Name: Jasmithasai Panuganti
WSU ID: A584V465

IMAGE ANALYSIS AND COMPUTER VISION
(CS898BA) ASSIGNMENT-2

1.Detection of GAN generated images using LBP features along with two-class support vector machine (SVM). Report Accuracy, F1-score and Recall of the classifier.
Dataset: https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces

Training dataset (150 live and fake images) should be used for SVM training. Trained SVM is tested on test dataset (150 live and spoof images).

Code:

```
:
import numpy as np
from sklearn import svm
from sklearn.metrics import accuracy_score, f1_score, recall_score, confusion_matrix
from skimage import io
from skimage.feature import local_binary_pattern
from skimage.color import rgb2gray
import os
import cv2

# Paths to data files
real_train_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/train/real'
fake_train_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/train/fake'
real_test_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/test/real'
fake_test_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/test/fake'


img_size = (64, 64)
points = 8 # number of circularly symmetric neighbour set points
radius = 1 # radius of circle
#empty train lists
X_train = []
y_train = []
#computing lbp features
def lbp(img):
    img_lbp = local_binary_pattern(img, points, radius)
    hist,_ = np.histogram(img_lbp.ravel(), bins=np.arange(0, points+3), range=(0, points+2))
    return hist
```

```
#looping through real train images folder
for img_file in os.listdir(real_train_path)[:150]:#150 live images
    img = io.imread(os.path.join(real_train_path, img_file))
    img= rgb2gray(img)#converting to gray scale
    img = cv2.resize(img, img_size)#resizing to (64,64)
    img = lbp(img)

    X_train.append(img)
    y_train.append(1)#1 because image is real

#looping through fake train images folder
for img_file in os.listdir(fake_train_path)[:150]:
    img = io.imread(os.path.join(fake_train_path, img_file))
    img = rgb2gray(img)
    img = cv2.resize(img, img_size)
    img = lbp(img)

    X_train.append(img)
    y_train.append(0)#0 because image is fake

#empty test lists
X_test = []
y_test = []

# Loop through real test image folders
for img_file in os.listdir(real_test_path)[:150]:
    img = io.imread(os.path.join(real_test_path, img_file))
    img = rgb2gray(img)
    img = cv2.resize(img, img_size)
    img = lbp(img)#computing lbp features

    X_test.append(img)
    y_test.append(1)
```

```
# Loop through fake test image folders
for img_file in os.listdir(fake_test_path)[:150]:
    img = io.imread(os.path.join(fake_test_path, img_file))
    img = rgb2gray(img)
    img = cv2.resize(img, img_size)
    img = lbp(img)

    X_test.append(img)
    y_test.append(0)

#deploy  2- class SVM
svmachine = svm.SVC(kernel='linear')
svmachine.fit(X_train, y_train)
# Predictions on test set
y_pred = svmachine.predict(X_test)

# report evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print('LBP-Accuracy:', accuracy)
print('LBP_F1-score:', f1)
print('LBP-Recall:', recall)
print('LBP-Confusion matrix:', confusion)
```

Steps followed:

The code starts with importing the required libraries for the modelling followed bt giving paths to the folders that has the data image files in the local system. To extract LBP features we define points and radius. lbp(img) is the function written that implements extraction of LBP features from the image. Now loop through the images to preprocess the 150 live and fake images for the training and testing dataset,making them ready for the classification.Image undergoes resizing and LBP features extraction.It is assigned 1 for live images and 0 for fake images. Thats how we classify the output as two classes 0/1 for classification for the y_train and y_test i.e., the class variable.Now, the deployment.Fit the Support vector machine on training dataset and train the model.Predict the values by using the classifier on the testing dataset and report the accuracy,f1 scores and recall values.

```
LBP-Accuracy: 0.53
LBP_F1-score: 0.4946236559139785
LBP-Recall: 0.46
LBP-Confusion matrix: [[90 60]
 [81 69]]
```

2.Alternatively use HOG feature descriptors along with two-class support vector machine (SVM). Compare the accuracy, F1-score and recall of the classifier with those obtained in Exp#1.

```
real_train_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/train/real'
fake_train_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/train/fake'
real_test_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/test/real'
fake_test_path = 'C:/Users/panug/Downloads/ML/real_vs_fake/real-vs-fake/test/fake'
```

```
# HOG parameters
orientations = 9
pixels_per_cell = (8, 8)
cells_per_block = (3, 3)

# computing HOG features for an image
def hogg(img):
    img_gray = rgb2gray(img)
    img_resized = cv2.resize(img_gray, (128*4, 64*4))
    hoggimg = hog(img_resized, orientations=orientations, pixels_per_cell=pixels_per_cell,
              cells_per_block=cells_per_block, )

    return hoggimg
```

```
# empty train lists
X_train = []
y_train = []

# Loop through real training image folder
for img_file in os.listdir(real_train_path)[:150]:
    img = io.imread(os.path.join(real_train_path, img_file))

    img_hog = hogg(img)#computer hog features

    X_train.append(img_hog)
    y_train.append(1)

# Loop through fake training image folders
for img_file in os.listdir(fake_train_path)[:150]:
    img = io.imread(os.path.join(fake_train_path, img_file))

    img_hog = hogg(img)

    X_train.append(img_hog)
    y_train.append(0)
```

```
# empty test lists
X_test = []
y_test = []

# Loop through real test image folder
for img_file in os.listdir(real_test_path)[:150]:
    img = io.imread(os.path.join(real_test_path, img_file))

    img_hog = hogg(img)

    X_test.append(img_hog)
    y_test.append(1)

# Loop through fake test image folder
for img_file in os.listdir(fake_test_path)[:150]:
    img = io.imread(os.path.join(fake_test_path, img_file))

    img_hog = hogg(img)

    X_test.append(img_hog)
    y_test.append(0)
```

```
# Training SVM
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)

# Testing SVM on test data and predicting values
y_pred = clf.predict(X_test)

# reporting metrics
acchog = accuracy_score(y_test, y_pred)
f1hog = f1_score(y_test, y_pred)
recallhog = recall_score(y_test, y_pred)
conf_mathog = confusion_matrix(y_test, y_pred)

print("HOG- Accuracy: ",acchog)
print("HOG- F1-score: ",f1hog)
print("HOG- Recall: ",recallhog)
print(f"HOG- Confusion matrix:\n{conf_mathog}")
```

Steps performed:

Just like we did the process for exp1, we follow the same procedure for this experiment too. But we compute HOG features for image to compare between LBP and HOG.

Define orientation values, pixels per cell and cells per block and write a method to compute HOG features for an image.

Reporting evaluation metrics:

```
HOG- Accuracy:  0.7333333333333333
HOG- F1-score:  0.7241379310344827
HOG- Recall:  0.7
HOG- Confusion matrix:
[[115  35]
 [ 45 105]]
```

Comparison:

```
print("The comparison:")

print(f"Accuracy of LBP-SVM is: {accuracy} while the accuracy of HOG-SVM is:{acchog} ")
print(f"F1 Score of LBP-SVM is: {f1} while the F1 score of HOG-SVM is: {f1hog} ")
print(f"recall of LBP-SVM is: {recall} while the recall of HOG-SVM is: {recallhog} ")
```

```
The comparison:
Accuracy of LBP-SVM is: 0.53 while the accuracy of HOG-SVM is:0.7333333333333333
F1 Score of LBP-SVM is: 0.4946236559139785 while the F1 score of HOG-SVM is: 0.7241379310344827
recall of LBP-SVM is: 0.46 while the recall of HOG-SVM is: 0.7
```

The accuracy for the SVM with extracting LBP features from image is 0.53 and f1 score, recall values are 0.49 and 0.46 respectively.
On the other hand, the accuracy for HOG extracted features alongside with SVM is 73% and f1 score, recall values are 0.72 and 0.7 correspondingly.
Since, HOG has higher accuracy,f1score and recall values,it suggests that HOG is better at extracting fveatures for the classification.