Task A

Part 1: Load and prepare your dataset

將 datapath /car 的資料匯入 dataset:

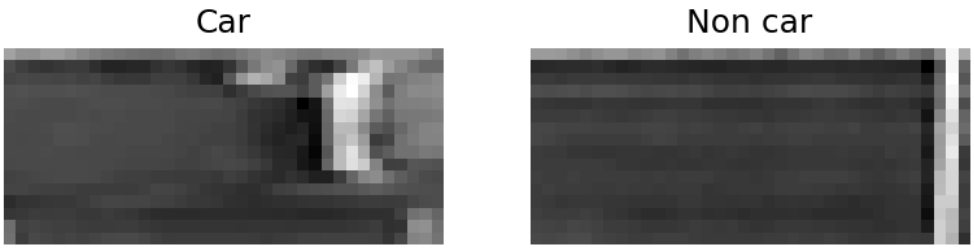
```
path1=data_path+'/car'
dataset=[]

for filename in os.listdir(path1):

    if filename.endswith('.png'):
        image=cv2.imread(os.path.join(path1,filename))
        image=cv2.resize(image,(36,16), interpolation=cv2.INTER_AREA)
        img=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        label=1

        dataset.append((img,label))
```

調整 image 大小並轉成 grayscale，變成 36*16 的 numpy array

將 label 設 1

最後把這兩者存入 dataset

Non-car 的資料也用相同的方式，只是 label=0



Part 2: Build and Train Models

2-1. __init__ :

(train data)

```
Xtrain=[]
Ytrain=[]
test=[]
for i in range(len(train_data)):

    Xtrain.append(np.ravel(train_data[i][0]))
    Ytrain.append(train_data[i][1])


self.x_train=np.asarray(Xtrain)
self.y_train=np.asarray(Ytrain)
```

Test data 也是用相同的方式。

2-2.build_model:依照 model name 做出相應的 model

```
if model_name=="KNN":
    clf=KNeighborsClassifier(n_neighbors=3)
    return clf

elif model_name=="RF":
    clf=RandomForestClassifier(n_estimators=100,max_depth=2)
    return clf

else:
    clf=AdaBoostClassifier(n_estimators=100,random_state=0)
    return clf
```

2-3. train:使用 fit 進行 train

```
self.model.fit(self.x_train, self.y_train)
```

○ Explain the difference between parametric and non-parametric models.

Parametric models 的數據分布可以用一些參數決定，在機器學習時有指定目標函數，而 non-parametric models 對目標函數沒有做太多假設，經過訓練數據多次後推出才推出函數。

Parametric model 執行的速度較快，需要的數據較少，較容易發生 underfitting；non-parametric models 彈性較高，可運算較複雜的數據，較容易發生 overfitting。

○ What is ensemble learning? Please explain the difference between bagging, boosting and stacking.

Ensemble learning 一次使用多個學習演算法，以找出比單獨使用一種演算法還好的預期表現。是由有限且可變動的 models 組成。

Bagging： 從原始的資料隨機抽取組成 bootstrap dataset，用不同的模型訓練，最後進行投票決定最終結果，每個模型有相同權重。

Boosting： 和 Bagging 不同的點是，每個模型之間有關聯。一開始每個數據權重相同，訓練完後，錯誤分類的資料會得到比較高的權重，也就是在下一個模型會加強錯誤資料的訓練。在某些情況，已證明 Boosting 的 accuracy 高於 Bagging，但也更容易 over-fitting。

Stack： 先把數據輸入多個 base learners，再把 base learner 的輸出匯入 combiner algorithm，由他訓練出最終預測結果。Stack 產生的性能通常比其他模型好。

○ Explain the meaning of the “n_neighbors” parameter in KNeighborsClassifier, “n_estimators” in RandomForestClassifier and AdaBoostClassifier.

n_neighber in KNeighborClassifier： 進行 kneighbors 查詢時，需考慮的 neighbors 數量。

n_estimators in RandomForestClassifier： 在 forest 裡有幾個 trees

n_estimators in AdaBoostClassifier : boosting 結束的 estimater 的最大數量。

○ Explain the meaning of four numbers in the confusion matrix.

[[True Positive(預測正實際正), False Negative(預測負實際正)]

[False Positive(預測正實際負), True Negative(預測負實際負)]]

○ In addition to “Accuracy”, “Precision” and “Recall” are two common metrics in classification tasks, how to calculate them, and under what circumstances would you use them instead of “Accuracy

Accuracy = (True Positive+ True Negative) /total

Precision =True Positive/(True Positive + False Positive)

Recall =True Negative/(True Negative + False Negative)

當正負樣本數量不平均時， Accuracy 高可能是因為 model 對單一類型的辨識能力較好，同時那個類型比例很高，此時 Precision 和 Recall 更能反映 model 的表現。

Part 3: Additional experiments

KNN:

n_neighbors=3

```
Accuracy: 0.9133
Confusion Matrix:
[[300   0]
 [ 52 248]]
```

n_neighbors=2

```
Accuracy: 0.8833
Confusion Matrix:
[[300   0]
 [ 70 230]]
```

n_neighbors=4

```
Accuracy: 0.86
Confusion Matrix:
[[300   0]
 [ 84 216]]
```

KNN 選擇用 n_neighbors=3 detect

RF:

n_estimators=100, max_depth=2, random_state=0

```
Accuracy: 0.9233
Confusion Matrix:
[[282  18]
 [ 28 272]]
```

n_estimators=90, max_depth=2, random_state=0

```
Accuracy: 0.925
Confusion Matrix:
[[282  18]
 [ 27 273]]
```

n_estimators=80, max_depth=2, random_state=0

```
Accuracy: 0.9217
Confusion Matrix:
[[282  18]
 [ 29 271]]
```

RF 選擇用 n_estimators=90, max_depth=2, random_state=0 detect

AB:

n_estimators=3 ,random_state=0

```
Accuracy: 0.92
Confusion Matrix:
[[261  39]
 [  9 291]]
```

n_estimators=4 ,random_state=0

```
Accuracy: 0.9067
Confusion Matrix:
[[295   5]
 [ 51 249]]
```

n_estimators=2 ,random_state=0

```
Accuracy: 0.9017
Confusion Matrix:
[[261  39]
 [ 20 280]]
```

AB 選擇用 n_estimators=4 ,random_state=0 detect

Part 4: Detect car

在 Gif 的每一幕，將 detectData.txt 裡讀到的座標跟 frame 送到 crop 剪出停車場，把停車場照片調整大小，轉成 grayscale，變成 1d，傳到 clasify

如果回傳的結果是 1，代表有偵測到車，在該停車位畫綠框。並把結果存下來。

Gif 全部讀完後，建立 ML_Models_pred.txt，把儲存的結果寫入 txt 裡。

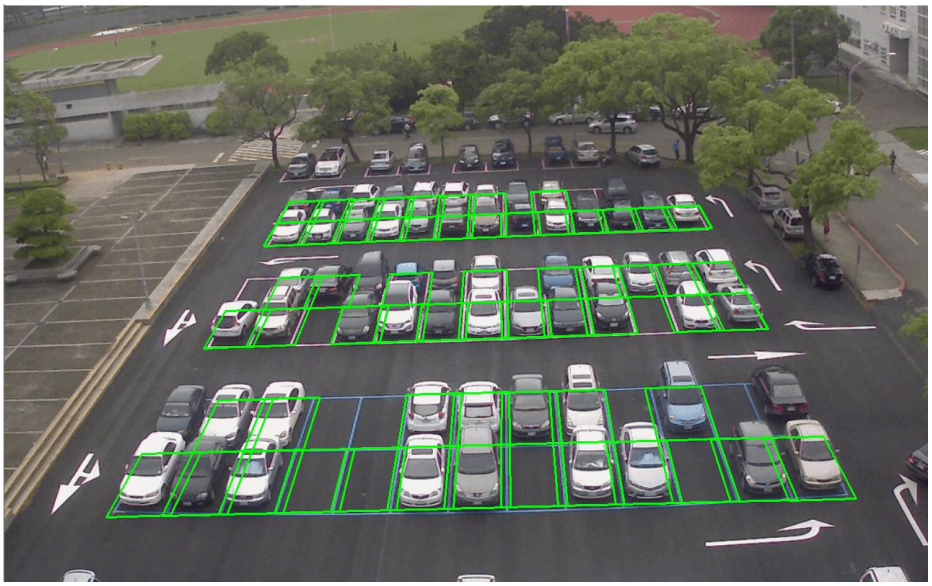
```
cap=cv2.VideoCapture("data/detect/video.gif")
txt=[]
while True:
    f = open(data_path, 'r')
    ret,frame=cap.read()
    if ret:
        n=int(f.readline())

        temp=[]
        for i in range(n):
            #print(n)
            L=f.readline()
            cr=list(map(int, L.split()))

            parkingspace=crop(cr[0],cr[1],cr[2],cr[3],cr[4],cr[5],cr[6],cr[7],frame)
            parkingspace=cv2.resize(parkingspace,(36,16), interpolation=cv2.INTER_AREA)
            img=cv2.cvtColor(parkingspace,cv2.COLOR_BGR2GRAY)
            parkingspace_test=[]
            parkingspace_test.append(np.ravel(img))
            car=clf.classify(parkingspace_test)
            temp.append(car)
            if car==1:
                green_color = (0, 255, 0)
                cv2.line(frame, (cr[0],cr[1]), (cr[2], cr[3]), green_color, 2)
                cv2.line(frame, (cr[2],cr[3]), (cr[6], cr[7]), green_color, 2)
                cv2.line(frame, (cr[6],cr[7]), (cr[4], cr[5]), green_color, 2)
                cv2.line(frame, (cr[0],cr[1]), (cr[4], cr[5]), green_color, 2)


        txt.append(temp)
        cv2.imshow("show",frame)
        f.close()

        key=cv2.waitKey(1) & 0xff
        if key==ord('q'):
            break
    else:
        break
```

Conservation:

RandomForestClassifier(n_estimators=90, max_depth=2, random_state=0
)

```
Accuracy: 0.9267
Confusion Matrix:
[[282  18]
 [ 26 274]]
```

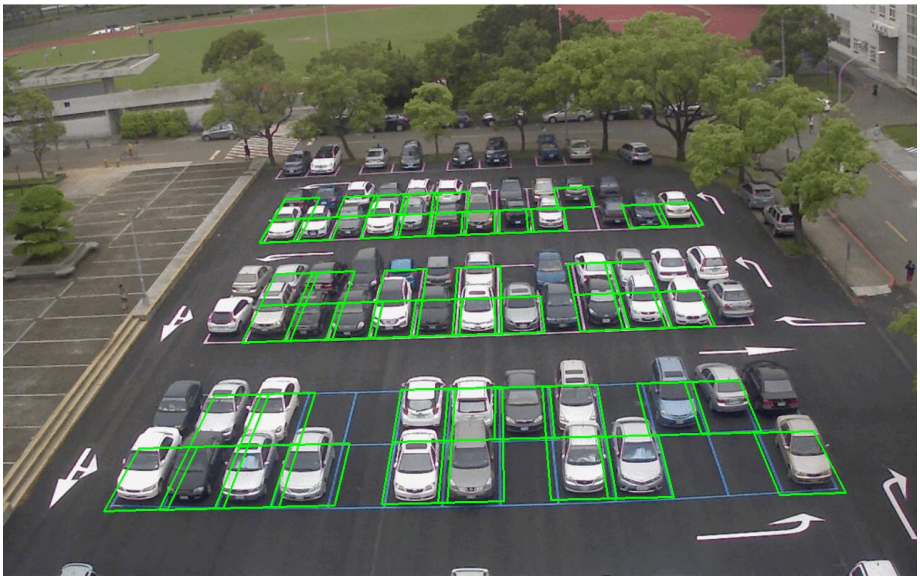
雖然 accuracy 高，但會把部分空的停車場判斷成有車，有些黑色的車偵測不到，會判斷成空位。



KNeighborsClassifier(n_neighbors=3)

```
Accuracy: 0.9133
Confusion Matrix:
[[300   0]
 [ 52 248]]
```

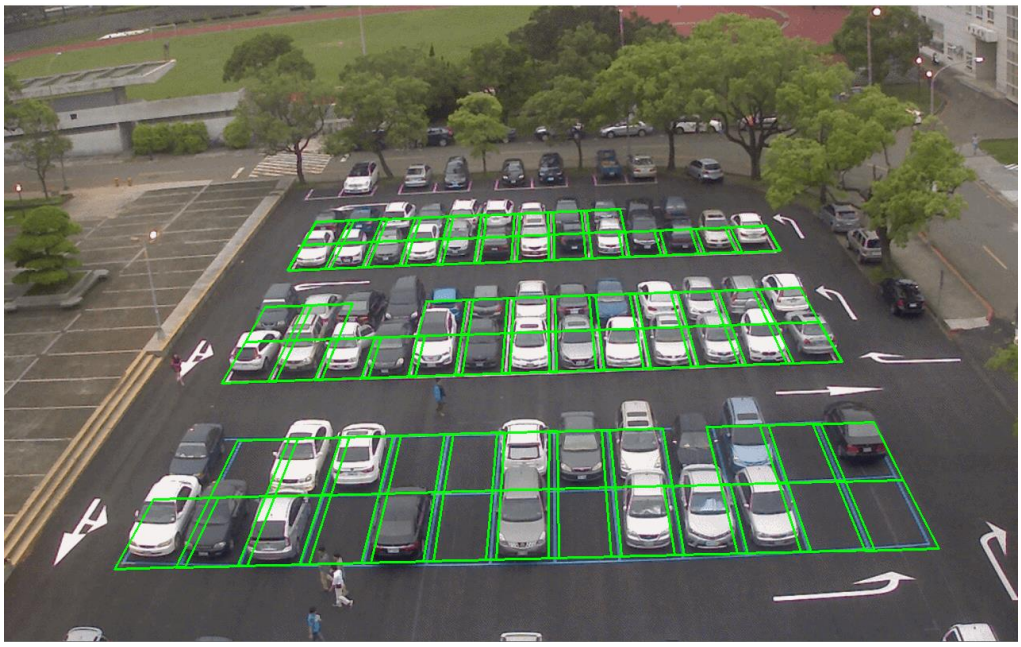
不會把空的停車位判斷成有車，但相對於上者，把有車的停車場判斷成沒車的比例較高，比對兩者的 Confusion Matrix，可以發現這個 model 的 FP 是 0，但 FN 比較大。



AdaBoostClassifier(n_estimators=3,random_state=0)

```
Accuracy: 0.92
Confusion Matrix:
[[261  39]
 [  9 291]]
```

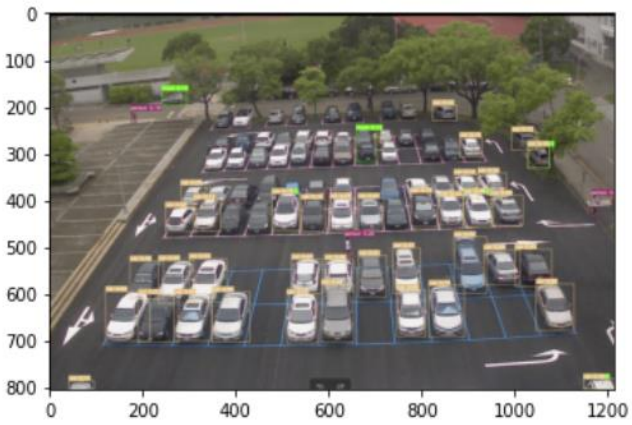
比前面 Random 的 model，有更多被判斷成有車的空車位



Task B

Part 1: Load a pretrained model and directly apply it to the .png to detect the object.

修改路徑後，執行儲存格，得到結果:



Part 2: Learn to fine tuning yolov7 model

將參數設為：

```
!python train.py  --device 0 --epoch 5 --batch-size 1 --img-size 360 160 --data HW1_material/hw1.yaml --weights 'yolov7-tiny-custom' --exist-ok --cfg cfg/training/yolov7-tiny.yaml --name yolov7-tiny --hyp data/hyp.scratch.custom.yaml
```

得到結果 ：

Apply on training data:

```
False Positive Rate: 30/300 (0.100000)
False Negative Rate: 25/300 (0.083333)
Training Accuracy: 545/600 (0.908333)
```

Apply on testing data:

```
False Positive Rate: 25/300 (0.083333)
False Negative Rate: 13/300 (0.043333)
Training Accuracy: 562/600 (0.936667)
```

Problems I met and solution:

1. 若把圖片的 numpy array 直接匯入 self.x_train，執行到 fit 時，會有維度不符合規定的錯誤訊息。

解決方法 ： 用 ravel 將圖片變成 1d array

2.不管怎麼改變參數，Calculate 的結果都沒改變

解決方法：發現忘記在 `detect.py` 加上 `–weights`