Homework 1: Face Detection

Homework Report

By 109550093 黃得誠

Part I. Implementation (6%):

Screenshot of Part 1:

```
    '''
    explaintation of part 1:
    First, set the datapath to dataPath/face.
    Second, use a for loop to put all the files under dataPath/face in to the list(dataset).
    For each file, use cv2.imread to get the image's information, which gives us a numpy array(img)
    Notice that I give imread an argument "cv2.IMREAD_GRAYSCALE", in order to get grayscale data.
    For files in dataPath/face, set the tuple's second element to 1.
    Reapeat the above steps with datapath set to dataPath/non-face.
    Finally, return dataset.
    '''
    dataset = []
    dataPath1 = dataPath+"/face"
    for filename in os.listdir(dataPath1):
        img = cv2.imread(os.path.join(dataPath1,filename),cv2.IMREAD_GRAYSCALE)
        if img is not None:
            tup = (img,1)
            dataset.append(tup)

    dataPath1 = dataPath+"/non-face"
    for filename in os.listdir(dataPath1):
        img = cv2.imread(os.path.join(dataPath1,filename),cv2.IMREAD_GRAYSCALE)
        if img is not None:
            tup = (img,0)
            dataset.append(tup)
    # End your code (Part 1)
    return dataset
```

Screenshot of Part 2:

```
    '''
    explaination of part2:
    First, initialize bestError with a large number, bestClf with WeakClassifier().
    Second, for each feature j in rows of featureVals, evaluate the error respect to weight[j].
    Third, find the feature j with the smallest Error(bestError).
    Finally, return the error and the best classifier bestClf.
    '''
```

Set temp to 1 if fetureVals[j,i] <0; otherwise, set to 0.

```
    bestError=10000000
    Error = 0
    bestClf = WeakClassifier(-1)
    for j in range(len(featureVals)):
        Error = 0
        for i in range(len(featureVals[j])):
            if featureVals[j,i]>=0:
                temp =0
            else:
                temp =1

            Error+=(weights[i]*abs(temp-labels[i]))
        if Error<bestError:
            bestError = Error
            bestClf = WeakClassifier(features[j])
    # raise NotImplementedError("To be implemented")
    # End your code (Part 2)
    #print("bestError:",bestError)
    return bestClf, bestError
```

Screenshots of Part 4:

```
'''
explanation of Part 4:
First, open the txt file which contains the filenames, number of faces, position/width/height of faces.
set the variable filename as filename, nFace as number of faces.
Second, since there will be nFace lines of information, use a for loop to run nFace times.
For each loop, get the information from the txt file, x,y,w,h.
Turn them into integers, so as to use them to slice lists.
Crop the file(with the filename) into 19x19 gray scale list.(crop_img1)
Third, check if clf.classify(crop_img1) is True.
If it's true, then draw a green grid on the formal image;otherwise, red grid.
Finally, show the picture with grids drawn.
'''
```

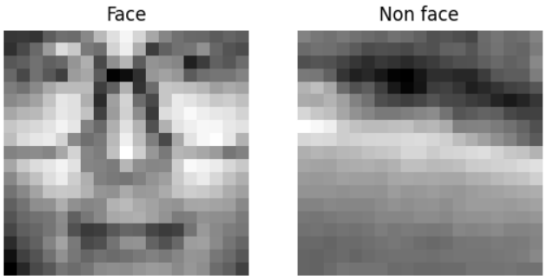
```
fd = open(dataPath, 'r')
while True:
    line = fd.readline()
    if len(line)==0:
        break
    filename, nFace = line.split()
    filename = 'data/detect/'+filename
    img = cv2.imread(filename,cv2.IMREAD_GRAYSCALE)
    img1 = cv2.imread(filename)

    dataset = []
    for times in range(int(nFace)):
        line = fd.readline()
        x,y,w,h = line.split()
        x=int(x)
        y=int(y)
        w=int(w)
        h=int(h)
        crop_img = img[y:y+h, x:x+w]
        crop_img1 = cv2.resize(crop_img,(19,19), interpolation=cv2.INTER_AREA)
        if clf.classify(crop_img1):
            for i in range(h):
                for j in range(w):
                    if i!=0 and i!=h-1:
                        if j!=0 and j!=w-1:
                            continue
                        else:
                            img1[y+i][x+j] = [0,255,0]
                    else:
                        img1[y+i][x+j] = [0,255,0]

        else:
            for i in range(h):
                for j in range(w):
                    if i!=0 and i!=h-1:
                        if j!=0 and j!=w-1:
                            continue
                        else:
                            img1[y+i][x+j] = [0,0,255]
                    else:
                        img1[y+i][x+j] = [0,0,255]

    img_rgb = img1[:,:,::-1]
    plt.imshow(img_rgb)
    plt.show()
```

P.S. After a little bit more research of cv2, I changed the draw grid part in to:
cv2.rectangle(img1, (x, y), (x+w, y+h), (0, 255, 0), 4, cv2.LINE_AA)

Part II. Results & Analysis (12%):

Part1 result:

```
PS D:\eric\university\AY21-2\IntroductiontoAI\AI_HW1>
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
```



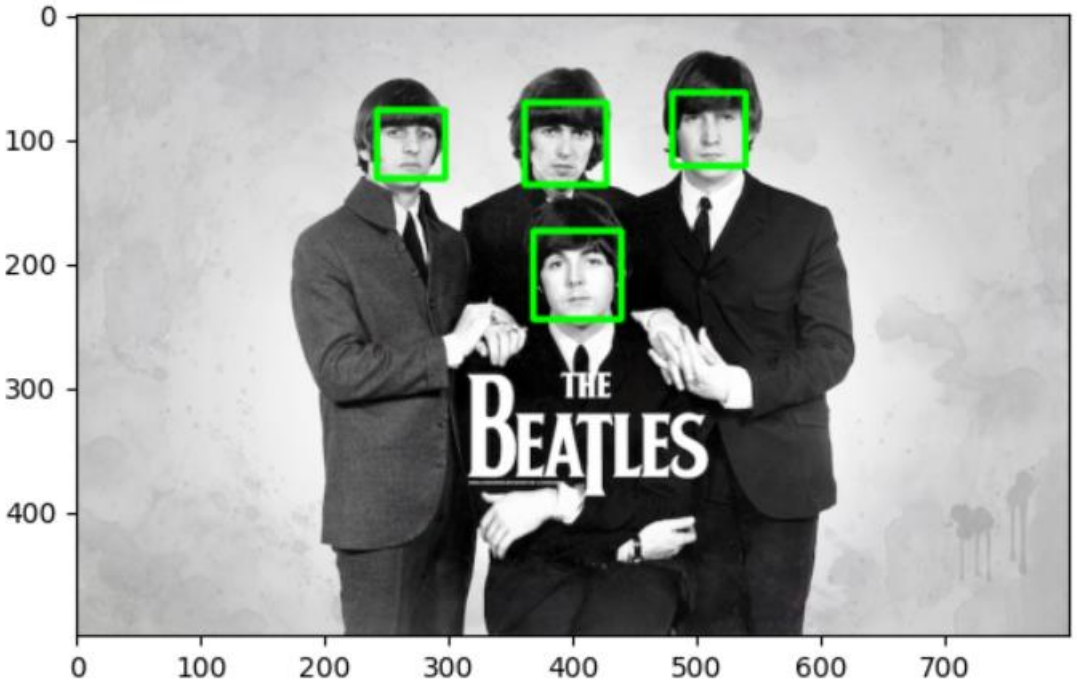
Part2 result(T=10):

```
Run No. of Iteration: 10
bestError: 0.4609309358527533
0.855049874883657
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(3, 2, 1, 1)
ions=[RectangleRegion(2, 2, 1, 1), RectangleRegion(3, 3, 1, 1)]) with accuracy: 105.000000 and alpha: 0.156595

Evaluate your classifier with training dataset
False Positive Rate: 26/100 (0.260000)
False Negative Rate: 19/100 (0.190000)
Accuracy: 155/200 (0.775000)

Evaluate your classifier with test dataset
False Positive Rate: 16/100 (0.160000)
False Negative Rate: 53/100 (0.530000)
Accuracy: 131/200 (0.655000)
```

Part3 results:

100 faces/100 not-faces	train data accuracy(%)	test data accuracy(%)	Correct Faces in image 1	Correct Faces in image 2	Correct Faces in image 3
method 1 T=1	71.5	62.5	2	2	10
method 1 T=2	71.5	62.5	2	2	10
method 1 T=3	79.5	66	4	4	12
method 1 T=4	72	63	3	2	11
method 1 T=5	72	63	3	2	11
method 1 T=6	72.5	63			
method 1 T=7	72.5	63			
method 1 T=8	78	66			
method 1 T=9	72	62.5			
method 1 T=10	77.5	65.5	4	7	21
method 1 T=50	73.5	62.5	4	5	20
method 1 T=100	73.5	62.5	4	5	20

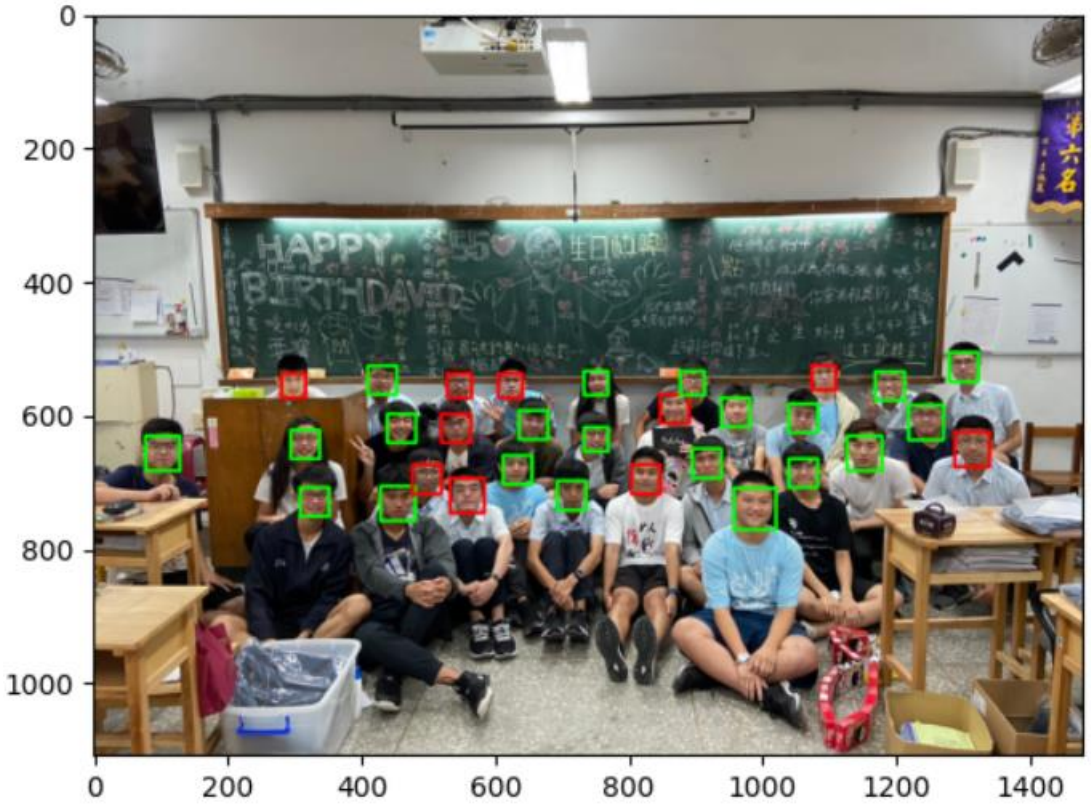
Part4 results(T=10):



Picture 1

Picture2

Part5 results(T=10): (Picture 3)



Analysis:
Chart that is same as part3.

100 faces/100 not-faces	train data accuracy(%)	test data accuracy(%)	Correct Faces in image 1	Correct Faces in image 2	Correct Faces in image 3
method 1 T=1	71.5	62.5	2	2	10
method 1 T=2	71.5	62.5	2	2	10
method 1 T=3	79.5	66	4	4	12
method 1 T=4	72	63	3	2	11
method 1 T=5	72	63	3	2	11
method 1 T=6	72.5	63	3	2	11
method 1 T=7	72.5	63	3	2	11
method 1 T=8	78	66	4	4	12
method 1 T=9	72	62.5	3	5	20
method 1 T=10	77.5	65.5	4	7	21
method 1 T=50	73.5	62.5	4	5	20
method 1 T=100	73.5	62.5	4	5	20

Please discuss the performance difference between the training and testing dataset, and present the results using a table or chart as follows.

1.The more iterations I run, does not mean the higher data accuracy.

2.The higher the training data accuracy, means the higher test data accuracy.

3. It seems that T=8 has the highest accuracy in train/test data, but in detect phase, T>8 still perform better than T=8.

4. It seems that the accuracy rate will tend to stability after T pass some number. (Can see that the different between T=50 and T=100 can't be visualize.)

Part III. Answer the questions (12%):

1. Please describe a problem you encountered and how you solved it.

   Problem: I didn't know how to get the face information of my own picture.
   Solution: use cv2 and dlib library to get face information. Then paste it to a txt file in order to test my classifier.

2. What are the limitations of the **Viola-Jones' algorithm**?

   In my opinion, **Viola-Jones' algorithm** can't do well if the face are not frontal faces.

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

   Apply a series of classifier, or adjust the threshold.

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.

   Start by searching for eyes, then other features on human faces.
   Pros: Maybe my algorithm can detect faces with masks put on better than Adaboost algorithm.
   Cons: Adaboost algorithm should be more stable than my algorithm, since my algorithm search for specific features.