Homework 1: Face Detection Report

109550087 單宇晟

Implementation

Part1:

```python
import os
import cv2

def loadImages(dataPath):
    """
    load all Images in the folder and transfer a list of tuples. The first
    element is the numpy array of shape (m, n) representing the image.
    The second element is its classification (1 or 0)
      Parameters:
        dataPath: The folder path.
      Returns:
        dataset: The list of tuples.
    """
    # Begin your code (Part 1)

    dataset = []
    path = dataPath + "/face"
    for filename in os.listdir(path):
        img = cv2.imread(path + "/" + filename, cv2.IMREAD_GRAYSCALE)
        dataset.append((img, 1))
    path = dataPath + "/non-face"
    for filename in os.listdir(path):
        img = cv2.imread(path + "/" + filename, cv2.IMREAD_GRAYSCALE)
        dataset.append((img, 0))

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

```

In part 1, I use os.listdir to find the files, and cv2.imread to read them. Then I use cv2.IMREAD_GRAYSCALE to read those images in gray scale. Also, I create a list called “dataset” to save those tuples which contain numpy arrays and classifications.

Part 2:

```
            bestClf: The best WeakClassifier Class
            bestError: The error of the best classifer
        """
        # Begin your code (Part 2)
        # raise NotImplementedError("To be implemented")

        # PDF 115
        # row = np.size(x, 0)
        # column = np.size(x, 1)
        dataLen = np.size(featureVals, 1)
        bestError = 1e8
        for j in range(len(features)):
            # for each feature j, train a classifier hj
            """
            x = np.array(featureVals[j])
            for element in x:
                if x < 0:
                    x = 1
                else:
                    x = 0
            """
            temp = WeakClassifier(features[j], 0, 1)
            error = 0
            for i in range(dataLen):
                # evaluate error for each feature
                error += weights[i] * abs(temp.classify(iis[i]) - labels[i])
            if error < bestError:
                bestError = error
                bestClf = temp

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

In part 2, I use np.size to find out how many data I need to count for each feature, and I use the function WeakClassifier to determine which classifier I am evaluating its error. In the computing part, I just simply convert the formula into python code. Finally, if the error is smaller than the bestError so far, update the value of bestError and the bestClf(classifier)

Part 4:

```
    txt = open(dataPath, "r") #read mode
    n = 0
    while n < 2:
        info = txt.readline()
        name = info.split(" ")[0]
        faces = info.split(" ")[1]
        boxes = []
        path = "data/detect/" + name
        img = cv2.imread(path)
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        for i in range(int(faces)):
            boxes.append(txt.readline())
            x, y, w, h = boxes[i].split(" ")
            x = int(x)
            y = int(y)
            w = int(w)
            h = int(h)
            img_crop = img_gray[y : y + h, x : x + w]
            img_resize = cv2.resize(img_crop, (19, 19), interpolation = cv2.INTER_NEAREST)
            start_point = (x, y)
            end_point = (x + w, y + h)
            # color
            green = (0, 255, 0) #face
            red = (0, 0, 255) #non-face
            # thickness
            if (clf.classify(img_resize) == 1):
                img = cv2.rectangle(img, start_point, end_point, green, 2)
            else:
                img = cv2.rectangle(img, start_point, end_point, red, 2)

        cv2.imshow(name, img)
        n += 1
```

In part 4, I open detectData.txt first. Then I use readline repeatedly to access the information in the txt file. In the first line, I will read the image file name, and the number of faces. Let's say, n faces. Then I can know that in the next n lines, the txt file will tell me the x, y coordinate, width and height of the block. Since I know where blocks are, I resize and convert them into 19 x 19 grayscale images. After finishing processes above, I use the function "classify" to see if the image in the block is detected as a face or not. Last, according to the result, put a certain colored box onto the image.

Part II. Results & Analysis:

Program result

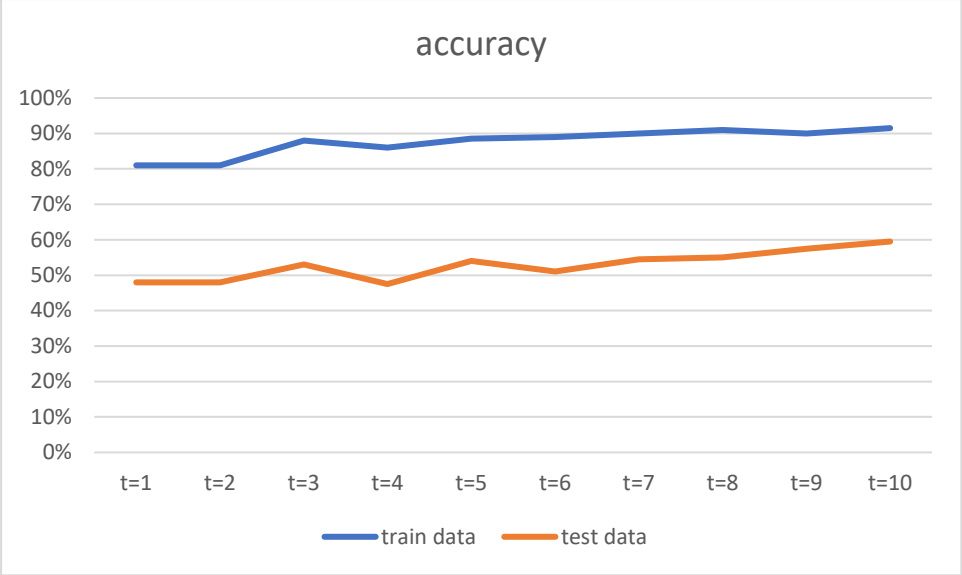
```
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature
(positive regions=[RectangleRegion(10, 4, 1, 1)], negative
regions=[RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and
alpha: 0.707795
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, 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)
```

Data chart

	train data	test data
t=1	81%	48%
t=2	81%	48%
t=3	88%	53%
t=4	86%	47.50%
t=5	88.50%	54%
t=6	89%	51%
t=7	90%	54.50%
t=8	91%	55%
t=9	90%	57.50%
t=10	91.50%	59.50%



According to the result, I found out that the accuracy tends to increase while T is also increasing. However, in image2, the number of green boxes, which means that the block is detected as a face, is higher at t = 1 but lower at t = 10. Although it seems that when t = 1, the result is better. Nevertheless, I think it means that the program wasn't trained well. In other word, when t = 1, the program would consider any object a face easily, so if we circle a non-face object for it to detect, it would give a green box as well.

Image1 (t = 10)

Image2 (t = 10)

Image2 (t = 1)

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, so some simple code such as “for”, I need to google for its format. Second, some functions took time to understand. Also, I have never used libraries such as opencv and matplotlib, so it really took me a lot of time to google how to use them. However, thanks to Google and my roommates, they help me with this homework a lot, which let me run my program successfully.

2. What are the limitations of the Viola-Jones’ algorithm?
   In this algorithm, the face’s direction is very important. When the face is in frontal view, the algorithm can detect it correctly. However, when the face is in side view, it would be detected as non-face. Second, Viola-Jones’ algorithm doesn’t put brightness into consideration, so same picture with different brightness may lead to different results.

3. Based on Viola-Jones’ algorithm, how to improve the accuracy except increasing the training dataset and changing the parameter T?
   We can change the threshold, because the threshold may not always be 0, sometimes it can have a negative value. This way, we can improve our algorithm.

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.
   After finishing this homework, I came a with another method. I think we should put different angles of face while training so that the program can detect more various faces later. Next, I would put hair into consideration, because most human have hair. If an object satisfy both harr features and hair appearance, it is probably a face.
   Pros: can detect faces at different angle, more accurate
   Cons: hair is hard to detect, and sometimes mixes with the background, such as black hair, black background.