Image Analysis and Computer Vision Assignment-2

Name: Sai Mounica Chenuri Venkata V lakshmi Phalguna

WSU ID:Z963A577

Report:

Title: Fingerprint-based Spoof Detector based on LBP

Abstract: Fake fingers can be easily fabricated using commonly available materials, such as latex, silicone and gelatin, with the fingerprint ridges of an individual engraved on it. These fake fingers can then be used by an adversary to launch a spoof attack by placing them on a fingerprint sensor and claiming the identity of another individual. The success rate of such spoof attacks can be upto 70%. Fingerprint spoof detection algorithms have been proposed as a countermeasure against spoof attacks. A fingerprint spoof detector is a pattern classifier that is used to distinguish a live finger from a fake (spoof) one in the context of an automated fingerprint recognition system. Most liveness detectors are learning-based and rely on a set of training images.

Project Goal: To develop a two-class fingerprint spoof detector that uses Local Binary Patterns(LBP) and Histogram of Oriented Gradients (HOG) features along with Support Vector Machines(SVM) to distinguish live fingerprints images from spoof samples.

Steps Followed for preforming HOG Features:

- Import necessary pandas from sklearn

```
from sklearn import svm
from skimage.feature import local_binary_pattern
from sklearn.metrics import classification_report,accuracy_score
from skimage.feature import hog
import os
import cv2
import numpy as np
from sklearn.datasets import load_digits
```

- Read the data of Live and spoof training and test image sets

```
live_trainpath = "C:/Users/chvsa/Desktop/Image Analysis/Spoof_data/Training Biometrika Live/live/"
live_testpath = "C:/Users/chvsa/Desktop/Image Analysis/Spoof_data/Testing Biometrika Live/live/"
spoof_trainpath = "C:/Users/chvsa/Desktop/Image Analysis/Spoof_data/Training Biometrika Spoof/Training Biometrika Spoof/spoof/"
spoof_testpath = "C:/Users/chvsa/Desktop/Image Analysis/Spoof_data/Testing Biometrika Spoof/Testing Biometrika Spoof/spoof/"
```

- Performing HOG on the image set and returning features of live and spoof data from training and test set.

```
def fingerdata(live, spoof, descp):
#array which stores labels in lab=[]
   lab = []
#array wich stores features in fea[]
   fea = []

#storing the data of finger print images in live and spoof
   finger_data = [live, spoof]

   for images_path in finger_data:


      folder = os.listdir(images_path)



      for images in folder:

#reading the images from the file and converting them into gray scale images
         images = cv2.imread(images_path + images, cv2.IMREAD_GRAYSCALE)

#resizing the image to fit out requiremts
         scaled_image = cv2.resize(images, (64, 64))
#performing HOG features on the resized image
         fd, img_HOG = hog(scaled_image, orientations=9, pixels_per_cell=(8, 8),
                           cells_per_block=(2, 2), visualize=True, multichannel=False)

#live image label=1 spoof image label=0
#           label = 1
         if 'spoof' in images_path:
            label = 0
         else:
            label = 1

#fitting the features and label values
         fea.append(fd)



         lab.append(label)



   return fea,lab
```

- Training the SVM classifier using HOG features
- Testing the SVM with the same and predicting the values

```
#dividing the data into Xtrain and Ytrain

Xtrain,Ytrain = fingerdata(live_trainpath,spoof_trainpath, 'HOG')

#feading the trained data to SVM classifier

HOGSVM_clf= svm.SVC()

#fitting the Xtrain and Ytrain data

HOGSVM_clf.fit(Xtrain,Ytrain)


#testing the SVM classifier on Xtest and Ytest

Xtest,Ytest = fingerdata(live_testpath,spoof_testpath, 'HOG')

HOG_SVMpred = HOGSVM_clf.predict(Xtest)

#predicting the values

HOG_SVMpred
```

- Printing accuracy report for HOG
  print(" HOG Accuracy: "+str(accuracy_score(Ytest, HOG_SVMpred)))
- Printing precision, recall, f1-score, support
  print(classification_report(Ytest, HOG_SVMpred))

Output:

```
Out[2]: array([1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
               1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
               0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,
               1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1,
               1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1,
               0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
               1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0])
```

```
[3]: print(" HOG Accuracy: "+str(accuracy_score(Ytest, HOG_SVMpred)))
```

```
 HOG Accuracy: 0.8175
```

	precision	recall	f1-score	support
0	0.75	0.96	0.84	200
1	0.95	0.67	0.79	200
accuracy			0.82	400
macro avg	0.85	0.82	0.81	400
weighted avg	0.85	0.82	0.81	400

LBP features:

- Performing HOG on the image set and returning features of live and spoof data from training and test set.

```
def fingerdata(live, spoof, descp):
#array which stores labels in lab=[]
   lab = []
#array wich stores features in fea[]
   fea = []
#assigning radius for the circle for LBP

   lbp_radius = 3

#giving number of neighbours to the actual image
   k = 8 * lbp_radius

#sorting finger print images for live and spoof

   finger_data = [live, spoof]

   for images_path in finger_data:

      folder = os.listdir(images_path)

      for images in folder:
#reading the images from the file and converting them into gray scale images
         images = cv2.imread(images_path + images, cv2.IMREAD_GRAYSCALE)
#resizing the image to fit out requiremts
         scaled_image = cv2.resize(images, (64, 64))

#performing local binary pattern on the resized images

         lbp = local_binary_pattern(scaled_image, k, lbp_radius, 'uniform')
         fd=lbp.flatten()

 ##live image label=1 spoof image label=0
         if 'spoof' in images_path:
            label = 0
         else:
             label = 1
#fitting the features and label values
         fea.append(fd)
         lab.append(label)

   return fea,lab
```

- Training the SVM classifier using HOG features
- Testing the SVM with the same and predicting the values

```
#dividing the data into Xtrain and Ytrain
Xtrain,Ytrain= fingerdata(live_trainpath,spoof_trainpath,'LBP')

#feading the trained data to SVM classifier
LBPSVM_clf= svm.SVC()
#fitting the Xtrain and Ytrain data
LBPSVM_clf.fit(Xtrain,Ytrain)

#testing the SVM classifier on Xtest and Ytest
Xtest,Ytest= fingerdata(live_testpath,spoof_testpath,'LBP')
LBP_SVMpred = LBPSVM_clf.predict(Xtest)
#predicting the values
LBP_SVMpred
```

- Printing accuracy score for LBP

```
print("Accuracy: "+str(accuracy_score(Ytest, LBP_SVMpred)))
```

- Printing precision, recall, f1-score, support

```
print(classification_report(Ytest, LBP_SVMpred))
```

Output for LBP:

```
Out[5]: array([0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
               0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
               1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
               1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
               1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
               1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
               0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
               0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1,
               1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0])
```

```
[6]: print("Accuracy: "+str(accuracy_score(Ytest, LBP_SVMpred)))
```

```
Accuracy: 0.8575
```

	precision	recall	f1-score	support
0	0.79	0.96	0.87	200
1	0.96	0.75	0.84	200
accuracy			0.86	400
macro avg	0.87	0.86	0.86	400
weighted avg	0.87	0.86	0.86	400