Homework 1: Face Detection

Report

110652019 林楷傑

Part I. Implementation:

- Part 1:
  Load the images from the dataset.

```
    # Begin your code (Part 1)
    dataset = []
    for file_name in os.listdir(os.path.join(dataPath, "face")):
        img = cv2.imread(os.path.join(dataPath, "face", file_name))[: ,: , 0]
        dataset.append((img, 1))
    for file_name in os.listdir(os.path.join(dataPath, "non-face")):
        img = cv2.imread(os.path.join(dataPath, "non-face", file_name))[:, :, 0]
        dataset.append((img, 0))
    # End your code (Part 1)
```

- Part 2:
  Select the lowest error classifier.

```
            # Begin your code (Part 2)
            clfs = [WeakClassifier(feature=feature) for feature in features]
            bestClf = None
            bestError = sum(weights)
            for clf in clfs:
                error = 0
                for i in range(len(iis)):
                    if clf.classify(iis[i]) != labels[i]:
                        error += weights[i]
                if error < bestError:
                    bestError = error
                    bestClf = clf
            # End your code (Part 2)
            return bestClf, bestError
```

- Part 4:

  Load the coordinates from the file and plot the bounding box.

```python
    # Begin your code (Part 4)
    # Load the txt file
    cords1 = []
    cords2 = []
    img_path = []
    with open(dataPath) as file:
        tmp = file.readline()
        tmp = tmp.split(' ')
        img_path.append(tmp[0])
        n_cord = int(tmp[1])
        for _ in range(n_cord):
            cor = file.readline()
            cor = cor.split(' ')
            res = tuple(map(int, cor))
            cords1.append(res)
        tmp = file.readline()
        tmp = tmp.split(" ")
        img_path.append(tmp[0])
        n_cord = int(tmp[1])
        for _ in range(n_cord):
            cor = file.readline()
            cor = cor.split(" ")
            res = tuple(map(int, cor))
            cords2.append(res)

    # Process on first image
    img1 = cv2.imread(os.path.join('data/detect/', img_path[0]))
    for (x, y ,w, h) in cords1:
        face_img = img1[y:y+h, x:x+w]
        face_img = cv2.resize(face_img, (19, 19))
        face_img_gray = cv2.cvtColor(face_img, cv2.COLOR_BGR2GRAY)
        res = clf.classify(face_img_gray)
        if res == 1:
            cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 255, 0), 2)
        else:
            cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # Process on second image
    img2 = cv2.imread(os.path.join('data/detect/', img_path[1]))
    for (x, y ,w, h) in cords2:
        face_img = img2[y:y+h, x:x+w]
        face_img = cv2.resize(face_img, (19, 19))
        face_img_gray = cv2.cvtColor(face_img, cv2.COLOR_BGR2GRAY)
        res = clf.classify(face_img_gray)
        if res == 1:
            cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 255, 0), 2)
        else:
            cv2.rectangle(img2, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # Show the images
    plt.imshow(cv2.cvtColor(img1, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.show()
    plt.imshow(cv2.cvtColor(img2, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.show()
    # End your code (Part 4)
```

Part II. Results & Analysis:

- Result T = 1

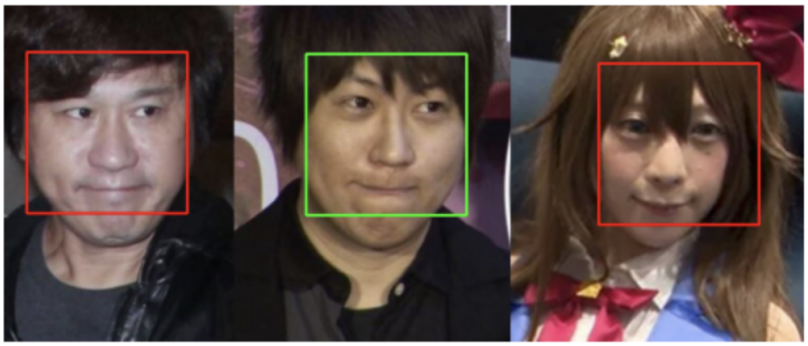
```
Start training your classifier
Computing integral images
Building features
Applying features to dataset
Selecting best features
Selected 5171 potential features
Initialize weights
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polarity=1, Haar feature (positive regions=[RectangleRegion(8, 0, 1, 3), Rec
tangleRegion(7, 3, 1, 3)], negative regions=[RectangleRegion(7, 0, 1, 3), RectangleRegion(8, 3, 1, 3)]) with accurac
y: 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)
```

- The performance between the training and testing dataset with different T.

Find that when T is bigger, the algorithm performs better.When T=10, adaboost works best.

- Detect on images

Test on my own image:



Implement the “selectBest” function in adaboost.py with another method.

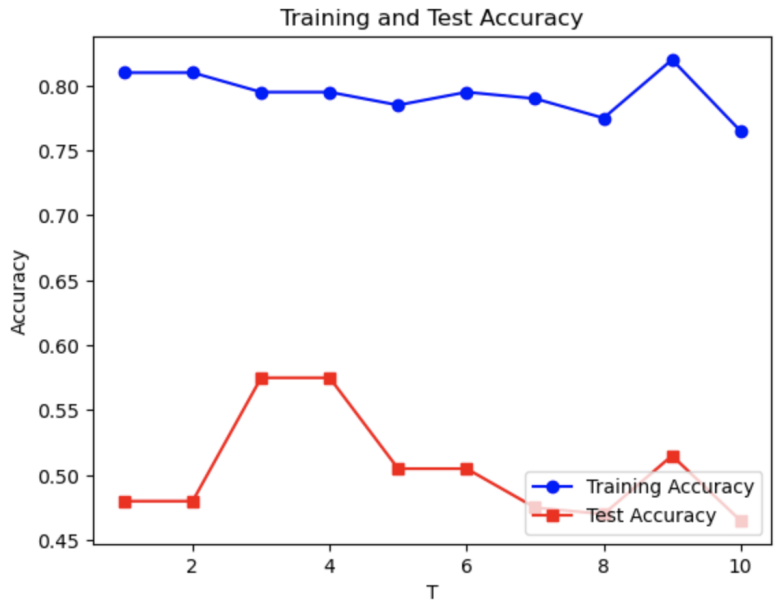
Use mean square error instead of 0-1 loss.

```
# Begin your code (Part 2)
clfs = [WeakClassifier(feature=feature) for feature in features]
bestClf = None
bestError = 0
for w in weights:
    bestError += w**2
bestError = np.sqrt(bestError)
for clf in clfs:
    error = 0
    for i in range(len(iis)):
        if clf.classify(iis[i]) != labels[i]:
            #error += weights[i]
            error += weights[i]**2 # Use mean square error
    error = np.sqrt(error)
    if error < bestError:
        bestError = error
        bestClf = clf
# End your code (Part 2)
```

Result analysis:

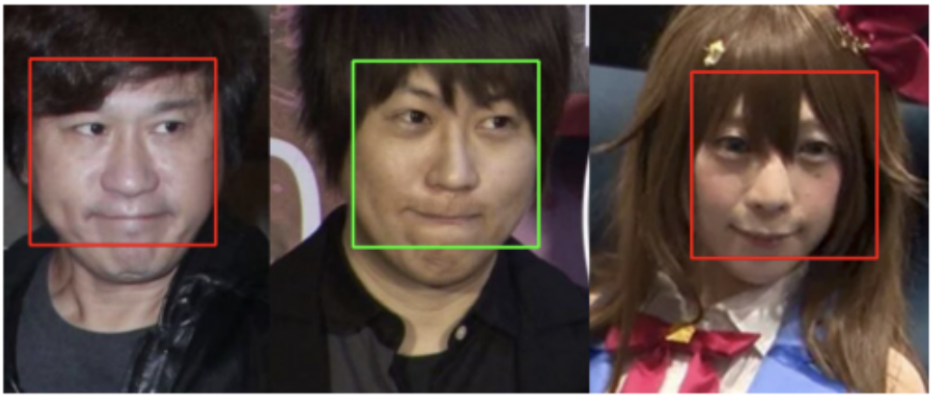
It looks like changing it did not improve anything.

```
When T = 3, the adaboost works best
```



Detect faces at the assigned lacation using your classifier

Detect faces on your own images



Part III. Answer the question:

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

I found that my algorithm was doing badly in the detection part. Then I solve it by preprocessing the detection image to grayscale. This makes the results more reasonable.

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

I think it has limited accuracy since Viola-Jones' algorithm can struggle to detect objects with complex shapes or textures. It tends to be overfitting on the training data. The training process for the Viola-Jones algorithm can be computationally expensive and time-consuming.

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

We can add more Haar-like features to the classifier to improve the performance. But this could cause overfitting issues.

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 CNN based deep neural networks to do this kind of classification task. The pros of CNNs include high accuracy and robustness, which can handle variations in lighting, pose, and expression. And it can get decent performance. The cons of CNNs are computationally expensive and require large amounts of training data. Compared to Adaboost, CNNs have higher accuracy and robustness, but require more time to train and more training data.