory_path.split(os.sep)[-1]
        print(label)
        for img_path in glob.glob(os.path.join(directory_path, "*.jpg")):
            img = cv2.imread(img_path, cv2.IMREAD_COLOR)
            img = cv2.resize(img, (SIZE, SIZE))
            images.append(img)
            labels.append(label)

    images = np.array(images)
    labels = np.array(labels)
    return images, labels
```

```
# Load train and validation images and labels
train_images, train_labels = load_images_labels(train_path)
val_images, val_labels = load_images_labels(val_path)

# Encode labels
le = LabelEncoder()
train_labels_encoded = le.fit_transform(train_labels)
val_labels_encoded = le.transform(val_labels)

# Split data into train and test datasets
x_train, x_test, y_train, y_test = train_test_split(train_images, train_labels_encoded, test_size=0.2, random_state=42)

# Normalize pixel values to between 0 and 1
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
spongebob
r2d2
vader
r2d2
spongebob
vader
```

```
# Create a simple CNN
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(SIZE, SIZE, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(np.unique(train_labels_encoded)), activation='softmax'))



# Compile the model
model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

# Train the model
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

# Evaluate the model
val_loss, val_acc = model.evaluate(val_images, val_labels_encoded)
print('Validation accuracy:', val_acc)

# Predict on validation dataset
val_predictions = model.predict(val_images)
val_predictions = np.argmax(val_predictions, axis=1)
```

```
Epoch 1/10
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5585: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True
  output, from_logits = _get_logits(
2/2 [==============================] - 12s 546ms/step - loss: 2.0342 - accuracy: 0.4333 - val_loss: 2.9905 - val_accuracy: 0.4375
Epoch 2/10
2/2 [==============================] - 0s 49ms/step - loss: 2.5561 - accuracy: 0.4000 - val_loss: 1.0056 - val_accuracy: 0.5625
Epoch 3/10
2/2 [==============================] - 0s 48ms/step - loss: 0.9599 - accuracy: 0.5500 - val_loss: 0.9025 - val_accuracy: 0.7500
Epoch 4/10
2/2 [==============================] - 0s 44ms/step - loss: 0.7149 - accuracy: 0.9500 - val_loss: 0.7638 - val_accuracy: 0.6875
Epoch 5/10
2/2 [==============================] - 0s 43ms/step - loss: 0.5052 - accuracy: 0.9167 - val_loss: 0.5770 - val_accuracy: 0.8125
Epoch 6/10
2/2 [==============================] - 0s 46ms/step - loss: 0.2978 - accuracy: 0.9667 - val_loss: 1.3608 - val_accuracy: 0.5625
Epoch 7/10
2/2 [==============================] - 0s 44ms/step - loss: 0.5227 - accuracy: 0.7833 - val_loss: 1.4049 - val_accuracy: 0.6250
Epoch 8/10
2/2 [==============================] - 0s 47ms/step - loss: 0.6756 - accuracy: 0.8000 - val_loss: 0.8746 - val_accuracy: 0.7500
Epoch 9/10
2/2 [==============================] - 0s 47ms/step - loss: 0.3399 - accuracy: 0.9000 - val_loss: 0.2559 - val_accuracy: 0.8750
Epoch 10/10
2/2 [==============================] - 0s 44ms/step - loss: 0.1813 - accuracy: 0.9500 - val_loss: 1.7054 - val_accuracy: 0.6875
2/2 [==============================] - 0s 139ms/step - loss: 79.1997 - accuracy: 0.8644
Validation accuracy: 0.8644067645072937
2/2 [==============================] - 0s 6ms/step
```

```
# Print confusion matrix
cm = confusion_matrix(val_labels_encoded, val_predictions)
fig, ax = plt.subplots(figsize=(6, 6))
sns.set(font_scale=1.6)
sns.heatmap(cm, annot=True, ax=ax)
```

```
<Axes: >
```