

國立雲林科技大學  
機器學習專案作業三

結合 YOLO 與 OCR 深度學習模型進行物件偵測

成員：

M11021009 吳宥霆

M11021028 劉軒瑋

M11021035 黃堉豪

M11021052 邱守燦

指導教授：

許中川 教授

日期：

2022 年 6 月 2 日

## 摘要

影像辨識在實務上除了影像分類之外，物件偵測是當今影像辨識領域的重要技術，物件偵測在大量應用在日常生活當中，例如：車牌號碼偵測，本研究以貨櫃屋圖片資料集為例，起初利用 labelImg 軟體標示圖片欲偵測物件的相對位置，接著利用 Alexey 開發者建立在 darknet 架構下的 YOLOv4 進行物件偵測，YOLOv4 可以針對欲進行辨識的相對位置當作目標，選擇 yolo.conv.137 做為目標權重訓練的方法，調整參數後進行訓練，最後依照訓練結果的權重與類別，使用 OCR 進行文字與數字辨識來提升物件偵測的準確度。

關鍵字：物件偵測、YOLO、OCR

## 一、緒論

### 1.1 研究動機

貨櫃屋圖片資料集上方的主要特徵包括數字與文字，若針對這些特徵進行傳統影像辨識技術分類時，其準確度與精確度較低，無法有效進行辨識，但利用 YOLO 與 OCR 物件偵測技術，除了能定位物件之外，也讓數字與文字的辨識更加精確，並透過此份資料集進行物件偵測當作實例。

### 1.2 研究目的

此份資料集為物件偵測值得練習的資料集，在定義物件的類別與名稱時，可完整選出欲進行偵測的區域，並可以將圖片資料格式儲存為 YOLO 格式，方便研究者將資料帶入 YOLO 模型進行訓練，並使用 YOLO 測試顯示偵測位置，最後透過 OCR 進行文字與數字偵測，讓物件偵測經由這些技術得以落實。

## 二、資料集

### 2.1 資料集

#### 2.1.1 貨櫃 Dataset 說明

貨櫃資料集只有 1 個類別，此類別分別包含 20 張訓練資料與 20 張測試資料，每張圖為長\*寬不規則的像素彩色影像。

- 類別數量：1
- 類別標籤：label 數字
- 訓練資料：20 張
- 測試資料：20 張

表 1 貨櫃 Test 資料集示例

1.jpg	2.jpg	9.jpg
		

表 2 貨櫃 Train 資料集示例

3.jpg	4.jpg	14.jpg
		

### 三、方法

#### 3.1 程式架構

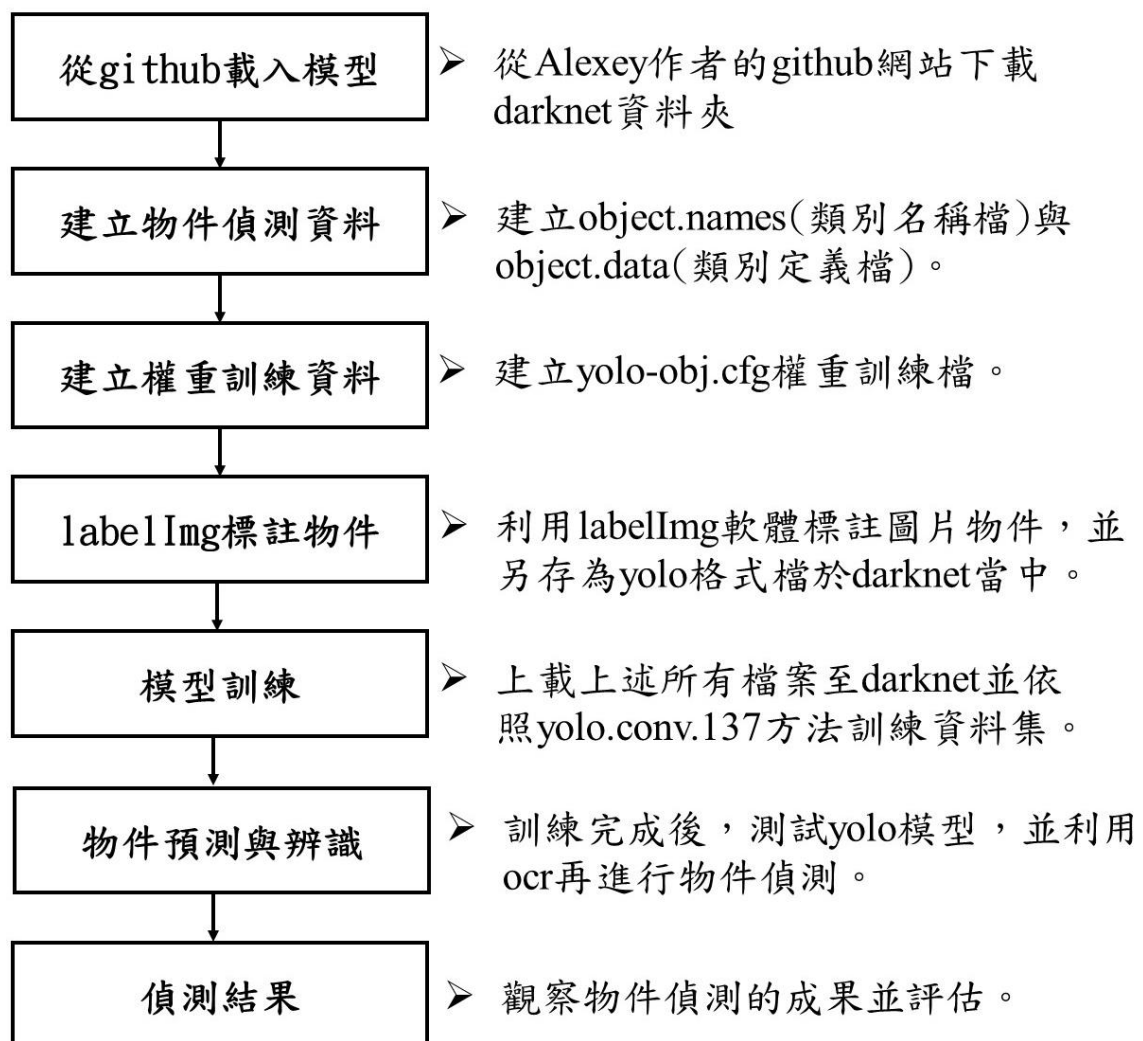


圖 1 程式架構流程圖及說明

## 3.2 程式執行方法

### 3.2.1 Yolo

下載 darknet 資料夾放入電腦中，在 cfg 資料夾中將 yolov4-custom.cfg 檔更名為 yolo-obj.cfg 為自己的資料訓練做事前準備，並更動其參數，接著在 data 資料夾上傳在 labelImg 標註特徵的類別與名稱(obj.names、obj.data)，並定義模型訓練的路徑與權重檔的位置，同時上傳資料集至 data 資料夾，編譯 darknet 並設定環境變數，撰寫程式下載 yolo.conv.137 使用 detector train 語法進行 YOLOv4 的訓練，訓練完成後，透過 detector test 語法進行測試集圖片的預測。

### 3.2.2 光學字元辨識 (OCR)

載入 pytesseract 為 ocr 環境套件、darknet 資料夾、測試資料、經過 YOLO 訓練的權重檔，定義 YOLO 模型、OCR 灰階轉換與字串辨識原理、YOLO 圖片物件辨識過程導入 OCR 字串辨識當中，最後利用測試資料集，透過 OpenCV 利用 OCR 辨識出貨櫃的文字與數字。

## 四、實驗

### 4.1 前置處理

- (1) 安裝 CUDA、Cudnn、Cmake、Opencv 套件
- (2) 使用 Visual Studio 2019 建構 Yolo 環境
- (3) 將 Train 的照片使用 Labelimg 標記特徵
- (4) 將標記好的圖片 VOC 檔轉成 Yolo 訓練時所需的 txt 檔案
- (5) 使用 Visual Studio 2019 建構 Yolo 環境
- (6) 調整 Yolov4 模型訓練參數 class = 80, filter = 255, max\_batch, 分別改為 1, 18, 6000
- (7) 使用顯卡：RTX-3060、CUDA：11.5、Cudnn：8.2.0、GPU：NVIDIA GeForce RTX 3060

```
CUDA-version: 11050 (11050), cuDNN: 8.2.0, GPU count: 1
OpenCV version: 4.5.5
0 : compute_capability = 860, cudnn_half = 0, GPU: NVIDIA GeForce RTX 3060
net.optimized_memory = 0
mini_batch = 1, batch = 16, time_steps = 1, train = 0
  layer  filters  size/strd(dil)    input          output
0 Create CUDA-stream - 0
Create cudnn-handle 0
```

圖 2 Yolov4 設備參數

### 4.2 實驗設計

- (1) 模型訓練：以 YOLOv4-Custom.cfg 架構與 yolov4.vonv.137 預訓練的模型進行訓練
- (2) 模型測試：使用 Test 照片測試 Yolo 正常運作，並將 Yolo 所框選的範圍以 jpg 另外儲存
- (3) 特徵圖片前處理：調整圖片的大小及轉成灰階
- (4) OCR 數字辨識：使用 tesseract 套件進行辨識
- (5) 績效評估：評估模型準確度

表 3 yolov4 網路架構圖

conv 34	3 x 3 / 1	416 x 416	→	208 x 208	64	0.354
1 conv	3 x 3 / 2	416 x 416	→	208 x 208	64	0.354
14 conv	1 x 1 / 1	208 x 208	→	208 x 208	64	0.177
3 route	1	208 x 208	→	208 x 208	64	0.177
5 conv	3 x 3 / 1	208 x 208	→	208 x 208	64	0.354
6 conv	3 x 3 / 1	208 x 208	→	208 x 208	64	0.354
7 Shortcut Layer: 4	vt = 0, va = 0, output: 208 x 208	64	0.003			
8 conv	1 x 1 / 1	208 x 208	→	208 x 208	64	0.177
9 route	8	208 x 208	→	208 x 208	64	0.177
10 conv	1 x 1 / 1	208 x 208	→	208 x 208	64	0.177
11 conv	3 x 3 / 2	208 x 208	→	104 x 104	128	1.595
12 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
13 route	11	104 x 104	→	104 x 104	128	0.177
14 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
15 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
16 conv	3 x 3 / 1	104 x 104	→	104 x 104	128	0.797
17 Shortcut Layer: 14	vt = 0, va = 0, output: 104 x 104	128	0.001			
18 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
19 conv	3 x 3 / 1	104 x 104	→	104 x 104	128	0.797
20 Shortcut Layer: 17	vt = 0, va = 0, output: 104 x 104	128	0.001			
21 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
22 route	21	104 x 104	→	104 x 104	128	0.177
23 conv	1 x 1 / 1	104 x 104	→	104 x 104	128	0.177
24 conv	3 x 3 / 2	104 x 104	→	52 x 52	256	1.595
25 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
26 route	24	52 x 52	→	52 x 52	256	0.177
27 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
28 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
29 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
30 Shortcut Layer: 27	vt = 0, va = 0, output: 52 x 52	256	0.000			
31 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
32 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
33 Shortcut Layer: 30	vt = 0, va = 0, output: 52 x 52	256	0.000			
34 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
35 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
36 Shortcut Layer: 33	vt = 0, va = 0, output: 52 x 52	256	0.000			
37 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
38 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
39 Shortcut Layer: 36	vt = 0, va = 0, output: 52 x 52	256	0.000			
40 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
41 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
42 Shortcut Layer: 39	vt = 0, va = 0, output: 52 x 52	256	0.000			
43 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
44 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
45 Shortcut Layer: 42	vt = 0, va = 0, output: 52 x 52	256	0.000			
46 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
47 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
48 Shortcut Layer: 45	vt = 0, va = 0, output: 52 x 52	256	0.000			
49 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
50 conv	3 x 3 / 1	52 x 52	→	52 x 52	256	0.797
51 Shortcut Layer: 48	vt = 0, va = 0, output: 52 x 52	256	0.000			
52 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
53 route	52	52 x 52	→	52 x 52	256	0.177
54 conv	1 x 1 / 1	52 x 52	→	52 x 52	256	0.177
55 conv	3 x 3 / 2	52 x 52	→	26 x 26	512	1.595
56 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
57 route	55	26 x 26	→	26 x 26	512	0.177
58 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
59 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
60 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
61 Shortcut Layer: 58	vt = 0, va = 0, output: 26 x 26	512	0.000			
62 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
63 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
64 Shortcut Layer: 61	vt = 0, va = 0, output: 26 x 26	512	0.000			
65 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
66 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
67 Shortcut Layer: 64	vt = 0, va = 0, output: 26 x 26	512	0.000			
68 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
69 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
70 Shortcut Layer: 67	vt = 0, va = 0, output: 26 x 26	512	0.000			
71 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
72 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
73 Shortcut Layer: 70	vt = 0, va = 0, output: 26 x 26	512	0.000			
74 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
75 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
76 Shortcut Layer: 73	vt = 0, va = 0, output: 26 x 26	512	0.000			
77 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
78 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
79 Shortcut Layer: 76	vt = 0, va = 0, output: 26 x 26	512	0.000			
80 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
81 conv	3 x 3 / 1	26 x 26	→	26 x 26	512	0.797
82 Shortcut Layer: 79	vt = 0, va = 0, output: 26 x 26	512	0.000			
83 conv	1 x 1 / 1	26 x 26	→	26 x 26	512	0.177
84 route	83	26 x 26	→	26 x 26	512	0.177
85 conv	3 x 3 / 2	26 x 26	→	13 x 13	1024	1.595
86 conv	1 x 1 / 1	13 x 13	→	13 x 13	1024	0.177
87 route	86	13 x 13	→	13 x 13	1024	0.177
88 conv	3 x 3 / 1	13 x 13	→	13 x 13	512	0.797
89 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
90 conv	3 x 3 / 1	13 x 13	→	13 x 13	512	0.797
91 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
92 Shortcut Layer: 89	vt = 0, va = 0, output: 13 x 13	512	0.000			
93 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
94 conv	3 x 3 / 1	13 x 13	→	13 x 13	512	0.797
95 Shortcut Layer: 92	vt = 0, va = 0, output: 13 x 13	512	0.000			
96 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
97 conv	3 x 3 / 1	13 x 13	→	13 x 13	512	0.797
98 Shortcut Layer: 95	vt = 0, va = 0, output: 13 x 13	512	0.000			
99 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
100 conv	3 x 3 / 1	13 x 13	→	13 x 13	512	0.797
101 Shortcut Layer: 98	vt = 0, va = 0, output: 13 x 13	512	0.000			
102 conv	1 x 1 / 1	13 x 13	→	13 x 13	512	0.177
103 route	102	13 x 13	→	13 x 13	512	0.177
104 conv	1024	13 x 13	→	13 x 13	1024	1.595
105 conv	512	13 x 13	→	13 x 13	512	0.177
106 conv	1024	13 x 13	→	13 x 13	1024	1.595
107 conv	512	13 x 13	→	13 x 13	512	0.177
108 aa	5x 5 / 1	13 x 13	→	13 x 13	512	0.002
109 route	107	13 x 13	→	13 x 13	512	0.177
110 aa	13x13 / 1	13 x 13	→	13 x 13	512	0.015
111 route	109	13 x 13	→	13 x 13	512	0.177
112 aa	110	13 x 13	→	13 x 13	1024	1.595
113 route	112	13 x 13	→	13 x 13	1024	1.595
114 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
115 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
116 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
117 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
118 spsample	2x	13 x 13	→	26 x 26	256	0.000
119 route	85	26 x 26	→	26 x 26	512	0.177
120 conv	256	1 x 1 / 1	→	26 x 26	512	0.177

121 conv	32	1 x 1 / 1	→	26 x 26	512	0.177
122 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
123 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
124 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
125 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
126 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
127 route	125	26 x 26	→	26 x 26	512	0.177
128 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
129 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
130 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
131 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
132 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
133 route	131	26 x 26	→	26 x 26	512	0.177
134 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
135 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
136 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
137 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
138 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
139 route	137	26 x 26	→	26 x 26	512	0.177
140 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
141 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
142 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
143 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
144 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
145 route	143	26 x 26	→	26 x 26	512	0.177
146 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
147 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
148 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
149 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
150 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
151 route	149	26 x 26	→	26 x 26	512	0.177
152 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
153 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
154 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
155 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
156 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
157 route	155	26 x 26	→	26 x 26	512	0.177
158 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
159 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
160 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
161 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
162 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
163 route	161	26 x 26	→	26 x 26	512	0.177
164 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
165 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
166 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
167 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
168 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
169 route	167	26 x 26	→	26 x 26	512	0.177
170 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
171 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
172 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
173 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
174 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
175 route	173	26 x 26	→	26 x 26	512	0.177
176 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
177 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
178 conv	1024	1 x 1 / 1	→	26 x 26	512	1.595
179 conv	512	1 x 1 / 1	→	26 x 26	512	0.177
180 conv	256	1 x 1 / 1	→	26 x 26	256	0.044
181 route	179	26 x 26	→	26 x 26	512	0.177
182 conv	256	1 x 1 / 1	→	26 x 26	512	0.177
183 conv	512					

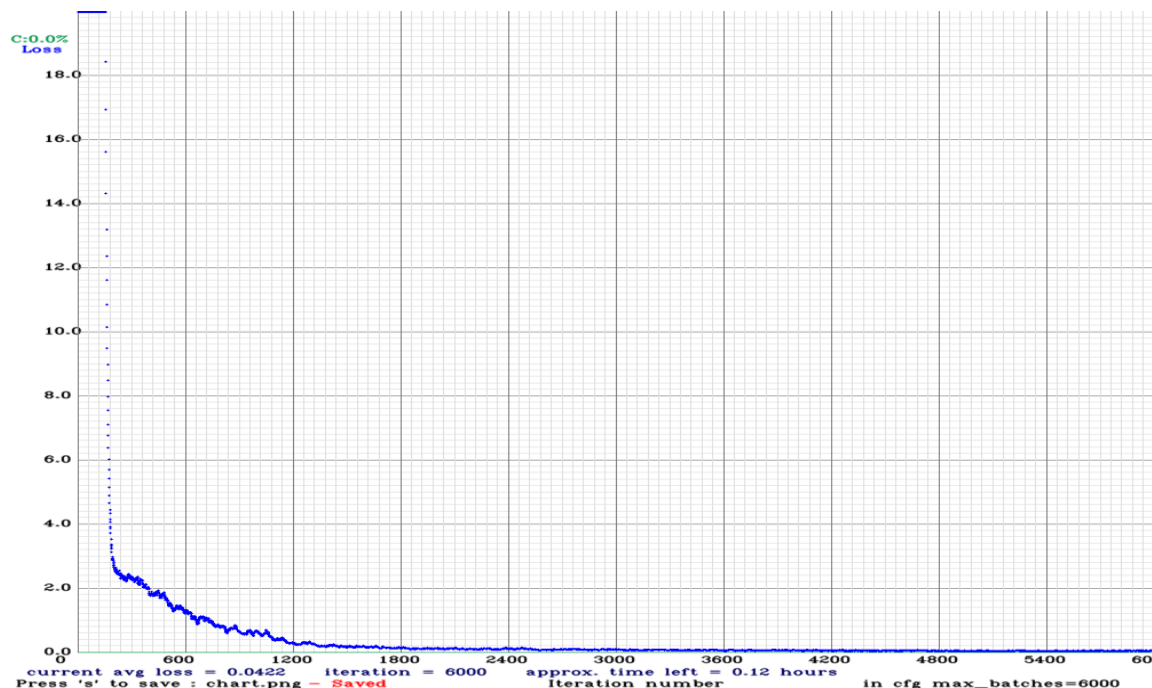




表 4 YOLOv4 貨櫃特徵辨識訓練流程

原始	Yolo Output
	
Yolo 特徵擷取	灰階
	

#### 4.3 實驗結果

利用正確率（ACC）進行 OCR 績效評估，評估 YOLO 模型的準確度。

準確度：號碼辨識正確 15 張，測試資料 20 張，績效為 0.75

## 五、 結論

在本研究中使用貨櫃資料集，當作辨識預測實驗資料集，並分別使用 YOLO 模型與 OCR 中的 tesseract 套件，去進行模型的建構與物件的辨識，以完成辨識預測之任務。在貨櫃資料集中，進行辨識的預測任務，期望能藉由 YOLO 模型來偵測並框出本實驗資料集中的數字與英文字母標籤位置，並藉由 OCR 來精準辨識數字與英文字母，該集料集包含了 1 號至 20 號的貨櫃圖片，在該實驗中預測正確之數量為 15 張，錯誤張數為 5 張，準確率高達 75%，該辨識模型可用於貨櫃辨識任務中，判別出大部分的貨櫃編號。

## 參考文獻

[1] YOLOv4 訓練教學

<https://medium.com/ching-i/yolo-c49f70241aa7>

[2] Yolo v4, v3 and v2 for Windows and Linux

<https://github.com/alexeyab/darknet>

[3] YOLOv4 輕快準！邊緣即時偵測應用就靠它

<https://www.ithome.com.tw/news/148303>

[4] YOLOv4: Optimal Speed and Accuracy of Object Detection

<https://arxiv.org/abs/2004.10934>

[5] YOLOv4 win10 配置 + 訓練自己的數據 + 測試

<https://codingnote.cc/zh-tw/p/313407/>

[6] YOLOv4 實作教學(使用官方 repo)

<https://hackmd.io/@neverleave0916/YOLOv4>

[7] YOLOv4 win10 配置 + 訓練自己的資料 + 測試

<https://iter01.com/583441.html>

[8] YOLOv4 產業應用心得整理 - 張家銘

<https://aiacademy.tw/yolo-v4-intro/>

[9] 什麼是 OCR (光學字元辨識)？

<https://aws.amazon.com/tw/what-is/ocr/>

[10] Python 自動識別圖片文字—OCR 實戰教程

<https://ithelp.ithome.com.tw/articles/10222697>

[11] Python 影像辨識筆記(二)：中英文字 OCR 辨識(圖片、驗證碼 ... | Python 圖形 辨識

<https://igotojapan.com>

[12] 臺灣車牌規則用於車牌辨識 - HackMD

<https://hackmd.io/@CynthiaChuang/Taiwan-License-Plate-Rules-for-LPR>