Image Analysis

Chakradhar Reddy Donuri

E949F496

I. CODE

Recognize.py

```
# import the necessary packages
from pyimagesearch.localbinarypatterns import LocalBinaryPatterns
from sklearn.svm import LinearSVC
from imutils import paths
import argparse
import cv2
import os
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-t", "--training", required=True, help="images\training")
ap.add_argument("-e", "--testing", required=True, help="images\testing")
args = vars(ap.parse_args())
# initialize the local binary patterns descriptor along with
# the data and label lists
desc = LocalBinaryPatterns(24, 8)
data = []
labels = []
# loop over the training images
for imagePath in paths.list_images(args["training"]):
    # load the image, convert it to grayscale, and describe it
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = desc.describe(gray)
    # extract the label from the image path, then update the
    # label and data lists
    labels.append(imagePath.split(os.path.sep)[-2])
    data.append(hist)
# train a Linear SVM on the data
```

```
model = LinearSVC(C=100.0, random_state=42, max_iter=10000)
model.fit(data, labels)

y_test = []
y_predic = []
# loop over the testing images
for imagePath in paths.list_images(args["testing"]):
    # load the image, convert it to grayscale, describe it,
    # and classify it
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = desc.describe(gray)

    y_test.append(imagePath.split(os.path.sep)[-2])
    prediction = model.predict(hist.reshape(1, -1))
    y_predic.append(prediction[0])

    # display the image and the prediction
    #cv2.putText(image, prediction[0], (10, 30), cv2.FONT_HERSHEY_SIMPLEX,1.0, (0, 0, 255), 3)
    # plt.imshow(image)
    #cv2.imshow("Image", image)
    # cv2.waitKey(0)

print('precision: ')
print(precision_score(y_test, y_predic, pos_label='Live', average='macro'))
print('recall: ')
print(recall_score(y_test, y_predic, pos_label='Live', average='macro'))
print('accuracy: ')
print(accuracy_score(y_test, y_predic))
```

LocalBinaryPatterns.py

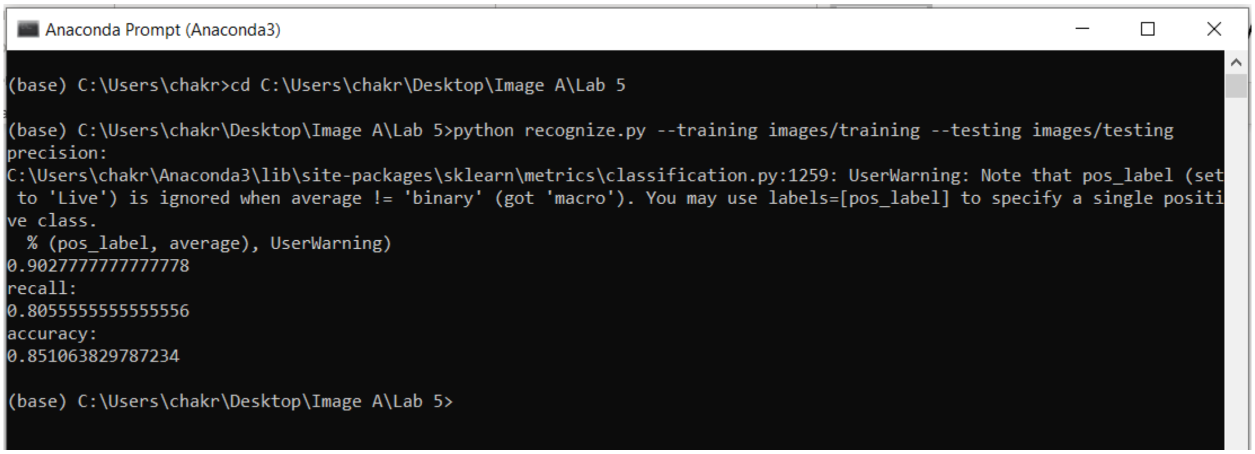
```
from skimage import feature
import numpy as np
class LocalBinaryPatterns:
    def __init__(self,numPoints, radius):
        self.numPoints =numPoints
        self.radius = radius

    def describe(self,image,eps=1e-7):
        lbp = feature.local_binary_pattern(image,self.numPoints,self.radius,method="uniform")
        (hist, _) = np.histogram(lbp.ravel(),
            bins=np.arange(0, self.numPoints + 3),range=(0, self.numPoints + 2))
    # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + eps)
        # return the histogram of Local Binary Patterns
        return hist
```

Steps to execute:

open Specific folder in anaconda prompt
run "pip install imutils"
now run "python recognize.py --training images/training --testing images/testing"

you'll see the outputs in the prompt

Results:



```
(base) C:\Users\chakr>cd C:\Users\chakr\Desktop\Image A\Lab 5

(base) C:\Users\chakr\Desktop\Image A\Lab 5>python recognize.py --training images/training --testing images/testing
precision:
C:\Users\chakr\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1259: UserWarning: Note that pos_label (set
 to 'Live') is ignored when average != 'binary' (got 'macro'). You may use labels=[pos_label] to specify a single positi
ve class.
  % (pos_label, average), UserWarning)
0.9027777777777778
recall:
0.8055555555555556
accuracy:
0.851063829787234

(base) C:\Users\chakr\Desktop\Image A\Lab 5>
```