Homework 1: Face Detection

109550134 梁詠晴

Part I. Implementation (6%):

Part 1 :

```
'''
use os.path.join to combine datapath of folder, then use os.listdir access to all img in the folder.
use cv2.imread to read each img, and use cv2.IMREAD_GRAYSCALE to convert it into grayscale.
then append img along with label into dataset[]
'''
dataset = list(tuple())
for file in os.listdir(os.path.join(dataPath,"face")):
    img = cv2.imread(os.path.join(dataPath,"face",file),cv2.IMREAD_GRAYSCALE)
    temp = ((img),1)
    dataset.append(temp)
for file in os.listdir(os.path.join(dataPath,"non-face")):
    img = cv2.imread(os.path.join(dataPath,"non-face",file),cv2.IMREAD_GRAYSCALE)
    temp = (img,0)
    dataset.append(temp)

# End your code (Part 1)
return dataset
```

Part 2 :

```
"""
set best error = infinity and best feature = none at the begining.
for each features, go through all training sample, decide each h base on its matching value with each sample.
calculate the error of feature by adding up all abs(h-label)*weight.
compare the error with current best error, if the performance is better, update new best error and best feature.
use WeakClassifier to get best clf with best feature.
"""
feature_count = 0
bestError = float('inf')
best_feature = None
#print(len(featureVals),len(features))
for f,feature in zip(featureVals,features):
  cur_error = 0
  for data_index in range(len(f)):
    if (f[data_index] < 0):
      h=1
    else:
      h=0
    cur_error += abs(h-labels[data_index])*weights[data_index]
  feature_count +=1
  if(cur_error<bestError):
    best_feature = feature
    bestError = cur_error

bestClf = WeakClassifier(feature=best_feature)
# End your code (Part 2)
return bestClf, bestError
```

Part 4 :

```
# Begin your code (Part 4)
# raise NotImplementedError("To be implemented")
"""
define red and green in BGR form.
use readline to read txt file in line, get img name and number of faces. read img 2 times, one in grayscale.
crop the img where face locates, resize it to a 19x19 square and classify the square.
if a face is detected, draw a green rectangle on the full colored img, else draw a red one.
convert the BGR img to RGB img and show.
"""
f = open(dataPath,'r')
red = (0,0,255)
green  = (0,255,0)

for a in range(2):
  correct = 0
  line = f.readline()
  str = line.split()
  img = cv2.imread(os.path.join('data','detect',str[0]),cv2.IMREAD_GRAYSCALE)
  color_img = cv2.imread(os.path.join('data','detect',str[0]))
  for i in range(int(str[1])):
    l = f.readline()
    s = l.split()
    pic = img[int(s[1]):int(s[1])+int(s[3]),int(s[0]):int(s[0])+int(s[2])]
    pic = cv2.resize(pic, (19, 19),interpolation=cv2.INTER_NEAREST)
    find = clf.classify(pic)
    if(find == 1):
      correct +=1
      cv2.rectangle(color_img,(int(s[0]),int(s[1])),(int(s[0])+int(s[3]),int(s[1])+int(s[2])),green,2)
    else:
      cv2.rectangle(color_img,(int(s[0]),int(s[1])),(int(s[0])+int(s[3]),int(s[1])+int(s[2])),red,2)

  print('find',correct,'faces in total')
  b,g,r = cv2.split(color_img)
  new_img = cv2.merge([r,g,b])
  fig, ax = plt.subplots(1, 1)
  ax.axis('off')
  ax.set_title(str[0])
  ax.imshow(new_img)
  plt.show()
f.close()
# End your code (Part 4)
```

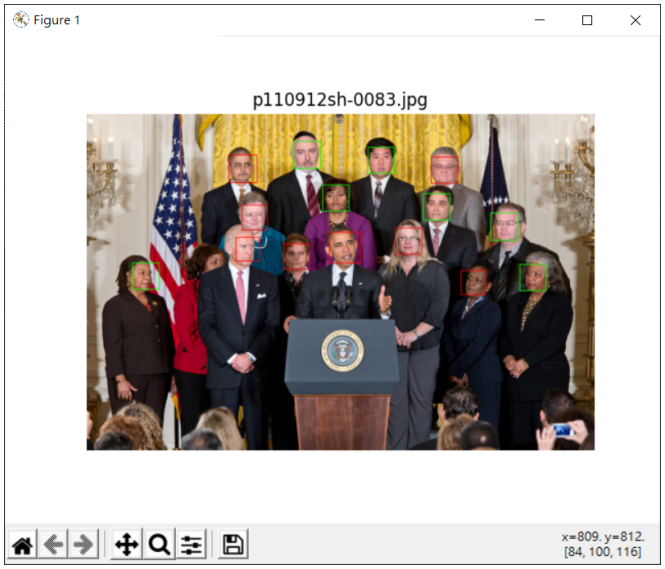
Part II. Results & Analysis (12%):

Result :

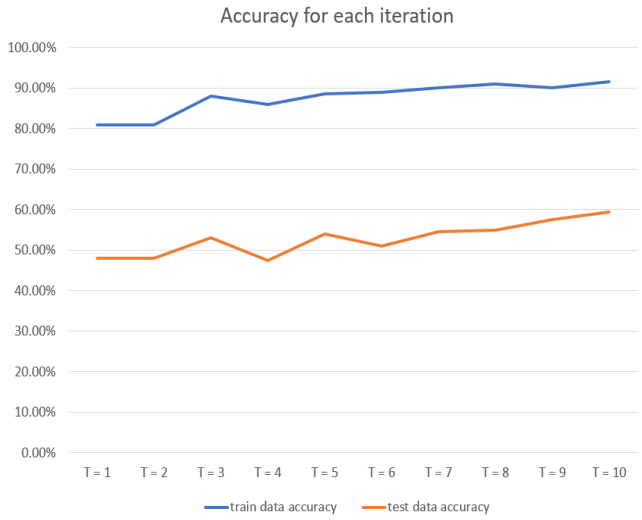
```
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[Rectan
RectangleRegion(2, 9, 2, 2), RectangleRegion(4, 11, 2, 2)]) with accuracy: 137.000000 and a
trained----------------------

Evaluate your classifier with training dataset
False Positive Rate: 17/100 (0.170000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 183/200 (0.915000)

Evaluate your classifier with test dataset
False Positive Rate: 45/100 (0.450000)
False Negative Rate: 36/100 (0.360000)
Accuracy: 119/200 (0.595000)
```



Analysis :



	train data accuracy	test data accuracy
T = 1	81.00%	48.00%
T = 2	81.00%	48.00%
T = 3	88.00%	53.00%
T = 4	86.00%	47.50%
T = 5	88.50%	54.00%
T = 6	89.00%	51.00%
T = 7	90.00%	54.50%
T = 8	91.00%	55.00%
T = 9	90.00%	57.50%
T = 10	91.50%	59.50%

When T becomes larger, both train data and test data's accuracy grows higher in trend. Test data is not as accurate as train data, it's because the best feature is chosen by its recognizing performance on train data, and for different sets of samples it may not be the most recognizable feature. Test data accuracy may rise if there's more iteration and more features chosen, which means it's more possible to get features that faces commonly have and can fit on both train data and test data.

Part III. Answer the questions (12%):

1. Please describe a problem you encountered and how you solved it.
One problem I met is that the images I showed were green instead of its original color. I found that's because I use opencv to read the image and use matplotlib to show it, and they have different formats to store the images (opencv: BGR, matplotlib: RGB). Thus I split the BGR image by color, then sort and merge to get a RGB image, so that the image became normal.

2. What are the limitations of the Viola-Jones' algorithm?
   1. Viola-Jones's algorithm is sensitive to brightness of the detected object, since it recognizes a face based on special features formed by different brightness on human faces.
   2. It may be harder for Viola-Jones' algorithm to recognize faces with darker skin tone, because the brightness features may not be obvious.
   3. Restricted to binary classification.

3. Based on Viola-Jones' algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?

1. Expand Haar-like features, for example, add 45°-rotatable rectangle features.
2. Pre-process the images (ex. Adjust brightness,contrast) before classifying to make the features more obvious.

4. Please propose another possible face detection method (no matter how good or bad, please come up with an idea). Please discuss the pros and cons of the idea you proposed, compared to the Adaboost algorithm.

Detect faces in 3D, get additional ‘depth’ information to help recognize faces. 3D face recognition may be more accurate than the Adaboost algorithm, because it has more information, by sensing the whole 3D face structure, it’s more possible to separate real faces with other objects with face-like features. However, training a 3D model is harder than training by Adaboost algorithm, it requires 3D data samples which cost much more than photos.