HW1_Report

Part1

```
dataset = []
car = os.listdir(dataPath + '/' + 'car')
count = 0
for i in car:
    img = cv2.imread(dataPath + '/' + 'car' + '/' + i, 0)
    img = cv2.resize(img, (36, 16))
    dataset.append((img, 1))

car = os.listdir(dataPath + '/' + 'non-car')
count = 0
for i in car:
    img = cv2.imread(dataPath + '/' + 'non-car' + '/' + i, 0)
    img = cv2.resize(img, (36, 16))
    dataset.append((img, 0))
```

car = os.listdir(dataPath + '/' + 'car')

用這一句指定檔案路徑

for 迴圈內將資料夾內的圖片轉為灰階 (cv2.imread 的第二個參數)和 36 x 16 的大小(cv2.resize)，將 car 的 label 標記為 1，non-car 的 label 標記為 0，並 append 在 dataset 內，在遍歷完 car 和 non-car 兩個資料夾後，return dataset

Part2/Adaboost

-Adaboost

Adaboost 會利用前面一次弱分類器分類錯誤的樣本學習，提高前面被分類錯誤樣本的權重，降低分類成功樣本的權重，並以修改過權值的資料集合訓練下一個弱分類器。整合這些弱分類器後，成為一個強分類器，每一步都在提升算法的準確度。

公式 ：

$$\alpha_k \leftarrow \frac{1}{2}\ln\frac{1-E_k}{E_k}$$

$$W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k}, & \text{if } h_k(x^i) = y_i \\ e^{\alpha_k}, & \text{if } h_k(x^i) \neq y_i \end{cases}$$

在 adaboost.py 內 的 train function:

```
accuracy = []
for x, y in zip(iis, labels):
    correctness = abs(clf.classify(x) - y)
    accuracy.append(correctness)
beta = error / (1.0 - error)
for i in range(len(accuracy)):
    weights[i] = weights[i] * (beta ** (1 - accuracy[i]))
alpha = math.log(1.0/beta)
self.alphas.append(alpha)
self.clfs.append(clf)
```

-Part2

這一部分的主要目的是找去擁有最佳效果 (即最小 error)的分類器

```
for value in featureVals[feature_index]:
    if value < 0:
        temp.append(1)
    else:
        temp.append(0)
```

這一段是用每一張圖片對應到的 feature value 分類，如果< 0，將那個 label 設為 1，代表有偵測到車，>= 0 時設為 0。而對 value 我也有調整過不同的值，例如 -1, 0.5 等等，最後發現設為 0 出現的結果較為準確。

```
bestError = 1000000000
bestFeature = -1
for i in range(len(result)):
    temp_error = 0
    for j in range(len(result[i])):
        temp_error += weights[j] * abs(result[i][j] - labels[j])
    if temp_error < bestError:
        bestError = temp_error
        bestFeature = i
bestClf = WeakClassifier(features[bestFeature])
#raise NotImplementedError("To be implemented")
# End your code (Part 2)
return bestClf, bestError
```

這個 for 迴圈裡的 I 是代表不同的 feature，而 j 則代表每一個 training sample，我把 error 的算法設定為

Weights[j] * abs(result[I][j] – labels[j])

將不同 feature 產生的 error 進行比較，當出現最小 error，就將該 feature 紀錄並傳進 WeakClassifier 內，得到最佳的 WeakClassifier class

Part3

T = 1

```
Run No. of Iteration: 1
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 25/300 (0.083333)
False Negative Rate: 88/300 (0.293333)
Accuracy: 487/600 (0.811667)

Evaluate your classifier with test dataset
False Positive Rate: 24/300 (0.080000)
False Negative Rate: 91/300 (0.303333)
Accuracy: 485/600 (0.808333)
```

T = 2

```
Run No. of Iteration: 2
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 25/300 (0.083333)
False Negative Rate: 88/300 (0.293333)
Accuracy: 487/600 (0.811667)

Evaluate your classifier with test dataset
False Positive Rate: 24/300 (0.080000)
False Negative Rate: 91/300 (0.303333)
Accuracy: 485/600 (0.808333)
```

T = 3

```
Run No. of Iteration: 3
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 14/300 (0.046667)
False Negative Rate: 67/300 (0.223333)
Accuracy: 519/600 (0.865000)

Evaluate your classifier with test dataset
False Positive Rate: 21/300 (0.070000)
False Negative Rate: 94/300 (0.313333)
Accuracy: 485/600 (0.808333)
```

T = 4

```
Run No. of Iteration: 4
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 28/300 (0.093333)
False Negative Rate: 48/300 (0.160000)
Accuracy: 524/600 (0.873333)

Evaluate your classifier with test dataset
False Positive Rate: 28/300 (0.093333)
False Negative Rate: 70/300 (0.233333)
Accuracy: 502/600 (0.836667)
```

T = 5

```
Run No. of Iteration: 5
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 20/300 (0.066667)
False Negative Rate: 45/300 (0.150000)
Accuracy: 535/600 (0.891667)

Evaluate your classifier with test dataset
False Positive Rate: 25/300 (0.083333)
False Negative Rate: 73/300 (0.243333)
Accuracy: 502/600 (0.836667)
```

T = 6

```
Run No. of Iteration: 6
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 23/300 (0.076667)
False Negative Rate: 39/300 (0.130000)
Accuracy: 538/600 (0.896667)

Evaluate your classifier with test dataset
False Positive Rate: 24/300 (0.080000)
False Negative Rate: 60/300 (0.200000)
Accuracy: 516/600 (0.860000)
```

T = 7

```
Run No. of Iteration: 7
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 21/300 (0.070000)
False Negative Rate: 39/300 (0.130000)
Accuracy: 540/600 (0.900000)

Evaluate your classifier with test dataset
False Positive Rate: 24/300 (0.080000)
False Negative Rate: 65/300 (0.216667)
Accuracy: 511/600 (0.851667)
```

T = 8

```
Run No. of Iteration: 8
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 17/300 (0.056667)
False Negative Rate: 41/300 (0.136667)
Accuracy: 542/600 (0.903333)

Evaluate your classifier with test dataset
False Positive Rate: 22/300 (0.073333)
False Negative Rate: 65/300 (0.216667)
Accuracy: 513/600 (0.855000)
```

T = 9

```
Run No. of Iteration: 9
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 21/300 (0.070000)
False Negative Rate: 34/300 (0.113333)
Accuracy: 545/600 (0.908333)

Evaluate your classifier with test dataset
False Positive Rate: 27/300 (0.090000)
False Negative Rate: 50/300 (0.166667)
Accuracy: 523/600 (0.871667)
```

T = 10

```
Run No. of Iteration: 10
Chose classifier: Weak Clf (threshold=0, polari

Evaluate your classifier with training dataset
False Positive Rate: 20/300 (0.066667)
False Negative Rate: 39/300 (0.130000)
Accuracy: 541/600 (0.901667)

Evaluate your classifier with test dataset
False Positive Rate: 25/300 (0.083333)
False Negative Rate: 60/300 (0.200000)
Accuracy: 515/600 (0.858333)
```

從上面的圖可以發現，隨著 T 的增加，在 training accuracy 跟 test accuracy 也呈正成長，代表 classifier 正在被訓練成具有更好分類效果的 classifier

Part4

-Part4

```python
with open(dataPath,'r') as fh:
    linelist = fh.readlines()
    for i in range(len(linelist)):
        linelist[i] = linelist[i].strip()

result = []
temp=[]
tt=[]
for i in linelist:
    temp= i.split()
    for j in temp:
        j = int(j)
        tt.append(j)
    result.append(tt)
    temp=[]
    tt=[]
result.pop(0)
```

首先讀取 detectData.txt2 的內容，在 video.gif 內總共有 76 格停車位，每一個停車位用四個座標點說明

```python
while capture.isOpened():
    ret, frame= capture.read()
    if frame is None:
        break
```

在 while 迴圈內對每一幀進行操作

```
for i in result:
  img = crop(i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7], frame)
  img = cv2.resize(img, (36,16))
  img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

利用 crop( )將 76 個停車個剪裁出來，並一樣轉為 36 x 16 的灰階圖片

```
if clf.classify(img) == 1:
  draw_box(frame, i[0], i[1], i[2], i[3], i[4], i[5], i[6], i[7])
  temp.append("1")
else:
  temp.append("0")
```

```
def draw_box(img, x1, y1, x2, y2, x3, y3, x4, y4):
    color = (0, 255, 0)
    thickness = 2
    cv2.line(img, (x1, y1), (x2, y2), color, thickness)
    cv2.line(img, (x2, y2), (x4, y4), color, thickness)
    cv2.line(img, (x3, y3), (x4, y4), color, thickness)
    cv2.line(img, (x1, y1), (x3, y3), color, thickness)
```

clf.classify(img)

使用 classify.py 的 classify function 判斷該車位是否有車，如果有，則回傳 true，並利用 draw_box( )畫出綠線格子，並且在 temp 這個 list 加入 ”1”，false 則加入”0”。

```
if count == 0 :
  f = open("Adaboost_pred.txt", "w")
  cv2.imwrite("data/detect/first.png", frame)
else :
  f = open("Adaboost_pred.txt", "a")
f.writelines(temp)
f.write("\n")
f.close()
count+=1
```

這裡我用 count 來判斷幀數，如果是第一幀，就將剛剛繪製好的圖片存取下來

f = open(“Adaboost_pred.txt”, “w”)

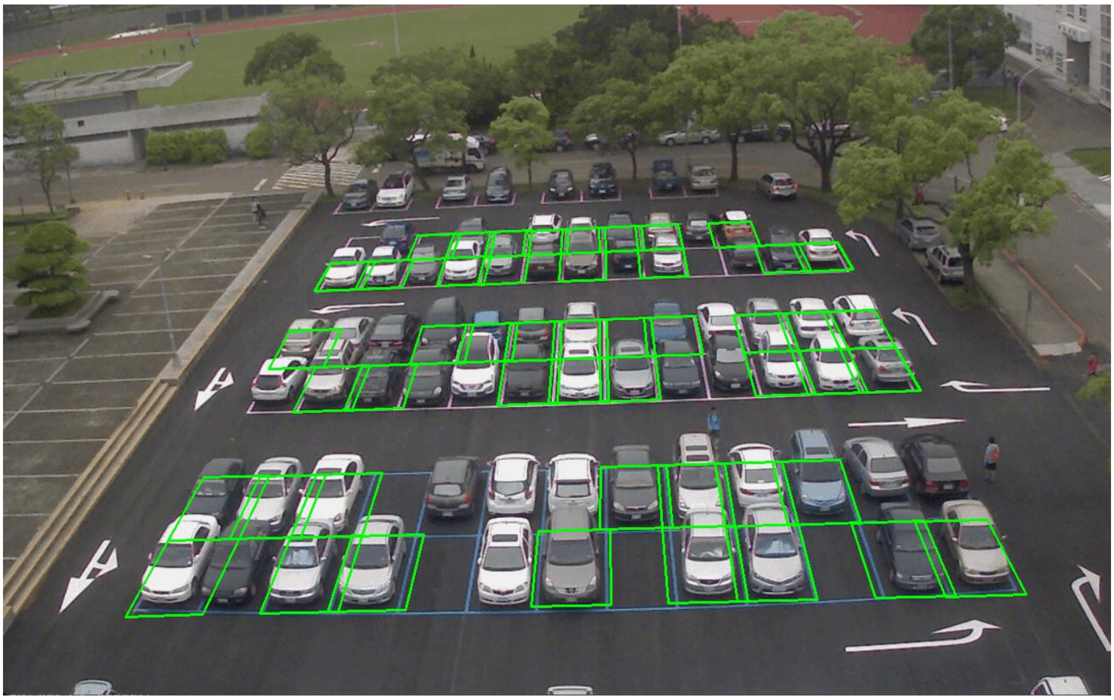
w 代表覆蓋之前的內容，另一行的 a 則是接著之前的內容。

寫入剛剛 temp 的內容，即為每一格車位是 1 或是 0

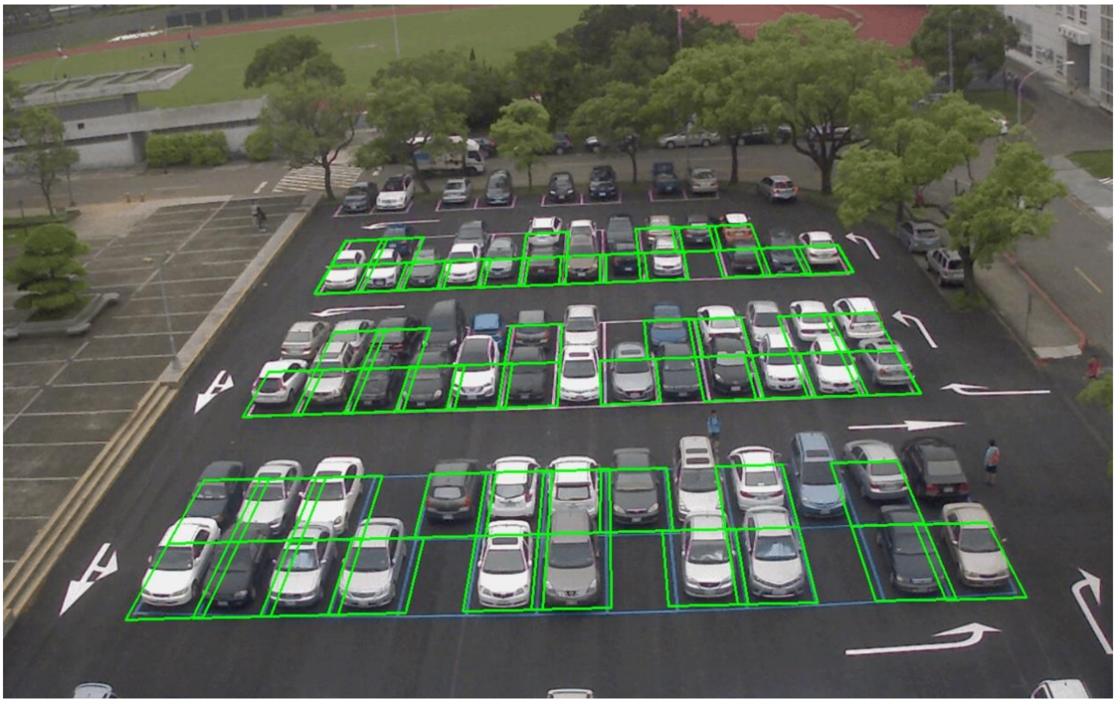
Part5

- **First frame**

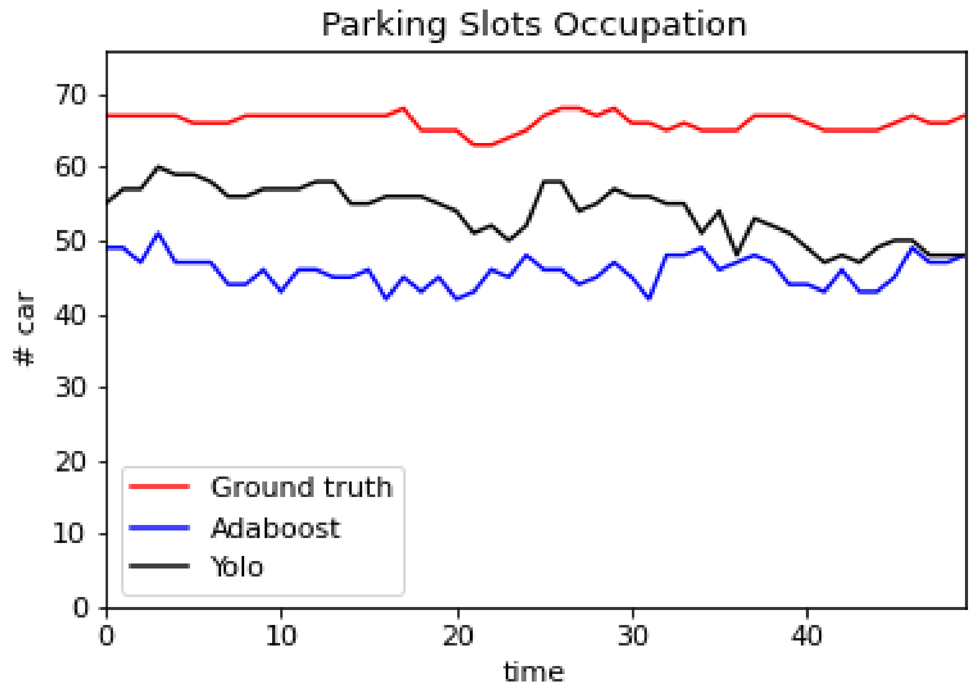
Adaboost



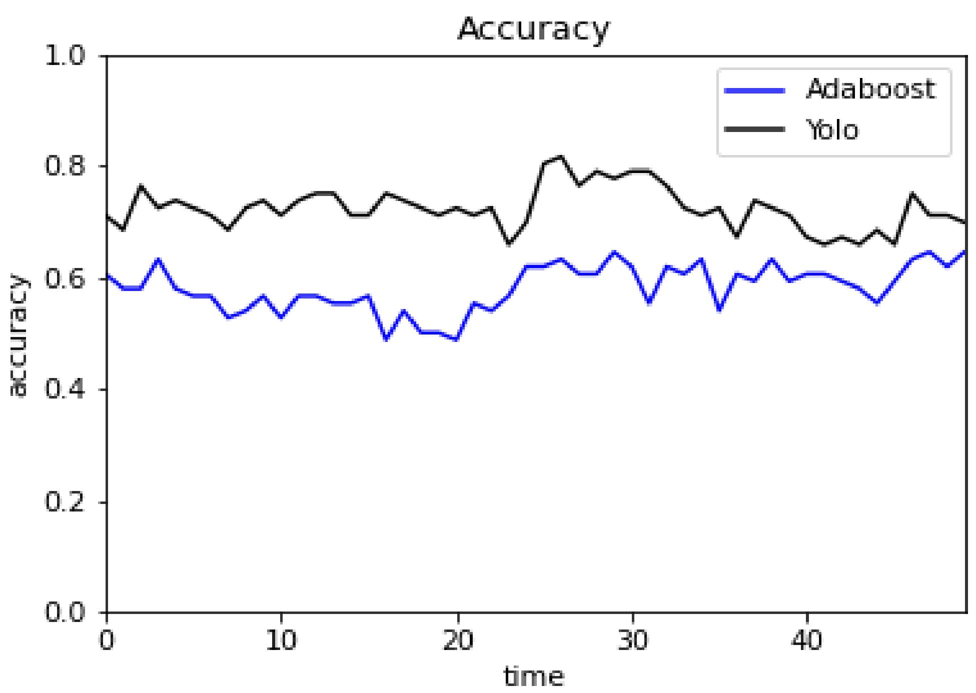
Yolov5



- **Occupation**



- **Accuracy**



Discussion

從 part5 實做出來的結果和繪製出來的兩張圖表可以發現，Yolov5 的準確度在每一張 frame 大概都比使用 adaboost 演算法高了 10%左右。而我也對第一幀照片進行計算，用得出的 accuracy 和 F1-score 做了比較：

```
Adaboost:
accuracy:  0.6052631578947368
f1-score:  0.7368421052631579

Yolov5:
accuracy:  0.7105263157894737
f1-score:  0.819672131147541
```

F1-score 的計算：

$$F - score \ = \ 2\,\frac{precision \ \times \ recall}{precision \ + \ recall}$$

演算法中常常會看到精確率 precision 和召回率 recall，precision 代表被判斷為真的樣本有多少為事實上為真(tp/tp+fp)，recall 代表事實為真的情形總和(tp/tp+fn)，f1-score 則是兩者的調和平均分數，可用來判斷演算法的好壞，並且 F1-score 的理想數值是趨近於 1。

從實做出的結果可以發現，Yolov5 的 f1-score 比 Adaboost 高了不少，這可能跟不同演算法使用的理論和我們設定的參數有關，我也有試著調整過助教給的 Yolov5 的 sample code，但得出來的結果並不比原本的好。

Problem

我覺得這次 homework 最大的挑戰就是要先讀懂 adaboost 的理論和使用到的各種公式，還有一些專有名詞，像是 viola-jones 演算法還有 Haar feature，在讀完之後，再去看已經寫好的 sample code，包含 feature.py, classifier.py 等等的內容，才能了解這次的 hw 到底需要寫哪些部分，和那些變數所代表的意義。在寫 select best 的部分花了我最多時間，因為在該 function 內有許多參數，像是 featureval, iis，為了要看懂他們，就必須從前面的幾個 function 讀起，例如 featureval 的計算方式、每一個 feature 是如何得出來的，再結合網路上查到的資訊，才終於把它寫出來。