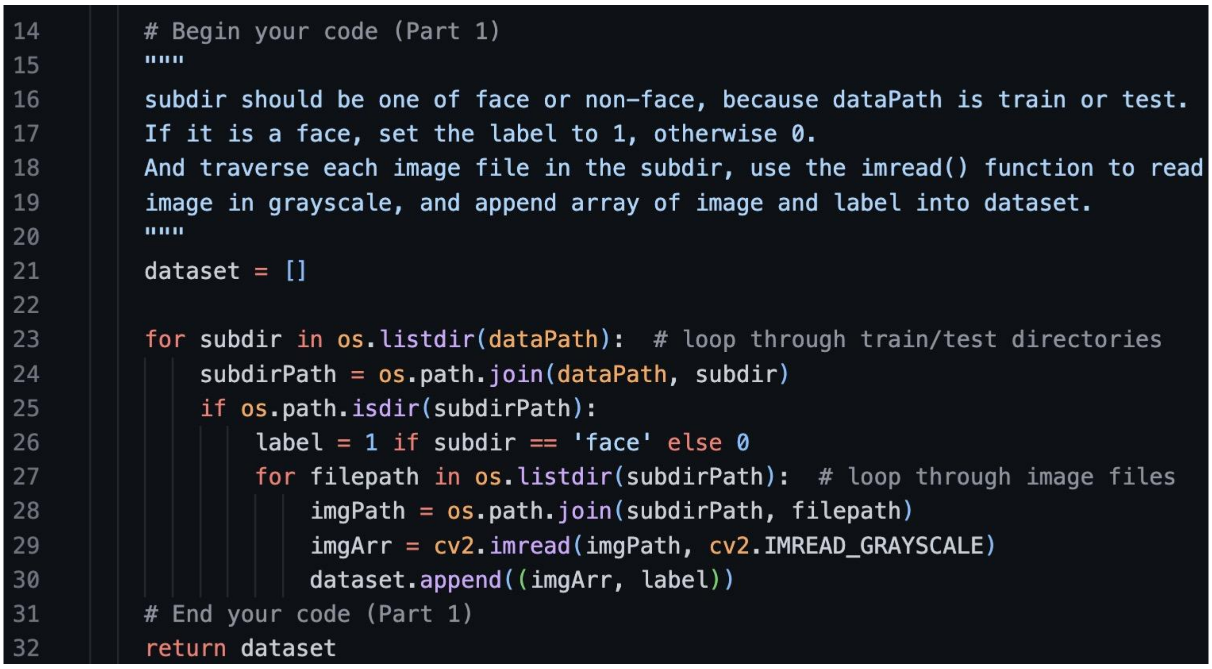
Homework 1: Face Detection Report

110550012 黃鵬軒

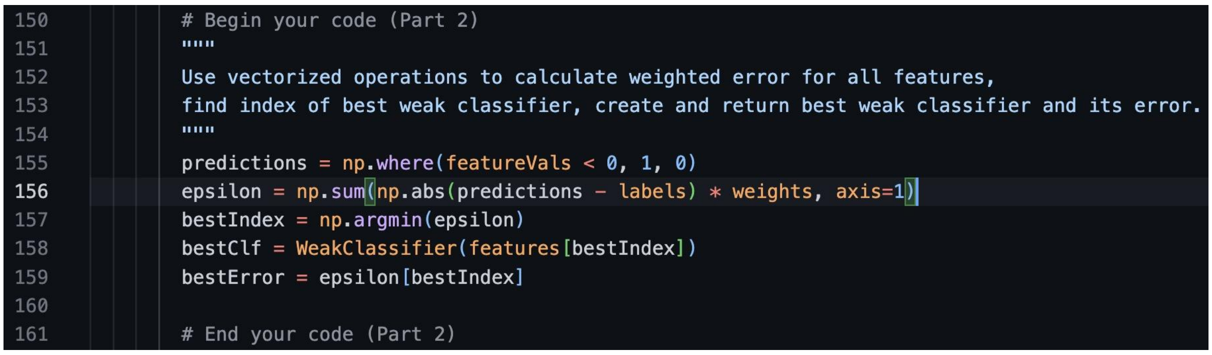
Part I. Implementation (6%):

Part 1



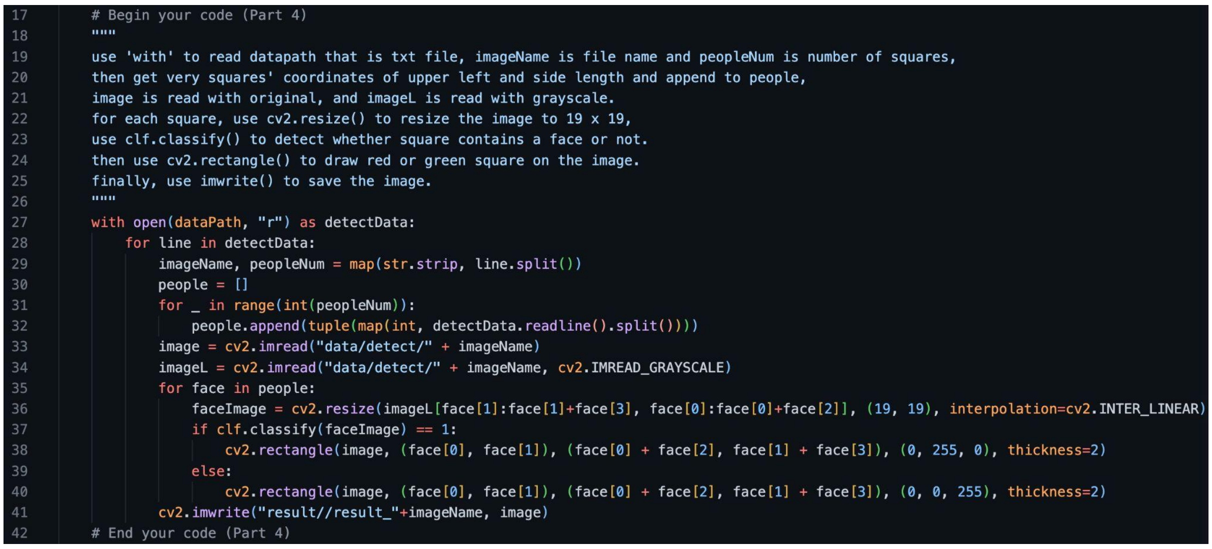
```
# Begin your code (Part 1)
"""
subdir should be one of face or non-face, because dataPath is train or test.
If it is a face, set the label to 1, otherwise 0.
And traverse each image file in the subdir, use the imread() function to read
image in grayscale, and append array of image and label into dataset.
"""
dataset = []

for subdir in os.listdir(dataPath):  # loop through train/test directories
    subdirPath = os.path.join(dataPath, subdir)
    if os.path.isdir(subdirPath):
        label = 1 if subdir == 'face' else 0
        for filepath in os.listdir(subdirPath):  # loop through image files
            imgPath = os.path.join(subdirPath, filepath)
            imgArr = cv2.imread(imgPath, cv2.IMREAD_GRAYSCALE)
            dataset.append((imgArr, label))
# End your code (Part 1)
return dataset
```

Part2



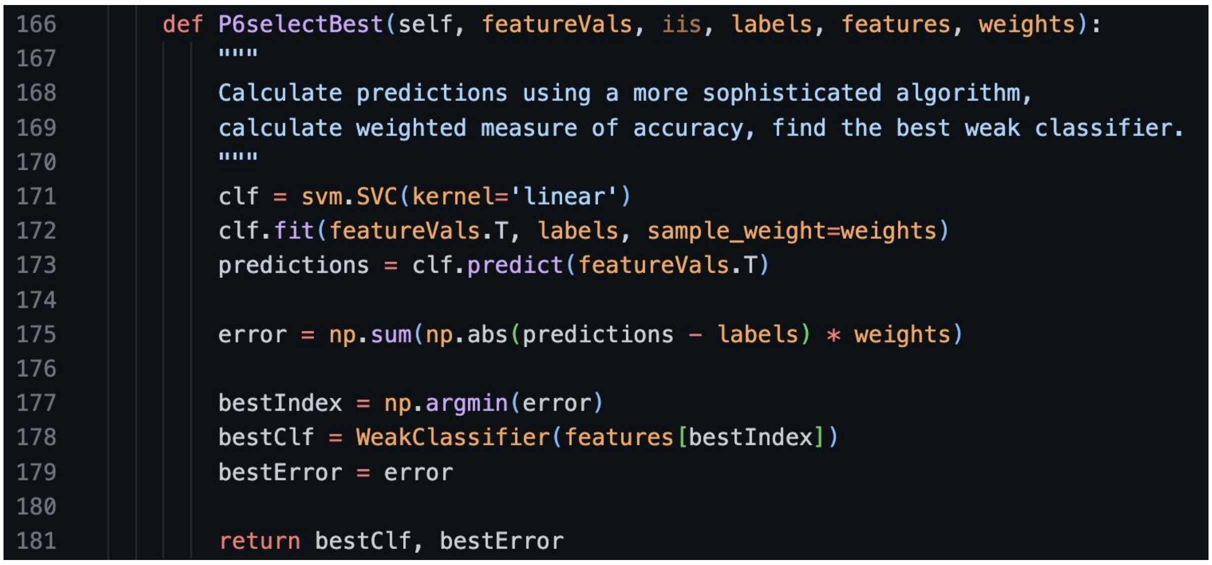
```
# Begin your code (Part 2)
"""
Use vectorized operations to calculate weighted error for all features,
find index of best weak classifier, create and return best weak classifier and its error.
"""
predictions = np.where(featureVals < 0, 1, 0)
epsilon = np.sum(np.abs(predictions - labels) * weights, axis=1)
bestIndex = np.argmin(epsilon)
bestClf = WeakClassifier(features[bestIndex])
bestError = epsilon[bestIndex]

# End your code (Part 2)
```

Part4



```python
# Begin your code (Part 4)
"""
use 'with' to read datapath that is txt file, imageName is file name and peopleNum is number of squares,
then get very squares' coordinates of upper left and side length and append to people,
image is read with original, and imageL is read with grayscale.
for each square, use cv2.resize() to resize the image to 19 x 19,
use clf.classify() to detect whether square contains a face or not.
then use cv2.rectangle() to draw red or green square on the image.
finally, use imwrite() to save the image.
"""
with open(dataPath, "r") as detectData:
    for line in detectData:
        imageName, peopleNum = map(str.strip, line.split())
        people = []
        for _ in range(int(peopleNum)):
            people.append(tuple(map(int, detectData.readline().split())))
        image = cv2.imread("data/detect/" + imageName)
        imageL = cv2.imread("data/detect/" + imageName, cv2.IMREAD_GRAYSCALE)
        for face in people:
            faceImage = cv2.resize(imageL[face[1]:face[1]+face[3], face[0]:face[0]+face[2]], (19, 19), interpolation=cv2.INTER_LINEAR)
            if clf.classify(faceImage) == 1:
                cv2.rectangle(image, (face[0], face[1]), (face[0] + face[2], face[1] + face[3]), (0, 255, 0), thickness=2)
            else:
                cv2.rectangle(image, (face[0], face[1]), (face[0] + face[2], face[1] + face[3]), (0, 0, 255), thickness=2)
        cv2.imwrite("result//result_"+imageName, image)
# End your code (Part 4)
```

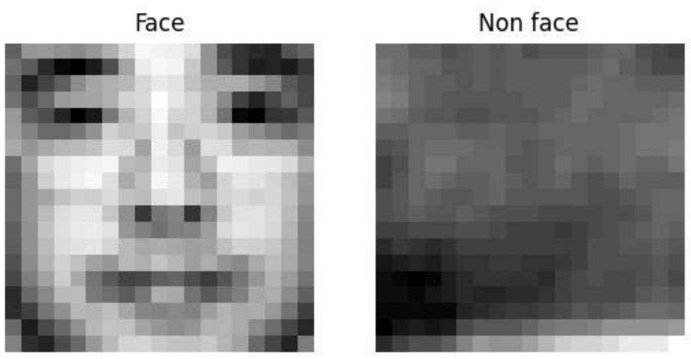
Part6



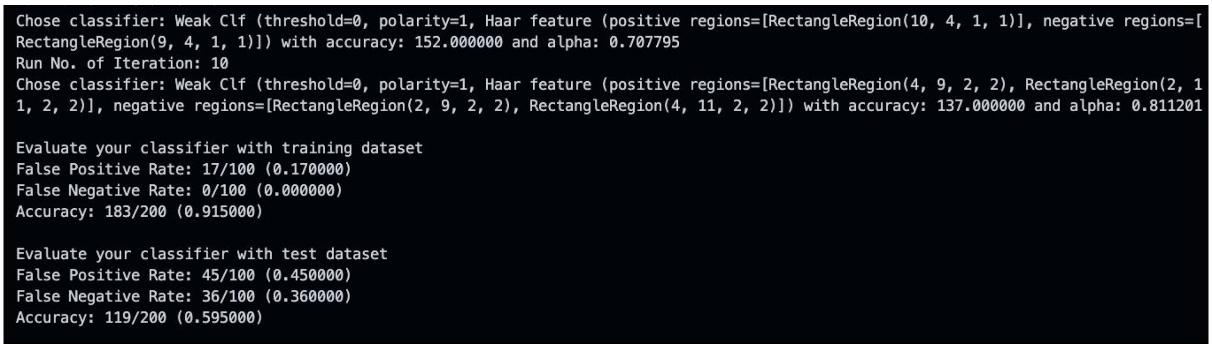
```python
def P6selectBest(self, featureVals, iis, labels, features, weights):
    """
    Calculate predictions using a more sophisticated algorithm,
    calculate weighted measure of accuracy, find the best weak classifier.
    """
    clf = svm.SVC(kernel='linear')
    clf.fit(featureVals.T, labels, sample_weight=weights)
    predictions = clf.predict(featureVals.T)

    error = np.sum(np.abs(predictions - labels) * weights)

    bestIndex = np.argmin(error)
    bestClf = WeakClassifier(features[bestIndex])
    bestError = error

    return bestClf, bestError
```

Part II. Results & Analysis (12%):

Part1



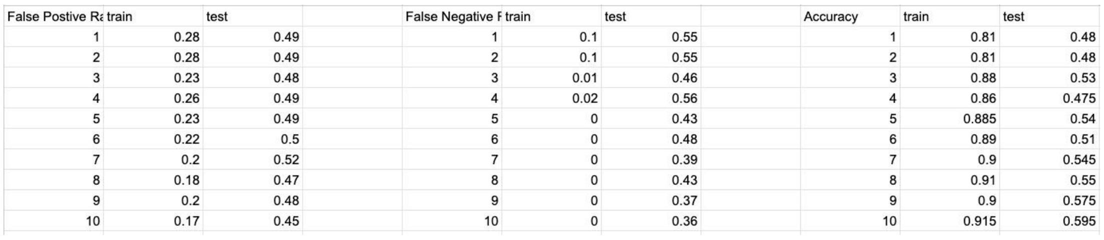
Part 2



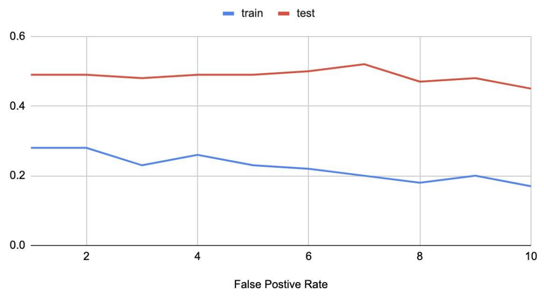
```
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(10, 4, 1, 1)], negative regions=[
RectangleRegion(9, 4, 1, 1)]) with accuracy: 152.000000 and alpha: 0.707795
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(4, 9, 2, 2), RectangleRegion(2, 1
1, 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)
```

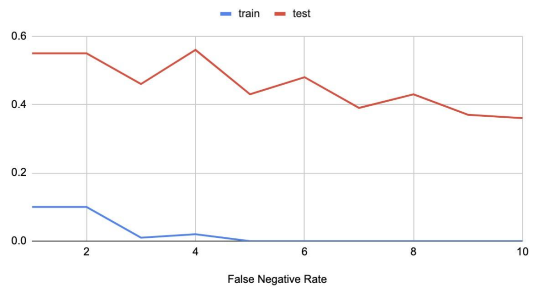
Part3

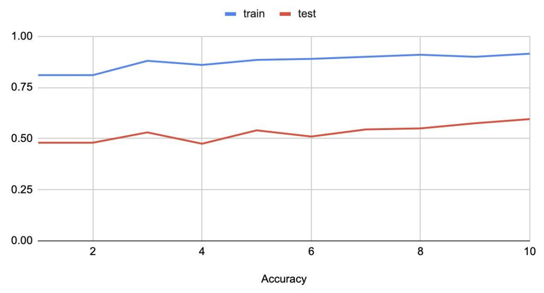
test the parameter T between 1 to 10, we can get the following table and line chart



False Postive Ra	train	test		False Negative F	train	test		Accuracy	train	test
1	0.28	0.49		1	0.1	0.55		1	0.81	0.48
2	0.28	0.49		2	0.1	0.55		2	0.81	0.48
3	0.23	0.48		3	0.01	0.46		3	0.88	0.53
4	0.26	0.49		4	0.02	0.56		4	0.86	0.475
5	0.23	0.49		5	0	0.43		5	0.885	0.54
6	0.22	0.5		6	0	0.48		6	0.89	0.51
7	0.2	0.52		7	0	0.39		7	0.9	0.545
8	0.18	0.47		8	0	0.43		8	0.91	0.55
9	0.2	0.48		9	0	0.37		9	0.9	0.575
10	0.17	0.45		10	0	0.36		10	0.915	0.595

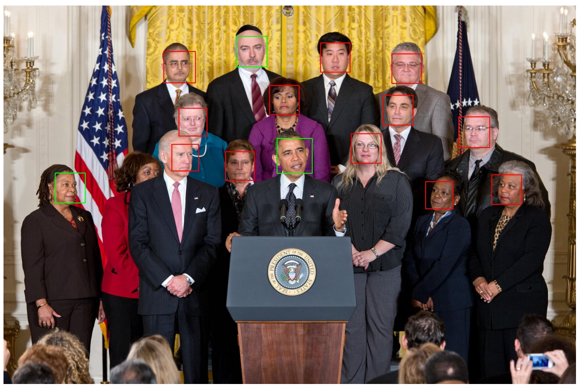


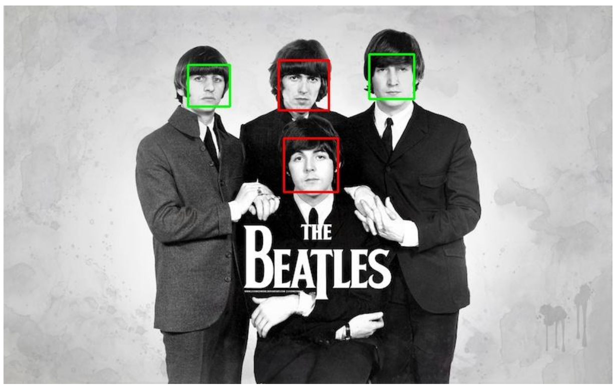




With more training, Accuracy increases and False Negative Rate decreases significantly, while False Positive Rate decreases at a slower rate. The model performs well on training data, but its ability to recognize faces in general images needs to be improved. Test results indicate that the model may require additional training. It has a high False Positive Rate, suggesting that it has learned important facial features but has overly loose conditions that identify non-faces as faces, while true faces are not easily misjudged.

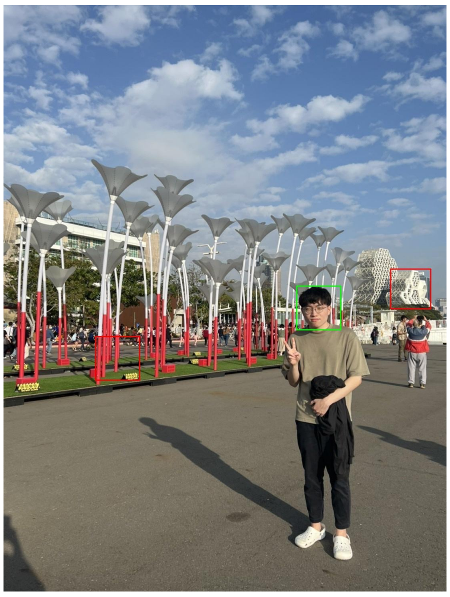
Part4

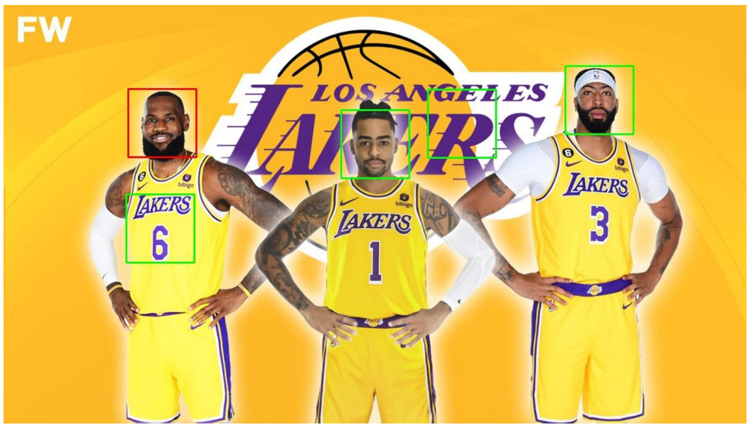




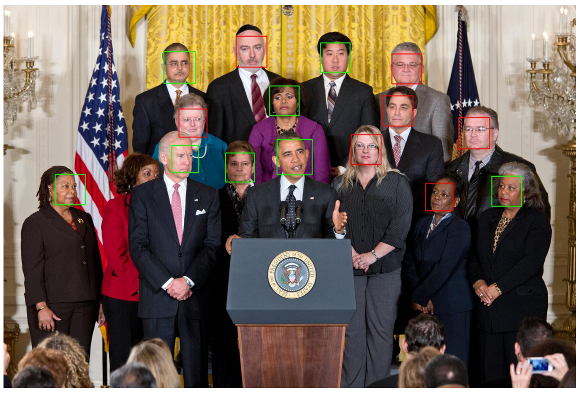
Part 5

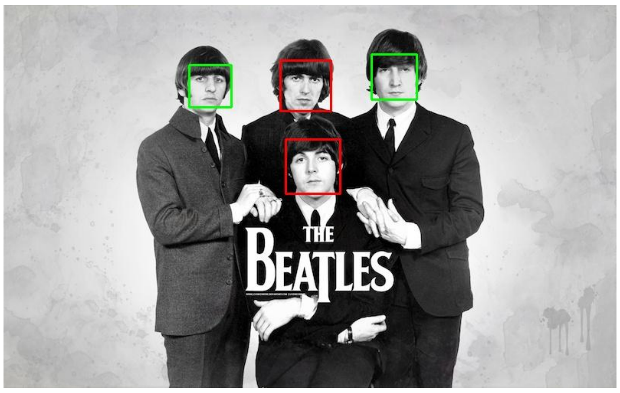
I use an online image coordinate analysis website to obtain coordinate data. Although the left image is fine, there are clear misjudgments in the right image. It can be seen that the recognition of non-face is highly inaccurate, and the recognition of face also can be improved.





Part6





Part III. Answer the questions (12%):

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

- I didn't set the directory part properly at the beginning, so the specified file could not be found, and I got an error message of file not found, so I reset the workspace of vscode.
- I didn't clearly understand the format of detectData.txt initially, and I don't know how to get the pixel coordination of the image. Finally, I just use an online image coordinate analysis website to obtain coordinate data.

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

- Sensitive to lighting conditions. It may fail in low light or high contrast conditions.
- Not very accurate in the general case and requires a large number of training samples.
- Front or side face will affect a lot.

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

- Augment the training data by applying transformations such as rotation, scaling to the images. This helps the algorithm learn to detect faces in different orientations and sizes.
- Select more relevant Haar-like features.
- Optimize the adaboost algorithm.

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.

We can use "skin color" as a feature to find faces.The idea is based on the assumption that human skin has a distinct color range that can be used to separate it from other objects in an image. but this method is sensitive to lighting conditions and skin tones variations. Apparently, it is not suitable for pictures in true color.

Compared to the Adaboost algorithm:

	skin color	Adaboost
Implementation	Simple and fast	Complex and slow
Data Requirement	No training data needed	Labeled data needed
Feature	Only color information used	Many Haar-like features
Accuracy	Lower	Higher
Background handling	Lower	Higher