Homework 1: Face Detection

Report Template

Please keep the title of each section and delete examples. Note that pleas e 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.
- As the example, the explanations are inside the code

- Part 1: Load and prepare your dataset

```
# Begin your code (Part 1)
"""
In this part1, we need to load and prepare our dataset
First thing first, direct the datapath to
images' folders and files using the os library imported
read the images using cv2.imread and get the numpy.ndarray of the image
store the numpy array of images to dataset by appending to list

Results:
Numpy array of images returned to main.py to load the images

"""
dataset = []
sets = os.listdir(dataPath)
for folders in sets:
    # print(folders)
    if folders == 'non-face':
        nffile = os.path.join(dataPath, 'non-face')
        nonface = os.listdir(nffile)
        for i in nonface:                              # i as the pgm filename
            nfdata = cv2.imread(os.path.join(nffile, i), 0)   # data as ndarray file type
            dataset.append((nfdata, 0))
    elif folders == 'face':
        ffile = os.path.join(dataPath, 'face')
        face = os.listdir(ffile)
        for i in face:
            fdata = cv2.imread(os.path.join(ffile, i), 0)     #fdata -> numpy aray of m,n representing the image
            dataset.append((fdata, 1)) # no .shape as results is numpyarray with m n dimension (diganti)

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

- Part 2: Implement Adaboost algorithm

```
# Begin your code (Part 2)
"""
In this part, we need to select the best weak classifier
The equation of Adaboost taught in the class is very crucial for this part
The understanding of it are needed to continue the code given
As well as understanding this whole code are needed because
the variables are used to continue the code too
Like featureVals served as table of numbers, features is to use the HaarFeature function
iis contains all the images  and labels is for if its face or nonface (1 or 0)
We also need to modify and added a few lines in other function like train function for the features
Apply the WeakClassifier to all the features then classify each line of the results
We need to count the error based on the variables we have
and find the lowest error from each features, which is going to be the best error
Then, we found the best weak classifier too

Results:
Best classifier and best error returned to main.py and if it's run,
will show that the classifier is being trained for each T from 1 to 10
as well as the accuracy and other information about the classifier

"""
bestClf, bestError = None, float('inf')
classifiers=[]
for i in features:
    classifiers.append(WeakClassifier(i))
for clf in classifiers:
    error = 0
    for img, lbl, weight in zip(iis, labels, weights):
        error += weight * abs(clf.classify(img) - lbl)      # as h(x) in slides
    if error < bestError:
        bestClf, bestError = clf, error

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

- Part 3: Additional experiments

In this part, we are told to change the parameter T in the Adaboost algorithm and to compare the corresponding detection performance from 1 to 10. The pictures in the results sections are the performances of T from 1 to 10. Even though the whole process took very long, but it showed me how the classifier is trained for both datasets.

- Part 4: Detect face

```
# Begin your code (Part 4)
"""
Begin this part by reading the datapath given which contains a detectData.txt
  Start to divide between each lines and each spaces
  If that line consists of less than or equal to two things,
  take the first thing which is the image's file name and
  use the cv2.imread to extract the numpy.ndarray of the image
  Else which means the line consists of the coordinates of the faces' coordinates
  save the coordinates and add the width and height to the left top coordinate
Crop each of the faces' images by slicing and
Resize to the wanted size (19*19) using cv2.resize
Convert colorspace from bgr to grayscale using cv2.cvtColor method
Classify the results and if it is classified as face (=1)
use cv2.rectangle and draw a green box, else draw a red box
Last, we show the whole images using cv2.imshow including the rectangles we added

Results:
After trained by training datasets and tested by testing datasets
In this part, we are required to do it on images
and detect if the faces are detected as a face or not
For the pictures given to us, the number of faces detected decreased as the T increases
the same goes for part 5, which is using the pictures we chose ourselves
This might happened because as the T increases,
the learned features are getting more specific (overfitting),
which causes some faces not detected in the end

"""
newlines=[]
f = open(dataPath, "r")
lines = f.readlines()
imgs = []
list_of_boxes = []
wk = {}
```

```
for i in lines:
  newlines.append(i.split('\n')[0])
# print(newlines)
head = ""
wk["pic_name"] = []
for i in newlines:
  x = i.split(' ')
  # print(x)
  if(len(x)<=2):
    head = x[0]         # x[0] contains files' names
    wk[head] = []
    wk["pic_name"].append(x[0])
    wk["numpyarray"+x[0]] = cv2.imread('data/detect/' + x[0])
  else:
    box = [None] * 4
    for j in range (4):
      box[j] = x[j]
      # print(box[j])
    wk[head].append(box)
# print(wk[head])
# print(wk[wk["pic_name"][x]])

numparr = []
numofimgs = len(wk["pic_name"])
for x in range (numofimgs):
  # print("File's name: ", wk["pic_name"][x])
  # print("Coordinates: ")
  # for i in wk[wk["pic_name"][x]]:
    # print(i)
  # print("Image's numpy array: ")
  # print(wk["numpyarray"+wk["pic_name"][x]])
  numparr.append(wk["numpyarray"+wk["pic_name"][x]])
  # print(numparr[x])
```

```python
faces = []
for i in range (numofimgs):
  for j in wk[wk["pic_name"][i]]:
    x1 = 0
    y1 = 0
    x2 = 0
    y2 = 0
    x1 = int(j[0])              # coordinates of left top in x
    y1 = int(j[1])              # coordinates of left top in y
    x2 = int(j[0]) + int(j[2])  # the width added to produce the right bottom x
    y2 = int(j[1]) + int(j[3])  # the height added to produce the right bottom y
    #crop -> resize -> change to grayscale -> give cls.classify()
    crop = numparr[i][y1:y2, x1:x2]          # [y:y+h,x:x+w] and crop is nparray type
    resized = cv2.resize(crop, (19, 19), interpolation = cv2.INTER_AREA)
    gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)        # change from BGR to grayscale
    # print(type(gray))
    faces.append(gray)

for i in range (numofimgs):
  for j,face in zip(wk[wk["pic_name"][i]],faces):
    if(clf.classify(face)):                  # the face images(hasil potongan) are classified by clf.classify
      ni = cv2.rectangle(numparr[i], (int(j[0]), int(j[1])), (int(j[0])+int(j[2]), int(j[1])+int(j[3])), (0,255,0), 2)
    else:
      ni = cv2.rectangle(numparr[i], (int(j[0]), int(j[1])), (int(j[0])+int(j[2]), int(j[1])+int(j[3])), (0,0,255), 2)

  gray2 = cv2.cvtColor(numparr[i], cv2.COLOR_BGR2RGB)
  fig, ax = plt.subplots(1, 1)
  ax.axis('off')
  ax.imshow(gray2)
  plt.show()

  cv2.waitKey(0)
  cv2.destroyAllWindows()

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

- Part 5: Test classifier on your own images

In this part, I chose two images with faces and made a text file with coordinates like the example files. The classifier can classify face and non-face, but the number of faces classified in the end is decreasing just like the example images given to us.

Part II. Results & Analysis (12%):

- Please screenshot the results.

- T = 1

```
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, 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)
```

- T = 2

```
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 accurac
y: 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)
```

- T = 3

```
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 acc
uracy: 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)
```

- T = 4

```
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 accur
acy: 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)
```

- T = 5

```
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 accura
cy: 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)
```

- T = 6

```
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 accurac
y: 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)
```

- T = 7

```
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 accur
acy: 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)
```

- T = 8

```
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 acc
uracy: 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)
```

- T = 9

```
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 accura
cy: 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)
```

- T = 10

```
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=[RectangleReg
ion(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)
```

For faces detection:

- T = 1 (4/4 and 14/15 detected as face)
- T = 2 (4/4 and 14/15 detected as face)
- T = 3 (2/4 and 6/15 detected as face)
- T = 4 (4/4 and 10/15 detected as face)
- T = 5 (3/4 and 5/15 detected as face)
- T = 6 (3/4 and 5/15 detected as face)
- T = 7 (3/4 and 4/15 detected as face)
- T = 8 (3/4 and 4/15 detected as face)

- T = 9 (3/4 and 4/15 detected as face)
- T = 10 (3/4 and 4/15 detected as face)

The images given in the detect folder for us to detect the faces are detected as faces or not works quite well at the start with almost all faces are detected. However, as it is trained, the number of faces detected also decreasing sharply.

- myImages T = 1 (3/4 and 4/5 detected as face)

- myImages T = 10 (2/4 and 2/5 detected as face)

- Your analysis or observation.
  Please discuss the performance difference between the training and testing dataset, and present the results using a table or chart as follows.

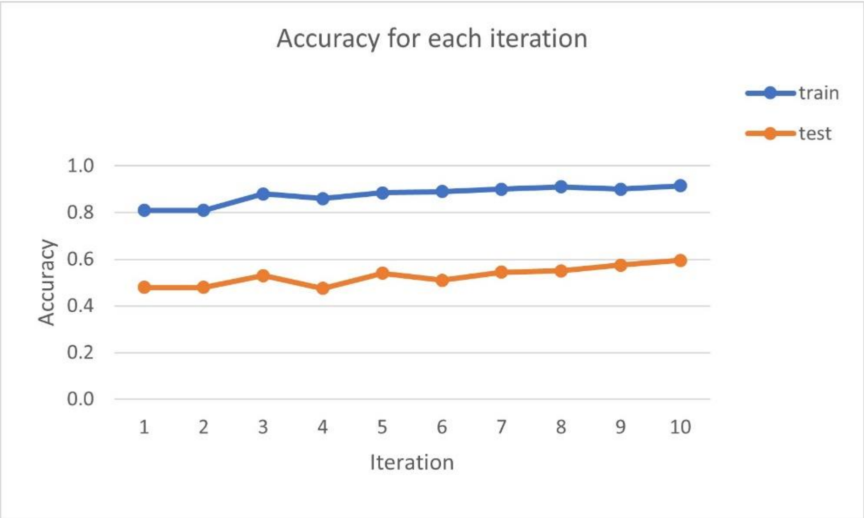
Discussion:

Please The performance difference between the training and testing dataset is quite big. The training data accuracy is getting more accurate every time it’s trained from T = 1 to T = 10, as we can see from the chart it went from 81% to 91.5%. The performance difference betwe en these two datasets is that big because it is trained by using the training dataset, which the data inside the dataset might still have features in common. However, when it is applied to testing dataset with some different features, the accuracy will go down. The accuracy still goes up from 48% to 59.5%, which means the testing dataset is also getting trained each time from T = 1 to T = 10.

Table:

200張	train data accurac	test data accuracy
method 1 t=1	81.0%	48.0%
method 1 t=2	81.0%	48.0%
method 1 t=3	88.0%	53.0%
method 1 t=4	86.0%	47.5%
method 1 t=5	88.5%	54.0%
method 1 t=6	89.0%	51.0%
method 1 t=7	90.0%	54.5%
method 1 t=8	91.0%	55.0%
method 1 t=9	90.0%	57.5%
method 1 t=10	91.5%	59.5%

Chart:



Part III. Answer the questions (12%):

1. Please describe a problem you encountered and how you solved it.
For this first assignment given to us, I can't deny that I faced a lot a lot of problems. One of the hardest problems I faced in this assignment is the part two, the adaboost part. We are required to understand every lines of equations in the slides and the variables in the codes to imp

lement this part. I spent and stucked in this part for a long time trying to figure out how to implement the Adaboost for feature selection in the adaboost.py as well as how to choose the best weak classifier and count the best error. I did searched in google and looking everywhere to find resources and references for any clue that can help me, read the code and slides several times to understand it.

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

One of the most obvious limitations of this algorithm is that the training time is very slow. Every time we run our code, we need to wait for a few minutes to get the results and it takes us a long time to do it. Other limitations might be the face needed to be seen clearly and being frontal for it to be detected. If some of the face features are covered, the Viola-Jones' algorithm might fail to detect the face. For example, if people have bangs that cover their eyebrows or have lighter shade of eyebrows, it will be harder to be detected as a face.

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

I do think the most important thing needed is increasing dataset and changing parameter T like how it is said in the question. We can broaden the learned and trained features from the training dataset. As it was said in the previous questions about Viola-Jones' algorithm limitations of detecting other than frontal faces, like facing sideways, upwards and downward. I think this is one of the things that we can do to improve the accuracy of the algorithm by taking the available Haar features of the sideways face.

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.

I thought of one possible face detection method which will still use Haar-features and use it to detect the shades between the face and non-face. However, I might add or make some more specific lines of code which can make the unclear faces or not frontal faces detected too. In this case, I think most faces that are not detected before will be detected, but the disadvantage is some non-face objects might get detected as face too.