Homework 1: Face Detection

Report

Please keep the title of each section and delete examples. Note that please keep the questions listed in Part III.

Part I. Implementation (6%):

- **Please screenshot your code snippets of Part 1, Part 2, Part 4, and explain your implementation.**
  - **Part 1**

```
# Begin your code (Part 1)
dataset = []
realpath = os.path.join(dataPath, 'face')
for filename in os.listdir(realpath):
  data = cv2.imread(os.path.join(realpath, filename), cv2.IMREAD_GRAYSCALE)
  datatuple = (data, 1)
  dataset.append(datatuple)
realpath = os.path.join(dataPath, 'non-face')
for filename in os.listdir(realpath):
  data = cv2.imread(os.path.join(realpath, filename), cv2.IMREAD_GRAYSCALE)
  datatuple = (data, 0)
  dataset.append(datatuple)
# Read the pgm file inside face and non-face folder in grayscale mode with cv2.imread()
# Append classification (face = 1, non-face = 0)
# Append (data, classification) tuple on dataset and return

# End your code (Part 1)
return dataset
```

  - **Part 2 and Part 6**

```
# Begin your code (Part 2)
e = []

for feature in featureVals:
  tempe = 0
  index = 0
  for value in feature:
    value = 1 if value < 0 else 0
    tempe += weights[index]*abs(value-labels[index])
    # Part 6: Bouns
    # Compute the error with mean square error
    # Uncomment it if you need to test it
    # tempe += weights[index]*pow(value-labels[index], 2)    #compute e
    index+=1
  # Part 6: Bouns
  # Compute the error with mean square error
  # Uncomment it if you need to test it
  # tempe = pow(tempe, 0.5)
  e.append(tempe)

bestError = min(e)    #get min e
bestClf = WeakClassifier(features[e.index(bestError)])

# Judge the value of fearureVals[x][y] and chage it to 0 or 1
# Compute the error of each feature and store it in a list (e)
# Assign bestError by extracting minimun value in the list (e)
# Assign bestClf with calling class WeakClassifier() and features[bestError's index]

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

- **Part 4**

```python
# Begin your code (Part 4)
with open(dataPath) as t:
  for i in range(0, 2):
    filename, facenum = t.readline().split()
    # print(filename, facenum)
    img = cv2.imread(os.path.join('data/detect', filename))
    for j in range(0, int(facenum)):
      x, y, xplus, yplus = t.readline().split()
      # print(x, y, xplus, yplus)
      crop_img = img[int(y):int(y)+int(yplus), int(x):int(x)+int(xplus)]
      gray_image = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
      final_image = cv2.resize(gray_image, (19, 19))
      result = clf.classify(final_image)
      if result == 1:
        cv2.rectangle(img, (int(x), int(y)), (int(x)+int(xplus), int(y)+int(yplus)), (0, 255, 0), 3)
      else:
        cv2.rectangle(img, (int(x), int(y)), (int(x)+int(xplus), int(y)+int(yplus)), (0, 0, 255), 3)
    cv2.imshow("Detected image", img)
    cv2.waitKey(0)
# Open detextData.txt and read the file with following format:
#------------------------------------------------------------
# file(image)_name number_of_face
# faceN_x faceN_y faceN_width faceN_height
#------------------------------------------------------------
# Read and crop the images with the coordinates
# Convert the image into grayscale and resize the image into 19*19
# Classify the image with clf.classify()
# Draw rectangles on the image with green or red color representing whether the face is detected or not
# Show the results

# End your code (Part 4)
```

Part II. Results & Analysis (12%):

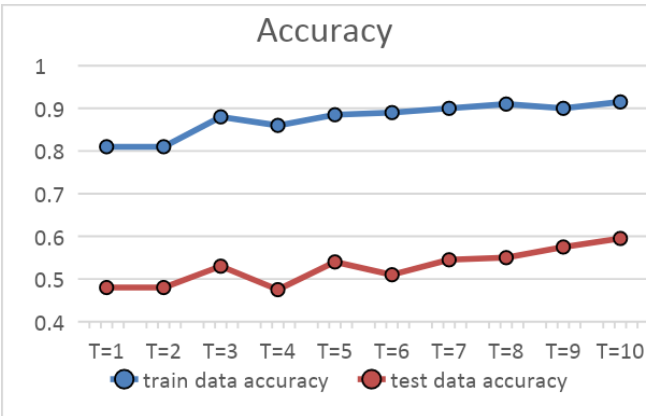
- **Part 2**

```
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(12, 11, 5, 1)], negative regions=[RectangleR
egion(12, 12, 5, 1)]) with accuracy: 72.000000 and alpha: 0.685227
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 4, 1, 1)], negative regions=[RectangleRe
gion(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)
```

200	train data accuracy	test data accuracy
T=1	0.81	0.48
T=2	0.81	0.48
T=3	0.89	0.53
T=4	0.86	0.475
T=5	0.885	0.54
T=6	0.89	0.51
T=7	0.9	0.565
T=8	0.91	0.55
T=9	0.9	0.575
T=10	0.915	0.595

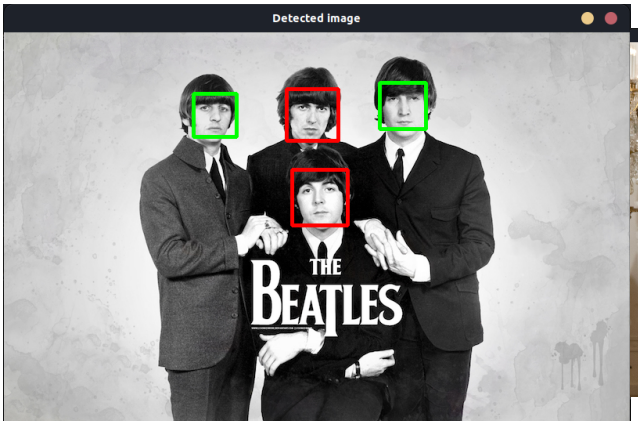


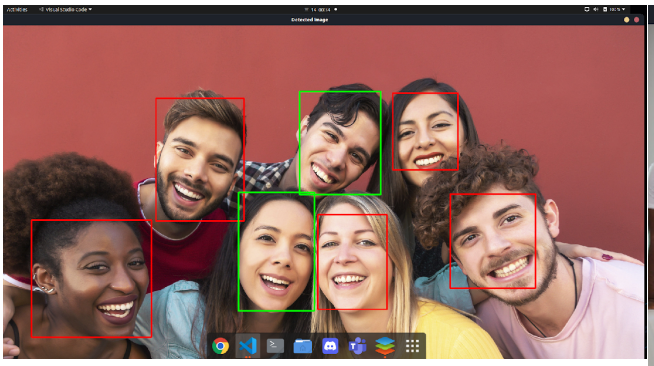
- **Your analysis or observation.**

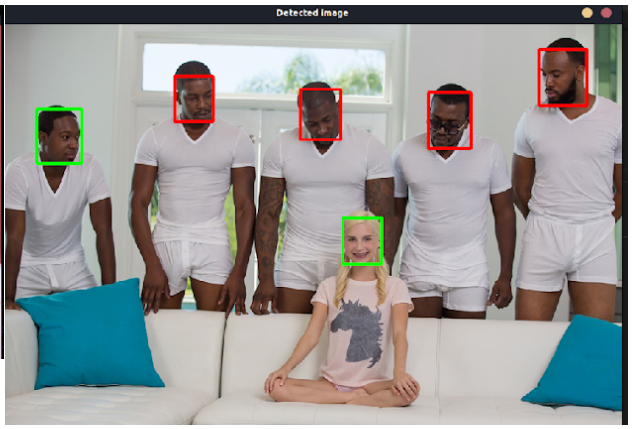
  According to the results, we can see that the accuracy of training data is higher than the accuracy of test data. It shows the difference between two data types.

On the other hand, with different iteration values, the detected results are also different.

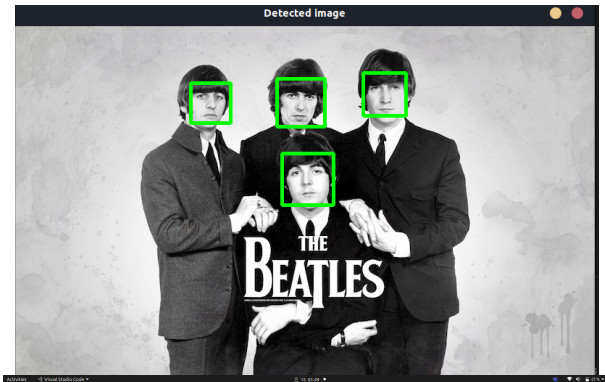
T=10:

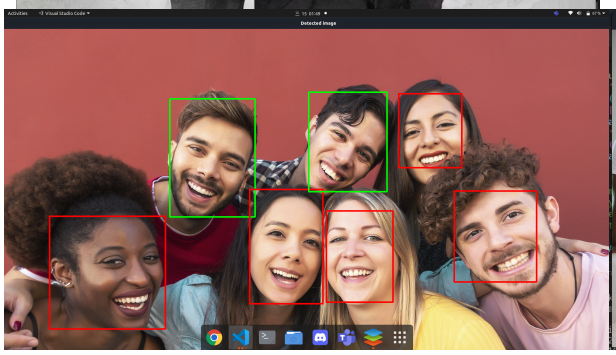


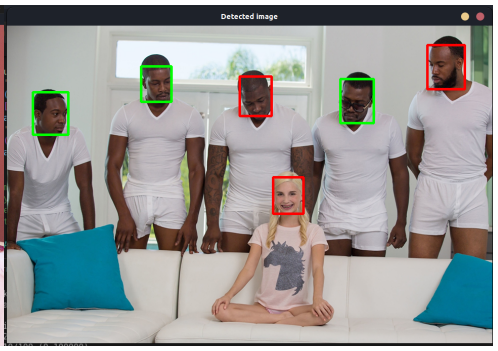




T=1:



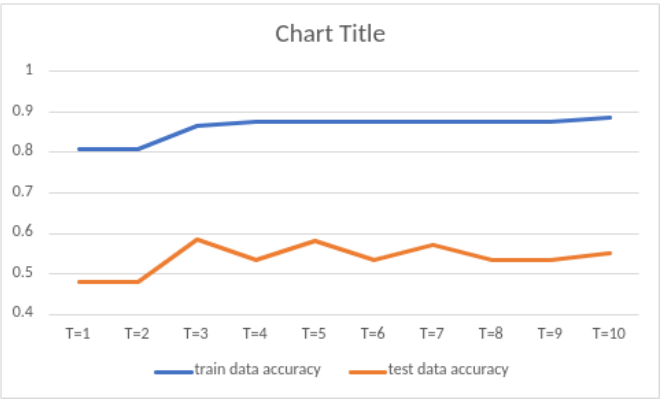




Although the accuracy at T=10 is higher than T=1, the detected results at T=1 are better than T=10, especially for grayscale images.

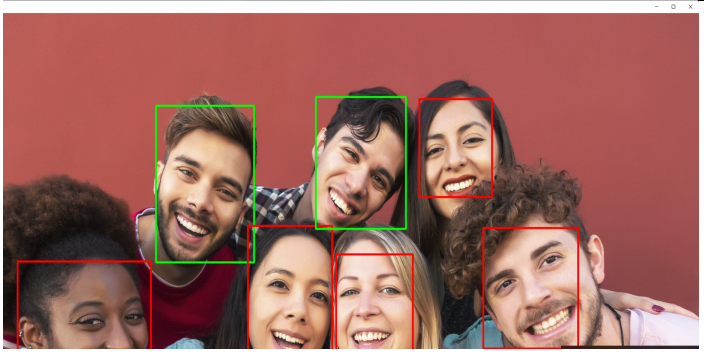
- **Part 6**

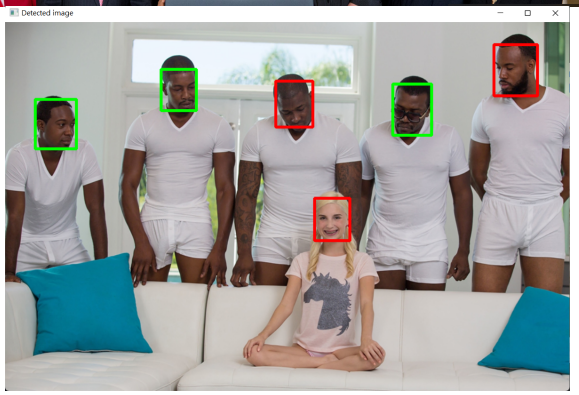
200	train data accuracy	test data accuracy
T=1	0.81	0.48
T=2	0.81	0.48
T=3	0.865	0.585
T=4	0.875	0.535
T=5	0.875	0.58
T=6	0.875	0.535
T=7	0.875	0.57
T=8	0.875	0.535
T=9	0.875	0.535
T=10	0.885	0.55



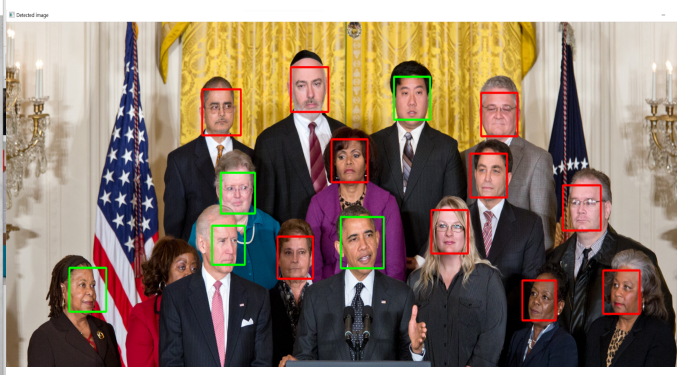
In this part, I change the way I calculate errors to Mean Square Error. According to the above statistics, the accuracy at T=3 and T=4 has risen, while others seem to make no big difference.

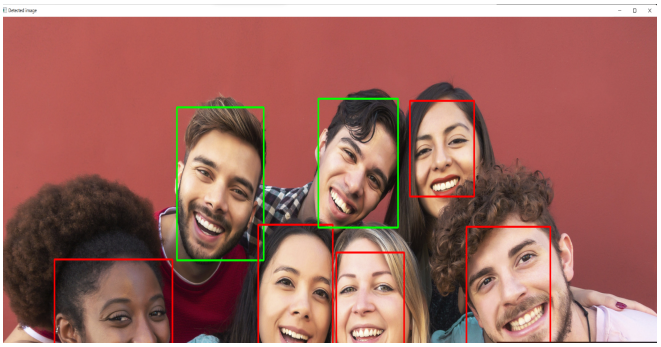
T=1:

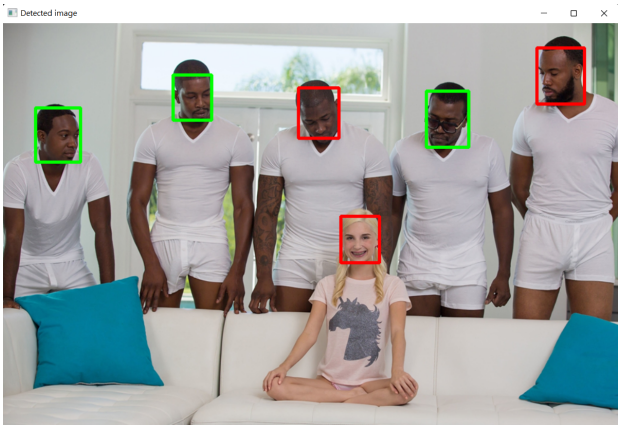




T=10:







The performance of grayscale images has improved in this part. Meanwhile, although the accuracy at T=10 is higher than T=1, the detected results at T=1 are better than T=1.

Part III. Answer the questions (12%):

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

   In the beginning, I did not quite understand the concept of adaboost algorithm. Moreover, time is needed for understanding the structure of the code. With more efforts on lecture's slides and notes, the problems are solved successfully.

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

   The limitations are listed as following:
   - Training time is very slow.
   - Restricted to binary classification.
   - Mostly effective when the face is in frontal view.
   - May be sensitive to very high/low exposure (brightness)
   - High true detection rate, but also high false detection rate.

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

   I tried to improve it by using a different method for calculating errors. However, the difference seems so small that it's not a really good idea. Then, I found an article online showing his implementation of Viola-Jones' algorithm. With his introduction, I realize that training the weakclassifier by checking threshold and polarity might be a good idea. But I don't really have enough time to test it.

   Source

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.**

   In my opinion, I would try to classify whether the face in the images is a front face or a side face. According to the classified result, we get different haar-like features for different situations. It will surely increase the accuracy of detection. But it will also need more time for the computer to classify which face type the face is.