



# 硬幣 分類識別

---

108034548 卓好庭

問題定義 01

YOLO介紹 02

研究架構 03

# Agenda

04 資料處理

05 模型架構

06 結果呈現

# 01

## 問題定義-5W1H

---

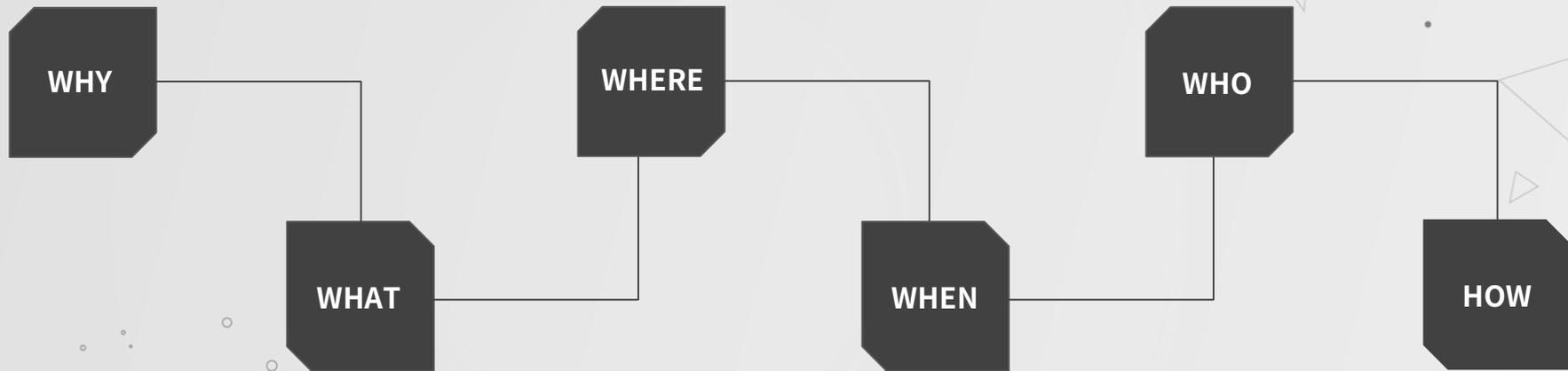


# 5W1H

不論是店家清點錢幣、個人整理錢幣，時常要逐一辨識，很難快速區分分別種類，尤其遇到不熟悉的外幣時，更是不易分辨。

任何需要做硬幣分類的場所，例如：家中、店面、銀行等等。

舉凡有在使用實體貨幣，需要整理硬幣的人們。



解決清點大量不同種類硬幣難分類的困擾。

需要做硬幣分類的時刻。

利用類神經網路訓練並建立一個可以識別硬幣種類的偵測系統。



# 02

## YOLO介紹

---

# YOLO

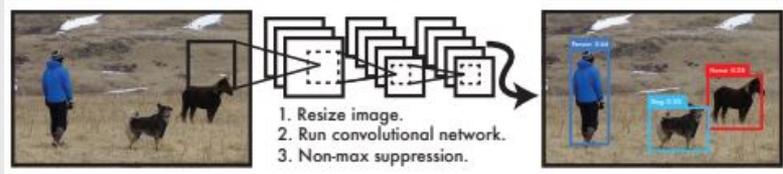
🎯 Yolo, You Only Look Once, 一種關於物件偵測的類神經網路演算法。

本次採用YOLOv3版本進行訓練

🎯 是one stage的物件偵測方法, 對輸入圖片的**物件位置偵測**和**物件辨識**同步處理, 就是只需要對圖片作一次CNN便能夠判斷圖形內的物體位置與類別。

🎯 執行步驟

1. 調整輸入的圖片大小416\*416
2. 執行一個CNN卷積神經網路
3. 基於模型輸出的信心程度(Confidence) 得到偵測結果

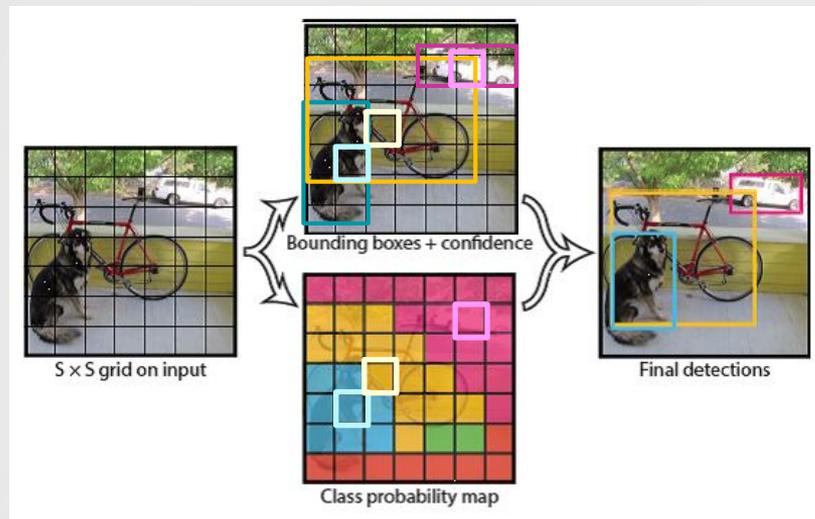


# YOLO

① **找物件**：由所有bounding box的信心程度，先刪除不是物件的bounding box，再刪除一些重疊的bounding box，重複執行後，剩下的bounding box就是選出來的物件，如右圖之中上圖。

② **找類別**：同時也輸出( $S \times S \times C$ )個值，代表每個grid cell內每一類別的機率，這時取每個grid cell中機率最大的數值當作此grid cell的類別，如右圖之中下圖。

③ 最後，結合選出的物件及對應grid cell的類別，即可判斷出物件所屬之類別。



先將一張圖片平均分成 $S \times S$ 格grid cell，總共要偵測 $C$ 個物件



# 03

## 研究架構

---

# 研究架構



## 資料收集

訓練集、驗證集、  
測試集資料收集



## 訓練、驗證集標記

利用labelImg套件  
標記硬幣所屬種類



## 模型訓練

將收集之照片套入  
YOLO進行訓練



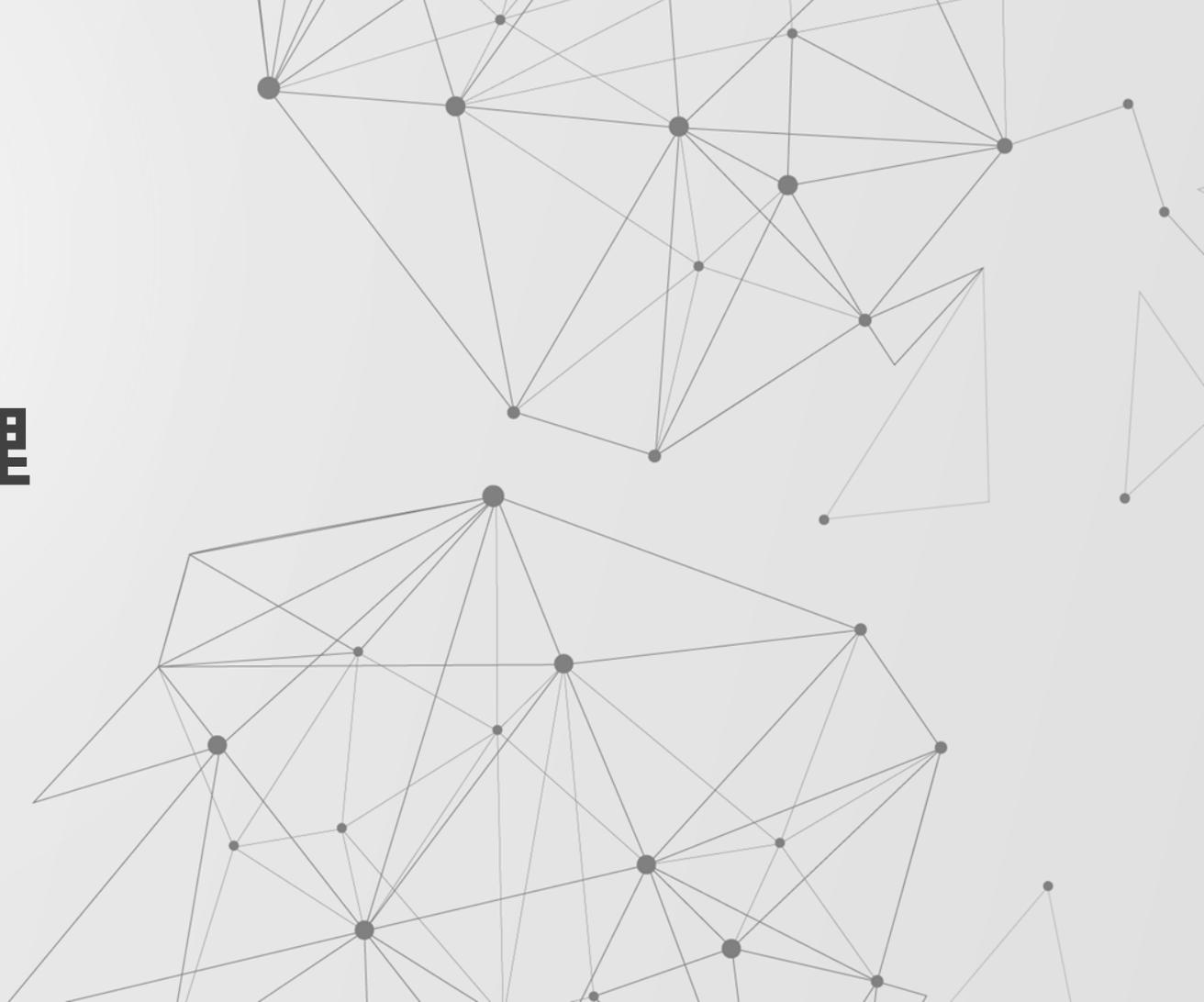
## 測試結果

訓練後測試系統  
準確度之照片、影片

# 04

## 資料處理

---



# 資料收集



## 自行拍攝

- 資料集為自行拍攝的新台幣硬幣群圖片，包含1元、5元、10元、50元之正反面
- 共80張照片作為訓練集
- 共20張作為驗證集
- 32張照片、5段影片做為測試集

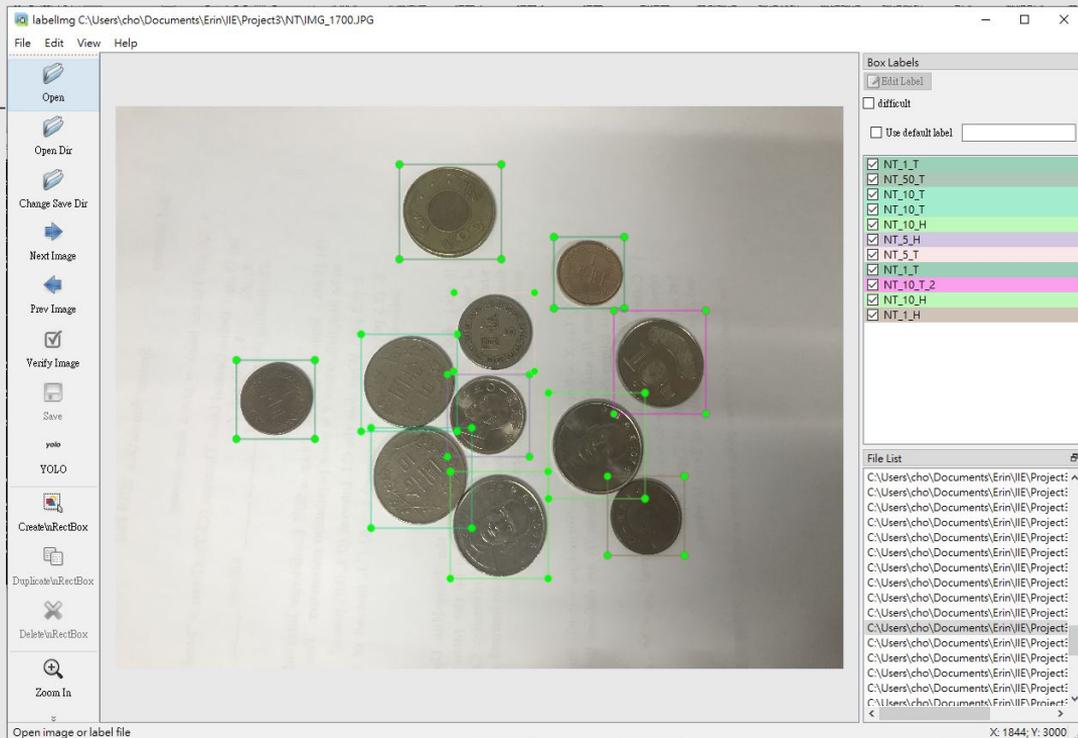


# 訓練、驗證集標記

## 類別標記

- 利用labelImg套件進行手動標記
- 將訓練集、驗證集共100張照片中的硬幣給予對應的標記項目

項目	標記名稱	項目	標記名稱
1元反面	NT_1_T	10元反面	NT_10_T
1元正面	NT_1_H	10元正面 (新款)	NT_10_T_2
5元反面	NT_5_T	10元反面	NT_10_H
5元正面	NT_5_H	50元反面	NT_50_T
		50元正面	NT_50_H



# 05

## 模型架構



# 卷積神經網路



## 神經網路架構

- 在訓練圖片前，先將輸入圖的bounding box長寬進行正規化，使得bounding box的長寬會介於0~1之間。
- 網路架構使用darknet-53結構，有75層卷積層且無全連結層，使得輸入的圖片不受限於固定的尺寸，任意輸入維度都可以在整個網絡上運行。
- 訓練過程中，每層之積活函數選擇leaky rectified linear activation。
- 損失採用 binary cross-entropy loss。

Type	Filters	Size	Output
Convolutional	32	$3 \times 3$	$256 \times 256$
Convolutional	64	$3 \times 3 / 2$	$128 \times 128$
1x	Convolutional	32	$1 \times 1$
	Convolutional	64	$3 \times 3$
Residual			$128 \times 128$
	Convolutional	128	$3 \times 3 / 2$
			$64 \times 64$
2x	Convolutional	64	$1 \times 1$
	Convolutional	128	$3 \times 3$
Residual			$64 \times 64$
	Convolutional	256	$3 \times 3 / 2$
			$32 \times 32$
8x	Convolutional	128	$1 \times 1$
	Convolutional	256	$3 \times 3$
Residual			$32 \times 32$
	Convolutional	512	$3 \times 3 / 2$
			$16 \times 16$
8x	Convolutional	256	$1 \times 1$
	Convolutional	512	$3 \times 3$
Residual			$16 \times 16$
	Convolutional	1024	$3 \times 3 / 2$
			$8 \times 8$
4x	Convolutional	512	$1 \times 1$
	Convolutional	1024	$3 \times 3$
Residual			$8 \times 8$
	Avgpool		Global
Connected		1000	
Softmax			

# YOLO執行程式碼

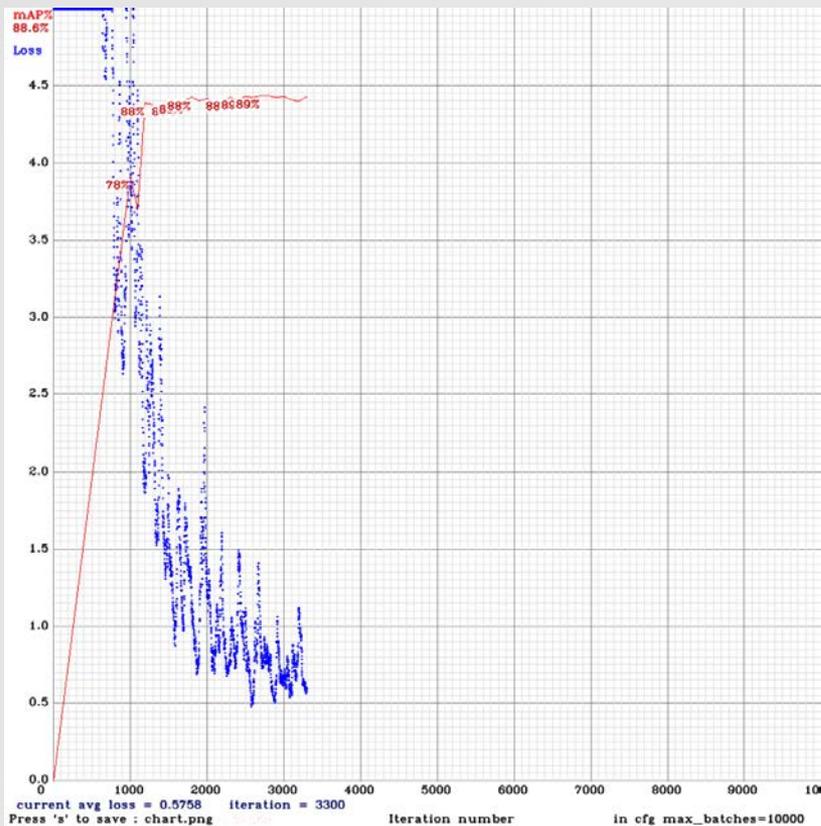
```
1 [net]
2 # Testing
3   batch=1
4   subdivisions=1
5 # Training
6   batch=64
7   subdivisions=16
8   width=608
9   height=608
10  channels=3
11  momentum=0.9
12  decay=0.0005
13  angle=0
14  saturation = 1.5
15  exposure = 1.5
16  hue=.1
17
18  learning_rate=0.001
19  burn_in=5000
20  max_batches = 500200
21  policy=steps
22  steps=400000,450000
23  scales=.1,.1
24
25 [convolutional]
26  batch_normalize=1
27  filters=32
28  size=3
29  stride=1
30  pad=1
31  activation=leaky
32
33 # Downsample
34
35 [convolutional]
36  batch_normalize=1
37  filters=64
38  size=3
39  stride=2
40  pad=1
41  activation=leaky
42
43 [convolutional]
44  batch_normalize=1
45  filters=32
46  size=1
47  stride=1
48  pad=1
49  activation=leaky
50
51 [convolutional]
52  batch_normalize=1
53  filters=64
54  size=3
55  stride=1
56  pad=1
57  activation=leaky
58
59 [shortcut]
60  from=-3
61  activation=linear
62
63 # Downsample
64
65 [convolutional]
66  batch_normalize=1
67  filters=128
68  size=3
69  stride=2
70  pad=1
71  activation=leaky
72
73 [convolutional]
74  batch_normalize=1
75  filters=64
76  size=1
77  stride=1
78  pad=1
79  activation=leaky
80
81 [convolutional]
82  batch_normalize=1
83  size=3
84  stride=1
85  pad=1
86  activation=leaky
87
88 [convolutional]
89  batch_normalize=1
90  size=3
91  stride=1
92  pad=1
93  activation=leaky
94
95 [convolutional]
96  batch_normalize=1
97  filters=128
98  size=3
99  stride=1
100 pad=1
101 activation=leaky
102
103 ...
104 [convolutional]
105  batch_normalize=1
106  size=3
107  stride=1
108  pad=1
109  activation=leaky
110
111 [convolutional]
112  batch_normalize=1
113  filters=256
114  activation=leaky
115
116 [convolutional]
117  batch_normalize=1
118  filters=128
119  size=1
120  stride=1
121  pad=1
122  activation=leaky
123
124 [convolutional]
125  batch_normalize=1
126  size=3
127  stride=1
128  pad=1
129  filters=256
130  activation=leaky
131
132 ...
133 [convolutional]
134  batch_normalize=1
135  size=3
136  stride=1
137  pad=1
138  filters=1818
139  activation=linear
140
141 [yolo]
142  mask = 0,1,2
143  anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
144  classes=601
145  num=9
146  jitter=.3
147  ignore_thresh = .7
148  truth_thresh = 1
149  random=1
```



# 06

## 結果呈現

# YOLO訓練成果



🎯 執行iteration 3300次

🎯 績效指標mAP為88.6%

🎯 損失loss為0.5758

# YOLO測試成果

```
calculation mAP (mean average precision)...
36
detections_count = 1486, unique truth count = 523
class_id = 0, name = NT_10_T_2, ap = 97.67% (TP = 53, FP = 21)
class_id = 1, name = NT_10_T, ap = 89.06% (TP = 47, FP = 30)
class_id = 2, name = NT_10_H, ap = 80.03% (TP = 54, FP = 24)
class_id = 3, name = NT_5_T, ap = 91.90% (TP = 80, FP = 10)
class_id = 4, name = NT_5_H, ap = 56.61% (TP = 8, FP = 1)
class_id = 5, name = NT_1_T, ap = 82.36% (TP = 68, FP = 14)
class_id = 6, name = NT_1_H, ap = 91.52% (TP = 65, FP = 36)
class_id = 7, name = NT_50_T, ap = 94.29% (TP = 33, FP = 0)
class_id = 8, name = NT_50_T_2, ap = 0.00% (TP = 0, FP = 0)
class_id = 9, name = NT_50_H, ap = 93.68% (TP = 19, FP = 0)

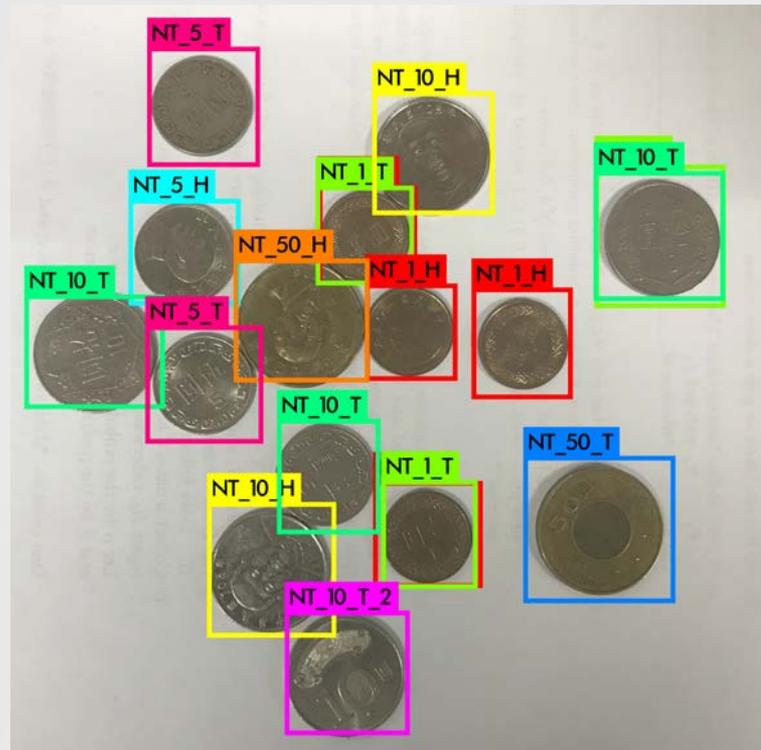
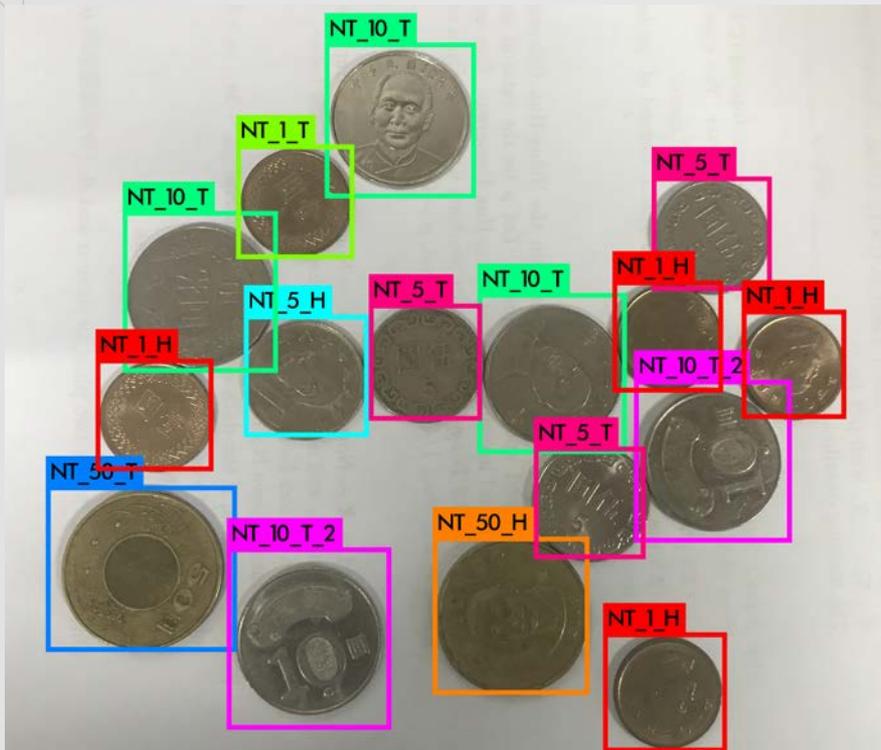
for conf_thresh = 0.25, precision = 0.76, recall = 0.82, F1-score = 0.79
for conf_thresh = 0.25, TP = 427, FP = 136, FN = 96, average IoU = 66.47 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.777105, or 77.71 %
Total Detection Time: 2.000000 Seconds
```

- 🎯 各類別之準確率皆大於80%
- 🎯 總體績效指標mAP為77.71%
- 🎯 precision = 0.76 · recall = 0.82

# YOLO測試成果

🎯 測試集中抽取2張照片展示

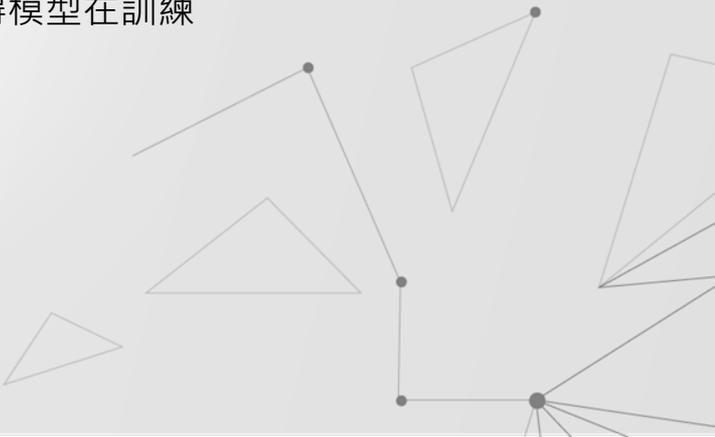


# YOLO測試成果



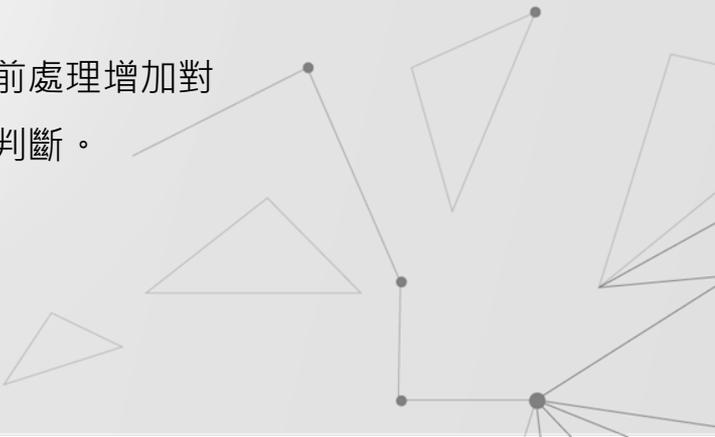


# 本次研究限制

- ④ 在資料集的數量上仍稍嫌不足，因此在訓練的類別辨識上受限於樣本過少無法達到百分百精準。
  - ④ 而在辨識硬幣方面的限制為，錢幣的本身變異過大，包括髒污程度影響硬幣圖像的呈現程度，進而使得模型在訓練或測試上皆會受到汙漬顏色影響判段結果。
- 



# 未來研究方向

- ◎ 本次僅針對台幣硬幣作訓練及系統建置，日後可以針對各國錢幣進行標記後訓練，使得此系統更加完善實用。
  - ◎ 在樣本數上可以再進行擴大，增加準確率。
  - ◎ 污漬顏色影響的問題，日後可以在增強圖像前處理增加對比度，使得硬幣之字樣刻痕更加顯現，幫助判斷。
- 



---

# THANKS

Does anyone have any questions?



---