

Homework1 Report: Face Detection

109550164 徐聖哲

PartI.Implement

Part1: Load and prepare your dataset

```
1  """
2  # Begin your code (Part 1)
3  # raise NotImplementedError("To be implemented")
4  """
5  1.create a list to store the data of face and non-face
6  2.use file1 and file 2 to get direct path of face and non-face
7  3.use os.listdir to iterate file in the folder
8  4.use cv2.imread to read the image
9  5. if path is face, Load 1, else 0
10 6.use list.append to load data
11 """
12 dataset = []
13 file1 = dataPath+"/face";
14 file2 = dataPath+"/non-face";
15
16 for i in os.listdir(file1):
17     img = cv2.imread(file1+'/'+i, cv2.IMREAD_GRAYSCALE);
18     dataset.append((img,1))
19
20 for i in os.listdir(file2):
21     img = cv2.imread(file2+'/'+i, cv2.IMREAD_GRAYSCALE);
22     dataset.append((img,0))
23 # End your code (Part 1)
```

Part2: Implement Adaboost algorithm

```
1  # Begin your code (Part 2)
2      # raise NotImplementedError("To be implemented")
3      """
4      1.featurevals is the value of f(x), each row is for a haarfeature class
5      2.if f(x) < 0 , h(x) = 1, else h(x) = 0
6      3.use the error function in Lecture2 to produce weights
7      4.store error in the list - clf
8      5.use min find the minimal value of error, store in bestError
9      6.use list.index find the index of bestError
10     7.stor the specify features to bestClf
11     """
12     clf = []
13     for i in range(len(featureVals)):
14         error = 0
15         for j in range(len(featureVals[i])):
16             if(featureVals[i][j] < 0):
17                 error += weights[j] * abs(1-labels[j])
18             else:
19                 error += weights[j] * abs(0-labels[j])
20         clf.append(error);
21
22     bestError = min(clf)
23     min_index = clf.index(bestError)
24     print(bestError)
25     print(min_index)
26     bestClf = WeakClassifier(features[min_index])
27     # End your code (Part 2)
```

Part3:Additional experiments

```
1  # Part 3: Modify difference values at parameter T of the Adaboost algorithm.
2  # And find better results. Please test value 1~10 at least.
3  print('Start training your classifier')
4
5  # classifier already been trained
6  for i in range(1,10):
7      clf = adaboost.Adaboost(i)
8      clf.train(trainData)
9
10     clf.save('clf_200_1_10')
11
12     clf = adaboost.Adaboost.load('clf_200_1_10')
13
14     print('\nEvaluate your classifier with training dataset')
15     utils.evaluate(clf, trainData)
16
17     print('\nEvaluate your classifier with test dataset')
18     utils.evaluate(clf, testData)
```

Part4: Detect face

```
1  # Begin your code (Part 4)
2  # raise NotImplementedError("To be implemented")
3  """
4  1.read information in detectData.txt
5  2.store image name, face_number, face_position in path, size, file
6  3.then use for Loop to classiy whether it is a face
7  4.first store face image in face1,2
8  5.second use cv2.cvtColor() and cv2.resize() to chage image to grayscale and 19 x 19
9  6.third check if image is clt.classify() as face
10 7.if is a face, use cv2.rectangle() to draw red rectangle, else draw red
11 8.plot image while change BGR to RGB
12  """
```

```
1  image1 = cv2.imread('data/detect/'+path1)
2  image2 = cv2.imread('data/detect/'+path2)
3  for i in range(size1):
4      face1 = image1[file1[i*4+1]: file1[i*4+1]+file1[i*4+3], file1[i*4]: file1[i*4]+file1[i*4+2]]
5      plt.imshow(cv2.cvtColor(face1, cv2.COLOR_BGR2RGB))
6      plt.show()
7      face1 = cv2.cvtColor(face1, cv2.COLOR_BGR2GRAY)
8      face1 = cv2.resize(face1, (19, 19), interpolation=cv2.INTER_NEAREST)
9      is_face = clf.classify(face1)
10     if(is_face):
11         green_color = (0, 255, 0) #BGR
12         cv2.rectangle(image1, (file1[i*4], file1[i*4+1]), (file1[i*4]+file1[i*4+2], file1[i*4+1]+file1[i*4+3]), green_color, 3, cv2.LINE_AA)
13     else:
14         red_color = (0, 0, 255) #BGR
15         cv2.rectangle(image1, (file1[i*4], file1[i*4+1]), (file1[i*4]+file1[i*4+2], file1[i*4+1]+file1[i*4+3]), red_color, 3, cv2.LINE_AA)
16
17  for i in range(size2):
18      face2 = image2[file2[i*4+1]: file2[i*4+1]+file2[i*4+3], file2[i*4]: file2[i*4]+file2[i*4+2]]
19      plt.imshow(cv2.cvtColor(face2, cv2.COLOR_BGR2RGB))
20      plt.show()
21      face2 = cv2.cvtColor(face2, cv2.COLOR_BGR2GRAY)
22      face2 = cv2.resize(face2, (19, 19), interpolation=cv2.INTER_NEAREST)
23      is_face = clf.classify(face2)
24      if(is_face):
25          green_color = (0, 255, 0) #RGB
26          cv2.rectangle(image2, (file2[i*4], file2[i*4+1]), (file2[i*4]+file2[i*4+2], file2[i*4+1]+file2[i*4+3]), green_color, 3, cv2.LINE_AA)
27      else:
28          red_color = (0, 0, 255) #RGB
29          cv2.rectangle(image2, (file2[i*4], file2[i*4+1]), (file2[i*4]+file2[i*4+2], file2[i*4+1]+file2[i*4+3]), red_color, 3, cv2.LINE_AA)
30
31  plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
32  plt.show()
33  plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
34  plt.show()
35  # End your code (Part 4)
```

```

1 file1 = []
2 file2 = []
3
4 with open(dataPath) as f:
5     line = f.readline()
6     line = line.strip()
7     s = line.split(' ')
8     path1 = (s[0])
9     size1 = int(s[1])
10    for i in range(size1):
11        line = f.readline()
12        line = line.strip()
13        s = line.split(' ')
14        for j in range(len(s)):
15            file1.append(int(s[j]))
16
17    line = f.readline()
18    line = line.strip()
19    s = line.split(' ')
20    path2 = (s[0])
21    size2 = int(s[1])
22    for i in range(size2):
23        line = f.readline()
24        line = line.strip()
25        s = line.split(' ')
26        for j in range(len(s)):
27            file2.append(int(s[j]))


```

Part5: Test classifier on your own images

```

1 Part 5: Test classifier on your own images
2 1.I use warrios.jpg to detect
3 2.the face position is stored in image.jpg
4 """
5
6 file3 = []
7
8 with open(dataPath) as f:
9     line = f.readline()
10    line = line.strip()
11    s = line.split(' ')
12    path3 = (s[0])
13    size3 = int(s[1])
14    for i in range(size3):
15        line = f.readline()
16        line = line.strip()
17        s = line.split(' ')
18        for j in range(len(s)):
19            file3.append(int(s[j]))


```



```

1 image3 = cv2.imread('data/detect/'+path3)
2 for i in range(size3):
3     face3 = image3[file3[i*4+1]: file3[i*4+1]+file3[i*4+3], file3[i*4]: file3[i*4]+file3[i*4+2]]
4     plt.imshow(cv2.cvtColor(face3, cv2.COLOR_BGR2RGB))
5     plt.show()
6     face3 = cv2.cvtColor(face3, cv2.COLOR_BGR2GRAY)
7     face3 = cv2.resize(face3, (19, 19), interpolation=cv2.INTER_NEAREST)
8     is_face = clf.classify(face3)
9     if(is_face):
10         green_color = (0, 255, 0) #RGB
11         cv2.rectangle(image3, (file3[i*4], file3[i*4+1]), (file3[i*4]+file3[i*4+2], file3[i*4+1]+file3[i*4+3]), green_color, 3, cv2.LINE_AA)
12     else:
13         red_color = (0, 0, 255) #RGB
14         cv2.rectangle(image3, (file3[i*4], file3[i*4+1]), (file3[i*4]+file3[i*4+2], file3[i*4+1]+file3[i*4+3]), red_color, 3, cv2.LINE_AA)
15
16 plt.imshow(cv2.cvtColor(image3, cv2.COLOR_BGR2RGB))
17 plt.show()
18 # End your code (Part 5)

```



```

1 # Part 5: Test classifier on your own images
2 print('\nDetect faces on your own images')
3 detection_own.detect('data/detect/image.txt', clf)

```

PartII

```
In [37]: Runfile('C:/Users/danzel/徐望哲/课程/大一/
AI_HW1')
Reloaded modules: dataset, feature, classifier, util
Loading images
The number of training samples loaded: 200
The number of test samples loaded: 200
Show the first and last images of training dataset
Computing integral images
Building features
Applying features to dataset
Selecting best features
Selected 5171 potential features
Initialize weights
```

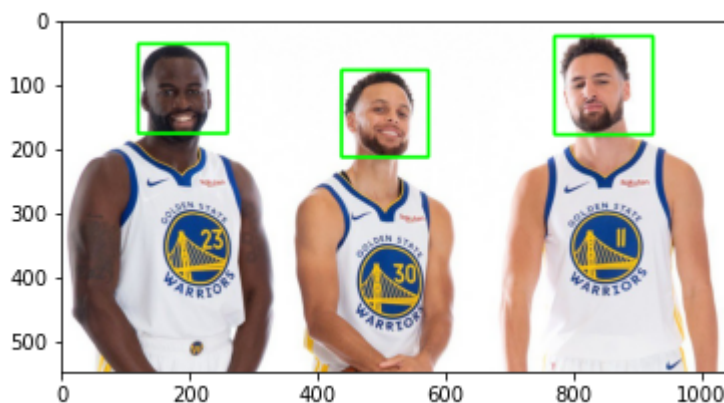
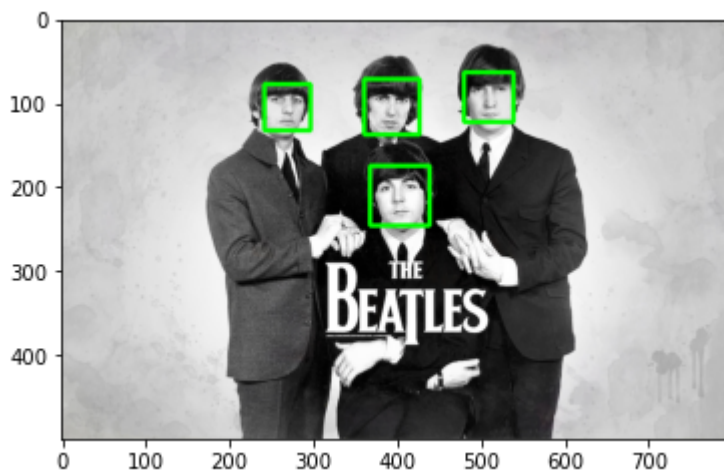
```
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9, 2, 2),
RectangleRegion(2, 11, 2, 2)], negative regions=[RectangleRegion(2, 9, 2, 2), RectangleRegion(4, 11, 2, 2)]) with accuracy:
137.000000 and alpha: 0.811201
```

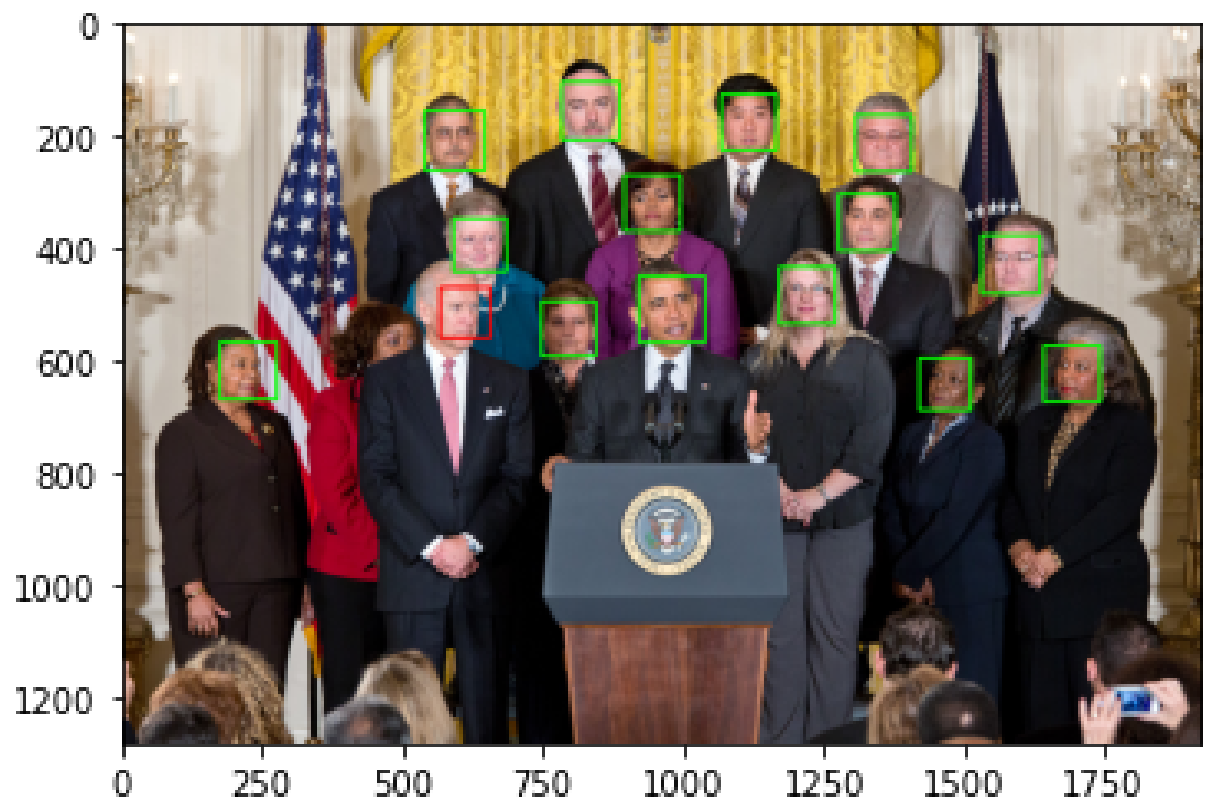
```
Evaluate your classifier with training dataset
False Positive Rate: 17/100 (0.170000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 183/200 (0.915000)
```

```
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)
```

```
Detect faces at the assigned location using your classifier
```

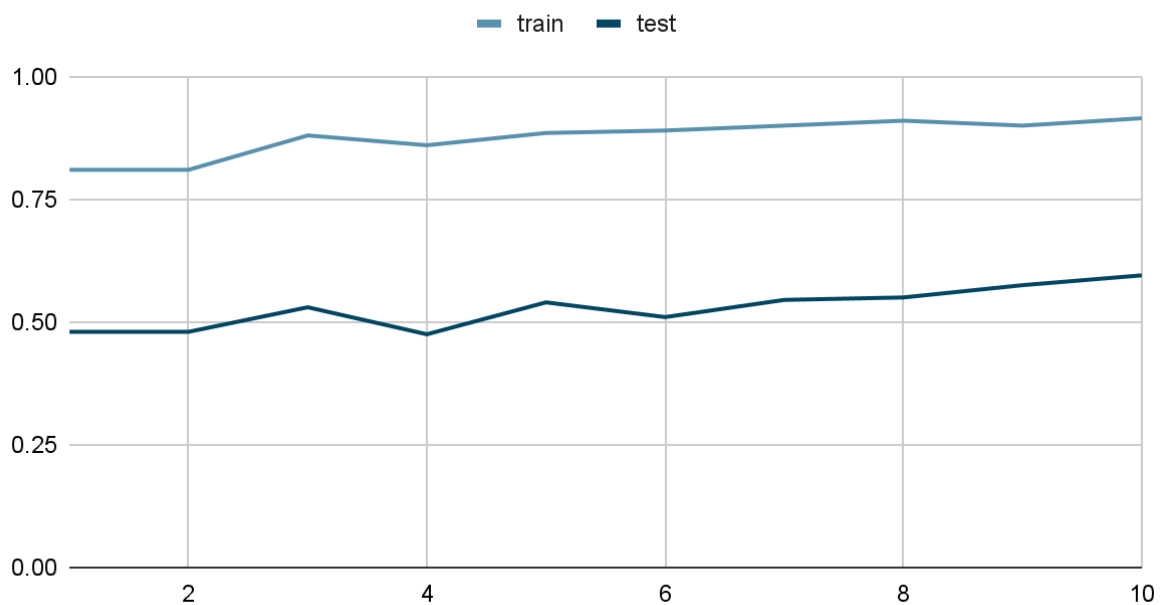
```
Detect faces on your own images
```





200張	train data accuracy	test data accuracy
method 1 t = 1	0.810000	0.480000
method 1 t = 2	0.810000	0.480000
method 1 t = 3	0.880000	0.530000
method 1 t = 4	0.860000	0.475000
method 1 t = 5	0.885000	0.540000
method 1 t = 6	0.890000	0.510000
method 1 t = 7	0.900000	0.545000
method 1 t = 8	0.910000	0.550000
method 1 t = 9	0.900000	0.575000
method 1 t = 10	0.915000	0.595000

Accuracy for each iteration



My observation:

From the sheet and the line chart above, the train data accuracy and test data accuracy both increase steadily, which reach to 91.5% and 59.5% separately.

Moreover, the two Obama's pictures show that when $t = 1$, the algorithm views 14 pictures as faces, while $t = 10$ views 7 pictures as faces. So, I guess as the classifier trains more times and reads more data, it will become more accurate and precise, thus it will finally recognize less pictures as faces.

PartIII

1. In HW1, I do encounter some problem. First, I was not familiar with python and anaconda, thus it took me some time to search for function such as `cv2.cvtColor` or `os.listdir`, I even need to figure out how to iterate numpy array. Secondly, the parameter sometimes confused me. For example, in part 2, `features` and `featuresval` was a little hard to understand the meaning, I had to ask for help. However, I finished part 4 pretty fast.

2. First, different haar-like features may produce different information, which will affect positive or negative values. Second, if faces are not in frontal view, the algorithm will be less effective. Third, too high or too low brightness will affect the detection

3. First, produce more haar-like features, which can be more useful in different situations. Second, make the angle of face into consideration, because people mostly found other in far away, while people's face are mostly not in front of them. Third, shadow is an important part of face detection for people, so I think algorithm may add this part.

4. I have a raw idea which is to collect different angles of a face and use the information to find out whether it is a face or not. This way, brightness is a significant parameter. I think brightness can combine with the portion of eyes, nose and mouth. Based on my face detection method, I think it will be more precise than the adaboost algorithm, but the angle I choose will be very important.