Introduction to AI

109350008　張詠哲

Task A

- **Part 1**

這裡 data_path 讀進來後還要再看是 car 還是 non-car 資料夾，之後要記得把接下來的檔名加再 data_path 上，一直到讀到 png 檔，再把圖片轉為灰階以及把它變為 36 x 16 的圖像，存進 dataset 中並且加上 label (car = 1, non-car = 0)。

```
dataset = []
for filename in os.listdir(data_path):
    if(filename == "car"):
        dir = os.path.join(data_path, filename)
        for img_name in os.listdir(dir):
            img_dir = os.path.join(dir, img_name)
            img = cv2.imread(img_dir)
            img_gray = cv2.cvtColor(img, cv2.COLOR_RGBA2GRAY)
            img_gray_resize = cv2.resize(img_gray, (36, 16))

            dataset.append((img_gray_resize, 1))
    if(filename == "non-car"):
        dir = os.path.join(data_path, filename)
        for img_name in os.listdir(dir):
            img_dir = os.path.join(dir, img_name)
            img = cv2.imread(img_dir)
            img_gray = cv2.cvtColor(img, cv2.COLOR_RGBA2GRAY)
            img_gray_resize = cv2.resize(img_gray, (36, 16))

            dataset.append((img_gray_resize, 0))
# End your code (Part 1)
return dataset
```

這是把 dataset 的第一張以及最後一張印出來的結果:

Car

Non car

- **Part 2**

由於傳進 model 的 dataset 為 list 因此要把他轉為 sklearn 中 model 可以接受的 nparray，要記得把 X reshape 為一個 2D 的資料集(sample, width*width)。

```
self.x_train, self.y_train, self.x_test, self.y_test = None, None, None, None

# Begin your code (Part 2-1)
#raise NotImplementedError("To be implemented")
self.x_train = np.array([data[0] for data in train_data]).reshape(len(train_data), -1)
self.y_train = np.array([data[1] for data in train_data])
self.x_test = np.array([data[0] for data in test_data]).reshape(len(test_data), -1)
self.y_test = np.array([data[1] for data in test_data])
# End your code (Part 2-1)

self.model = self.build_model(model_name)
```

依照輸進來的 model 參數，回傳對應的 model

```
if model_name == "KNN":
    model = KNeighborsClassifier(n_neighbors= 1)
elif model_name == "RF":
    model = RandomForestClassifier()
elif model_name == "AB":
    model = AdaBoostClassifier(n_estimators=200, learning_rate=0.1)

return(model)
```

把資料以及 label 丟進選好的 model 訓練。

```
Fit the model on training data (self.x_train and
'''
# Begin your code (Part 2-3)
#raise NotImplementedError("To be implemented")
self.model.fit(self.x_train, self.y_train)
# End your code (Part 2-3)
```

以下為問題的回答:

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

參數模型需要假設數據的分布屬於某種特定的概率分布，而非參數模型不需要對數據分布做出任何假設。一般來說參數模型會比非參數模型更容易 overfitting，且非參數模型通常會有更好的表現，但是參數模型一般會有更好的解釋性。

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

是只由很多個 base model 所集合而成的大模型，用於提高表現。以下為實現方法的介紹:

1. bagging:

從 training dataset 中隨機挑出多種子樣本，再由每種 base model 去訓練出一個基本的模型，最後再將這些模型所輸出的結果平均或是多數決的方式得出最終預測。

2. boosting

和 bagging 最大的差異就是他的 base model 權重皆不一樣，給錯誤率小的模型較大的權重，給錯誤率較大的模型較小的權重。最後所有模型投票得出最後結果。

3. stacking

產出 m 種不同演算法的 base model(且彼此無相互關聯)，再將這些模型的輸出當作新模型的輸入再訓練一個新的模型。

○ **Explain the meaning of the "n_neighbors" parameter in KNeighborsClassifier, "n_estimators" in RandomForestClassifier and AdaBoostClassifier.**

n_neighbors:

決定每一個新的資料點所要找出離自己最近的 k 個 neighbor。

N_estimator:

n_estimator 決定說要用幾個 base model。

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

1. TP: 實際上為 True, 且預測為 Positive (和實際相同)
2. FP: 實際上為 False, 且預測為 Positive (和實際不同)
3. TN: 實際上為 True, 且預測為 Negative (和實際相同)
4. FN: 實際上為 False, 且預測為 Negative (和實際不同)

○ **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".**

1. **Precision:**

   TP / (TP + FP)，用於需要正例預測較好的時候的情況

2. **Recall**

   TP / (TP + FN)，用於需要反例預測較好的情況下，像是醫學診斷為了減少誤判，因此需要優先考慮 Recall

- **Part 3**

這部分要去試各種模型的超參數，使自己的 model 可以有更好的表現，以下是我測試幾次之後找出的比較好的超參數:

KNN:

調整 n_neighbor，是決定每一個新的資料點所要找出離自己最近的 k 個

neighbor，這裡我試了幾個後發現 n_neighbor = 1 時的結果(accuraccy)最高，n_neighbor 越大 accuraccy 會有稍微的下降。

RF:

要調整 n_estimator。由於 RandomForest 是集成模型，意思是由好幾個小 decision tree 所投票表決出來的，因此可以由 n_estimator 決定說要用幾棵樹來表決，越多的 training 結果會越好，但也因此容易 overfitting。這裡我從 100 開始試後發現當為 200 時 accuraccy 會較高(約為 0.97 多)，再更大效果差不多或是有點下降。

AB:

和 RF 一樣要調整 n_estimator，但是我也多調整了 learning rate，可以調整梯度收斂的速度，太大容易錯過最佳解或是不會收斂，而太小會訓練太慢，這裡我挑的是 0.1，以及 n_estimator = 200

且發現 RF、AB 的表現均比 KNN 好，主要是因為 RF 以及 AB 皆為集成學習。

```
if model_name == "KNN":
    model = KNeighborsClassifier(n_neighbors= 1)
elif model_name == "RF":
    model = RandomForestClassifier(n_estimators=200)
elif model_name == "AB":
    model = AdaBoostClassifier(n_estimators=200, learning_rate=0.1)

return(model)
```

- **Part 4**

讀檔後，讀取 GIF 檔的每一個 frame，並且對每一個 frame 的每一個停車格做偵測有沒有車，若是有車把停車格用綠線框起來(位置再 detectData.txt)，會有 f.seek(0,0)是因為當檔案讀取完後指針會指到檔案末尾，因此每到要偵測新的 frame 時需要把他的指針移到檔案開頭(0,0)的位置。而 first_time 是因為只有要顯示出第一張 frame 的結果，因此用這個做為 check。也要把預測的結果寫入 ML_Model_pred.txt 中

```
f = open(data_path)
gif = cv2.VideoCapture("data\detect\\video.gif")
pf = open("ML_Models_pred.txt", "a")
first_time = 1

while True:

  ret, frame = gif.read()
  if not ret:
    break

  predictions = ""
  f.seek(0,0)
  row = f.readline()
  for i in range(0, int(row) - 1):
    temp = f.readline()
    temp = temp.strip("\n").split(" ")
    x1, y1, x2, y2, x3, y3, x4, y4 = temp[0], temp[1], temp[2], temp[3], temp[4], temp[5], temp[6], temp[7]
    img = crop(x1, y1, x2, y2, x3, y3, x4, y4, frame)
    img = cv2.resize(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), (36, 16))
    if (clf.classify(img.flatten().reshape(1, -1)) == 1):
      if(first_time):
        pts = np.array([[int(x1), int(y1)], [int(x2), int(y2)], [int(x4), int(y4)], [int(x3), int(y3)]])
        cv2.polylines(frame, [pts], True, (0, 255, 0), thickness=2)
      predictions += "1 "
    else:
      predictions += "0 "
```

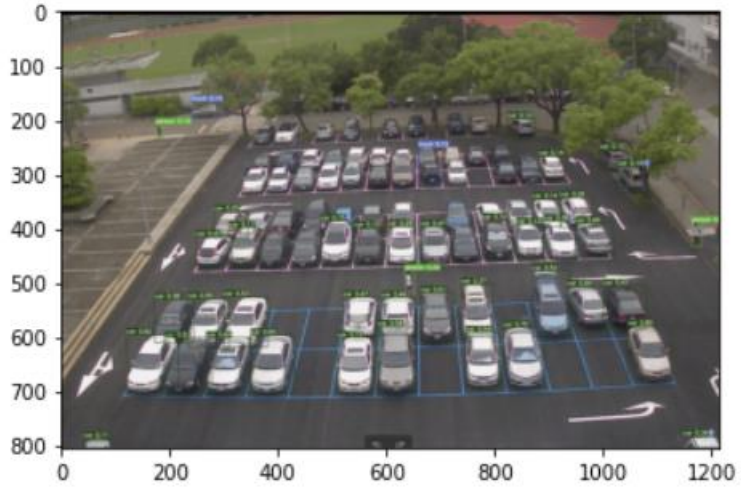
```
  predictions = predictions.strip(" ")
  predictions += "\n"
  #print(predictions)
  pf.write(predictions)
  if (first_time):
    fig, ax = plt.subplots(1, 1)
    ax.imshow(frame)
    plt.show()
    first_time = 0

pf.close()
gif.release()
cv2.destroyAllWindows()
# End your code (Part 4)
```

Task B

- **Part 1**

只要執行助教給的 code 就好。而這是產出的圖片:



- **Part 2**

在 finetune 中我使用的參數為:

--device 0 --epoch 200 --batch-size 64

但是在存最佳的 pt 時，不知道為什麼執行下面那個 cell 後 detect 都偵測不到，

因此我直接用 best.pt。示意圖如下:

```
!python  detect.py  --conf  0.1  --weights  /content/yolov7/runs/train/yolov7-tiny/weights/best.pt
'''
```

以下是最後的表現，皆超過 90%:

1. Training

```
Calculate('/content/yolov7/HW1_material/train/','/content/yolov7/runs/detect/train/labels/')
```

```
False Positive Rate: 21/300 (0.070000)
False Negative Rate: 24/300 (0.080000)
Training Accuracy: 555/600 (0.925000)
```

2. Testing

```
Calculate('/content/yolov7/HW1_material/test/','/content/yolov7/runs/detect/test/labels/')
```

```
False Positive Rate: 16/300 (0.053333)
False Negative Rate: 12/300 (0.040000)
Training Accuracy: 572/600 (0.953333)
```