Homework 1: Face Detection

Part I. Implementation (6%):

Part 1

Code

```
# Begin your code (Part 1)
# raise NotImplementedError("To be implemented")
dataset = []
data_path = dataPath + '/face'
for image_path in os.listdir(data_path):
    image = cv2.imread(data_path+'/'+image_path,cv2.IMREAD_GRAYSCALE)
    dataset.append((image,1))
data_path = dataPath + '/non-face'
for image_path in os.listdir(data_path):
    image = cv2.imread(data_path+'/'+image_path,cv2.IMREAD_GRAYSCALE)
    dataset.append((image,0))
# End your code (Part 1)
```

Explain

First, let `dataset` become a list which store data tuples.The first element in tuple is a numpy array that stores an image, and the second element in tuple is it label.

Second, iterate every image in `dataPath/face/`. The function use opencv to convert the image into a numpy array and put the numpy array and labels (labels are 0 here) into the list.

Third, do the same operations in second step, but change the labels into 1, and the path must be `dataPath/non-face/`.

Part 2

Code

```python
# Begin your code (Part 2)
# raise NotImplementedError("To be implemented")
## bestError = -np.inf ## my method
bestError = np.inf ##
bestClf_i = 0
for i in range(len(features)):
    ## epsilon = 1 ## my method
    epsilon = 0 ##
    for j in range(len(iis)):
        h = (featureVals[i][j] < 0)
        ## epsilon = epsilon * (1 - weights[j] * abs(h-labels[j])) ## my method
        epsilon = epsilon + weights[j] * abs(h-labels[j]) ##
    if bestError > epsilon: ##
    ## if bestError < epsilon: ## my method
        bestError = epsilon
        bestClf_i = i
bestClf = WeakClassifier(features[bestClf_i])
## bestError = 1 - bestError ## my method
# End your code (Part 2)
```

Explain

First, initialize the `bestError` and `bestClf_i` into $-\infty$ and 0.

Second, iterate every feature in `features`. For every feature, the function check its value in every integral image. If the value is negative then $h = -1$; Otherwise, $h = 1$.

Third, after apply the weights, the function calculate the sum of ϵ. If ϵ of smaller than `bestError`, then I record it.

After all the operations, the function returns the classifier with the smallest ϵ.

Part 4

Code

```python
# Begin your code (Part 4)
# raise NotImplementedError("To be implemented")
clf = adaboost.Adaboost.load('clf_200_1_10')
with open(dataPath,'r') as detect_file:
    while Line := detect_file.readline():
        [file_name,number_of_rec] = Line.split()
        number_of_rec = int(number_of_rec)
        image = cv2.imread(dataPath[:-14]+file_name,cv2.IMREAD_GRAYSCALE)
        print(dataPath[:-14])
        image_draw = cv2.imread(dataPath[:-14]+file_name,cv2.IMREAD_COLOR)
        for id in range(1,number_of_rec+1):
            Line = detect_file.readline()
            (x,y,x_len,y_len) = map(int,Line.split())
            # print(x,y,x+x_len,y+y_len)
            image_face = image[y:y+y_len,x:x+x_len]
            # cv2.imshow("image_face",image_face)
            if clf.classify(image_face) == True:
                color = (0,255,0)
            else:
                color = (0,0,255)
            cv2.rectangle(image_draw, (x, y), (x+x_len, y+y_len), color, 2)
        cv2.imwrite('new_'+file_name,image_draw)
# End your code (Part 4)
```

Explain

First, the program load the classifier which saved before.

Second, read the data in `detectData.txt`. It will tell the program images' name and the position of faces.

Third, generate numpy arrays which represent faces, then use the classifier the program load before to classify that numpy arrays.

If it returns true, then use the red rectangle to frame the area. Otherwise, use the green rectangle to frame the area.

Part II. Results & Analysis (12%):

Original Result

- T = 1

```
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), RectangleRegion(7, 3, 1, 3)], negative regions=[RectangleRegion(7, 0
, 1, 3), RectangleRegion(8, 3, 1, 3)]) with accuracy: 162.000000 and alpha: 1.450010

Evaluate your classifier with training dataset
False Positive Rate: 28/100 (0.280000)
False Negative Rate: 10/100 (0.100000)
Accuracy: 162/200 (0.810000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 55/100 (0.550000)
Accuracy: 96/200 (0.480000)
```

- T = 2

```
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 8, 2, 9)], negative regions=[RectangleRegion(2, 8, 2, 9)]) with accuracy: 156.
000000 and alpha: 1.286922

Evaluate your classifier with training dataset
False Positive Rate: 28/100 (0.280000)
False Negative Rate: 10/100 (0.100000)
Accuracy: 162/200 (0.810000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 55/100 (0.550000)
Accuracy: 96/200 (0.480000)
```

- T = 3

```
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(16, 16, 1, 2)], negative regions=[RectangleRegion(15, 16, 1, 2)]) with accuracy:
155.000000 and alpha: 1.011738

Evaluate your classifier with training dataset
False Positive Rate: 23/100 (0.230000)
False Negative Rate: 1/100 (0.010000)
Accuracy: 176/200 (0.880000)

Evaluate your classifier with test dataset
False Positive Rate: 48/100 (0.480000)
False Negative Rate: 46/100 (0.460000)
Accuracy: 106/200 (0.530000)
```

- T = 4

```
Run No. of Iteration: 4
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 14, 8, 2)], negative regions=[RectangleRegion(4, 16, 8, 2)]) with accuracy: 15
3.000000 and alpha: 0.908680

Evaluate your classifier with training dataset
False Positive Rate: 26/100 (0.260000)
False Negative Rate: 2/100 (0.020000)
Accuracy: 172/200 (0.860000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 56/100 (0.560000)
Accuracy: 95/200 (0.475000)
```

- T = 5

```
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 8, 1, 1)], negative regions=[RectangleRegion(9, 8, 1, 1)]) with accuracy: 155
.000000 and alpha: 0.924202

Evaluate your classifier with training dataset
False Positive Rate: 23/100 (0.230000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 177/200 (0.885000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 43/100 (0.430000)
Accuracy: 108/200 (0.540000)
```

- T = 6

```
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 3, 3, 8)], negative regions=[RectangleRegion(4, 3, 3, 8)]) with accuracy: 78.0
00000 and alpha: 0.769604

Evaluate your classifier with training dataset
False Positive Rate: 22/100 (0.220000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 178/200 (0.890000)

Evaluate your classifier with test dataset
False Positive Rate: 50/100 (0.500000)
False Negative Rate: 48/100 (0.480000)
Accuracy: 102/200 (0.510000)
```

- T = 7

```
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(5, 2, 10, 2)], negative regions=[RectangleRegion(5, 4, 10, 2)]) with accuracy: 14
5.000000 and alpha: 0.719869

Evaluate your classifier with training dataset
False Positive Rate: 20/100 (0.200000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 180/200 (0.900000)

Evaluate your classifier with test dataset
False Positive Rate: 52/100 (0.520000)
False Negative Rate: 39/100 (0.390000)
Accuracy: 109/200 (0.545000)
```

- T = 8

```
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12, 11, 5, 1)], negative regions=[RectangleRegion(12, 12, 5, 1)]) with accuracy:
72.000000 and alpha: 0.685227

Evaluate your classifier with training dataset
False Positive Rate: 18/100 (0.180000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 182/200 (0.910000)

Evaluate your classifier with test dataset
False Positive Rate: 47/100 (0.470000)
False Negative Rate: 43/100 (0.430000)
Accuracy: 110/200 (0.550000)
```

- T = 9

```
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 4, 1, 1)], negative regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152
.000000 and alpha: 0.707795

Evaluate your classifier with training dataset
False Positive Rate: 20/100 (0.200000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 180/200 (0.900000)

Evaluate your classifier with test dataset
False Positive Rate: 48/100 (0.480000)
False Negative Rate: 37/100 (0.370000)
Accuracy: 115/200 (0.575000)
```

- T = 10

```
Run No. of Iteration: 10
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)
```

- Original Detect

- Additional Detect


聯合頒獎典禮
第46屆國際大學生程式設計競賽亞洲區台北站
110年度全國大專
軟體設計競賽

Part 6 Result

- T = 1

```
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), RectangleRegion(7, 3, 1, 3)], negative regions=[RectangleRegion(7, 0
, 1, 3), RectangleRegion(8, 3, 1, 3)]) with accuracy: 162.000000 and alpha: 1.561476

Evaluate your classifier with training dataset
False Positive Rate: 28/100 (0.280000)
False Negative Rate: 10/100 (0.100000)
Accuracy: 162/200 (0.810000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 55/100 (0.550000)
Accuracy: 96/200 (0.480000)
```

- T = 2

```
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 8, 2, 9)], negative regions=[RectangleRegion(2, 8, 2, 9)]) with accuracy: 156.
000000 and alpha: 1.417495

Evaluate your classifier with training dataset
False Positive Rate: 28/100 (0.280000)
False Negative Rate: 10/100 (0.100000)
Accuracy: 162/200 (0.810000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 55/100 (0.550000)
Accuracy: 96/200 (0.480000)
```

- T = 3

```
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 1, 2, 14)], negative regions=[RectangleRegion(8, 1, 2, 14)]) with accuracy: 1
52.000000 and alpha: 1.167039

Evaluate your classifier with training dataset
False Positive Rate: 34/100 (0.340000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 166/200 (0.830000)

Evaluate your classifier with test dataset
False Positive Rate: 45/100 (0.450000)
False Negative Rate: 34/100 (0.340000)
Accuracy: 121/200 (0.605000)
```

- T = 4

```
Run No. of Iteration: 4
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 1, 9, 4)], negative regions=[RectangleRegion(7, 5, 9, 4)]) with accuracy: 158.
000000 and alpha: 1.136797

Evaluate your classifier with training dataset
False Positive Rate: 30/100 (0.300000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 170/200 (0.850000)

Evaluate your classifier with test dataset
False Positive Rate: 53/100 (0.530000)
False Negative Rate: 27/100 (0.270000)
Accuracy: 120/200 (0.600000)
```

- T = 5

```
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 3, 2, 10)], negative regions=[RectangleRegion(5, 3, 2, 10)]) with accuracy: 68
.000000 and alpha: 0.965220

Evaluate your classifier with training dataset
False Positive Rate: 24/100 (0.240000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 176/200 (0.880000)

Evaluate your classifier with test dataset
False Positive Rate: 49/100 (0.490000)
False Negative Rate: 25/100 (0.250000)
Accuracy: 126/200 (0.630000)
```

- T = 6

```
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(16, 17, 1, 1)], negative regions=[RectangleRegion(15, 17, 1, 1)]) with accuracy:
156.000000 and alpha: 1.131634

Evaluate your classifier with training dataset
False Positive Rate: 24/100 (0.240000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 176/200 (0.880000)

Evaluate your classifier with test dataset
False Positive Rate: 46/100 (0.460000)
False Negative Rate: 26/100 (0.260000)
Accuracy: 128/200 (0.640000)
```

- T = 7

```
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12, 11, 3, 1)], negative regions=[RectangleRegion(12, 12, 3, 1)]) with accuracy:
81.000000 and alpha: 1.147041

Evaluate your classifier with training dataset
False Positive Rate: 19/100 (0.190000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 181/200 (0.905000)

Evaluate your classifier with test dataset
False Positive Rate: 46/100 (0.460000)
False Negative Rate: 23/100 (0.230000)
Accuracy: 131/200 (0.655000)
```

- T = 8

```
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(5, 14, 8, 2)], negative regions=[RectangleRegion(5, 16, 8, 2)]) with accuracy: 15
5.000000 and alpha: 1.018697

Evaluate your classifier with training dataset
False Positive Rate: 21/100 (0.210000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 179/200 (0.895000)

Evaluate your classifier with test dataset
False Positive Rate: 48/100 (0.480000)
False Negative Rate: 30/100 (0.300000)
Accuracy: 122/200 (0.610000)
```

- T = 9

```
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 8, 1, 1)], negative regions=[RectangleRegion(9, 8, 1, 1)]) with accuracy: 155
.000000 and alpha: 0.983342

Evaluate your classifier with training dataset
False Positive Rate: 18/100 (0.180000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 182/200 (0.910000)

Evaluate your classifier with test dataset
False Positive Rate: 48/100 (0.480000)
False Negative Rate: 26/100 (0.260000)
Accuracy: 126/200 (0.630000)
```

- T = 10

```
Run No. of Iteration: 10
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: 0.975036

Evaluate your classifier with training dataset
False Positive Rate: 16/100 (0.160000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 184/200 (0.920000)

Evaluate your classifier with test dataset
False Positive Rate: 46/100 (0.460000)
False Negative Rate: 25/100 (0.250000)
Accuracy: 129/200 (0.645000)
```

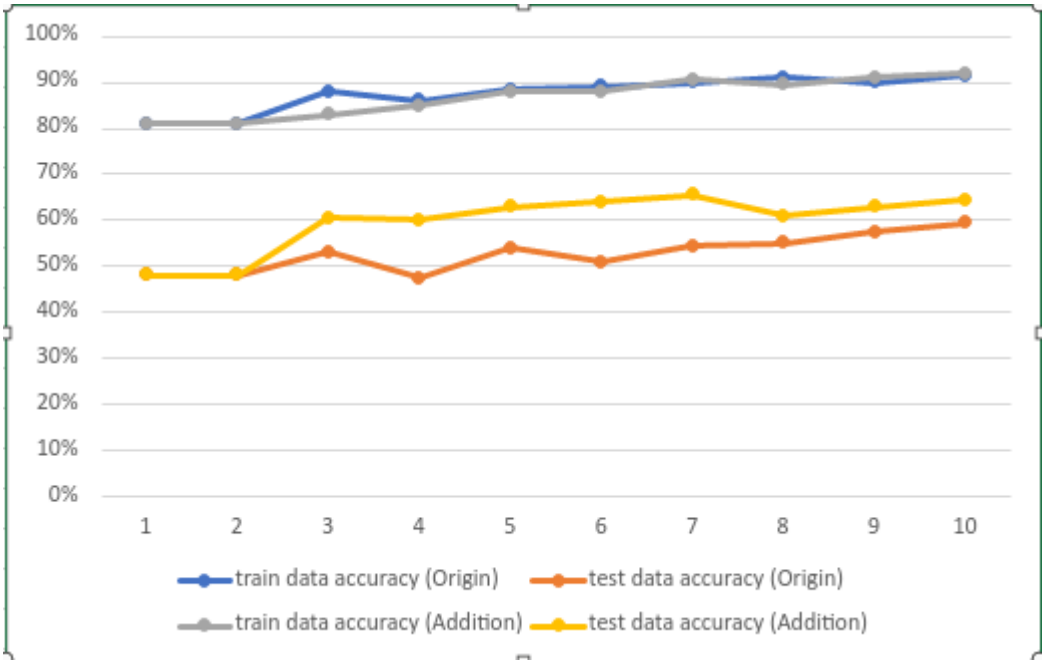
- Original Detect

- Additional Detect


聯合頒獎典禮
第46屆國際大學生程式設計競賽亞洲區台北站
110年度全國大專
軟體設計競賽

Analysis and Observation

T	train data accuracy (Origin)	test data accuracy (Origin)	train data accuracy (Addition)	test data accuracy (Addition)
1	81.0%	48.0%	81.0%	48.0%
2	81.0%	48.0%	81.0%	48.0%
3	88.0%	53.0%	83.0%	60.5%
4	86.0%	47.5%	85.0%	60.0%
5	88.5%	54.0%	88.0%	63.0%
6	89.0%	51.0%	88.0%	64.0%
7	90.0%	54.5%	90.5%	65.5%
8	91.0%	55.0%	89.5%	61.0%
9	90.0%	57.5%	91.0%	63.0%
10	91.5%	59.5%	92.0%	64.5%



▼ Original method

It is obvious that when T become large, there is an upward trend in performance.

On the other hand, It do great performance on training data, but it is only 59.5% on test data. This result imply that Adaboost has a long way to go. Adaboost doesn't conquer face detection problem.

▼ My method

I do a little modification on calculating ϵ.

It do a significant improve on original method in test data, and its performance on training data is close to original method.

Though the performance in test data improve, the performance in test data is still lower than 70%, which means this method cannot work correctly on any situation.

Part III. Answer the questions (12%):

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

   First, I am not familiar with Python at the beginning. I didn't know how to open the file, transform the image into a numpy array. To solve those problem, I did a lot of search on official documents. In the end, I know how to use those powerful tools in Python.

   Second, every time I get a bad performance it is difficult for me to distinguish the reason of it. I am not sure that the bad performance is the limitation of algorithm or I wrote the wrong algorithm. It is a great challenge for me, and my solution is to look inside every iteration's performance. If the performance change in a strange way such as decrease significantly or changeless at any time, then I know there is something wrong in my code.

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

   First, It is slow when the algorithm is searching features in images.

   Second, It can not reconginze colors every well, since it transform images into gray scale.

   Third, the feature is limited in haar-like features, which means that we cannot get some subtle feature which can recognize complex shapes.

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

   First, My method is one of the methods. You can try to change the way algorithm calculate ϵ.

   Second, try to consider RGB scale, instead of gray scale. Therefore, the algorithm can get more information from the image.

   Third, the algorithm can try to reflex and rotate the image it is going to train to get more data. Also, it can be used on classification. It can use the vote to decide whether the image is face or not. For example, if the image is the face then the classifier should say "yes" to half of images generated from the original one (rotate or reflex).

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

   - My method

     First, I need a large database which contains many shapes of face. This part should be generated by experts.

Second, when the algorithm do the classification, it compares every shape in the database. If there is a shape that is familiar to the image, then the algorithm will say “yes it is a face!”. Otherwise, it will say “no”.

Additional: we use the difference of color to recognize the boundary of shape.

- Pros
    1. I believe that it is possible to include most of shapes of face, therefore, it may get great performance if the database is large enough.
    2. It can solve the image which has color.
- Cons
    1. This method I propose above needs lots of expert generate the database, and this means that it may cost more than Adaboost.
    2. Since it need to search the whole database, the algorithm must slower than Adaboost.