交大資工
110550062 朱致伶

Homework 1: Face Detection

I. Implementation (6%):

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

Part 1 (dataset.py):

1. Declare a list to store data, which data in (gray scale 19x19 pixels image, label) format
2. Use 2 for loops to run through the 'train/face' folder and 'train/non-face' folder, os.listdir function can traverse all folder under the input datapath
3. The cv2.imread function turn image into numpy array format, with parameter (image_path, flag) , flag is set to 0 responding to cv2.imread_GRAYSCALE
4. os.path.join function with parameter (folder_path, filename), connect the two string together, return a image file datapath
5. Then, pass the pre-processed image and label as a tuple element into the dataset list

```
# Begin your code (Part 1)
    dataset = []
    for filename in os.listdir(dataPath+"/face"):
        file = cv2.imread(os.path.join(dataPath + "/face", filename), 0)
        if file is not None:
            image = (file, 1)
            dataset.append(image)
    for filename in os.listdir(dataPath+"/non-face"):
        file = cv2.imread(os.path.join(dataPath + "/non-face", filename), 0)
        if file is not None:
            image = (file, 0)
            dataset.append(image)
    return dataset
# End your code (Part 1)
```

Part 2 (adaboost.py – selectBest):

1. Declare a list errors, store error for each feature
2. Outer for loop run through all features, as to inner for loop, it runs through all integral images
3. Declare a WeakClassifier class for each feature, and classify all iis values to assign value to hvalue variable
4. Error for each feature is the sum of the product of the feature weight and the absolute value of (hvalue – label), append the calculated error to the errors list
5. Take the minimum value in the errors list as the bestError, and the corresponding classifier as the bestClassifier

```
# Begin your code (Part 2)
      errors = []
      for i in range(len(features)):
         error = 0
         hclass = WeakClassifier(features[i])
         for j in range(len(iis)):
            hvalue = hclass.classify(iis[j])
            error += weights[j] * abs( hvalue - labels[j])
         # print("error: ", error)
         errors.append(error)
      bestError = min(errors)
      idxbestClf = errors.index(bestError)
      bestClf = WeakClassifier(feature = features[idxbestClf])
      return bestClf, bestError
 # End your code (Part 2)
```

Part 4 (detection.py):

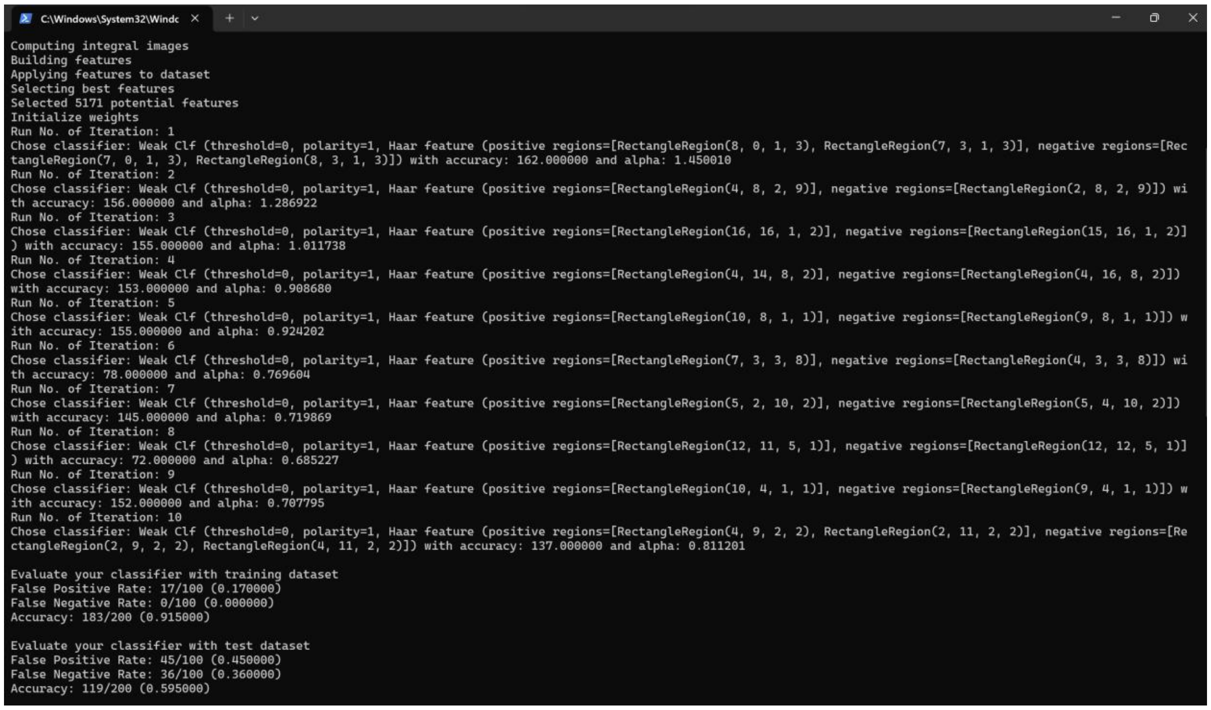
1. Open the dataDetect.txt file.
   Since the image variable used to store the image run later can’t be none, declare a black 512x512 pixels image with np.zero function
2. Read the line in dataDetect.txt file, one line once with .read.splitlines() function.
   Split the line with white space and store the elements into lst
3. If the number of elements in lst is 2, then read the image by the image name string store in the first element. Use os.walk() function to traverse through the detect folder, and check if the image is in the folder read the image with cv2.imread and os.path.join mentioned above.
4. If the number of elements in lst is 4, the 4 elements corresponds to (x, y, w, h).
   Delacre and assign startpoint, endpoint, thichness.
5. Crop and resize the face image to gray scale 19x19 pixels image, remind the face image is a copy of the cropped image, iin order not to affect the original image
6. Classify whether the gray face image is a face or not, assign the color green if is face, otherwise assign it red
7. Draw the rectangle on the original image with cv2.recctangle function
8. Show the original image with rectangle on it

```
# Begin your code (Part 4)
   # raise NotImplementedError("To be implemented")
   with open(dataPath) as dtxt:
      image = np.zeros(shape=[512, 512], dtype=np.uint8)
      imgname = ""
      for line in dtxt.read().splitlines():
         lst = line.split(" ")
         if len(lst) == 2:
            imgname = lst[0]
            for root, dirs, files in os.walk(str(Path(dataPath).parent)):
```

```
        elif len(lst) == 4:
          x, y, w, h = int(lst[0]), int(lst[1]), int(lst[2]), int(lst[3])
          startpoint = (x, y)
          endpoint = (x+w, y+h)
          thickness = 3
          crop = image[y:y+h, x:x+w].copy()
          face = cv2.resize(crop, (19, 19))
          gray = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
          if clf.classify(gray):
            color = (0, 255, 0)  #  print("Is Face!")
          else:
            color = (0, 0, 255)  #  print("Not face!")
          image = cv2.rectangle(image, startpoint, endpoint, color, thickness)
          cv2.imshow(imgname, image)
          cv2.waitKey(0)
      cv2.destroyAllWindows()
    dtxt.close()
    # End your code (Part 4)
```

II. Results & Analysis (Screenshot the result) (12%):

- **<u>Main Result:</u>**

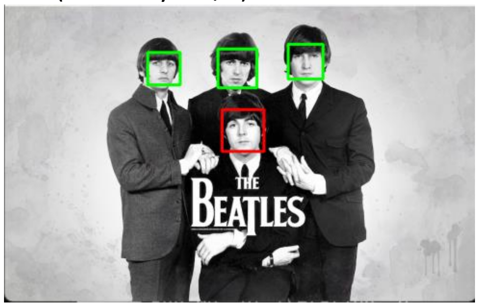


```
Computing integral images
Building features
Applying features to dataset
Selecting best features
Selected 5171 potential features
Initialize weights
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), RectangleRegion(7, 3, 1, 3)], negative regions=[Rec
tangleRegion(7, 0, 1, 3), RectangleRegion(8, 3, 1, 3)]) with accuracy: 162.000000 and alpha: 1.450010
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)]) wi
th accuracy: 156.000000 and alpha: 1.286922
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 accuracy: 155.000000 and alpha: 1.011738
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 accuracy: 153.000000 and alpha: 0.908680
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)]) w
ith accuracy: 155.000000 and alpha: 0.924202
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)]) wi
th accuracy: 78.000000 and alpha: 0.769604
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 accuracy: 145.000000 and alpha: 0.719869
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 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=[RectangleRegion(9, 4, 1, 1)]) w
ith 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=[Re
ctangleRegion(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)
```

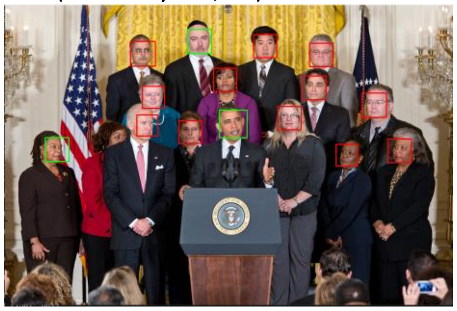
交大資工
110550062 朱致伶

- **Face Classifier on Detect Data:**

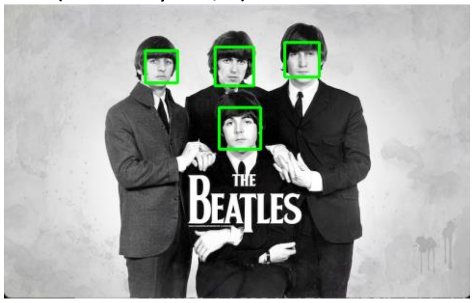
T=5 (Accuracy = 3/4)



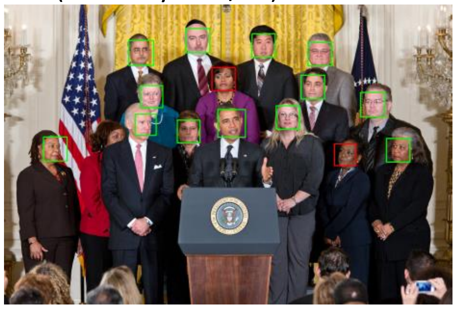
T=9 (Accuracy = 3/15)



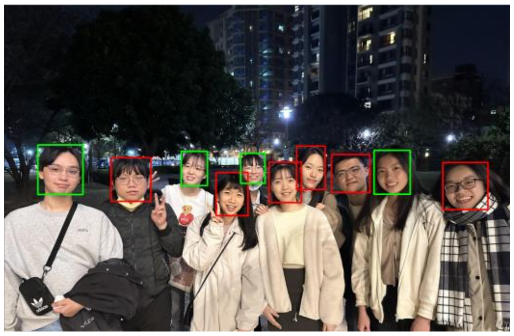
T=4 (Accuracy = 4/4)

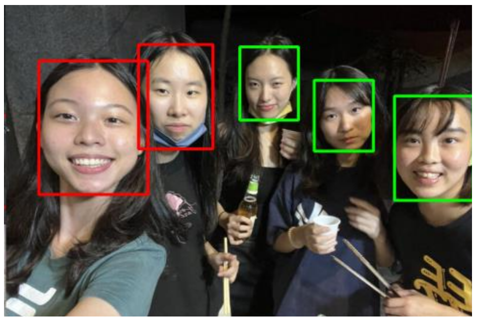


T=2 (Accuracy = 13/15)

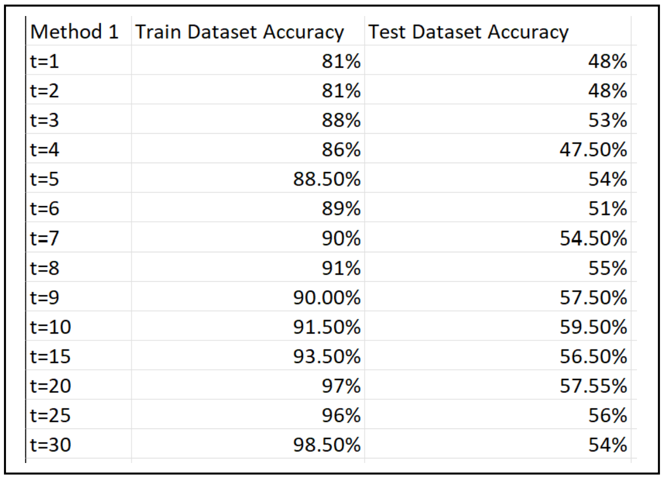


- **Face Classifier on My Own Image (T=6):**





- **Performance of different T value:**



Method 1	Train Dataset Accuracy	Test Dataset Accuracy
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	90.00%	57.50%
t=10	91.50%	59.50%
t=15	93.50%	56.50%
t=20	97%	57.55%
t=25	96%	56%
t=30	98.50%	54%

交大資工
110550062 朱致伶

Analysis / Observation:

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

From the previous result, we can see that training accuracy is higher than testing accuracy, whatever the T value is. As T increases in the range of 1 to 8, the training and testing accuracy both rise, in other words, T values and the two accuracy are positive correlated.

However, when T value is larger than 8, the training accuracy rises. Meanwhile, the testing accuracy stays around 58%, even decreases as T value gets over 20. We may guess the trained model is overfitting when T value gets over 10.

Therefore, the classifying result of my own testing data and given detected data is more precise when T=4 than T=10. The trained model of Viola-Jones' algorithm has the best performance on my own images at T=6.

III. Answer the questions (12%):

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

Since I am not familiar with extracting the line messages from a .txt file, I got stuck in how to read detectdata.txt at part 4 of the coding session for a while. The key point of extracting the message is to classify the line message into two types by the number of elements in the line, one including image name and face number, the other stores the rectangle position of the faces. Also, the function of jumping from the present file to parent folder takes me a while to figure how to operate.

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

- Restricted to binary classification.
- Mostly effective when face is in frontal view, bad at detecting tilted and turned face, due to the Haar-like feature format.
- Since the training dataset is turn to gray scale, the model may be sensitive to extremely high/low exposure (brightness).
- High false positive rate, but also high false negative rate.

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

Application of Composite Features to Reduce False Detection Rate

These composite features are an adaptation of the original Haar-like rectangular features. Although the basic Haar-like features keep a high flexibility for the model to learn classifying face and non-face, it makes the training process more unpredictable than with a composite feature of detailed eye, lip, nose feature.

Instead of using just one rectangular feature, we can utilize composite features, or vectors of simple features as the weak learner in their algorithm. Composite features with detailed facial components can reduce the difficulties of the training process and the false rate of the result.

4. 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

交大資工
110550062 朱致伶

The classical Histogram of Gradients (HoG) feature, usually trained with support vector machine, is also a common supervised algorithm used on face detection.
When working on face detection, HoG maintains good invariance to the geometric and optical deformation of the image, since HoG operates on the local grid unit of the image, instead of bigger pattern. However, it takes a great amount of training time for model to learn one feature, compared with Viola-Jones' algorithm. Also, since its main concept based on gradient, the training model is sensitive with the noise of the image.

Bonus:

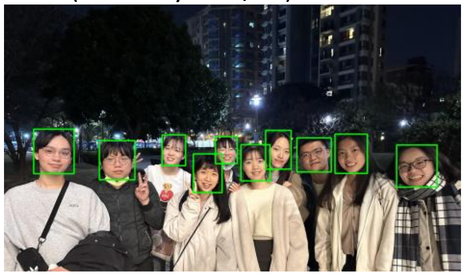
- **Method Introduction:**

To implement the "selectBest" function in another method, I come up with an idea that slightly modify the classify function of WeakClassifier class. Instead of assigning the h value with only two values, 0 and 1, when classifying the integral image, we can assign a value between 0 and 1. With printing the value of (self.polarity*self.feature.computeFeature(x)), I observed the value is in the range between -15000 and 15000, therefore (self.polarity*self.feature.computeFeature(x)) is divided by 15000.

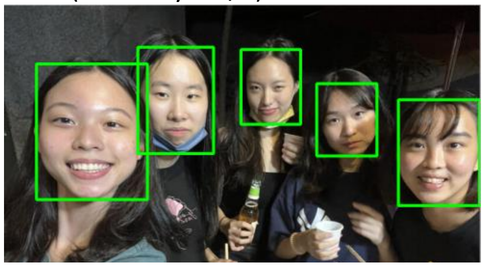
Assigning the h value in the range of 0 to 1, allows h value to represent the probability of the integral image to be a face or non-face. With the linear transformation in the code below, we know as the h value near 1, the integral image is highly classified as face, and for h value near 0, it is highly classified as non-face. The difference between the method I proposed and the original one, is that h value becomes continuous rather than discrete. The method makes the h value to indicate the degree of the face and non-face, showing stronger relationship of the h value and whether it is a face, compared with original method only assigns 0 and 1. Below is the modified code and result:

```
# Only modify classifier.py by adding function classify2, function selectBest is unchanged
def classify2(self, x):
    ori = - ( self.polarity * self.feature.computeFeature(x) ) / 15000
    h  = ori * 0.5 + 0.5
    return  h
```

T=10 (Accuracy = 15/15)

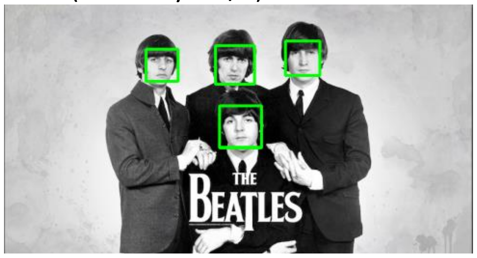


T=10 (Accuracy = 5/5)

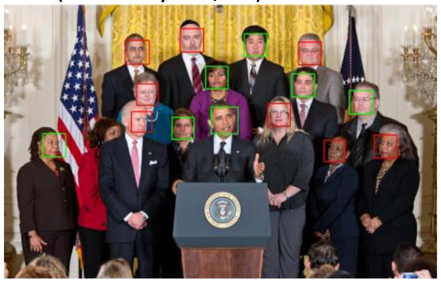


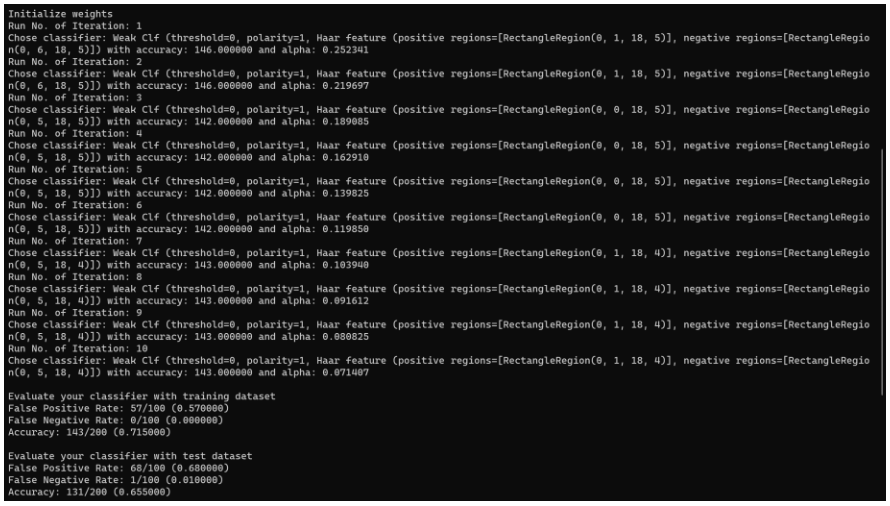
交大資工
110550062 朱致伶

T=10 (Accuracy = 4/4)

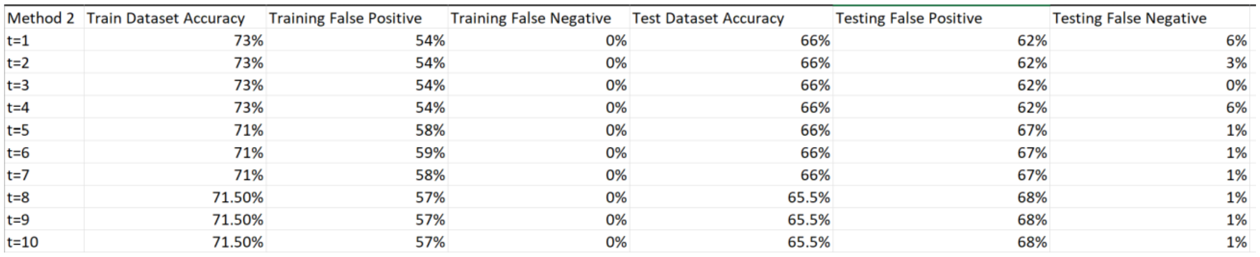


T=2 (Accuracy = 7/10)





```
Initialize weights
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 5)], negative regions=[RectangleRegio
n(0, 6, 18, 5)]) with accuracy: 146.000000 and alpha: 0.252341
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 5)], negative regions=[RectangleRegio
n(0, 6, 18, 5)]) with accuracy: 146.000000 and alpha: 0.219697
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 0, 18, 5)], negative regions=[RectangleRegio
n(0, 5, 18, 5)]) with accuracy: 142.000000 and alpha: 0.189085
Run No. of Iteration: 4
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 0, 18, 5)], negative regions=[RectangleRegio
n(0, 5, 18, 5)]) with accuracy: 142.000000 and alpha: 0.162910
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 0, 18, 5)], negative regions=[RectangleRegio
n(0, 5, 18, 5)]) with accuracy: 142.000000 and alpha: 0.139825
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 0, 18, 5)], negative regions=[RectangleRegio
n(0, 5, 18, 5)]) with accuracy: 142.000000 and alpha: 0.119850
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 4)], negative regions=[RectangleRegio
n(0, 5, 18, 4)]) with accuracy: 143.000000 and alpha: 0.103940
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 4)], negative regions=[RectangleRegio
n(0, 5, 18, 4)]) with accuracy: 143.000000 and alpha: 0.091612
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 4)], negative regions=[RectangleRegio
n(0, 5, 18, 4)]) with accuracy: 143.000000 and alpha: 0.080825
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 1, 18, 4)], negative regions=[RectangleRegio
n(0, 5, 18, 4)]) with accuracy: 143.000000 and alpha: 0.071407

Evaluate your classifier with training dataset
False Positive Rate: 57/100 (0.570000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 143/200 (0.715000)

Evaluate your classifier with test dataset
False Positive Rate: 68/100 (0.680000)
False Negative Rate: 1/100 (0.010000)
Accuracy: 131/200 (0.655000)
```



Method 2	Train Dataset Accuracy	Training False Positive	Training False Negative	Test Dataset Accuracy	Testing False Positive	Testing False Negative
t=1	73%	54%	0%	66%	62%	6%
t=2	73%	54%	0%	66%	62%	3%
t=3	73%	54%	0%	66%	62%	0%
t=4	73%	54%	0%	66%	62%	6%
t=5	71%	58%	0%	66%	67%	1%
t=6	71%	59%	0%	66%	67%	1%
t=7	71%	58%	0%	66%	67%	1%
t=8	71.50%	57%	0%	65.5%	68%	1%
t=9	71.50%	57%	0%	65.5%	68%	1%
t=10	71.50%	57%	0%	65.5%	68%	1%

- **Analysis / Observation:**

  Same as the original method, the new method has training accuracy higher than testing accuracy, whatever the T value is. Surprisingly, although the method I proposed has lower training accuracy compared with the original one, it has higher testing accuracy. It is also a surprise that the probability method I proposed has a better result on random test image of my own.

  However, the training and testing accuracy seems to have weak relation with T value. As T value increases, the training and testing accuracy slightly changes.