Homework 1: Face Detection

109550135 范恩宇

Part I. Implementation :

↓ ↓ Code and explanation of “ Part 1 ” ↓ ↓

```python
    # Begin your code (Part 1)
    """
            First create an array for dataset and get paths of "face" and
        "non-face" , then read the images as grayscale . For "face" , label
        it as "1" while labeling "non-face" as "0" .
            Finally , add the images' numpy array and label into "dataset"
        and return "dataset"
    """
    dataset = []
    isFace = dataPath + '/face'
    notFace = dataPath + '/non-face'
    L1 = os.listdir(isFace)
    L2 = os.listdir(notFace)

    for files in L1:
      p_file = os.path.join(isFace,files)
      img = cv2.imread(p_file ,0)
      dataset.append([img,1])

    for files in L2:
      p_file = os.path.join(notFace,files)
      img = cv2.imread(p_file ,0)
      dataset.append([img,0])

    # raise NotImplementedError("To be implemented")
    # End your code (Part 1)
    return dataset
```

↓ ↓ Code and explanation of “ Part 2 ” ↓ ↓

```python
        # Begin your code (Part 2)
        """
                First , set curr_th(current threshold) as "0" and curr_pl(current polarity)
            as "1" , then initialize "bestClf" by "WeakClassifier" with parameters I just
            set and set "bestError" as infinity .
                Then read the array of HaarFeature class . During the process , first
            set currClf(current Clf) and currError(current error) , but initialize
            currError as 0 . Also in HaarFeature class reading process , set variable
            "tmp" and traverse the array of "label" , if currClf.polarity * featureVals
            " < " currClf.polarity * currClf.threshold , then make "tmp" as "1" (if not ,
            make it as "0") . Current Error keeps iterating with corresponding weights and
            absolute value of ( corresponding label - "tmp" ) , leading to the final error.
                Finally , if currError " < "  bestError , then replace bestError with the
            smaller one , so we can get a result with lower error rate .

        """
        curr_th = 0
        curr_pl = 1
        bestClf = WeakClassifier(features[0], threshold = curr_th, polarity = curr_pl)
        bestError = math.inf
```

```
        for i in range(len(features)):
            currClf = WeakClassifier(features[i], threshold = curr_th, polarity = curr_pl)
            currError = 0
            tmp = 0
            for j in range(len(labels)):
                sample_val = featureVals[i,j]
                if currClf.polarity * sample_val < currClf.polarity * currClf.threshold:
                    tmp = 1
                else:
                    tmp = 0

                currError += ( weights[j] * abs(labels[j] - tmp) )

            if currError < bestError :
                bestError = currError
                bestClf = currClf

        # raise NotImplementedError("To be implemented")
        # End your code (Part 2)
        return bestClf, bestError
```

↓ ↓ Code and explanation of “ Part 4 ” ↓ ↓

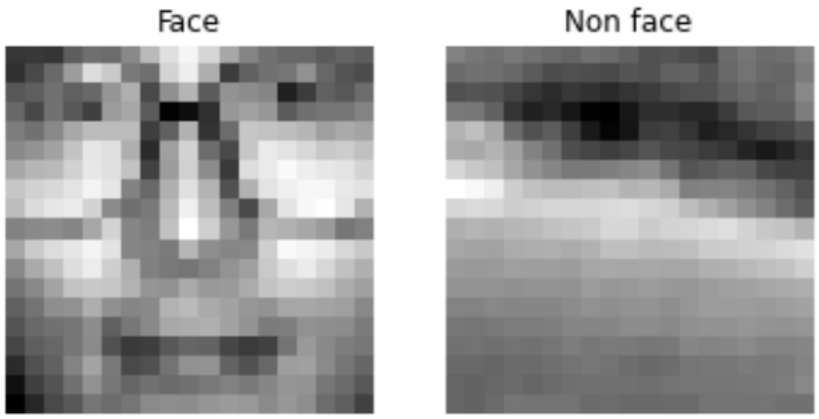
```
    # Begin your code (Part 4)
    """
            First , read the detectData.txt through given datapath in " main.py ".
        In "detectData.txt" , if a line's length equals to 2 , then collect its line[0] as file name
        of image and line[1] as number of faces , and set these two data in the "data" array. While a
        line's length equals to 4 , collect line[0]~[3] for locating the rectangles , then get a 19x19
        grayscale version of original image . After that, use " clf.classify() " to detect faces and
        get the result of "whether it's a face".
             If the result obtained above equals to 1 , then mark it with green lines , while using
        red lines for results that equals to 0 . After doing several turns of marking , the images marked
        with green/red rectangle in desired size will be shown .
    """

    data = []
```

```
    data = []

    for line in open(dataPath, "r"):
        line = line.split()
        if len(line) == 2 :
            file_path = line[0]
            img = cv2.imread(dataPath[0:12] + file_path)
            pic = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            data.append((img, int(line[1])))

        else :
            for i in range(4) :
                line[i] = int(line[i])

            tmp_pic = pic[line[1] : line[1] + line[3], line[0] : line[0] + line[2]]
            tmp_pic = cv2.resize(tmp_pic,(19, 19))
            tmp_res = clf.classify(tmp_pic)
            if tmp_res == 1:
                cv2.rectangle(img, (line[0], line[1]), (line[0] + line[2], line[1] + line[3]), (0, 255, 0), 2)
            else:
                cv2.rectangle(img, (line[0], line[1]), (line[0] + line[2], line[1] + line[3]), (0, 0, 255), 2)


    for i in data:
        ims = cv2.resize(i[0], (960, 638))
        cv2.imshow('My Image',ims)
        cv2.waitKey(0)

    # raise NotImplementedError("To be implemented")
    # End your code (Part 4)
```

Part II. Results & Analysis :

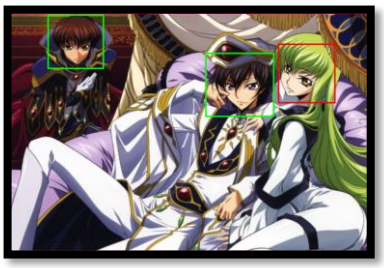
↓ ↓ First training results ↓ ↓

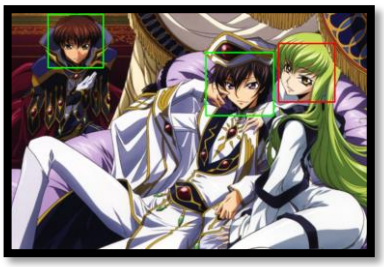
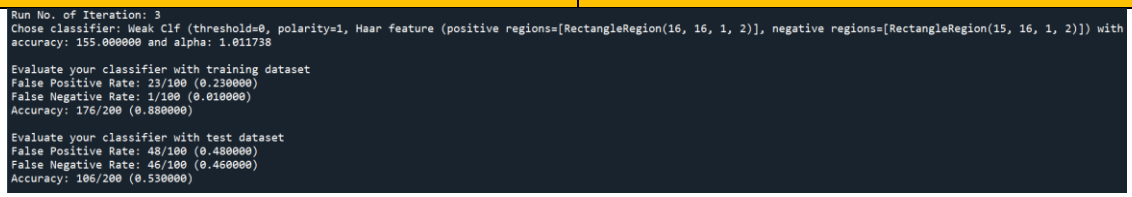
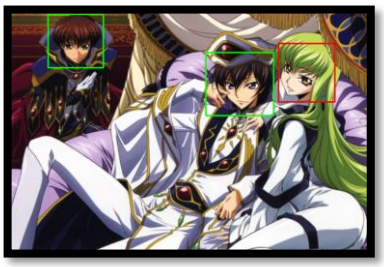
```
In [2]: runfile('D:/學業/人工智慧概論/作業/AI_HW1/AI_HW1/main.py', wdir='D:/學業/人工智慧概論/作業/AI_HW1/AI_HW1')
Reloaded modules: feature, classifier, utils, dataset, adaboost, detection
Loading images
The number of training samples loaded: 200
The number of test samples loaded: 200
Show the first and last images of training dataset
```



↓ ↓ Full training results ↓ ↓

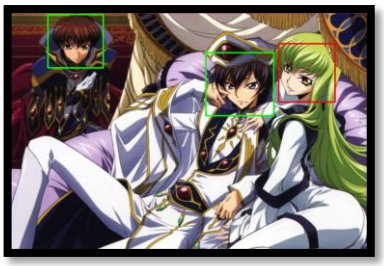
	Train data accuracy	Test data accuracy
Original	----------	----------
Test image		

Own image			
T = 1	81.0%		48.0%
Result	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, 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)		
Test image			
Own image			
T = 2	81.0%		48.0%
Result	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)		

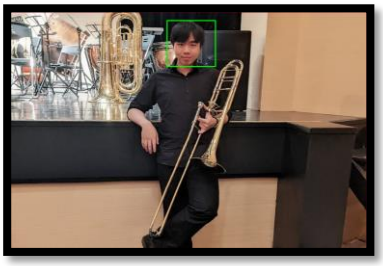
Test image			
Own image			
T = 3	88.0%	53.0%	
Result			
Test image			
Own image			
T = 4	86.0%	47.5%	

```
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)
```

Result	<pre>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: 153.000000 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)</pre>	

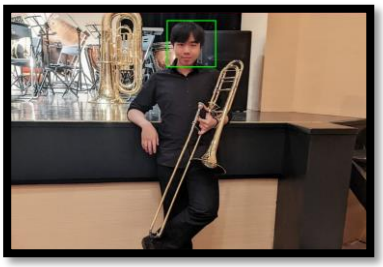
Test image		
Own image	 	
T = 5	88.5%	54.0%

Result	<pre>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)</pre>	

Test image		
Own image	 	

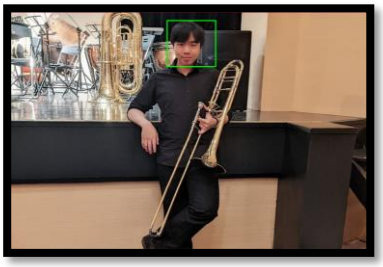
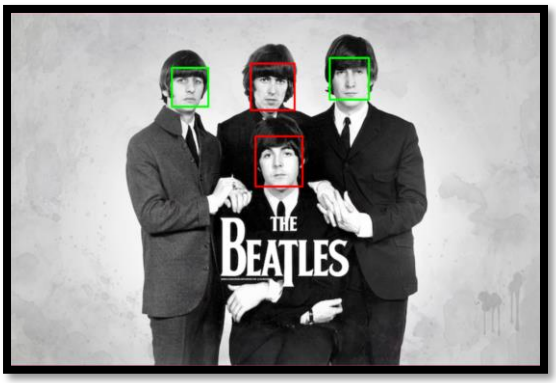
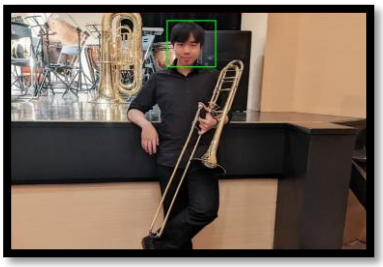
<table>
<tr><td>T = 6</td><td>89.0%</td><td>51.0%</td></tr>
<tr><td>Result</td><td colspan="2">

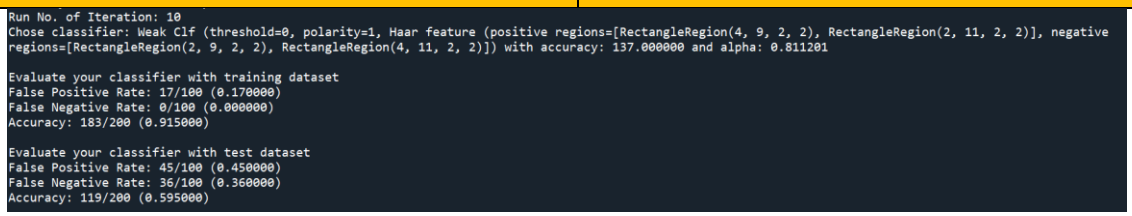
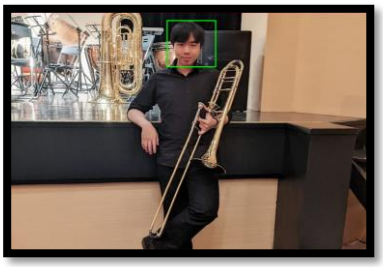
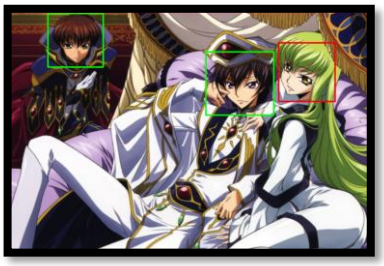
```
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(7, 3, 3, 8)], negative regions=[RectangleRegion(4, 3, 3, 8)]) with
accuracy: 78.000000 and alpha: 0.769604

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)
```

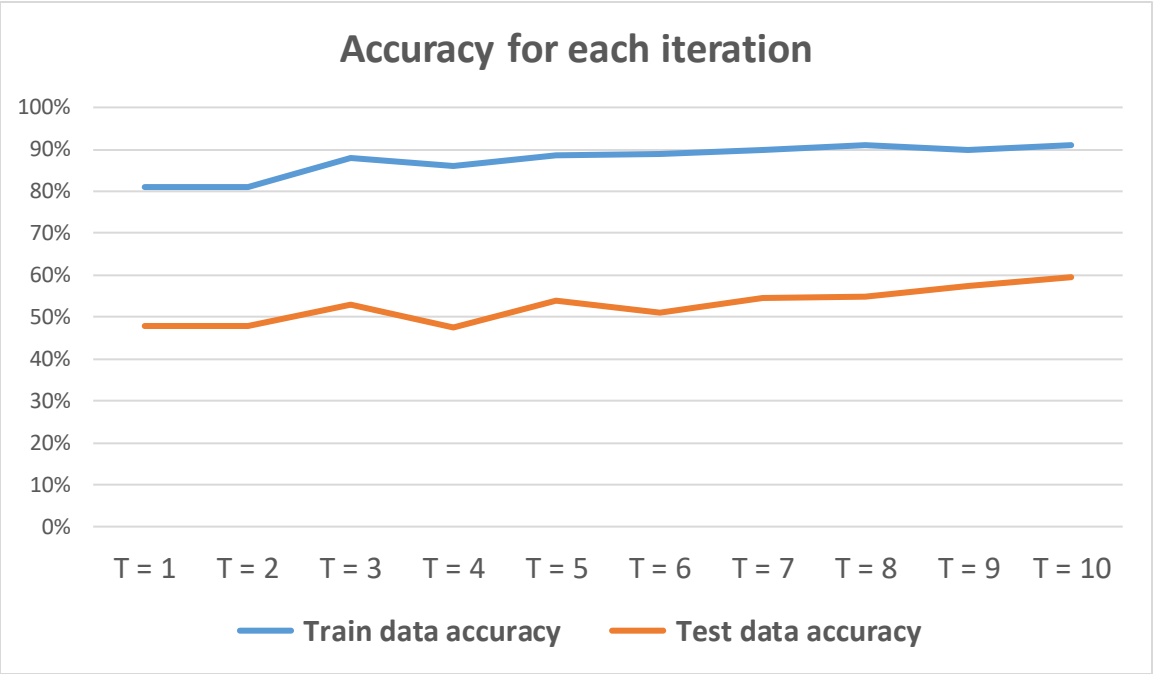
</td></tr>
<tr><td>Test image</td><td>  </td><td>  </td></tr>
<tr><td>Own image</td><td colspan="2">    </td></tr>
<tr><td>T = 7</td><td>90.0%</td><td>54.5%</td></tr>
<tr><td>Result</td><td colspan="2">

```
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: 145.000000 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)
```

</td></tr>
<tr><td>Test image</td><td>  </td><td>  </td></tr>
</table>

Own image			
T = 8	91.0%		55.0%
Result	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)		
Test image			
Own image			
T = 9	90.0%		57.5%
Result	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)		

Test image		
Own image	 	
T = 10	91.5%	59.5%
Result		
Test image		
Own image	 	



From the results received , we can find out that train data accuracy is always higher than test data accuracy , not only that , they have similar changes in the process . I think its because the former is the one that has been trained several times , making it perform higher accuracy . Not only that , I found out that those with smaller “T”s detect faces with higher accuracy , I believe that its because smaller “T”s result in smaller error .

Part III. Answer the questions :

1. The most memorable problem I encountered happened at “part 4” , I didn’t use another temporary variable to store the grayscale image at first , making me unable to load all images I put in . After debugging for a moment , I tried to use another variable as I said before , making the original image unmodified . Fortunately , this solution worked .

2. Viola-Jones’ algorithm has slow training time and is restricted to binary classification . It may also be sensitive to very high/low exposure . In addition , it’s mostly effective when face is in frontal view and with both high true detection rate and high false detection rate .

3. We can use “Attentional Cascade” , which is a series of weak classifiers we trained, making a strong classifier when used together . The cascade is sorted with classifiers from strongest to weakest , being able to eliminate negative samples earlier , which allows faster detection and decreases computation time on inference.

4. XGBoost algorithm . Comparing to AdaBoost algorithm , XGBoost is generally more accurate and faster, since it has sparsity penalties . However , AdaBoost is a lot more flexible with respect to distributions and baselearner selection .