Homework 1: Face Detection Report

110550080 何曉嫻

**Part I. Implementation (6%):**

Part 1

1. Find the path of two folders (face, non-face)
2. Read all files in the folder by os.
3. Read all images by cv2 in grayscale.
4. Append the label and image path into dataset.
5. Return dataset.

```
# Begin your code (Part 1)

dataset = []
face=os.path.join(dataPath,'face')
non=os.path.join(dataPath,'non-face')
for filename in os.listdir(face):
    img = cv2.imread(os.path.join(face,filename),cv2.IMREAD_GRAYSCALE)
    if img is not None:
        dataset.append((img,1))

for filename in os.listdir(non):
    img = cv2.imread(os.path.join(non,filename),cv2.IMREAD_GRAYSCALE)
    if img is not None:
        dataset.append((img,0))


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

Part 2

1. Get the size of features and dataset.
2. Create an array with 0's, use to record $\varepsilon$ for each feature.
3. Evaluate $\varepsilon_j$ for each feature.

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right| \quad \text{and} \quad h_j(x_i) = \begin{cases} 1 & f_j(x_i) < 0 \\ 0 & else \end{cases}.$$

4. Choose the classifier with lowest $\varepsilon$.
5. Return $\varepsilon$ and classifier.

```
# Begin your code (Part 2)
features_num=featureVals.shape[0]
dataset_num=featureVals.shape[1]
err=np.zeros(features_num)

for j in range(features_num):
    for i in range(dataset_num):
        err[j]+= weights[i]*abs((1 if featureVals[j][i] < 0 else 0)-labels[i])

bestError = err[0]
bestClf = WeakClassifier(features[0])

for i in range(features_num):
    if err[i] < bestError:
        bestClf = WeakClassifier(features[i])
        bestError = err[i]

# End your code (Part 2)
return bestClf, bestError
```

Part 4

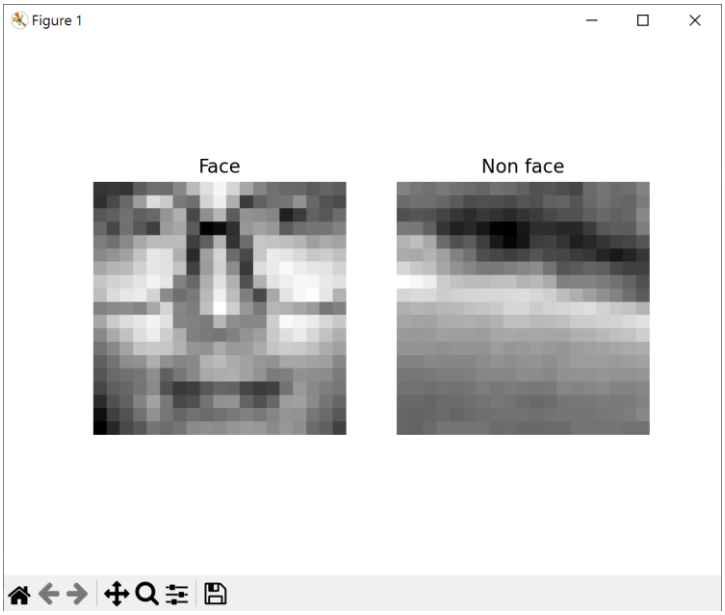
1. Read the file and split each lines into a list.
2. Run a loop to get info of each two pictures.
   Use while loop run each lines.
   For the first line, use split( ) to separate the image name and the number of people. Then run step.3
   Run for loop and use split( ) to separate the info and get the parameters of the square (x, xx, y, yy ). Then run step.4,5,6.
3. Read the image with grayscale and original by cv2.
4. Use the grayscale image and parameters to convert the face images to 19 x 19 grayscale images.
5. Use clf.classify( ) to detect the faces (if it is a face return 1, else return 0).
6. Draw the red and green square on original image.
7. Output the image.

```
# Begin your code (Part 4)
with open(dataPath,'r') as f:
  content = f.read().splitlines()
line=0
while line<len(content):
  fline=content[line].split( )
  name=fline[0]
  people=fline[1]
  line+=1
  img=cv2.imread("data/detect/"+name)
  img_gray=cv2.imread("data/detect/"+name,0)
  for i in range (int(people)):
    loc=content[line].split( )
    x=int(loc[0])
    y=int(loc[1])
    xx=int(loc[2])
    yy=int(loc[3])
    line+=1
    face=cv2.resize(img_gray[y:y+yy,x:x+xx],(19,19),interpolation=cv2.INTER_LINEAR )
    if clf.classify(face)==1:
      cv2.rectangle(img,(x,y),(x+xx,y+yy),(0,255,0),2)
    else:
      cv2.rectangle(img,(x,y),(x+xx,y+yy),(0,0,255),2)

  cv2.imshow("result-"+name,img)
  cv2.waitKey(0)
# End your code (Part 4)
```

## Part II. Results & Analysis (12%):

Part 1




Part 2

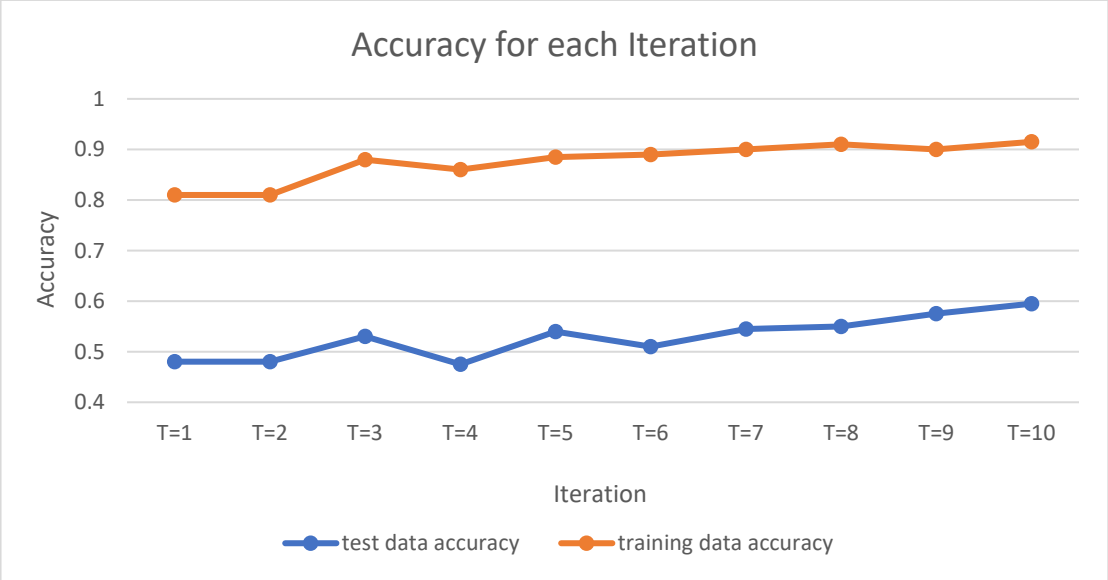
```
Selecting best features
Selected 5171 potential features
Initialize weights
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), Rectangl
eRegion(7, 3, 1, 3)], negative regions=[RectangleRegion(7, 0, 1, 3), RectangleRegion(8, 3, 1, 3)]) with accuracy: 162.0000
00 and alpha: 1.450010
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 8, 2, 9)], negativ
e regions=[RectangleRegion(2, 8, 2, 9)]) with accuracy: 156.000000 and alpha: 1.286922
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(16, 16, 1, 2)], negat
ive regions=[RectangleRegion(15, 16, 1, 2)]) with accuracy: 155.000000 and alpha: 1.011738
Run No. of Iteration: 4
ve regions=[RectangleRegion(4, 16, 8, 2)]) with accuracy: 153.000000 and alpha: 0.908680
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 8, 1, 1)], negati
ve regions=[RectangleRegion(9, 8, 1, 1)]) with accuracy: 155.000000 and alpha: 0.924202
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 3, 3, 8)], negativ
e regions=[RectangleRegion(4, 3, 3, 8)]) with accuracy: 78.000000 and alpha: 0.769604
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(5, 2, 10, 2)], negati
ve regions=[RectangleRegion(5, 4, 10, 2)]) with accuracy: 145.000000 and alpha: 0.719869
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12, 11, 5, 1)], negat
ive regions=[RectangleRegion(12, 12, 5, 1)]) with accuracy: 72.000000 and alpha: 0.685227
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 4, 1, 1)], negati
ve regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and alpha: 0.707795
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9, 2, 2), Rectangl
eRegion(2, 11, 2, 2)], negative regions=[RectangleRegion(2, 9, 2, 2), RectangleRegion(4, 11, 2, 2)]) with accuracy: 137.00
0000 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)
```

Part 3

| | test data accuracy | training data accuracy |
|---|---|---|
| T=1 | 0.48 | 0.81 |
| T=2 | 0.48 | 0.81 |
| T=3 | 0.53 | 0.88 |
| T=4 | 0.475 | 0.86 |
| T=5 | 0.54 | 0.885 |
| T=6 | 0.51 | 0.89 |
| T=7 | 0.545 | 0.90 |
| T=8 | 0.55 | 0.91 |
| T=9 | 0.575 | 0.90 |
| T=10 | 0.595 | 0.915 |




As the number of train increase, the accuracy also increases. This result tells us that we can make the machine better by training more times.

Besides, the accuracy of training data is obviously better than test data because the classifier of the machine is trained by these data. However, the test data accuracy can show whether the machine is good, because it detects other images not just the training data.

Part 4


result-the-beatles.jpg
THE
BEATLES


result-p110912sh-0083.jpg

Part 5


result-my.jpg

**Part III. Answer the questions (12%):**

1. Please describe a problem you encountered and how you solved it.
   I check my part 2 code and think it is right, but it still can't run.
   → I didn't use grayscale in part 1.

2. What are the limitations of the Viola-Jones' algorithm?
   It only applicable for frontal face detection, it may fail if the face is facing towards or away from the screen or rotated on the screen plane."

3. Based on Viola-Jones' algorithm, how to improve the accuracy except changing the training dataset and parameter T?
   By expanding Haar-like rectangle feature, the integral image of rotated rectangle can be figured out rapidly to calculate feature value of face detection under different states.

4. Other than Viola-Jones' algorithm, 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.
   Using skin color information to detect faces in an image.
   Pros:
   The method can be used to detect faces in images where the faces are not well-defined, such as low-resolution images.

   Cons:
   The method may not work well with people of different skin tones. Also, it is sensitive to variations in lighting conditions.

   Compared to the Adaboost algorithm, skin color-based detection can be fast and computationally efficient. However, it may have lower accuracy and may require more careful tuning of parameters.