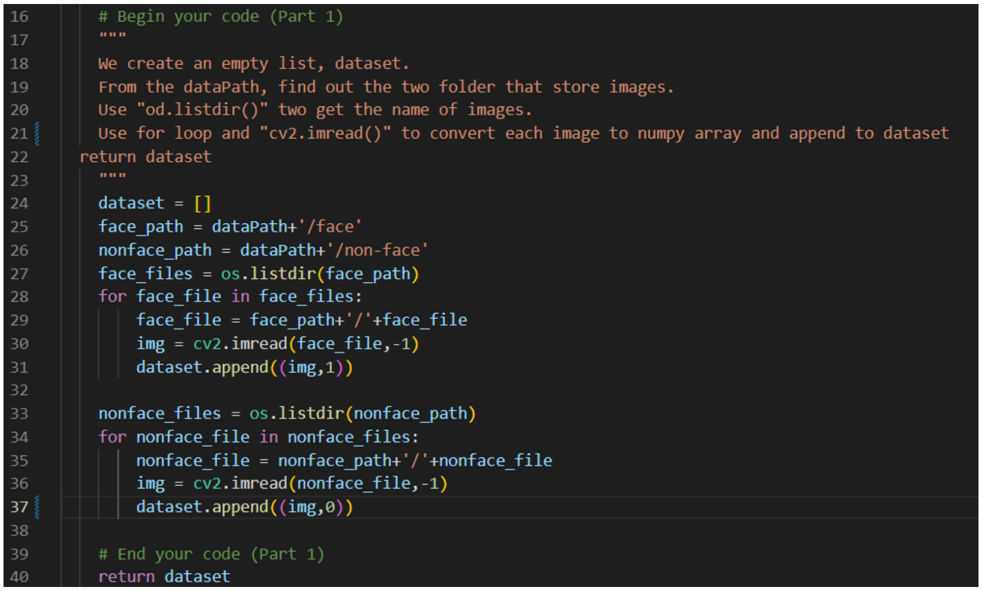
110550126 曾家祐

Homework 1: Face Detection

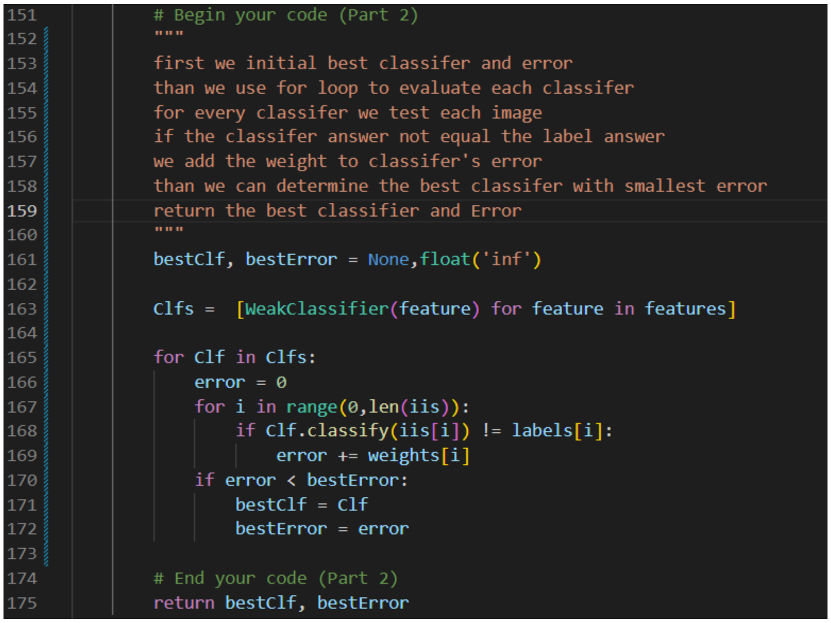
Part I. Implementation :

- **Part 1: Load and prepare your dataset: dataset.py -> loadImages()**



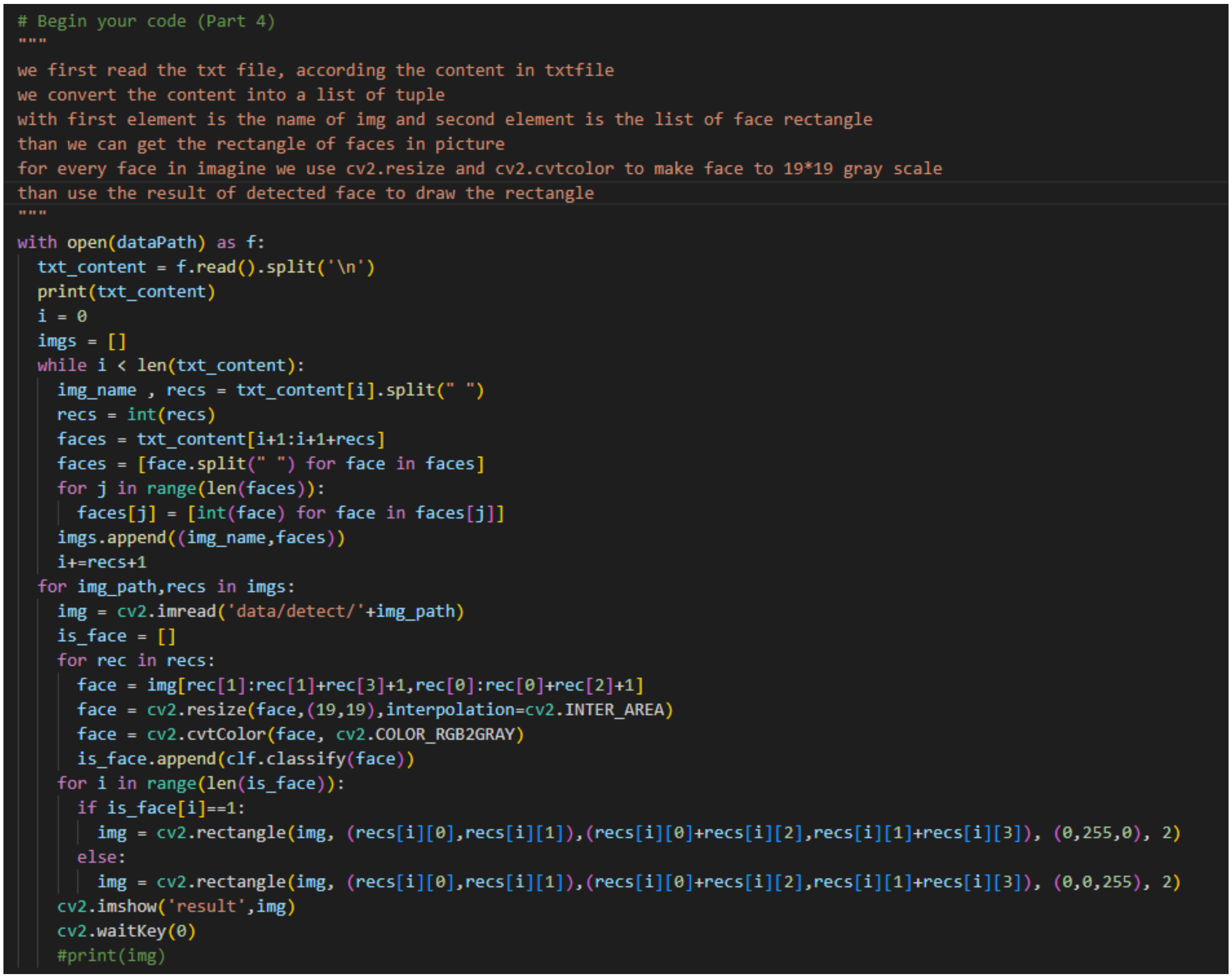
```
    # Begin your code (Part 1)
    """
    We create an empty list, dataset.
    From the dataPath, find out the two folder that store images.
    Use "od.listdir()" two get the name of images.
    Use for loop and "cv2.imread()" to convert each image to numpy array and append to dataset
  return dataset
    """
    dataset = []
    face_path = dataPath+'/face'
    nonface_path = dataPath+'/non-face'
    face_files = os.listdir(face_path)
    for face_file in face_files:
        face_file = face_path+'/'+face_file
        img = cv2.imread(face_file,-1)
        dataset.append((img,1))

    nonface_files = os.listdir(nonface_path)
    for nonface_file in nonface_files:
        nonface_file = nonface_path+'/'+nonface_file
        img = cv2.imread(nonface_file,-1)
        dataset.append((img,0))

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

- **Part 2: Implement Adaboost algorithm: adaboost.py-> selectbest()**



```
        # Begin your code (Part 2)
        """
        first we initial best classifer and error
        than we use for loop to evaluate each classifer
        for every classifer we test each image
        if the classifer answer not equal the label answer
        we add the weight to classifer's error
        than we can determine the best classifer with smallest error
        return the best classifier and Error
        """
        bestClf, bestError = None,float('inf')

        Clfs =  [WeakClassifier(feature) for feature in features]

        for Clf in Clfs:
            error = 0
            for i in range(0,len(iis)):
                if Clf.classify(iis[i]) != labels[i]:
                    error += weights[i]
            if error < bestError:
                bestClf = Clf
                bestError = error

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

- **Part 4: Detect face: Detection.py-> detect()**

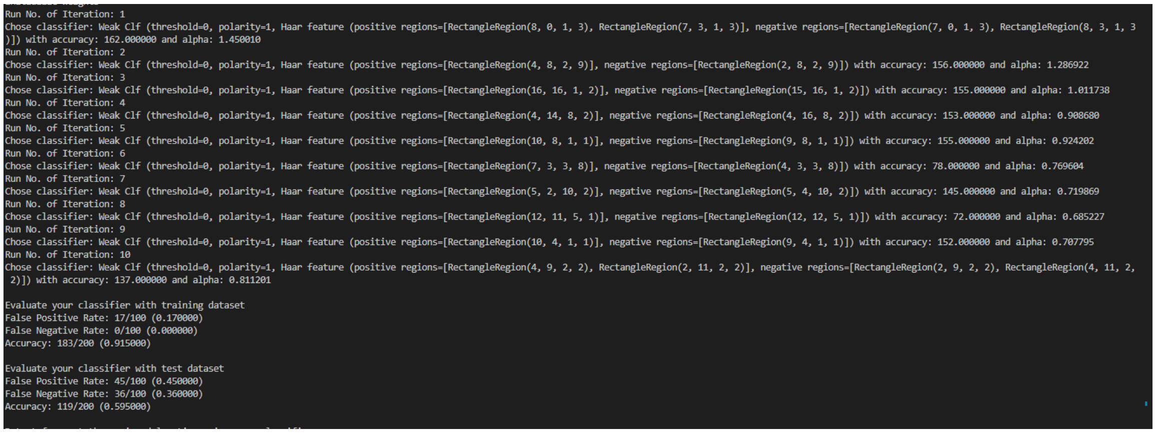


```python
# Begin your code (Part 4)
"""
we first read the txt file, according the content in txtfile
we convert the content into a list of tuple
with first element is the name of img and second element is the list of face rectangle
than we can get the rectangle of faces in picture
for every face in imagine we use cv2.resize and cv2.cvtcolor to make face to 19*19 gray scale
than use the result of detected face to draw the rectangle
"""
with open(dataPath) as f:
  txt_content = f.read().split('\n')
  print(txt_content)
  i = 0
  imgs = []
  while i < len(txt_content):
    img_name , recs = txt_content[i].split(" ")
    recs = int(recs)
    faces = txt_content[i+1:i+1+recs]
    faces = [face.split(" ") for face in faces]
    for j in range(len(faces)):
      faces[j] = [int(face) for face in faces[j]]
    imgs.append((img_name,faces))
    i+=recs+1
  for img_path,recs in imgs:
    img = cv2.imread('data/detect/'+img_path)
    is_face = []
    for rec in recs:
      face = img[rec[1]:rec[1]+rec[3]+1,rec[0]:rec[0]+rec[2]+1]
      face = cv2.resize(face,(19,19),interpolation=cv2.INTER_AREA)
      face = cv2.cvtColor(face, cv2.COLOR_RGB2GRAY)
      is_face.append(clf.classify(face))
    for i in range(len(is_face)):
      if is_face[i]==1:
        img = cv2.rectangle(img, (recs[i][0],recs[i][1]),(recs[i][0]+recs[i][2],recs[i][1]+recs[i][3]), (0,255,0), 2)
      else:
        img = cv2.rectangle(img, (recs[i][0],recs[i][1]),(recs[i][0]+recs[i][2],recs[i][1]+recs[i][3]), (0,0,255), 2)
    cv2.imshow('result',img)
    cv2.waitKey(0)
    #print(img)
```

Part II. Results & Analysis :

- **Part 3: Additional experiments**

Method 1 with threshold = 0, polarity = 1

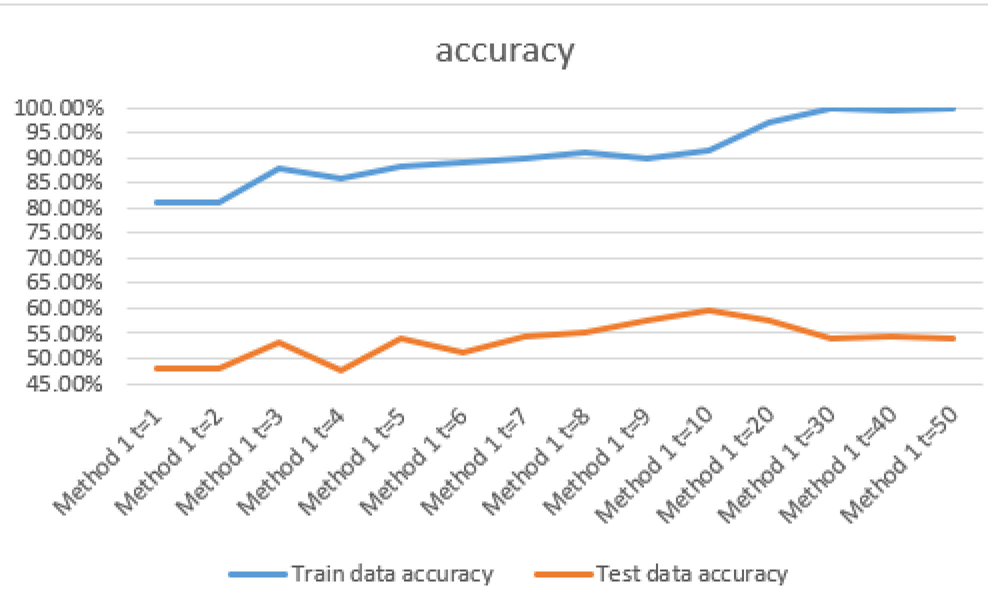


```
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), RectangleRegion(7, 3, 1, 3)], negative regions=[RectangleRegion(7, 0, 1, 3), RectangleRegion(8, 3, 1, 3
)]) with accuracy: 162.000000 and alpha: 1.450010
Run No. 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 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)]) with 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)]) with 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)]) 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)
```

Training model with T = 10

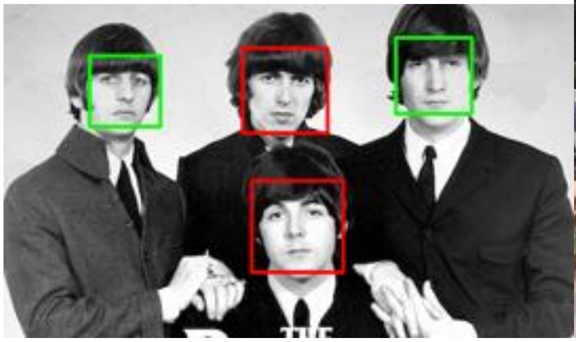
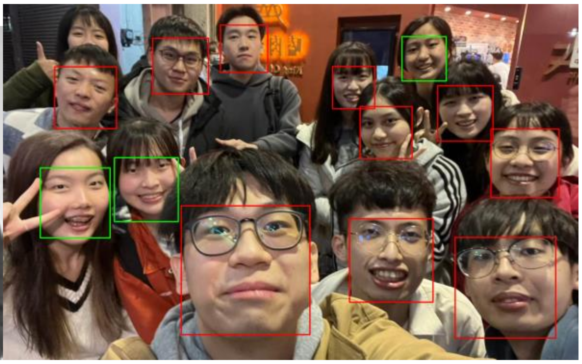
The table below is the result of our training model with t from 1~10,20,30,40,50. We can see that when t increase train data accuracy increasing , but t increase and have peak value at t = 10

200 張	Train data accuracy	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%
Method 1 t=20	97.0%	57.5%
Method 1 t=30	100.0%	54.0%
Method 1 t=40	99.5%	54.5%
Method 1 t=50	100.0%	54.0%



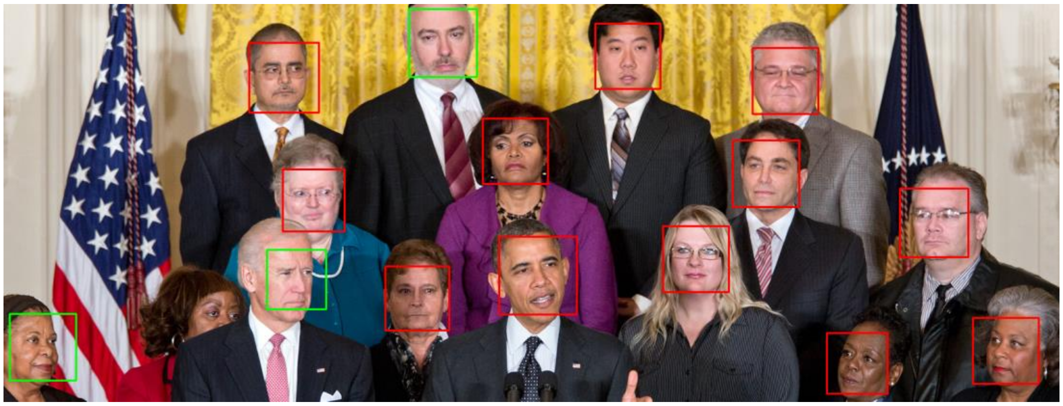
However when testing using image, the result with t = 2 is more higher than t = 10.

- **Part 5: Test classifier on your own images**

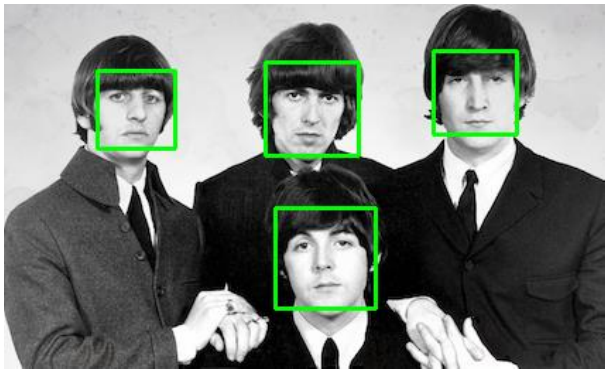
-  

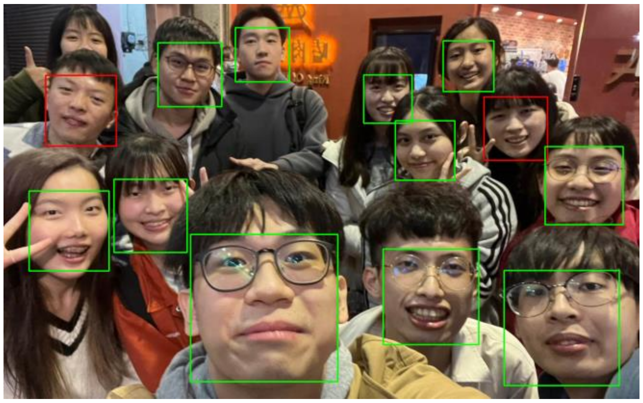
⬆Face detection “the-beatles.jpg” Face detection “my_pic.jpg” ⬆

⬇Face detection “p110912sh-0083.jpg”



With value t = 10

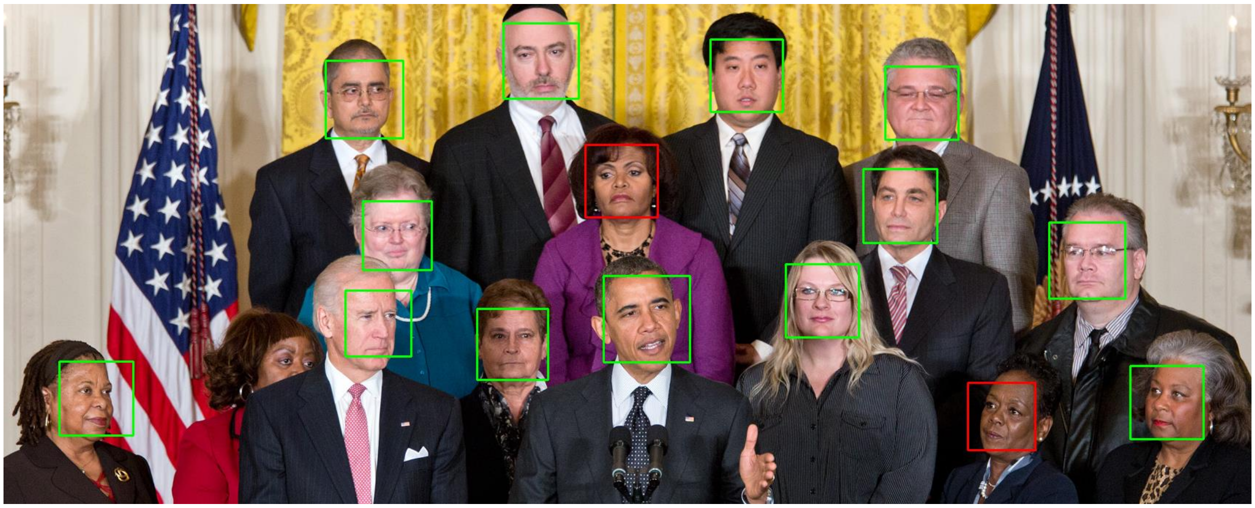




⬆Face detection “the-beatles.jpg”

Face detection “my_pic.jpg” ⬆

⬇Face detection “p110912sh-0083.jpg”



With value t = 2

- **Part III. Answer the questions**

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

   The problem I encountered is the datatype of each variable and the whole process of the algorithm.

   The scale of the program is much greater than I write before. So I spend a lot of time on reading the code, and take the note about the input, result of each function. Further more I also search the articles Adaboost, Viola-Jones' algorithm to understand the each part of these algorithm and know the whole process of the code.

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

The limitation of Viola-Jones algorithm, is restricted to binary classification tasks, and since Viola-Jones' algorithm is based on weakclassifier the features grow rapidly when the size of image grow. This make the training time grow rapidly to. So the viola-Jones' algorithm is limited by the size of image.

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

We can increase the size of input image (ex: 19*19 ->25*25). When the size of image increase, the features of image increase rapidly and avoid the distortion of image, since our images are much bigger than 19*19 image in training dataset. With more features, Viola-Jones' algorithm can get higher accuracy.

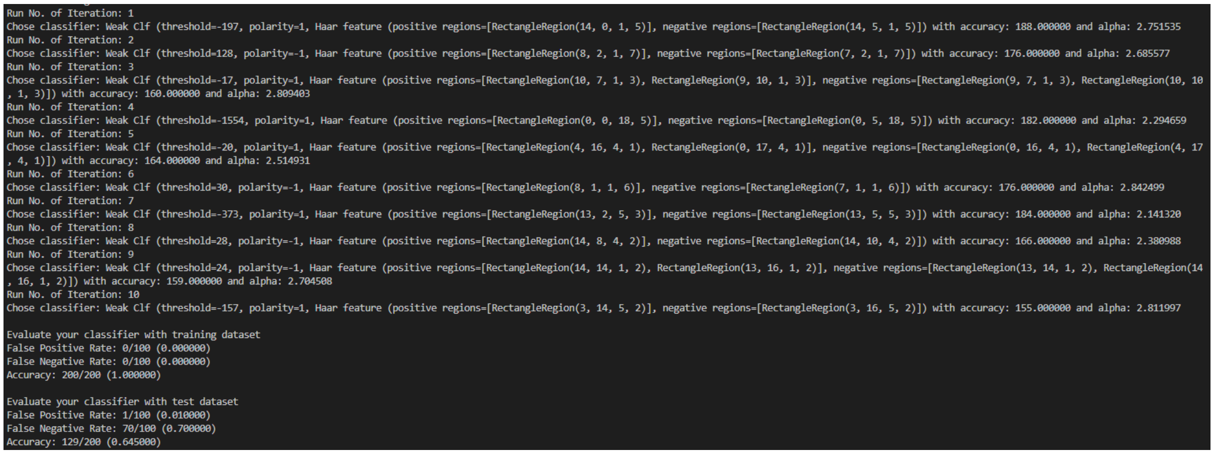
4. Other than Viola-Jones' algorithm, please propose another possible face detection method (no matter how good or bad, please come up with an idea). Please discuss the pros and cons of the idea you proposed, compared to the Adaboost algorithm

I think the face detection should not influenced by the color of people The Harra features used the brightness and darkness of image as features to recognize the face or not, however black people face image may not have such different part between brightness and darkness. Rather than only the color, my algorithm should also consider the relative position of facial and formatting the color of images, that can make the different between people not that much.

Compare to the Adaboost algorithm, my algorithm pros is can eliminate the weakness of detect black people’s face, but the cons is may detected the image that is not face but have the same relative position part as human face.

• Part 6: Implement another classifier (Bonus)

Method 2: with decided threshold and polarity by function “classifier.py -> train_weak()” to generate weakclassifier rather than default threshold = 0, polarity = 1

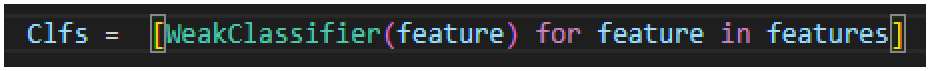


```
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=-197, polarity=1, Haar feature (positive regions=[RectangleRegion(14, 0, 1, 5)], negative regions=[RectangleRegion(14, 5, 1, 5)]) with accuracy: 188.000000 and alpha: 2.751535
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=128, polarity=-1, Haar feature (positive regions=[RectangleRegion(8, 2, 1, 7)], negative regions=[RectangleRegion(7, 2, 1, 7)]) with accuracy: 176.000000 and alpha: 2.685577
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=-17, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 7, 1, 3), RectangleRegion(9, 10, 1, 3)], negative regions=[RectangleRegion(9, 7, 1, 3), RectangleRegion(10, 10
, 1, 3)]) with accuracy: 160.000000 and alpha: 2.809403
Run No. of Iteration: 4
Chose classifier: Weak Clf (threshold=-1554, polarity=1, Haar feature (positive regions=[RectangleRegion(0, 0, 18, 5)], negative regions=[RectangleRegion(0, 5, 18, 5)]) with accuracy: 182.000000 and alpha: 2.294659
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=-20, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 16, 4, 1), RectangleRegion(0, 17, 4, 1)], negative regions=[RectangleRegion(0, 16, 4, 1), RectangleRegion(4, 17
, 4, 1)]) with accuracy: 164.000000 and alpha: 2.514931
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=30, polarity=-1, Haar feature (positive regions=[RectangleRegion(8, 1, 1, 6)], negative regions=[RectangleRegion(7, 1, 1, 6)]) with accuracy: 176.000000 and alpha: 2.842499
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=-373, polarity=1, Haar feature (positive regions=[RectangleRegion(13, 2, 5, 3)], negative regions=[RectangleRegion(13, 5, 5, 3)]) with accuracy: 184.000000 and alpha: 2.141320
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=28, polarity=-1, Haar feature (positive regions=[RectangleRegion(14, 8, 4, 2)], negative regions=[RectangleRegion(14, 10, 4, 2)]) with accuracy: 166.000000 and alpha: 2.380988
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=24, polarity=-1, Haar feature (positive regions=[RectangleRegion(14, 14, 1, 2), RectangleRegion(13, 16, 1, 2)], negative regions=[RectangleRegion(13, 14, 1, 2), RectangleRegion(14
, 16, 1, 2)]) with accuracy: 159.000000 and alpha: 2.704508
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=-157, polarity=1, Haar feature (positive regions=[RectangleRegion(3, 14, 5, 2)], negative regions=[RectangleRegion(3, 16, 5, 2)]) with accuracy: 155.000000 and alpha: 2.811997

Evaluate your classifier with training dataset
False Positive Rate: 0/100 (0.000000)
False Negative Rate: 0/100 (0.000000)
Accuracy: 200/200 (1.000000)

Evaluate your classifier with test dataset
False Positive Rate: 1/100 (0.010000)
False Negative Rate: 70/100 (0.700000)
Accuracy: 129/200 (0.645000)
```

Training model with T = 10

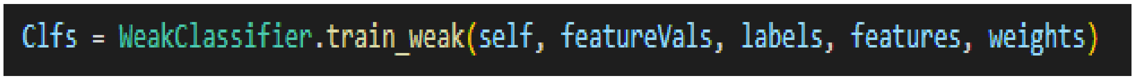
- **Code to train threshold and polarity**

Before:

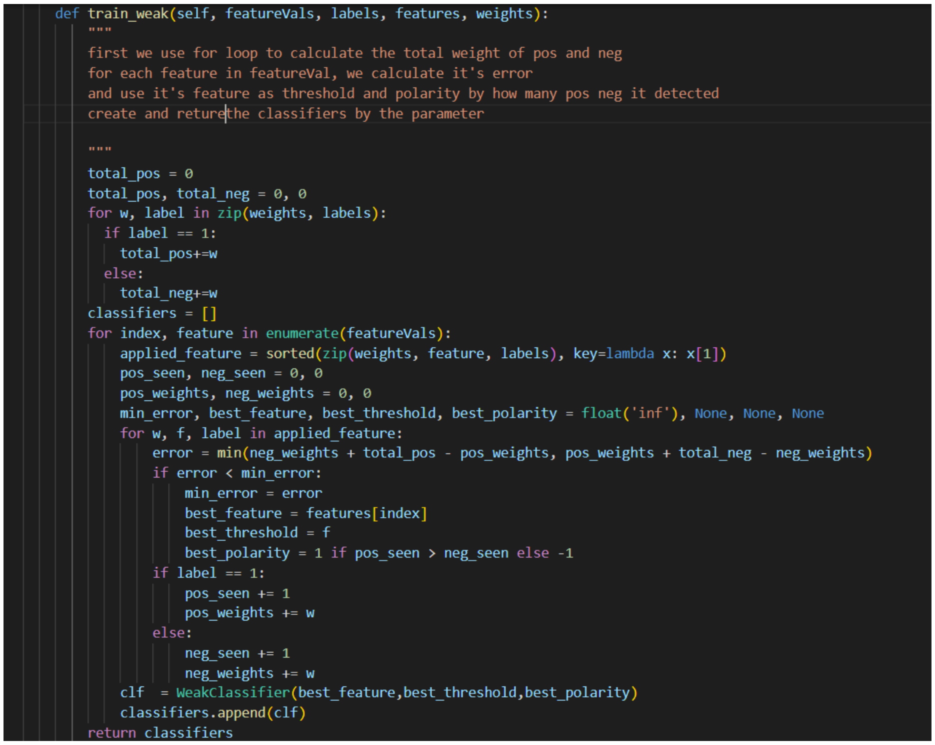


```
Clfs =  [WeakClassifier(feature) for feature in features]
```

After:



```
Clfs = WeakClassifier.train_weak(self, featureVals, labels, features, weights)
```

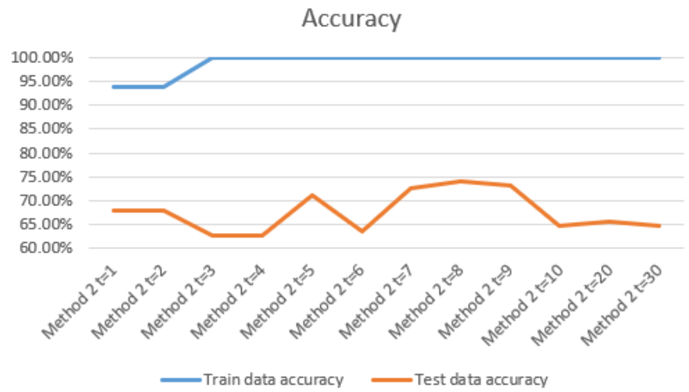


```
    def train_weak(self, featureVals, labels, features, weights):
        """
        first we use for loop to calculate the total weight of pos and neg
        for each feature in featureVal, we calculate it's error
        and use it's feature as threshold and polarity by how many pos neg it detected
        create and returethe classifiers by the parameter

        """
        total_pos = 0
        total_pos, total_neg = 0, 0
        for w, label in zip(weights, labels):
          if label == 1:
            total_pos+=w
          else:
            total_neg+=w
        classifiers = []
        for index, feature in enumerate(featureVals):
            applied_feature = sorted(zip(weights, feature, labels), key=lambda x: x[1])
            pos_seen, neg_seen = 0, 0
            pos_weights, neg_weights = 0, 0
            min_error, best_feature, best_threshold, best_polarity = float('inf'), None, None, None
            for w, f, label in applied_feature:
                error = min(neg_weights + total_pos - pos_weights, pos_weights + total_neg - neg_weights)
                if error < min_error:
                    min_error = error
                    best_feature = features[index]
                    best_threshold = f
                    best_polarity = 1 if pos_seen > neg_seen else -1
                if label == 1:
                    pos_seen += 1
                    pos_weights += w
                else:
                    neg_seen += 1
                    neg_weights += w
            clf  = WeakClassifier(best_feature,best_threshold,best_polarity)
            classifiers.append(clf)
        return classifiers
```

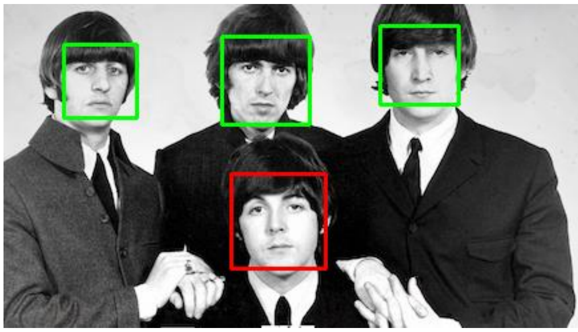
Code of train_weak

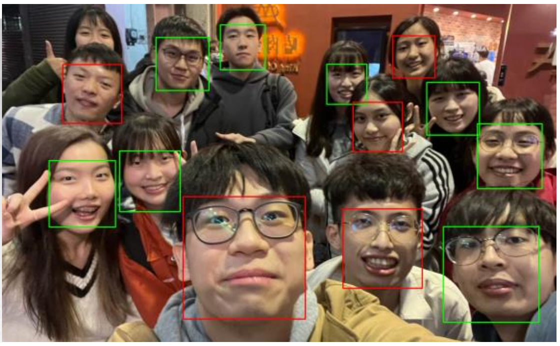
The table below is the result of our training model with t from 1~10,20,30,40,50. We can see that when t increase train data accuracy increasing , but t increase and have peak value at t = 8

200 張	**Train data accuracy**	**Test data accuracy**
Method 2 t=1	**94.0%**	**68.0%**
Method 2 t=2	**94.0%**	**68.0%**
Method 2 t=3	**100.0%**	**62.5%**
Method 2 t=4	**100.0%**	**62.5%**
Method 2 t=5	**100.0%**	**71.0%**
Method 2 t=6	**100.0%**	**63.5%**
Method 2 t=7	**100.0%**	**72.5%**
Method 2 t=8	**100.0%**	**74.0%**
Method 2 t=9	**100.0%**	**73.0%**
Method 2 t=10	**100.0%**	**64.5%**
Method 2 t=20	**100.0%**	**65.5%**
Method 2 t=30	**100.0%**	**64.5%**



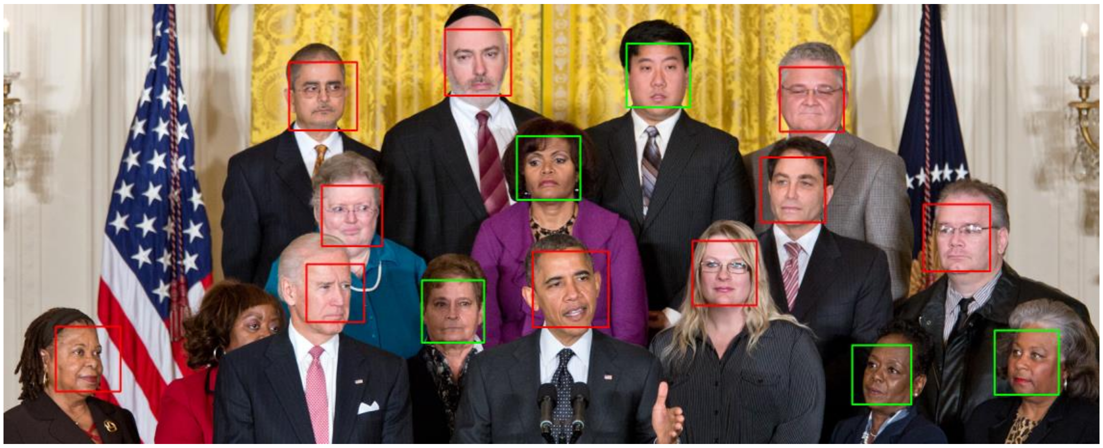
We can see the result of changing threshold and polarity is much better than before we only set threshold = 0, polarity = 0.
When testing using image, the best result is t = 8 same as the evaluate before we did.





↑Face detection “the-beatles.jpg” Face detection “my_pic.jpg” ↑

↓Face detection “p110912sh-0083.jpg”



With value t = 8