HW1 report

Part I. Implementation (5%):

- Part 1-1

```
23      """
24      line 29 : prepare data path
25      line 30 ~ 31 : prepare two lists for train dataset and test dataset
26      line 32 ~ 46 : use for loop to get the filename in each folder and read the image and append them to the
27      dataset they belong to with their labels, and I let face image to be appended first
28      """
29      path = "data/data_small"
30      train_dataset = []
31      test_dataset = []
32      for filename in os.listdir(f"{path}/train/face"):
33          image = cv2.imread(os.path.join(f"{path}/train/face/{filename}"), cv2.IMREAD_GRAYSCALE)
34          train_dataset.append((image, 1))
35
36      for filename in os.listdir(f"{path}/train/non-face"):
37          image = cv2.imread(os.path.join(f"{path}/train/non-face/{filename}"), cv2.IMREAD_GRAYSCALE)
38          train_dataset.append((image, 0))
39
40      for filename in os.listdir(f"{path}/test/face"):
41          image = cv2.imread(os.path.join(f"{path}/test/face/{filename}"), cv2.IMREAD_GRAYSCALE)
42          test_dataset.append((image, 1))
43
44      for filename in os.listdir(f"{path}/test/non-face"):
45          image = cv2.imread(os.path.join(f"{path}/test/non-face/{filename}"), cv2.IMREAD_GRAYSCALE)
46          test_dataset.append((image, 0))
```

  - line 29 : prepare data path
  - line 30 ~ 31 : prepare two lists for train dataset and test dataset
  - line 32 ~ 46 : use for loop to get the filename in each folder and read the image and append them to the dataset they belong to with their labels, and I let face image to be appended first

- Part 1-2

```
113 """
114   line 120 : pick the current face bounding box
115   line 122 ~ 125 : define non-face bounding box with small intersection with face box by moving the face box
116   with some random offset in x and y, and get the non-face box's left top and right bottom
117   line 127 ~ 128 : ensure non-face bounding box is within image bounds
118   line 130 : crop non-face region
119   """
120   face_box = face_box_list[i]
121
122   x_offset = np.random
123   y_offset = np.random
124   left_top_nonface = (face_box[0][0] + x_offset, face_box[0][1] + y_offset)
125   right_bottom_nonface = (face_box[1][0] + x_offset, face_box[1][1] + y_offset)
126
127   left_top_nonface = (max(left_top_nonface[0], 0), max(left_top_nonface[1], 0))
128   right_bottom_nonface = (min(right_bottom_nonface[0], img_gray.shape[1]), min(right_bottom_nonface[1], img_gray.shape[0]))
129
130   img_crop = img_gray[left_top_nonface[1]:right_bottom_nonface[1], left_top_nonface[0]:right_bottom_nonface[0]].copy()
```

(variable) face_box: Any

  - line 120 : pick the current face bounding box

- line 122 ~ 125 : define non-face bounding box with small intersection with face box by moving the face box with some random offset in x and y, and get the non-face box's left top and right bottom
- line 127 ~ 128 : ensure non-face bounding box is within image bounds
- line 130 : crop non-face region

- Part 2 & 6

```
    line 182 : method == 1 means use the formula in the slides to choose the best weak classifier
    line 183 ~ 184 : initialize bestClf and bestError
    line 185 : for each feature to choose a classifier
    line 186 : initialize error
    line 187 ~ 189 : for each image to calculate h and error
    line 190 ~ 192 : to choose the bestClf with bestError

    line 193 : method == 2 means use the formula I use for part 6 to choose the best weak classifier
    line 194 ~ 195 : initialize bestClf and bestError
    line 196 : for each feature to choose a classifier
    line 197 : choose the maximum value among all images with current feature
    line 198 : initialize error
    line 199 ~ 201 : for each image to calculate h and error, but I calculate h by new formula, if featureVal >= 0 it's 0,
    else h = featureVals[i][j] / (Max + 0.001) + 0.1
    line 202 ~ 204 : to choose the bestClf with bestError
    """
    if method == 1:
        bestClf = None
        bestError = 1e9
        for i in range(len(features)):
            error = 0
            for j in range(len(labels)):
                h = np.where(featureVals[i][j] < 0, 1, 0)
                error += weights[j] * abs(h - labels[j])
            if error < bestError:
                bestError = error
                bestClf = WeakClassifier(features[i])
    else:
        bestClf = None
        bestError = 1e9
        for i in range(len(features)):
            Max = np.max(featureVals[i])
            error = 0
            for j in range(len(labels)):
                h = np.where(featureVals[i][j] < 0, featureVals[i][j] / (Max + 0.001) + 0.1, 0)
                error += weights[j] * abs(h - labels[j])
            if error < bestError:
                bestError = error
                bestClf = WeakClassifier(features[i])
```

- line 182 : method == 1 means use the formula in the slides to choose the best weak classifier
- line 183 ~ 184 : initialize bestClf and bestError
- line 185 : for each feature to choose a classifier
- line 186 : initialize error
- line 187 ~ 189 : for each image to calculate h and error
- line 190 ~ 192 : to choose the bestClf with bestError
- line 193 : method == 2 means use the formula I use for part 6 to choose the best weak classifier

- line 194 ~ 195 : initialize bestClf and bestError
- line 196 : for each feature to choose a classifier
- line 197 : choose the maximum value among all images with current feature
- line 198 : initialize error
- line 199 ~ 201 : for each image to calculate h and error, but I calculate h by new formula, if featureVal >= 0 it's 0, else h = featureVals[i][j] / (Max + 0.001) + 0.1
- line 202 ~ 204 : to choose the bestClf with bestError
- Why I divide featureVals by (Max + 0.001) is because I find that Max may be 0, so I plus 0.001 to avoid being divided by 0. And I additionally plus 0.1 because I find its performace is better than others
- I add a parameter in selectBest which is method, you can choose method by yourself

- Part 4

```
"""
line 31 ~ 33 : prepare data structure to store filename, current file, and box region
line 34 ~ 45 : read the txt and store the filename and their box region
line 47 ~ 49 : for each filename, read the image in RGB and grayscale, RGB image is to show while gray image is to get the box
line 51 ~ 58 : for each box region, crop the region on the gray image and resize into 19 x 19, and check whether it's face or
non-face by classifier, and use rectangle to show the box in red or green on the RGB image
line 60 ~ 61 : show the result and wait
"""
filename = []
current_file = ""
rectangleRegion = dict()
with open(dataPath) as file:
    for lines in file:
        line = lines.split(' ')
        if len(line) == 2:
            current_file = line[0]
            filename.append(current_file)
            rectangleRegion[current_file] = []
        else:
            new_region = []
            for x in line:
                new_region.append(int(x))
            rectangleRegion[current_file].append(new_region)
for file in filename:
    image = cv2.imread(os.path.join(f"data/detect/{file}"), cv2.IMREAD_UNCHANGED)
    original_image = cv2.imread(os.path.join(f"data/detect/{file}"), cv2.IMREAD_GRAYSCALE)

    for region in rectangleRegion[file]:
        x, y, w, h = region
        face = original_image[y : y + h, x : x + w]
        face = cv2.resize(face, (19, 19), interpolation = cv2.INTER_CUBIC)
        if clf.classify(face) == 1:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
        else:
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

    cv2.imshow("windows", image)
    cv2.waitKey(0)
```

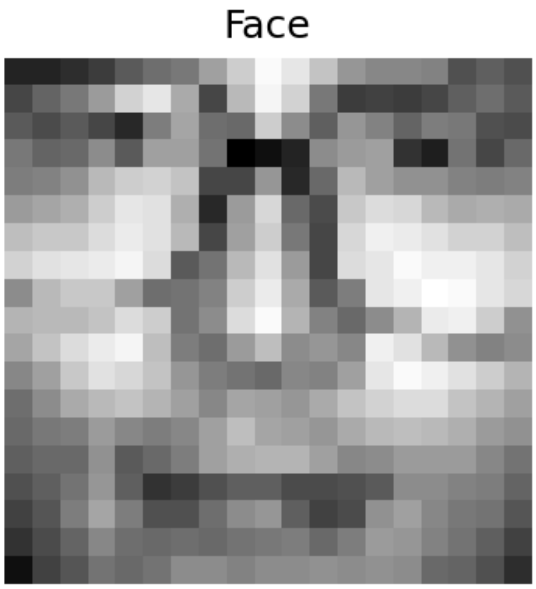
- line 31 ~ 33 : prepare data structure to store filename, current file, and box region

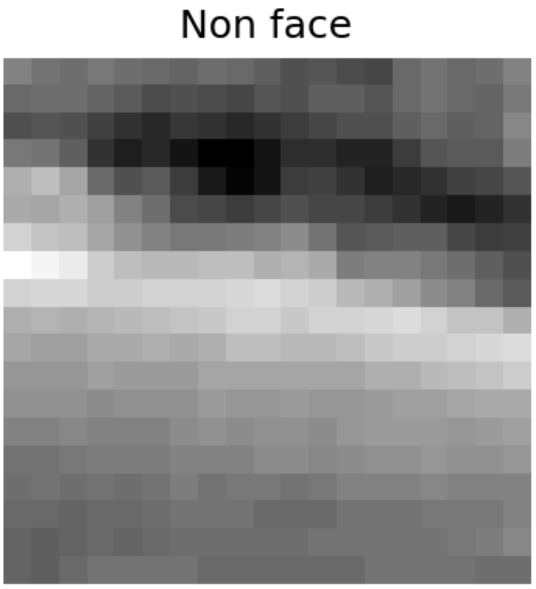
- line 34 ~ 45 : read the txt and store the filename and their box region
- line 47 ~ 49 : for each filename, read the image in RGB and grayscale, RGB image is to show while gray image is to get the box
- line 51 ~ 58 : for each box region, crop the region on the gray image and resize into 19 x 19, and check whether it's face or
- non-face by classifier, and use rectangle to show the box in red or green on the RGB image
- line 60 ~ 61 : show the result and wait

Part II. Results & Analysis (10%):

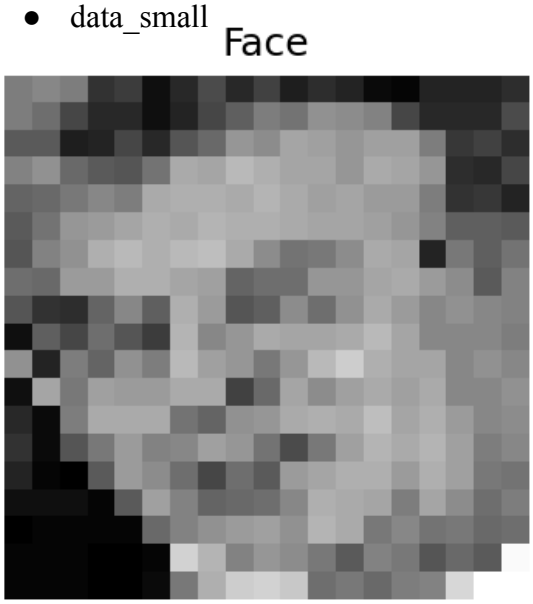
Screenshots for results :

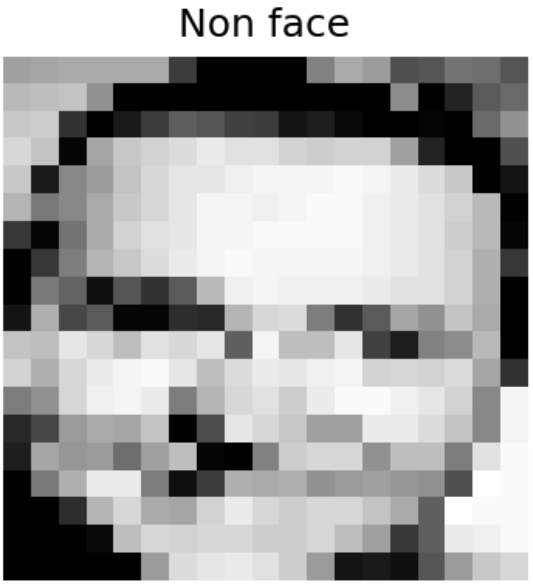
- Results for Part 1





- data_small





- data_FDDB

- Results for Part 2 (I choose the best results to be here, others will be in charts)

```
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_small, method 1, T = 10

```
Evaluate your classifier with training dataset
False Positive Rate: 61/100 (0.610000)
False Negative Rate: 3/100 (0.030000)
Accuracy: 136/200 (0.680000)

Evaluate your classifier with test dataset
False Positive Rate: 57/100 (0.570000)
False Negative Rate: 10/100 (0.100000)
Accuracy: 133/200 (0.665000)
```

  - data_small, method 2, T = 10

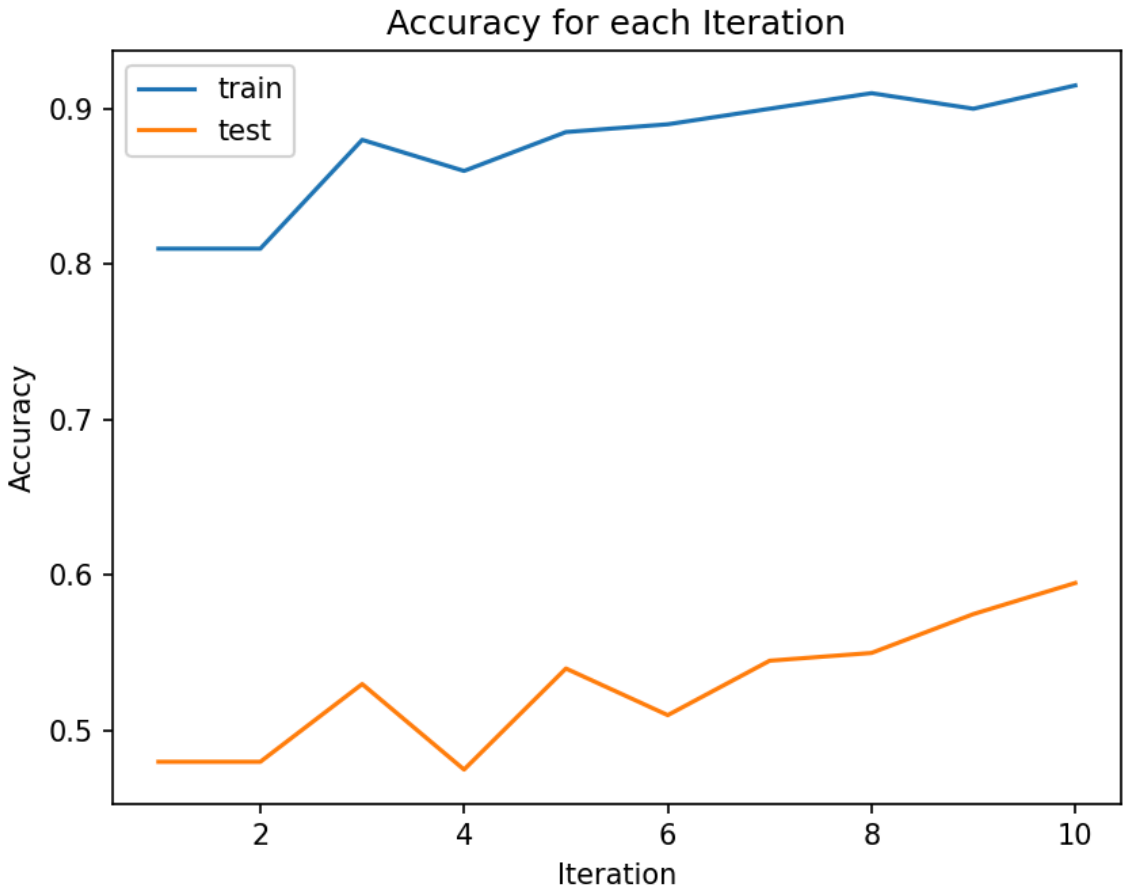
```
Evaluate your classifier with training dataset
False Positive Rate: 150/360 (0.416667)
False Negative Rate: 46/360 (0.127778)
Accuracy: 524/720 (0.727778)

Evaluate your classifier with test dataset
False Positive Rate: 75/155 (0.483871)
False Negative Rate: 22/155 (0.141935)
Accuracy: 213/310 (0.687097)
```

- data_FDDB, method 1, T = 9

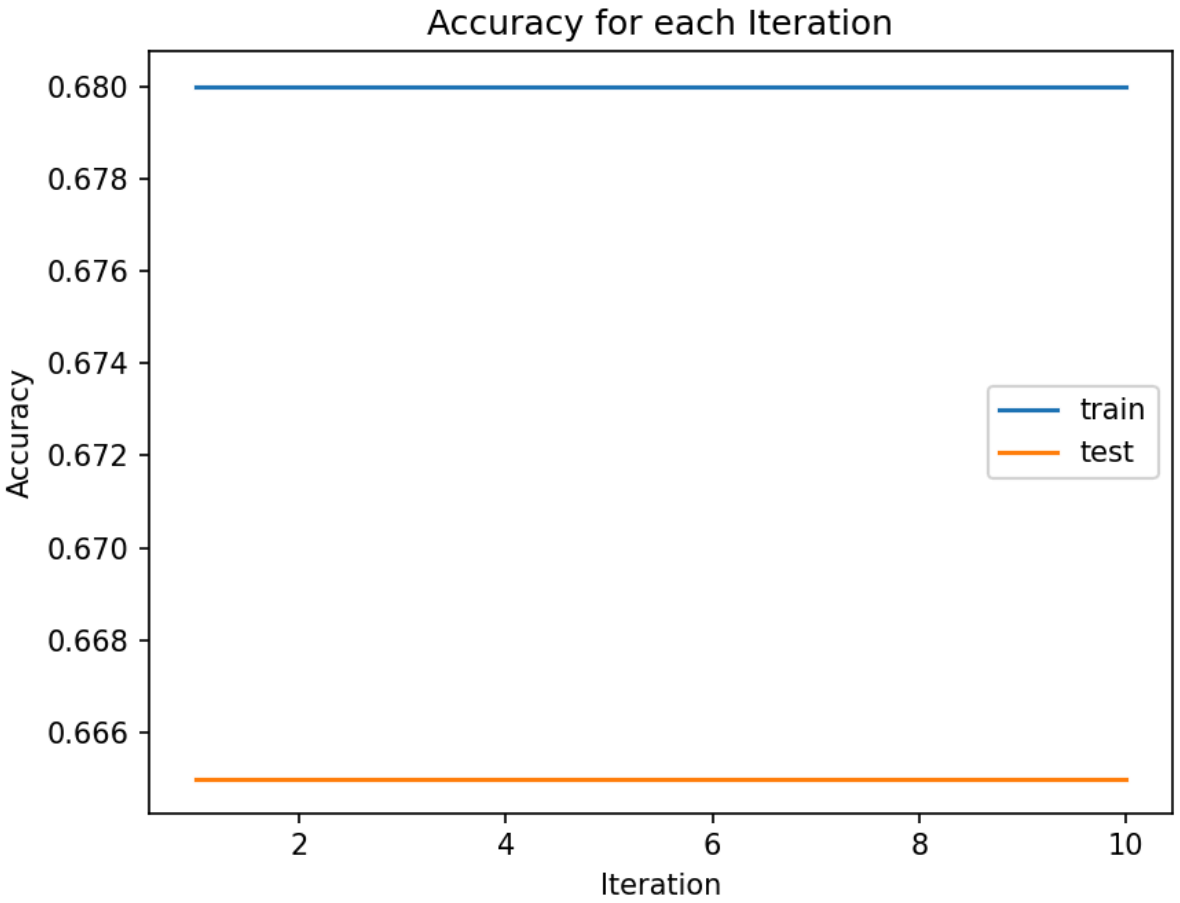
```
Evaluate your classifier with training dataset
False Positive Rate: 197/360 (0.547222)
False Negative Rate: 95/360 (0.263889)
Accuracy: 428/720 (0.594444)

Evaluate your classifier with test dataset
False Positive Rate: 91/155 (0.587097)
False Negative Rate: 34/155 (0.219355)
Accuracy: 185/310 (0.596774)
```

- data_FDDB, method 2, T =10

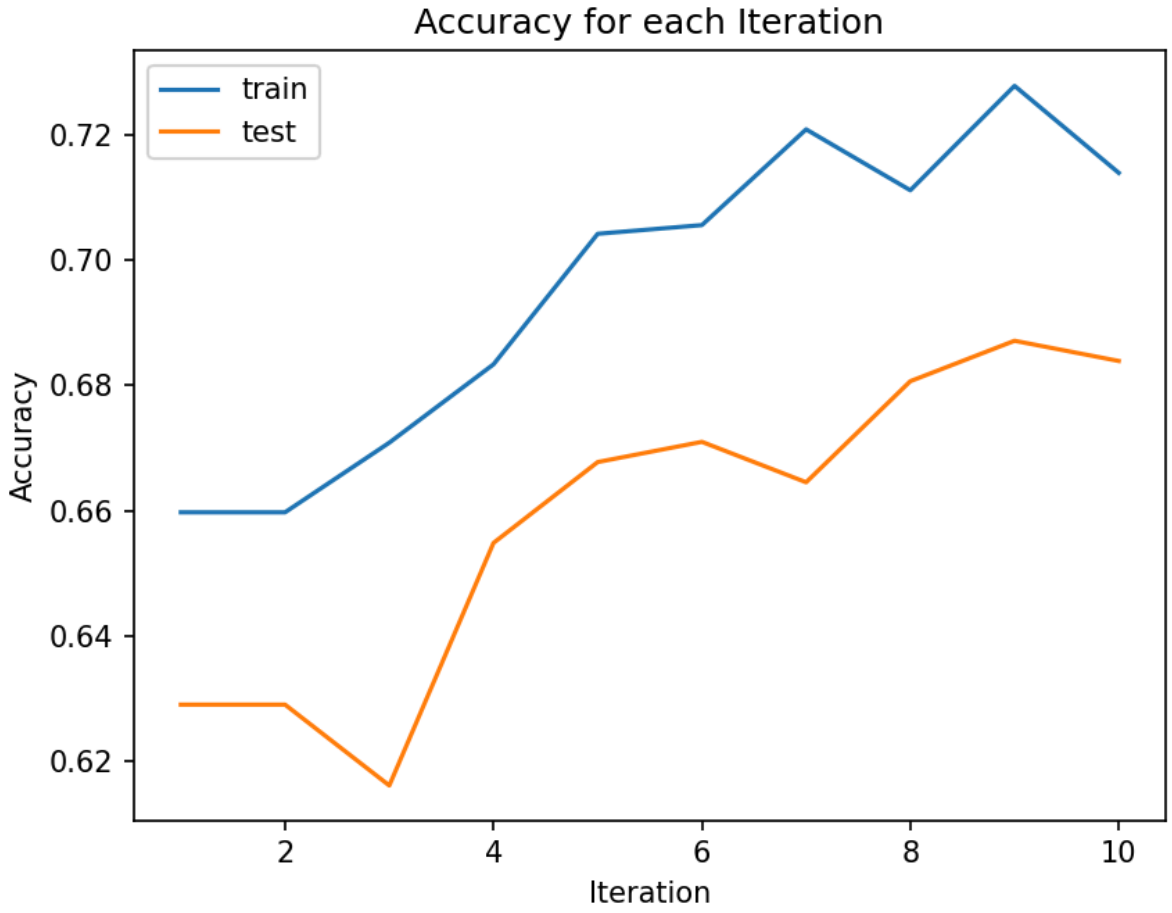
- Results for Part 3



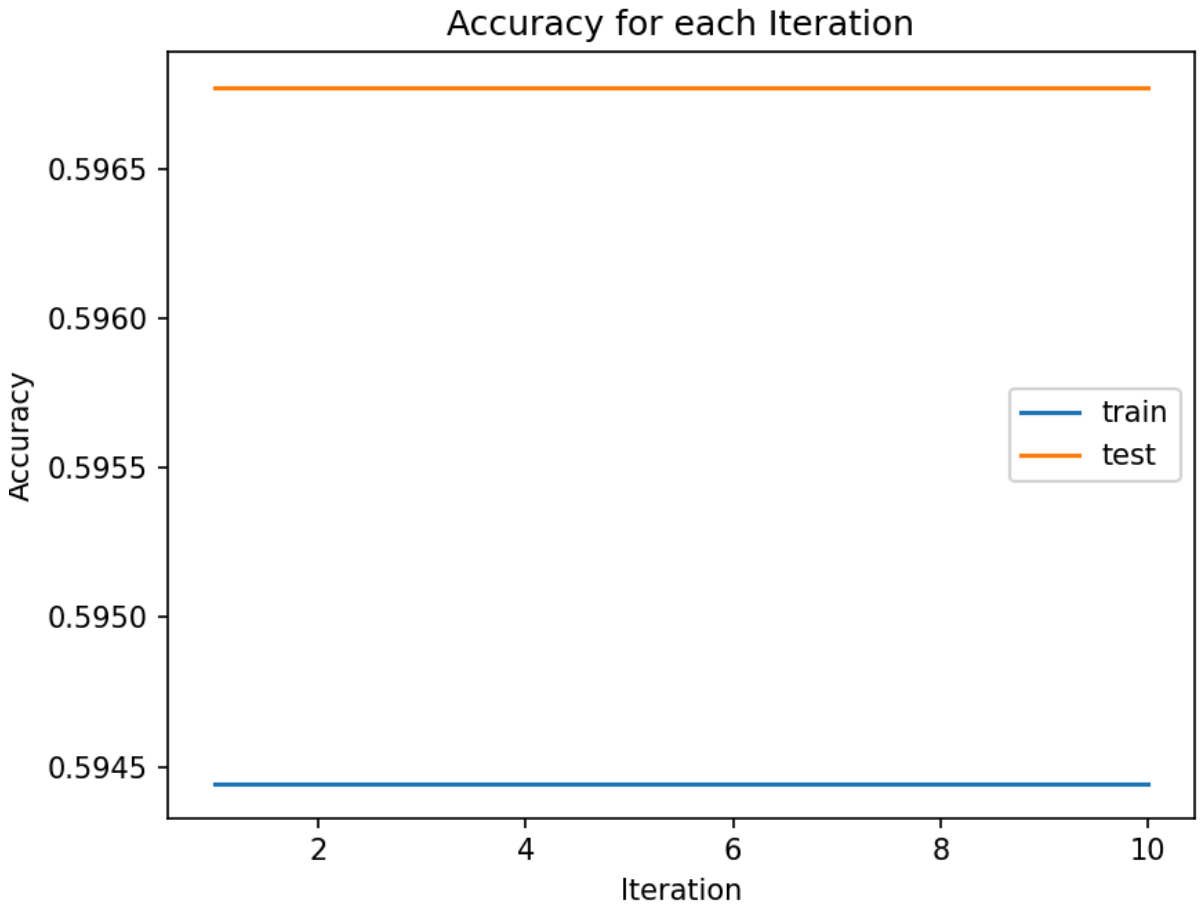
- data_small, method 1



- data_small, method 2

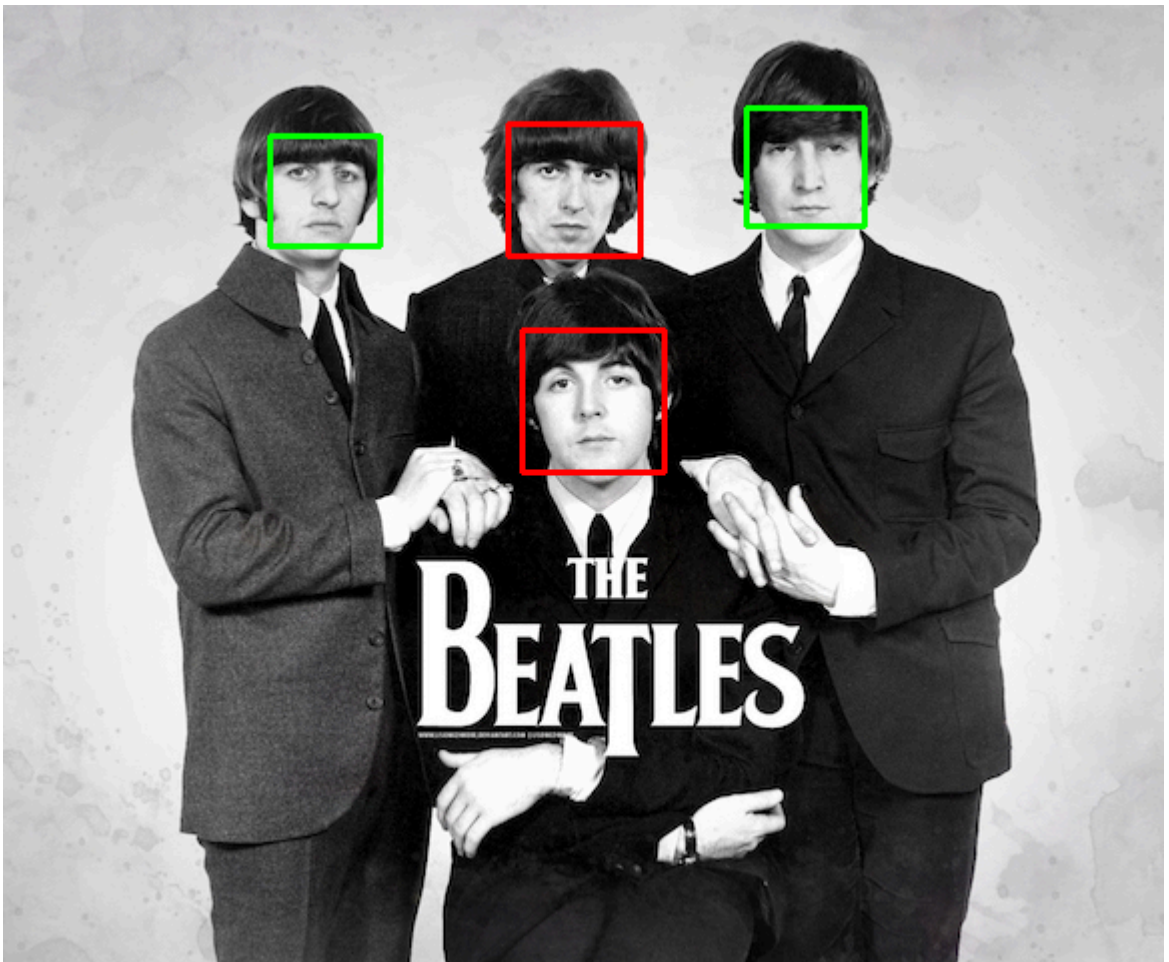


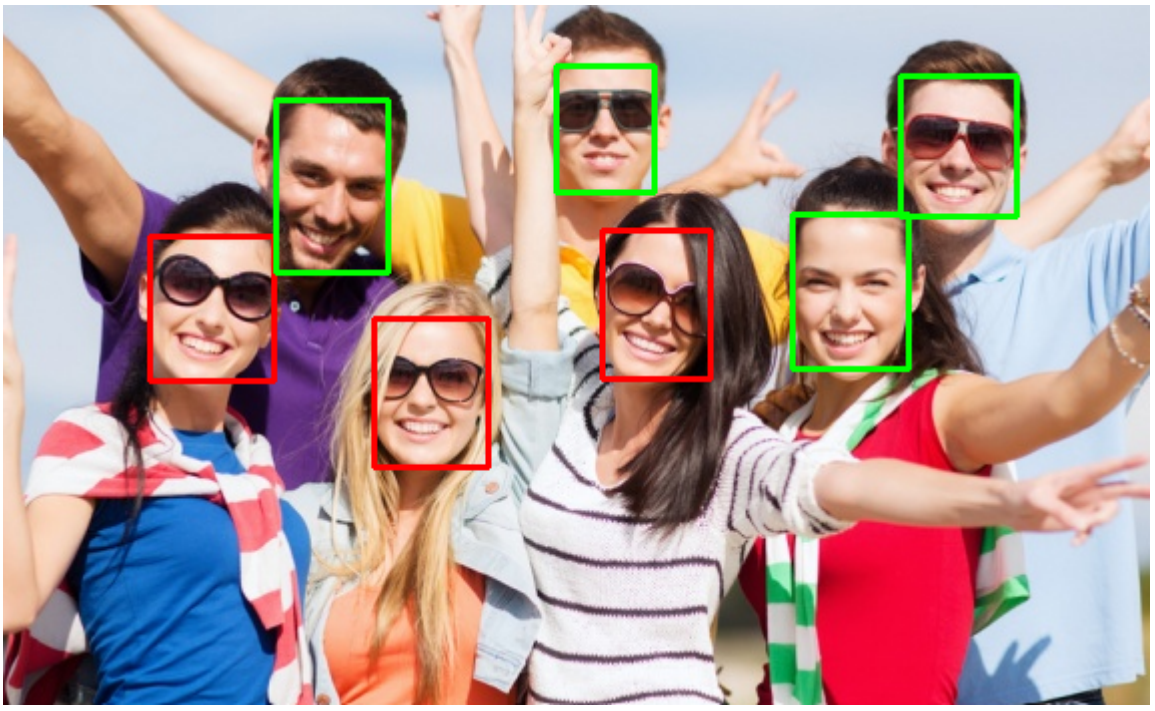
- data_FDDB, method 1



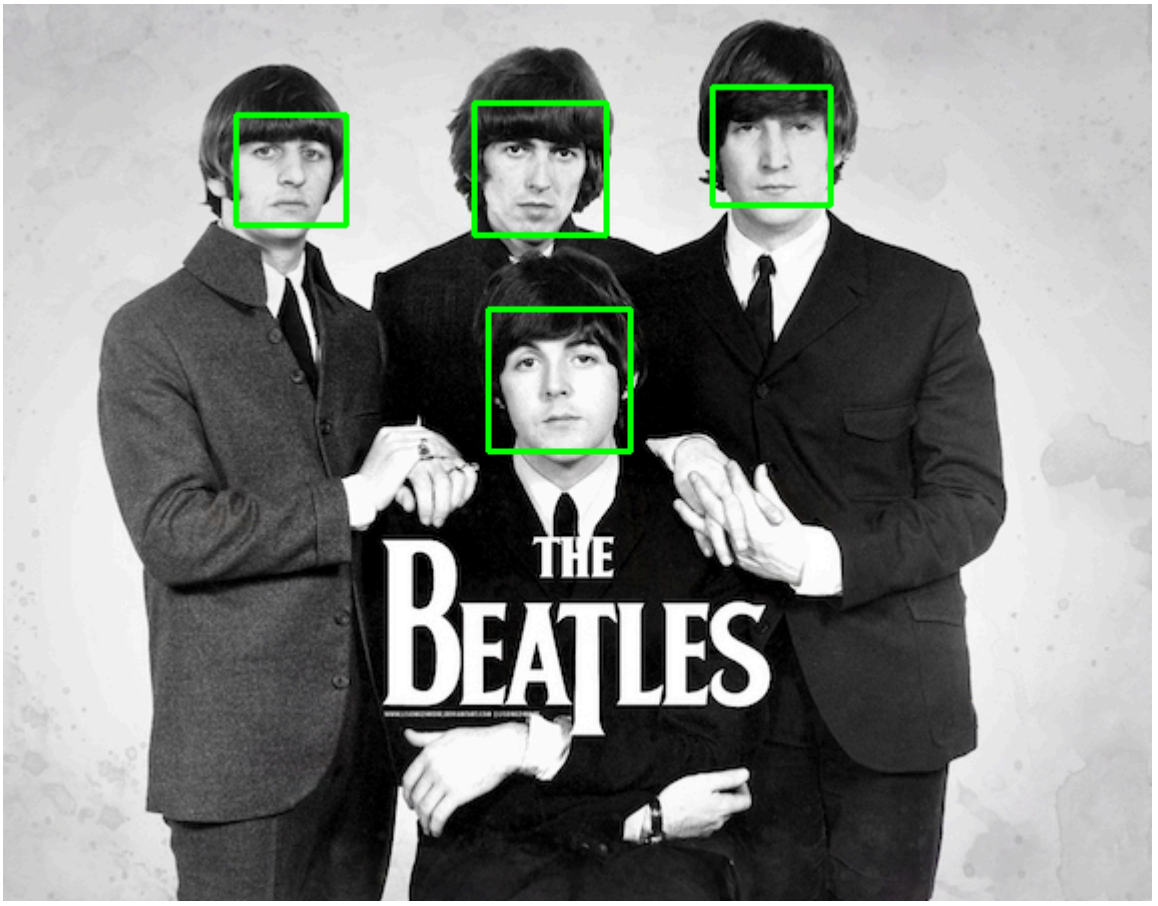
- data_FDDB, method 2

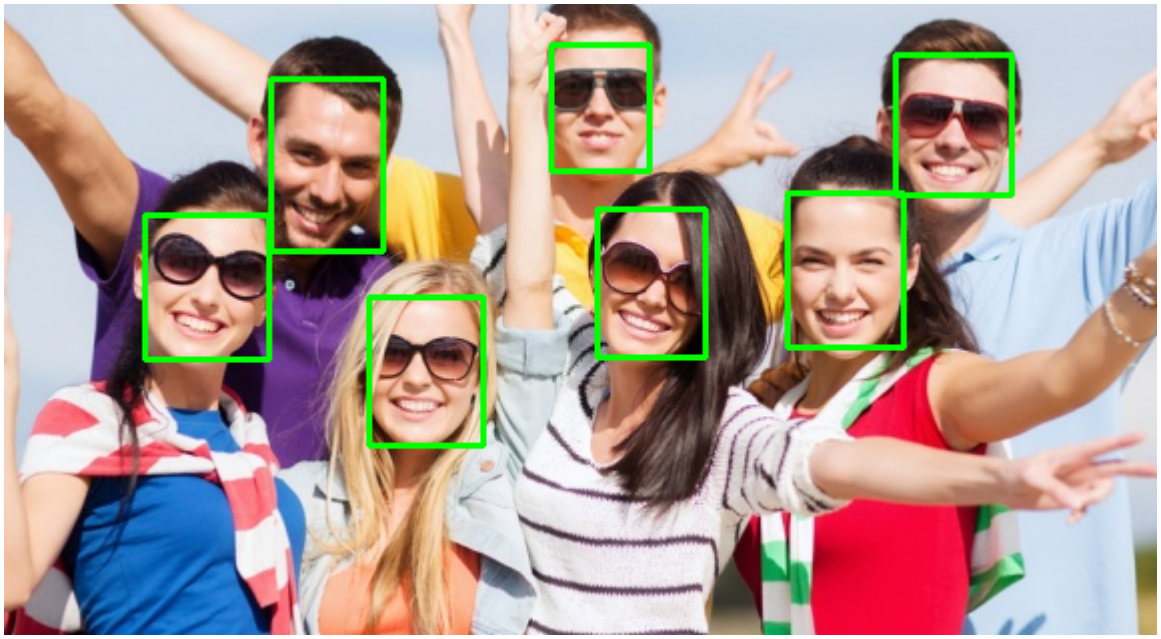
- Results for Part 4 & 5



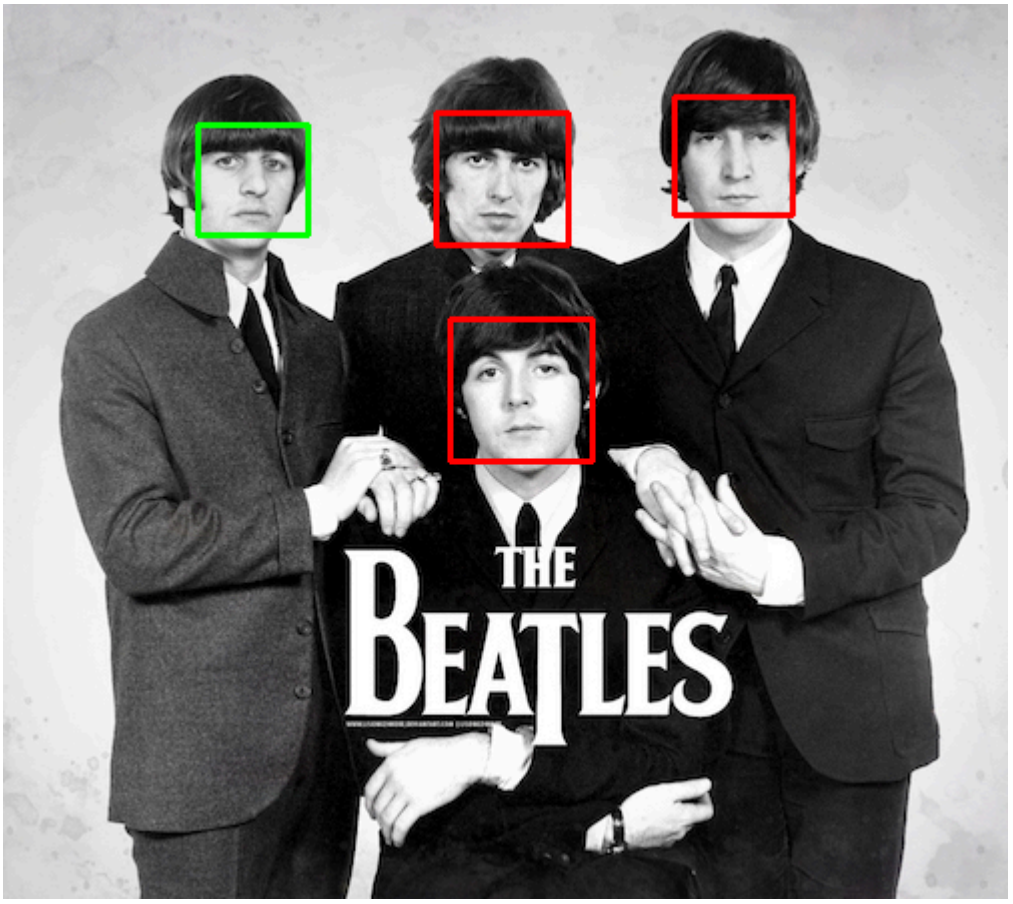


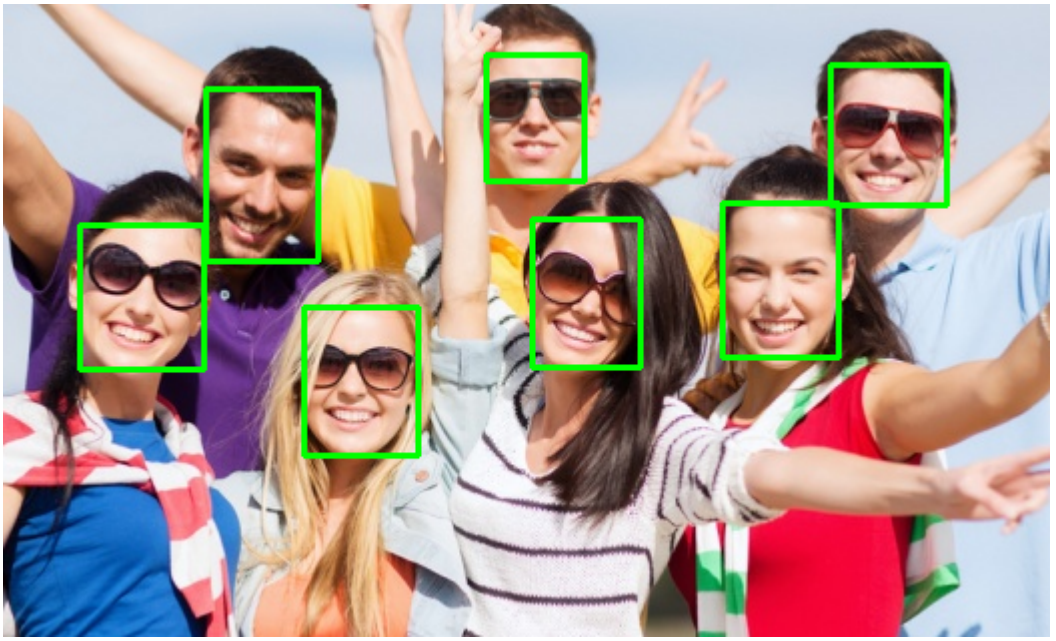
- data_small, method 1, T = 10

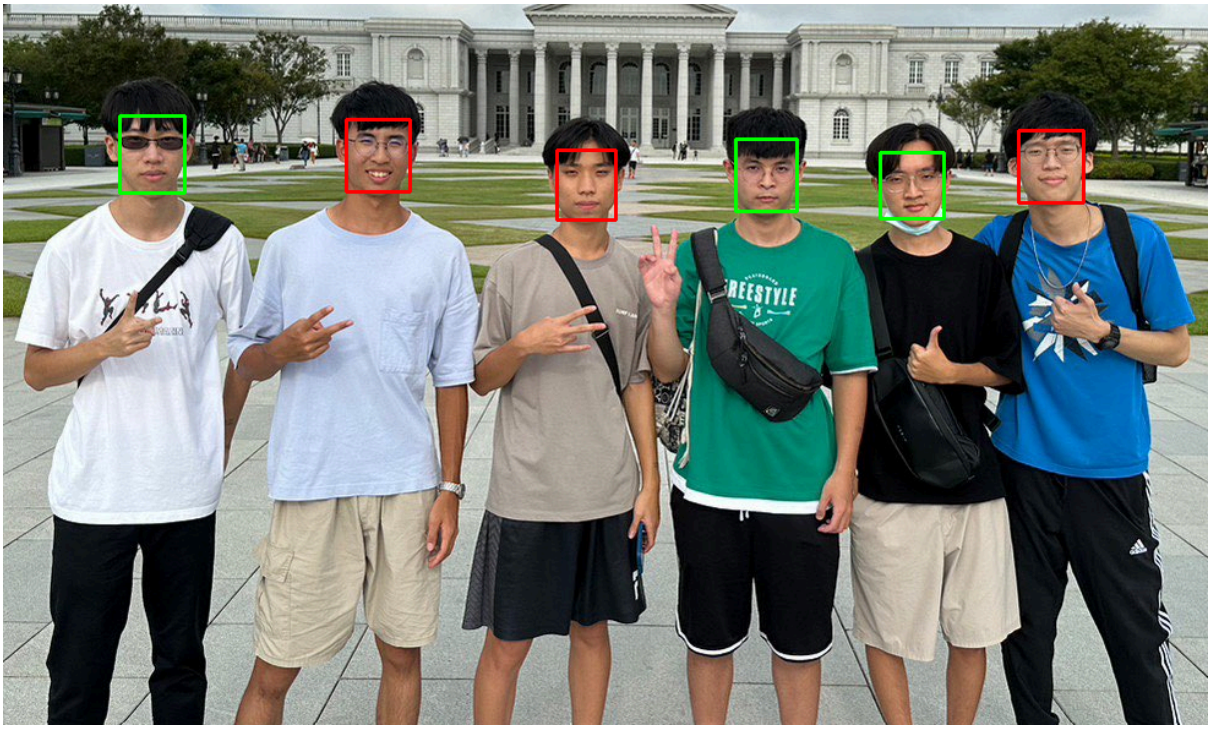

THE
BEATLES



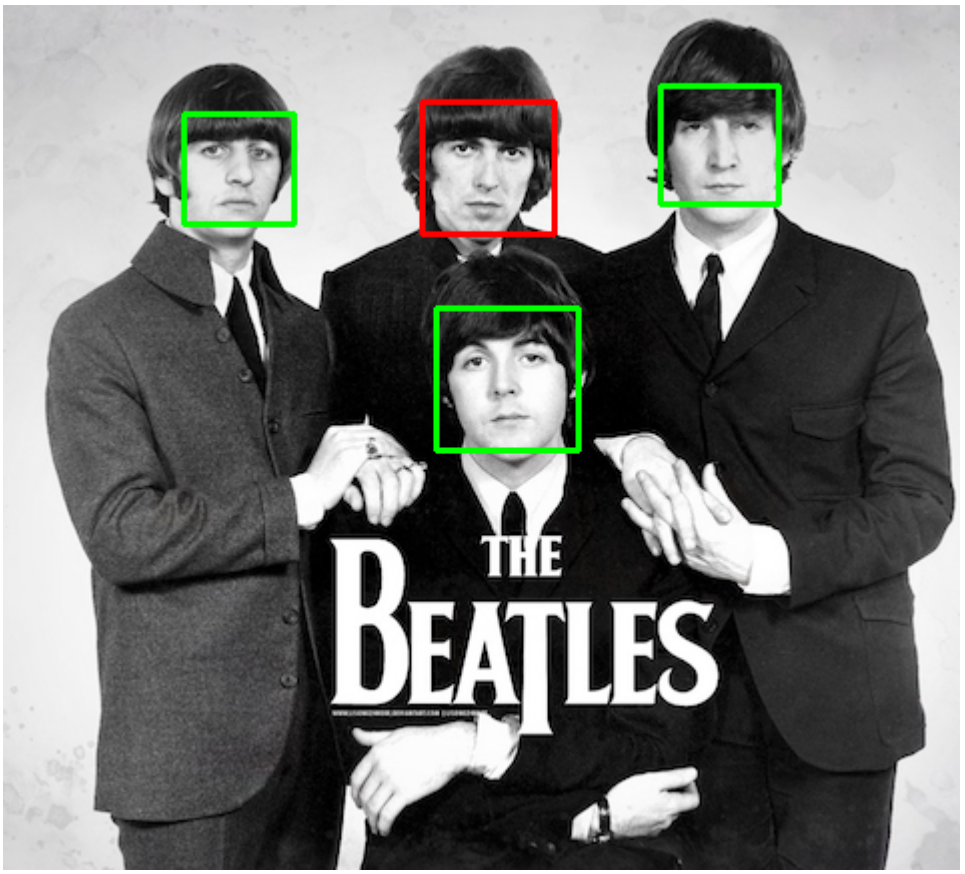
- data_small, method 2, T = 10

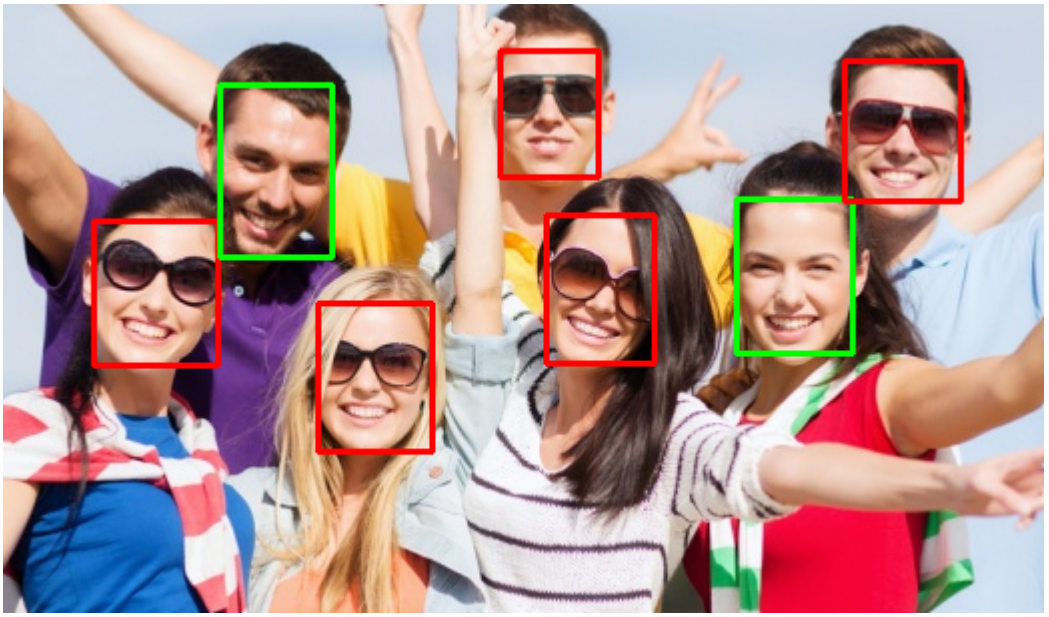


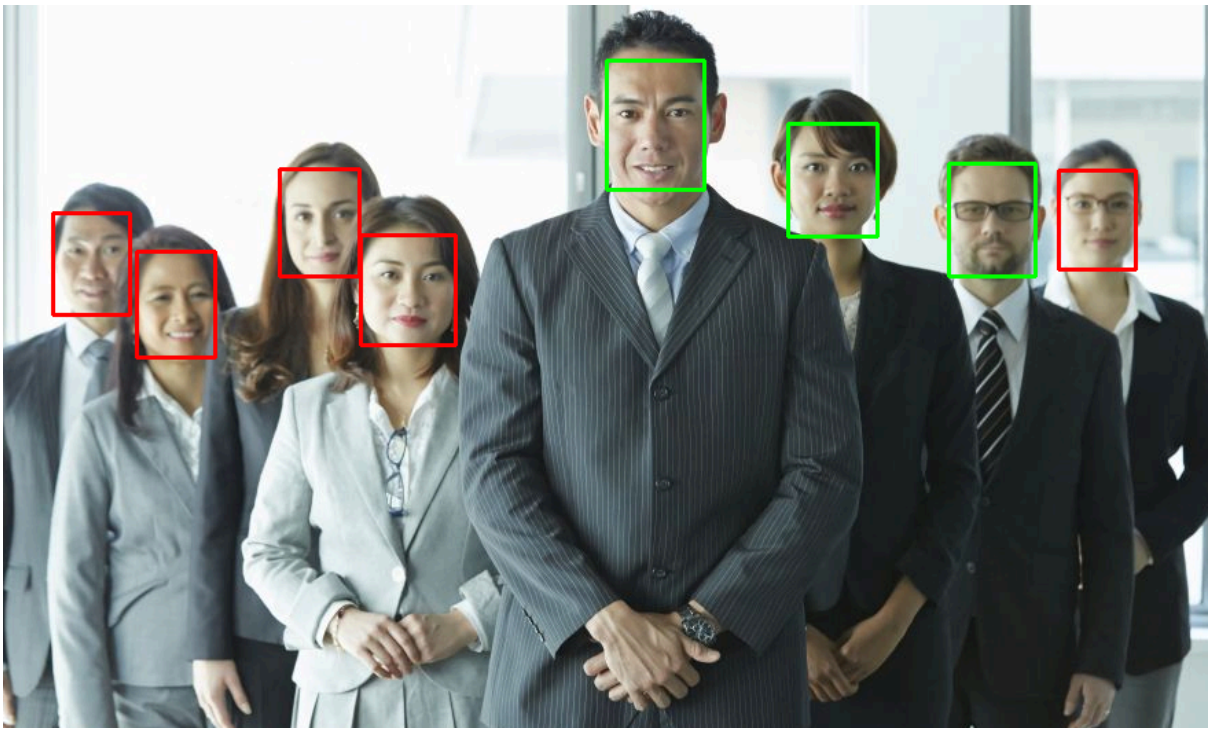


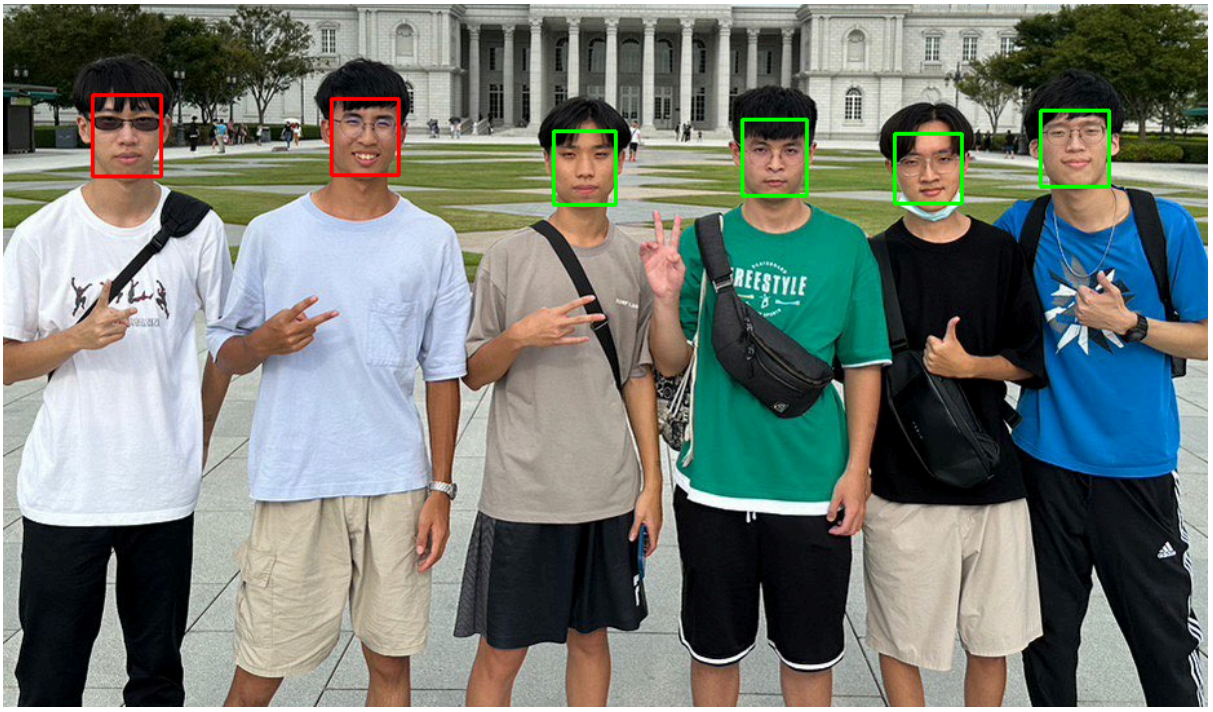


- data_FDDB, method 1, T = 9


THE
BEATLES







- data_FDDB, method 2, T = 10

Analysis or observation :

- By the charts above, I noticed that training dataset always has better performance than testing dataset. I think overfitting may be the main reason.
- I also noticed that as parameters T grows, the accuracy grows too. But method 2's accuracy always keeps the same. I found that at each iteration, it always chooses the same classifier. I think the reason is that the maximum value for a feature among all images is too big, so it dominates the whole selection.
- Compare method 1 and method 2 in data_small, method 1's training dataset's performance is better than method 2, but in testing dataset, method 2 is better. Moreover, the performance on detecting face also method 2 wins.
- Compare method 1 and method 2 in data_FDDB, method 1's training dataset and testing dataset's performance are better than method 2. And the performance on detecting face, I think they tied.
- Compare data_small and data_FDDB by method 1, the classifier trained by data_small has better performance on training dataset than that of data_FDDB, but on the performance of testing dataset, the classifier trained by data_FDDB is better. And I think on the performance of detecting face, the classifier of data_FDDB is better.

- Compare data_small and data_FDDB by method 2, the classifier trained by data_small has better performance on training dataset and testing dataset than that of data_FDDB. I noticed that though just a little bit, the classifier trained by data_FDDB has better performance on testing data than training data. And I think on the performance of detecting face, the classifier of data_small is better.

Part III. Answer the questions (15%):

1. **Please describe a problem you encountered and how you solved it.**
   When I need to draw a bounding box, I use the wrong color to draw it. cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2), I don't know the color is BGR, I think it is RBG. So my face is red, my non-face is blue. I changed the color order to solve this problem.
2. **How do you generate "nonface" data by cropping images?**
   I define a non-face bounding box with a small intersection with the face box by moving the face box with some random offset in x and y, and get the non-face box's left top and right bottom. This is how I generate non-face.
3. **What are the limitations of the Viola-Jones' algorithm?**
   Viola-Jones' algorithm has limitations such as its specificity to certain object classes, sensitivity to variations in lighting and pose, tendency for false positives in cluttered scenes, and computational intensity due to feature extraction. It may also struggle with occluded faces and requires preprocessing steps.
4. **Based on Viola-Jones' algorithm, how to improve the accuracy except changing the training dataset and parameter T?**
   Use more complex features other than Haar-like features or we can try more detectors to detect one object's different region.
5. **Other than Viola-Jones' algorithm, 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.**
   Convolutional Neural Networks compare to adaboost algorithm
   - Pros
     - High accuracy : CNNs has ability to recognize more complex patterns from data

- End to end learning : CNNs can learn hierarchical representations directly from raw pixel values, reducing the need for manual tuning
- Robustness: CNNs are more robust to variations in lighting, pose, and facial expressions compared to Adaboost, which may lead to better performance in challenging conditions.

- Cons

  - CNNs needs more training data to get well performance
  - CNNs needs more computational resource, especially for large models and high-resolution images