HW1: Face Detection Report

110550093 蔡師睿

Part I. Implementation

Part 1

```
    # Begin your code (Part 1)
    """
    1. Initialize an empty list called dataset.
    2. Use os.listdir() and for loop to make sure all the image can be read.
    3. Use cv2.imread() to read the image, and classify it into 1 for face and 0 for non-face.
    4. Add the image and its classification to a tuple and put the tuple into dataset.
    5. Return the dataset.
    """
    dataset=[]
    num=1
    for i in os.listdir(dataPath):
        root=dataPath+'/'+str(i)+'/'
        for k in os.listdir(root):
          dataset.append((cv2.imread(root+k, cv2.IMREAD_GRAYSCALE), num))
        num-=1
    # End your code (Part 1)
    return dataset
```

Part 2

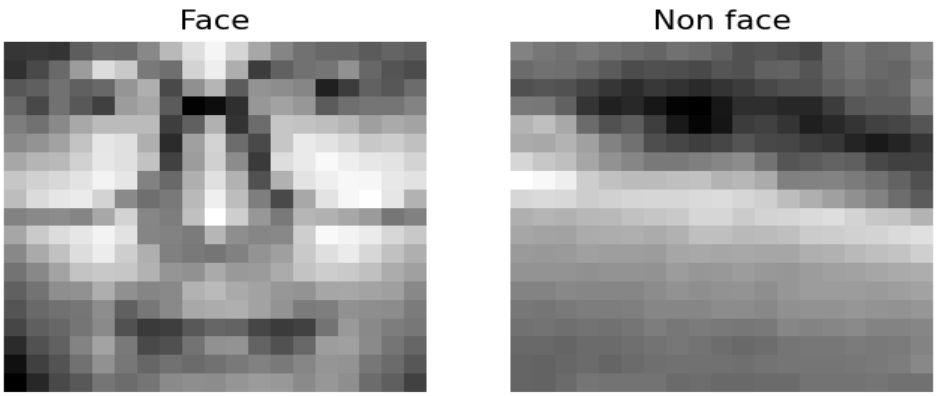
```
            # Begin your code (Part 2)
            """
            1. Initialize bestClf and bestError to none and infinity, respectively.
            2. For every column of featureVals, initialize eps=0.
            3. Check every raw whether featureVals and labels are correct
               (featureVals<=0, labels=1 or featureVals>0, label=0).
               If not, add its corresponding weights to eps.
            4. After scanning the whole row, if eps is less than bestError,
               make bestError equal to eps and bestClf become weakclassifier of the corresponding features.
            5. Return the bestClf and bestError.
            """
            col, row=featureVals.shape
            bestClf, bestError=None, float('inf')
            for i in range(col):
              eps=0
              for j in range(row):
                if (featureVals[i][j]<0 and labels[j]==0) or (featureVals[i][j]>=0 and labels[j]==1):
                    eps+=weights[j]
              if eps<bestError:
                  bestError=eps
                  bestClf=WeakClassifier(features[i])
            # End your code (Part 2)
```

Part 4

```python
# Begin your code (Part 4)
"""
1. Read the txt file, and open the image written in the txt.
2. Read the following lines and get the specific position of the image, which is a face.
3. Resize the face to 19x19.
4. Use clf.classify to detect faces.
   If the result is true, draw the green box on the image.
   Otherwise, draw the red box on the image.
"""
folderPath='data/detect/'
with open(dataPath, 'r') as file:
    text=file.readlines()
    while len(text):
        name, num=text[0].split()
        text.pop(0)
        img=cv2.imread(folderPath+name)
        gray_img=cv2.imread(folderPath+name, cv2.IMREAD_GRAYSCALE)
        for i in range(int(num)):
            x0, y0, x1, y1=map(int, text[0].split())
            x1+=x0
            y1+=y0
            text.pop(0)
            if clf.classify(cv2.resize(gray_img[y0:y1, x0:x1], (19, 19))):
                cv2.rectangle(img, (x0, y0), (x1, y1), (0, 255, 0), thickness=3)
            else:
                cv2.rectangle(img, (x0, y0), (x1, y1), (0, 0, 255), thickness=3)
        cv2.imshow('image', img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
# End your code (Part 4)
```

Part II. Results & Analysis

Part 1



The result after the function loadImages() in dataset.py is done.

Part 2

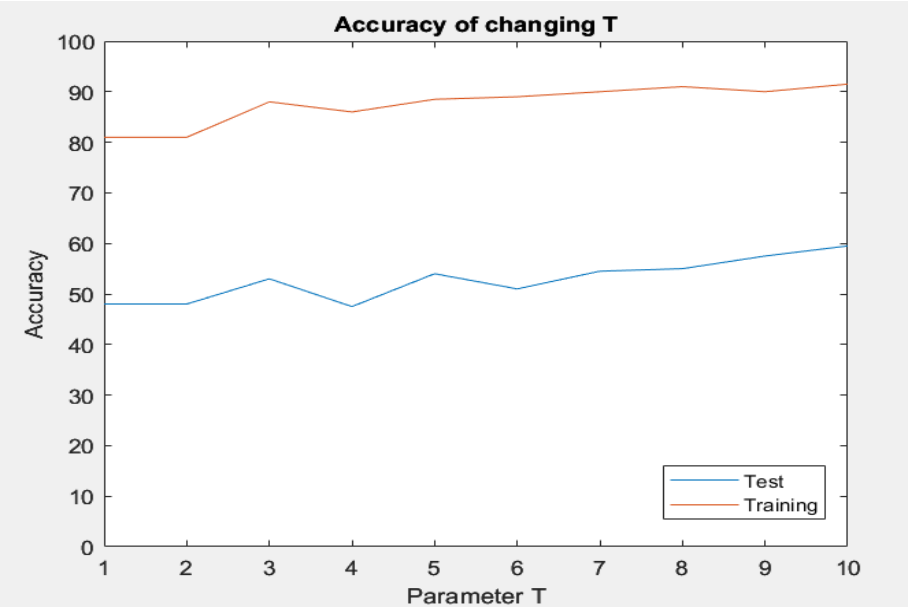
```
Evaluate your classifier with training dataset
False Positive Rate: 17/100 (0.170000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 183/200 (0.915000)

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)
```

The data after finishing selectBest() in adaboost.py

Part 3

Different accuracy by testing parameter T between 1 to 10.

200 images	**Training data accuracy**	**Test data accuracy**
$t = 1$	81.0%	48.0%
$t = 2$	81.0%	48.0%
$t = 3$	88.0%	53.0%
$t = 4$	86.0%	47.5%
$t = 5$	88.5%	54.0%
$t = 6$	89.0%	51.0%
$t = 7$	90.0%	54.5%
$t = 8$	91.0%	55.0%
$t = 9$	90.0%	57.5%
$t = 10$	91.5%	59.5%

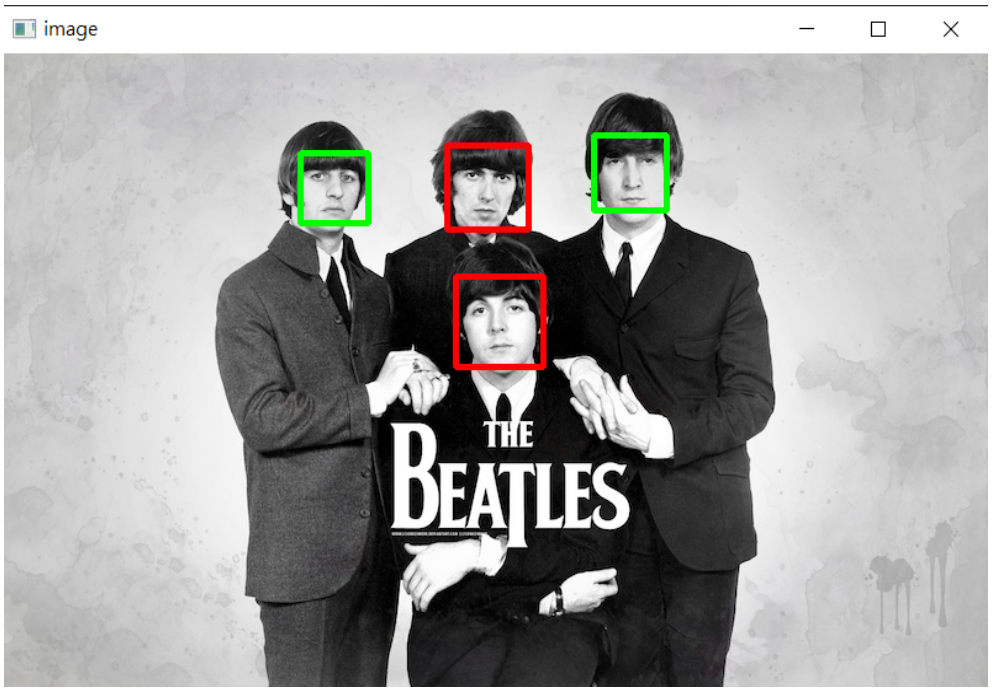


According to the results, we can reach the following conclusions :

1. The more time we train, the higher accuracy we can get.
2. The accuracy of training data is always higher than the test data, because the classifier is trained from training data.
3. Test data can tell whether the machine good or bad; hence, from the accuracy of test data we know the machine need to train more times or optimize its method for selecting classifier.

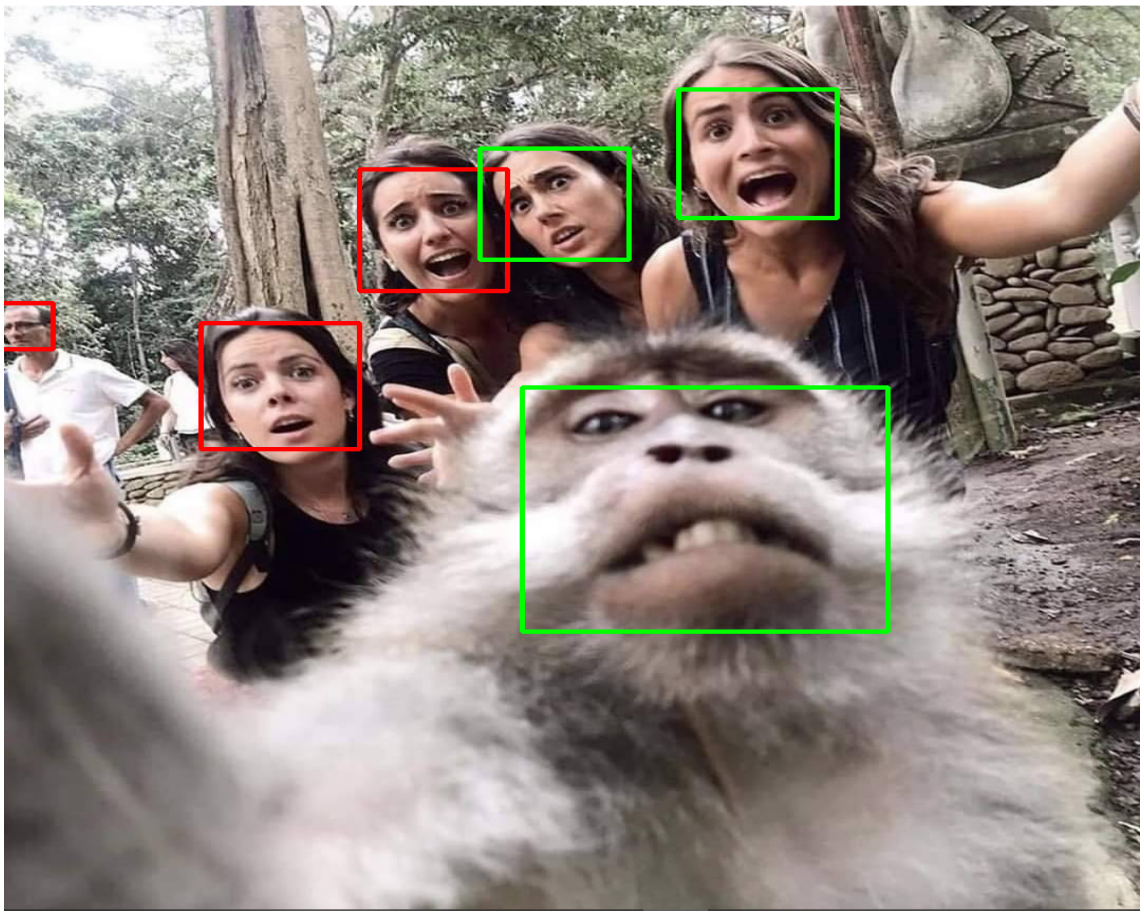
Part 4

After finishing the detect() in detection.py, we can obtain the 2 images.



Part 5

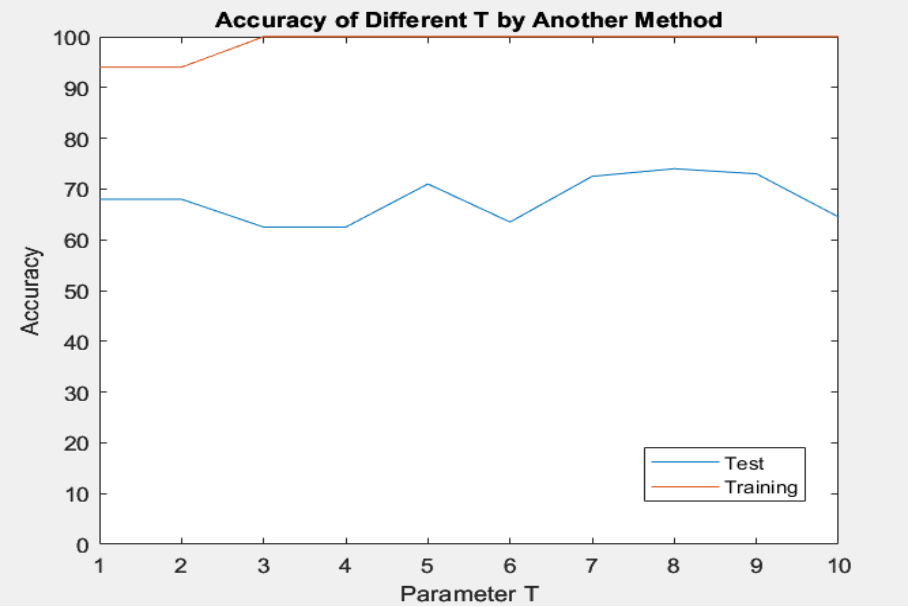
Select the points by Microsoft Paint and test. The following is the result.



Part 6

In the selectBest function, I calculate the best threshold and polarity of each classifier first, and then calculate their errors to find the smallest one.
According to the results, it indicates that we can get higher accuracy if we give each classifier its suitable threshold and polarity.

200 images	Training data accuracy	Test data accuracy
$t = 1$	94.0%	68.0%
$t = 2$	94.0%	68.0%
$t = 3$	100.0%	62.5%
$t = 4$	100.0%	62.5%
$t = 5$	100.0%	71.0%
$t = 6$	100.0%	63.5%
$t = 7$	100.0%	72.5%
$t = 8$	100.0%	74.0%
$t = 9$	100.0%	73.0%
$t = 10$	100.0%	64.5%



By giving the best thresholds for each classifier, we do get better performance. Still, the limit of the accuracy of test data is 74%, even we increasing the parameter T.

Part III. Answer the questions

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

1. Didn't know how to start Part 2 at beginning, but after I took a long time reading the slides and the whole code, I realize how adaboost works and how to select the best classifier to improve the accuracy.
2. The coordinate in the txt file makes me detect the wrong position. Since I understood what the four numbers represent to, I was able to detect the faces and draw boxes on the image which are at the right position.

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

1. Sensitive to light conditions.
2. Results may be wrong if the faces are rotated.
3. It takes a large amount of training time.

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

Modify a better way for machine to select the classifier.

4. Other than Viola-Jones' algorithm, please propose another possible face detection method. Please the discuss the pros and cons of the idea you proposed, compared to the Adaboost algorithm.

RetinaFace, a deep learning-based method for face recognition which was published in 2019.
Compare RetinaFace with Viola-Jones' algorithm :

Comparison \ Algorithm	Viola-Jones	RetinaFace
Accuracy	60%80%	Higher than Viola-Jones
Accuracy under challenging condition	Low	High
Required dataset for training	Less than RetinaFace	More than Viola-Jones
Detect faces at different scales and orientations	Mostly impossible	Able

In conclusion, due to the CNN and deep learning, RetinaFace is a robust and more accurate method compared to Viola-Jones, and that's why it is frequently applied in real world.