Report

109550031 李旻融

Part 1：load image

 # main：將 train 和 test 兩個資料夾的路徑分別傳進 dataset.loadImages()

```
# Part 1: Implement loadImages function in dataset.py and test the following code.
print('Loading images')
trainData = dataset.loadImages('data/train')
print(f'The number of training samples loaded: {len(trainData)}')
testData = dataset.loadImages('data/test')
print(f'The number of test samples loaded: {len(testData)}')
```

 # dataset.py：先創一個 list (dataset)，再分別打開 car 和 non-car 的資料夾，讀取其中的每張照片，接著利用 cv2.cvtColor()將照片轉成灰階並且 resize 成(36,16)，有車的話 label=1，沒車則是 label=0，最後將照片和 label 存進一個 tuple，再存進 dataset。

```
import os
import cv2
import numpy
def loadImages(dataPath):
    """
    Load all Images in the folder and transfer a list of tuples.
    The first element is the numpy array of shape (m, n) representing the image.
    (remember to resize and convert the parking space images to 36 x 16 grayscale images.)
    The second element is its classification (1 or 0)
      Parameters:
        dataPath: The folder path.
      Returns:
        dataset: The list of tuples.
    """
    # Begin your code (Part 1)
    dataset = []
    car = dataPath+'/car'
    allfiles = os.listdir(car)

    for file in allfiles:
        img = cv2.imread(car+'/'+file)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.resize(img,(36,16))
        label = 1
        tuples = (img,label)
        dataset.append(tuples)

    noncar = dataPath+'/non-car'
    allfiles = os.listdir(noncar)

    for file in allfiles:
        img = cv2.imread(noncar+'/'+file)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.resize(img,(36,16))
        label = 0
        tuples = (img,label)
        dataset.append(tuples)
    # End your code (Part 1)

    return dataset
```

Part 2：train data

 # main：呼叫 adaboost.Adaboost.train()

```
print('Start training your classifier')
clf = adaboost.Adaboost(T=10)
clf.train(trainData)
```

adaboost.Adaboost._init_()：以 T=10 進行初始化

```
def __init__(self, T = 10):
    """
      Parameters:
        T: The number of weak classifiers which should be used.
    """
    self.T = T
    self.alphas = []
    self.clfs = []
```

adaboost.Adaboost.train()：iis[]用來存照片，labels[]則是存是否有車，posNum 紀錄有幾張照片有車，negNum 紀錄幾張照片沒車，記錄完之後陸續跑 buildFeatures()、applyFeatures()（下面會講解），SelectPercentile()則會根據最高分數的百分位選擇特徵。接著這個 for 迴圈是在初始化權重，一開始，我們沒有任何可作為權重的基礎，所以每個 train 的權重都相同，最後則是要跑 T 次迴圈，在選擇最佳 classifier 前，要先對 normalize 權重，接著 SelectBest() 會回傳 bestClf 和 bestError，在每次的 iteration 裡選出最好的 weak classifier(clf)，如果 clf 分類出來的結果符合標準答案，correctness=0，不符合 correctness 則為 1，並將 correctness 存進 accuracy[]，最後再使用其誤差來更新權重。

```
def train(self, dataset):
    """
    Trains the Viola Jones classifier on a set of images.
      Parameters:
        dataset: A list of tuples. The first element is the numpy
          array with shape (m, n) representing the image. The second
          element is its classification (1 or 0).
    """
    print("Computing integral images")
    posNum, negNum = 0, 0
    iis, labels = [], []
    for i in range(len(dataset)):
        iis.append(utils.integralImage(dataset[i][0]))
        labels.append(dataset[i][1])
        if dataset[i][1] == 1:
            posNum += 1
        else:
            negNum += 1
    print("Building features")
    print(iis[0].shape)
    features = self.buildFeatures(iis[0].shape)
    print("Applying features to dataset")
    featureVals = self.applyFeatures(features, iis)
    print("Selecting best features")
    indices = SelectPercentile(f_classif, percentile=10).fit(featureVals.T, labels).get_support(indices=True)
    featureVals = featureVals[indices]
    features = features[indices]
    print("Selected %d potential features" % len(featureVals))

    print("Initialize weights")
    weights = np.zeros(len(dataset))
    for i in range(len(dataset)):
        if labels[i] == 1:
            weights[i] = 1.0 / (2 * posNum)
        else:
            weights[i] = 1.0 / (2 * negNum)
    for t in range(self.T):
        print("Run No. of Iteration: %d" % (t+1))
        # Normalize weights
        weights = weights / np.linalg.norm(weights)
        # Compute error and select best classifiers
        clf, error = self.selectBest(featureVals, iis, labels, features, weights)
        #update weights
        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)
        print("Chose classifier: %s with accuracy: %f and alpha: %f" % (str(clf), len(accuracy) - sum(accuracy), alpha))
```

adaboost.Adaboost.buildFeatures()：訓練需要選擇最好的 weak classifier，但是每種可能的特徵都有一個 weak classifier，所以在訓練之前，我們需要 build features，imageShape 是一組 tuple=(height, width)，一開始先跑過照片中全部的矩形，檢查是否能使用那個特徵，然後存在一個有兩個數組的 tuple 裡，前面是對特徵有正面貢獻的 RectangleRegions，後面是對特徵有負面貢獻的 RectangleRegions。

```
def buildFeatures(self, imageShape):
    """
    Builds the possible features given an image shape.
      Parameters:
        imageShape: A tuple of form (height, width).
      Returns:
        A numpy array of HaarFeature class.
    """
    height, width = imageShape
    features = []
    for w in range(1, width+1):
        for h in range(1, height+1):
            i = 0
            while i + w < width:
                j = 0
                while j + h < height:
                    #2 rectangle features
                    immediate = RectangleRegion(i, j, w, h)
                    right = RectangleRegion(i+w, j, w, h)
                    if i + 2 * w < width: #Horizontally Adjacent
                        features.append(HaarFeature([right], [immediate]))

                    bottom = RectangleRegion(i, j+h, w, h)
                    if j + 2 * h < height: #Vertically Adjacent
                        features.append(HaarFeature([immediate], [bottom]))

                    right_2 = RectangleRegion(i+2*w, j, w, h)
                    #3 rectangle features
                    if i + 3 * w < width: #Horizontally Adjacent
                        features.append(HaarFeature([right], [right_2, immediate]))

                    bottom_2 = RectangleRegion(i, j+2*h, w, h)
                    if j + 3 * h < height: #Vertically Adjacent
                        features.append(HaarFeature([bottom], [bottom_2, immediate]))

                    #4 rectangle features
                    bottom_right = RectangleRegion(i+w, j+h, w, h)
                    if i + 2 * w < width and j + 2 * h < height:
                        features.append(HaarFeature([right, bottom], [immediate, bottom_right]))

                    j += 1
                i += 1
    return np.array(features)
```

adaboost.Adaboost.applFeatures()：找到最好的 weak classifier 後，要評估每個 train 的每個 feature，所以跑過兩個 for 迴圈並且算出 feature 值，如果在開始訓練之前就跑一次這個部分，就可以少跑很多次，加快訓練速度，因為照片的每個 feature 值不會改變。

```
def applyFeatures(self, features, iis):
    """
    Maps features onto the training dataset.8
      Parameters:
        features: A numpy array of HaarFeature class.
        iis: A list of numpy array with shape (m, n) representing the integral images.
      Returns:
        featureVals: A numpy array of shape (len(features), len(dataset)).
          Each row represents the values of a single feature for each training sample.
    """
    featureVals = np.zeros((len(features), len(iis)))
    for j in range(len(features)):
        for i in range(len(iis)):
            featureVals[j, i] = features[j].computeFeature(iis[i])
    return featureVals
```

classifier.WeakClassifier()：一開始先初始化 feature、threshold、polarity 的值，classify()會回傳 0 或 1 分別代表沒車或有車。

```
class WeakClassifier:
    def __init__(self, feature, threshold=0, polarity=1):
        """
          Parameters:
            feature: The HaarFeature class.
            threshold: The threshold for the weak classifier.
            polarity: The polarity of the weak classifier.(1 or -1)
        """
        self.feature = feature
        self.threshold = threshold
        self.polarity = polarity

    def __str__(self):
        return "Weak Clf (threshold=%d, polarity=%d, %s" % (self.threshold, self.polarity, str(self.feature))

    def classify(self, x):
        """
        Classifies an integral image based on a feature f
        and the classifiers threshold and polarity.
          Parameters:
            x: A numpy array with shape (m, n) representing the integral image.
          Returns:
            1 if polarity * feature(x) < polarity * threshold
            0 otherwise
        """
        return 1 if self.polarity * self.feature.computeFeature(x) < self.polarity * self.threshold else 0
```

(before training weak)

adaboost.Adaboost.selectBest()：先創一個 classifier[]，負責存每個 feature 丟進 WeakClassifier 初始化得到的 clf，接著用一個 for 迴圈跑過整個 classifier[]，如果 clf 經過 classify()分類出來的結果符合標準答案(label)，correctness=0，不改變 error 值，但如果不符合的話，correctness=1，其權重會算進 error 裡，並持續更新 bestError 和 bestClf。

```
    # no train weak
    classifier = []
    for f in features:
        clf = WeakClassifier(f)
        classifier.append(clf)

    bestClf = None
    bestError = float('inf')

    for clf in classifier:
        error = 0
        for img,label,weight in zip(iis,labels,weights):
            correctness = abs(clf.classify(img)-label)
            error += weight*correctness
        error = error / len(features)
        if error < bestError:
            bestError = error
            bestClf = clf

    return bestClf, bestError

def classify(self, image):
```

(after training weak：經過 train weak 後，我發現精確度會提高)

adaboost.Adaboost.selectBest()：一開始先算出沒車(label=0)和有車(label=1)的總權重，接著將對應到的 feature 值對權重進行排序後才跑，找出錯誤率最小的，並更新 best_feature、best_threshold，還要比較 pos_seen 和 neg_seen 誰大，如果是 pos_seen 大，best_polarity=1，反之 best_polarity=0，將 best_feature、best_threshold、best_polarity 傳入 WeakClassifier()，並加進 classifier[]，接著和上面一樣，用一個 for 迴圈跑過整個 classifier[]，並持續更新 bestError 和 bestClf。

```
allpos = 0
allneg = 0
for label,weight in zip(labels,weights):
    if label == 0:
        allneg += weight
    elif label == 1:
        allpos += weight

classifier = []
allfeatures = featureVals.shape[0]

for index,feature in enumerate(featureVals):
    pos_seen = 0
    neg_seen = 0
    pos_weight = 0
    neg_weight = 0
    min_error = float('inf')
    best_feature = None
    best_threshold = None
    best_polarity = None

    applied_feature = sorted(zip(weights,feature,labels), key=lambda x: x[1])
    for weight,fea,label in applied_feature:
        error = min(neg_weight+allpos-pos_weight,pos_weight+allneg-neg_weight)

        if error < min_error:
            min_error = error
            best_feature = features[index]
            best_threshold = fea
            if pos_seen>neg_seen:
                best_polarity = 1
            else:
                best_polarity = -1

        if label == 1:
            pos_seen += 1
            pos_weight += weight
        else:
            neg_seen += 1
            neg_weight += weight

    clf = WeakClassifier(best_feature,best_threshold,best_polarity)
    classifier.append(clf)
```

Part 3：change parameter T

1. Weak Classifier 未經訓練的 Adaboost 在 T=3 時的準確率最高，Weak Classifier 經過訓練的 Adaboost 則是在 T=9 時的準確率最高，所以之後和 Yolov5 比較時，我會以這樣的結果下去進行比較。
2. 無論在 train data 還是 test data，甚至是 Weak Classifier 訓練與否，在 T=2,4,6,8,10(偶數)時，都會有 accuracy 較低的問題，我認為是因為在奇數次和偶數次的時候分別會讓結果往回歸直線的兩邊移動，然後剛好結果是比較靠近奇數次移動的那邊，所以精確度才會比較高，但隨著 train 的次數增加，最後應該會逐漸往中間靠近，在 Weak Classifier 經過訓練的 Adaboost 數據中，也可以發現這樣的現象，雖然偶數還是偏低，但其實有慢慢提升其準確率。
3. 在 Weak Classifier 未經訓練的 Adaboost 裡，T=2 的時候 FP rate 是 100%，但在 Weak Classifier 經過訓練後似乎好了些，沒有讓結果如此偏激。

(before training weak)

training	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
FP rate(%)	8.33	100	10.0	40.3	47.6	75.6	13.0	77.0	17.3	50.3
FN rate(%)	29.0	0.00	17.0	22.6	6.33	19.6	25.3	12.3	16.3	27.0
Accuracy(%)	81.3	50.0	86.5	68.5	73.0	52.3	80.8	55.3	83.1	61.3

test	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
FP rate(%)	8.00	100	12.3	36.6	44.3	69.3	20.6	71.0	24.3	52.3
FN rate(%)	30.3	0.33	16.6	20.0	6.66	16.0	20.6	9.00	14.3	22.3
Accuracy(%)	80.8	49.8	85.5	71.6	74.5	57.3	79.3	60.0	80.6	62.6

(after training weak)

training	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
FP rate(%)	3.33	57.6	1.00	30.6	0.00	25.3	0.33	22.6	0.00	24.0
FN rate(%)	19.0	2.66	6.33	4.00	4.33	1.66	4.66	0.66	4.33	0.00
Accuracy(%)	88.8	69.8	96.3	82.6	97.8	86.5	97.5	88.3	97.8	88.0

test	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
FP rate(%)	3.33	54.3	3.33	27.6	3.00	23.0	3.00	19.6	3.00	23.0
FN rate(%)	16.0	3.66	12.0	9.00	10.0	5.66	10.3	6.66	10.0	4.00
Accuracy(%)	90.3	71.0	92.3	81.6	93.5	85.6	93.3	86.8	93.5	86.5

Part 4：detection

main()：將 clf 進行偵測

```
# Part 4: Implement detect function in detection.py and test the following code.
print('\nUse your classifier with video.gif to get the predictions (one .txt and one .png)')
detection.detect('data/detect/detectData.txt', clf)
```

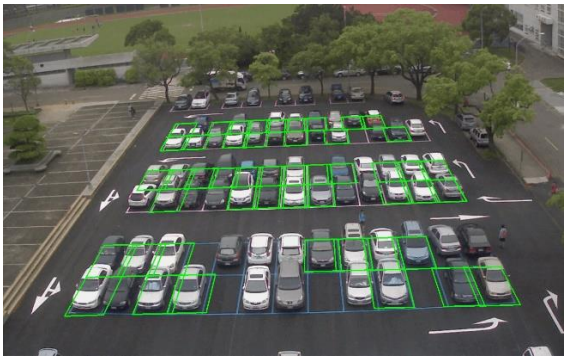
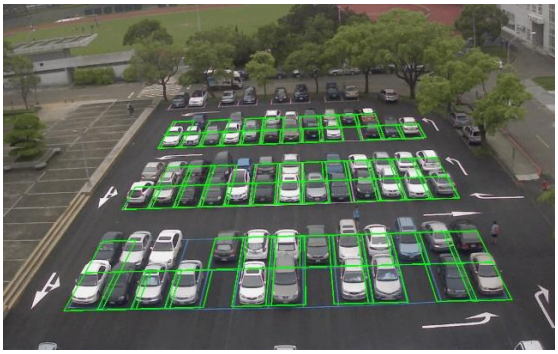
detection.detect()：讀取 detectData.txt，將每個座標都記下來(coordinate)，接著讀取 video.gif 的每個 frame，將 frame 進行 crop，再轉成灰階和 resize 成 36*12，接著丟進 classify()判斷是否有車，如果有車(label=1)根據座標畫出綠色框框，並將 label 紀錄在 Adaboost_pred.txt。

```
# Begin your code (Part 4)
coordinate = {}
coordinate['x1'] = []
coordinate['y1'] = []
coordinate['x2'] = []
coordinate['y2'] = []
coordinate['x3'] = []
coordinate['y3'] = []
coordinate['x4'] = []
coordinate['y4'] = []

file = open(dataPath,'r')
parking_space = int(file.readline())

for line in file.readlines():
    num = line.split(" ")
    coordinate['x1'].append(int(num[0]))
    coordinate['y1'].append(int(num[1]))
    coordinate['x2'].append(int(num[2]))
    coordinate['y2'].append(int(num[3]))
    coordinate['x3'].append(int(num[4]))
    coordinate['y3'].append(int(num[5]))
    coordinate['x4'].append(int(num[6]))
    coordinate['y4'].append(int(num[7]))

video = cv2.VideoCapture('data/detect/video.gif')
f = open('Adaboost_pred.txt','w')

while(1):
    ret, img = video.read()
    if ret == False:
        break
    for i in range(parking_space):
        x1 = coordinate['x1'][i]
        y1 = coordinate['y1'][i]
        x2 = coordinate['x2'][i]
        y2 = coordinate['y2'][i]
        x3 = coordinate['x3'][i]
        y3 = coordinate['y3'][i]
        x4 = coordinate['x4'][i]
        y4 = coordinate['y4'][i]
        crop_img = crop(x1,y1,x2,y2,x3,y3,x4,y4,img)
        crop_img = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
        crop_img = cv2.resize(crop_img,(36,16))
        label = clf.classify(crop_img)
        f.write(str(label))
        f.write(' ')
        if label == 1:
            points = np.array([ [x1,y1],[x2,y2],[x4,y4],[x3,y3]], np.int32)
            cv2.polylines(img,[points],True,(0,255,0),2)
    f.write('\n')
    cv2.imshow('img',img)
    cv2.waitKey(0)

video.release()
cv2.destroyAllWindows()
```

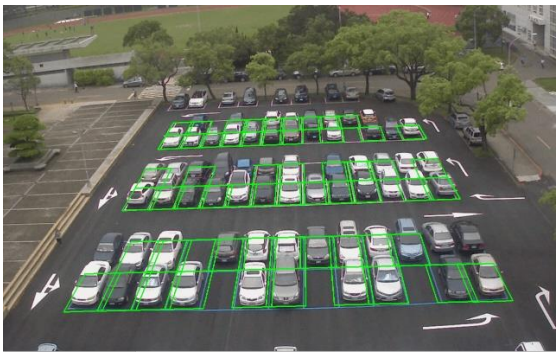
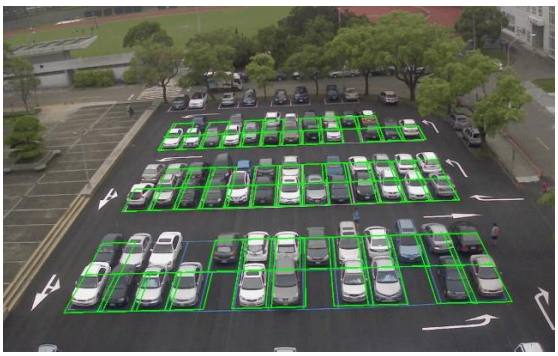
Part 5：Compare Adaboost and Yolov5

- First frame

下圖分別是 Adaboost 和 Yolov5 的第一張 frame，無論是在有車或是沒有車的情況下，都可以看到 Yolov5 很明顯比 Adaboost 還要來的精準許多。

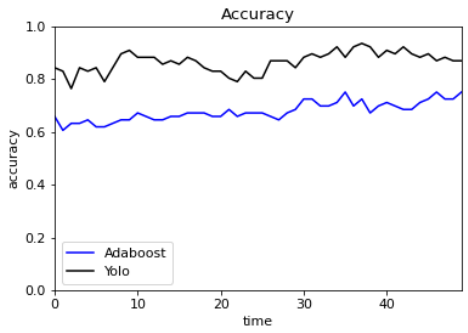
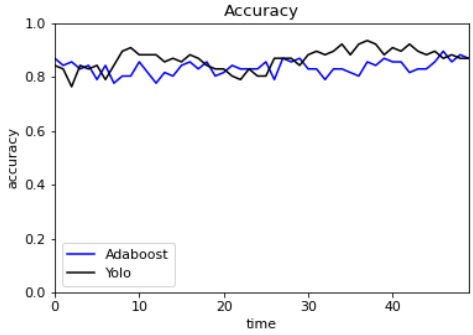
Adaboost	Yolov5
	

下圖兩張圖則是經過訓練的 Adaboost 和 Yolov5 的第一張 frame，無論是在有車或是沒有車的情況下，都可以看到 Yolov5 很明顯比 Adaboost 還要來的精準許多。

Adaboost(after training weak)	Yolov5
	

- Accuracy

下表是 Adaboost 的 Weak Classifier 經過訓練前和訓練後各自和 Yolov5 比較精準度，為了方便比較，我使用了同一個 Yolov5 的數據，可以發現經過訓練的 Adaboost 準確率幾乎和 Yolov5 差不多，大約是 90%，但未經訓練的明顯低於 Yolov5，約只有 60%，可見 weak classifier 經過訓練會提升不少的準確度。

Adaboost(before training weak)	Adaboost(after training weak)
	

- F1-score

$$recall = \frac{TP}{TP + FN} \qquad precision = \frac{TP}{TP + FP} = +P$$

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

觀察一下圖表，我發現 Accuracy 高的時候，基本上 F1-score 也會比較高。

(before training weak)

因為 T=2 的時候，TP 和 FN 都是 0，所以不能算 recall 和 F-score

training data	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
precision	0.9167	0	0.9	0.597	0.524	0.244	0.87	0.23	0.827	0.497
recall	0.759675	#DIV/0!	0.841121	0.725395	0.892219	0.554545	0.774711	0.651558	0.835354	0.647979
F-score(%)	83.08334	#DIV/0!	86.95652	65.49643	66.02407	33.88889	81.95949	33.99852	83.11558	56.25354

test data	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
precision	0.92	0	0.877	0.634	0.557	0.307	0.794	0.29	0.757	0.477
recall	0.752249	0	0.840844	0.760192	0.893201	0.657388	0.794	0.763158	0.841111	0.681429
F-score(%)	82.77103	#DIV/0!	85.85414	69.1385	68.61296	41.85412	79.4	42.02899	79.68421	56.11765

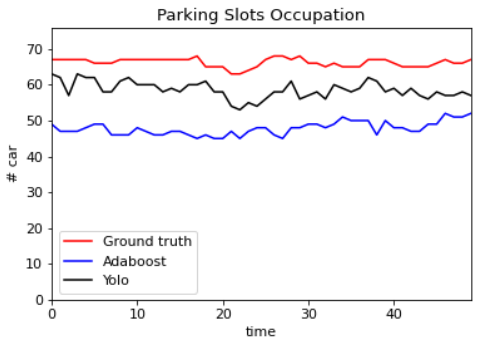
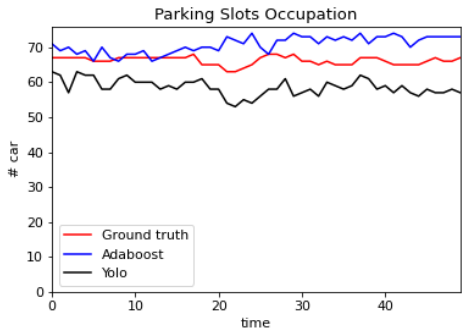
(after training weak)

training data	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
precision	0.9667	0.424	0.99	0.694	1	0.747	0.9967	0.774	1	0.76
recall	0.83574	0.940968	0.939903	0.945504	0.958497	0.978261	0.955334	0.991545	0.958497	1
F-score(%)	89.64622	58.45857	96.43014	80.04614	97.88088	84.71309	97.55787	86.93699	97.88088	86.36364

test data	T=1	T=2	T=3	T=4	T=5	T=6	T=7	T=8	T=9	T=10
precision	0.9667	0.457	0.9667	0.724	0.97	0.77	0.97	0.804	0.97	0.77
recall	0.857992	0.925851	0.889574	0.889435	0.906542	0.931527	0.904007	0.923501	0.906542	0.950617
F-score(%)	90.9108	61.19443	92.65347	79.82359	93.71981	84.30965	93.58418	85.96172	93.71981	85.08287

Parking slots occupation

紅色那條是 Ground truth，我們可以發現 Yolov5(黑色)的線略低於 Ground truth，這可能是因為 threshold 過高所導致結果，左圖是 Weak Classifier 未經過訓練的 Adaboost，右圖則是 Weak Classifier 有經過訓練的 Adaboost(藍色)，雖然說右圖的藍色略高於 Ground truth，但還是可以看出他和 Ground truth 的差距遠小於左圖。

Adaboost(before training weak)	Adaboost(after training weak)
	

Problem

1. 一開始在 Part1 一直沒辦法正確匯入圖片，找很久才發現是路徑的問題，for file in allfiles 那邊的 file 不是完整路徑，還是再加上前面那段。
2. part2 那邊一開始完全沒有想法，是後來慢慢看了很多有關 Adaboost 資料，還一行一行程式碼慢慢對照，才慢慢有個雛型出來，從一開始寫出 selectBest()那邊，到後來訓練 Weak Classifier，提升 Adaboost 的準確率，過程中測試花了蠻長的時間，才做出想要的結果。
3. 因為我想要比較 Weak Classifier 訓練與否和 FP 率、FN 率、Accuracy 是否有關，所以在跑參數(T=1~T=10)這方面我跑了兩次，花了很長的時間才得到需要的數據。
4. 因為之前沒用過 Colab，所以反覆開了檔案好幾次才弄好，後來在跑很像 part4 那個 cell 的時候，一直有個錯不知道該如何處理，後來看討論區有人問助教，才知道是版本的問題，後來加上!git checkout 9931f56 這行程式碼，果然可以順利執行了。
5. 第一次執行程式的時候，我的 Yolov5_pred.txt 是空白的，看到討論區助教說要等一下，但等了五分鐘，還是一直都沒出來，結果再跑一次就好了，很奇怪，不知道是環境的問題還是哪裡出了差錯。
6. 在算 Weak Classifier 未經訓練的 Adaboost 的 F1-score 時，因為 T=2 的 TP 和 FN 都是 0，所以不能算 recall 和 F-score。