2021 年臺灣國際科學展覽會
優勝作品專輯

作品編號　**190027**

參展科別　電腦科學與資訊工程

作品名稱　利用 Yolo 模型辨識台灣國語口手語之研究

就讀學校　基隆市私立二信高級中學

指導教師　張宏文

作者姓名　張庭宇、游翔任、蔡炅曄

關鍵詞　手語、深度學習、影像辨識

作者簡介

張庭宇(右)：我是基隆二信高中學生。從小我就對自然科學深感興趣，小學連續三年參加科展比賽，到了國中、高中也都有作品參賽。去年報名錄取【海大學術列車 AI 人工智慧專班】，接觸人工智慧的理論之後，深深覺得這是未來人類的生活必備工具，我希望將來能投入相關領域的研究工作。

游翔任(中)：我是二信高中二年級的學生。從小我爸爸就會在家中教我一些關於電腦的基本概念，也會帶著我寫一些簡單的程式、網頁，我也對這些領域有了興趣。到了高中，多了許多可以在這個領域學習的機會，在不停地探索中，我覺得電資領域是我喜歡的，也是我想發展的方向。

蔡炅曄(左)：我來自二信高中二年級，對於資訊科學領域有著強烈的興趣，這次是個人首度嘗試人工智慧領域的作品，雖然與想像中有點出入，也遇到了不少挫折，但我喜歡挑戰新鮮的事物，也期待未來能夠成為資訊領域的種子人才，為科技發展貢獻一份心力！

摘要

手語為聾啞人士日常溝通的工具，但對一般人來說這是一種難以理解的溝通方式。本實驗使用深度學習的 Yolov3 與 Yolov4 模型訓練 37 個國語注音符號手勢，然後再驗證模型對圖片、影片、即時(Real time)攝影辨識的正確率。

實驗結果顯示：Yolo v3 圖片辨識度效果還不錯，但影片辨識度很差，而 Yolo v4 不管在靜態的圖片或動態影片都有不錯的辨識率，另外在即時的影像辨識也有不錯的效果。

雖然有部分符號的辨識度很低，但這可能是訓練時照片拍攝的問題，如果可以改進拍攝的數量和技巧，相信可以大幅提升判讀的準確率。

Abstract

Sign language is a daily communication tool for deaf-mute people, but it is an incomprehensible way of communication for ordinary people. This experiment conducted the deep learning models Yolov3 and Yolov4 to recognize 37 Mandarin phonetic symbol gestures and verifies the accuracy of the model's recognition of pictures, videos, and real time photography.

The results show that: Yolo v3 had higher performance in image recognition while it was not good in video recognition. Yolo v4 had good recognition in both static pictures and dynamic videos, and also did well in real-time image recognition.

Lower recognition in some specific symbols may due to the lower quality of photo shooting. If the quality and quantity of shooting samples can be improved, the accuracy of interpretation can be greatly improved.

一、　前言（含研究動機、目的）

每當電視轉播重大演說，螢幕的右下角總有一位手語老師即時翻譯著演說內容；走在路上，偶爾也會看到兩個路人邊走邊用著手語交談著。這些都是我們看到、接觸到手語的日常，其實手語一直圍繞在我們身邊。到了高中，參加了康輔社，也在因緣際會下學習了手語，也開始漸漸認識了手語。在有一天上網查有關手語的相關資料時，赫然發現全球竟有近4億人為聽障人士，並不算少數，而他們因為種種原因必須要用手語溝通，這也使得手語是重要的溝通語言之一。

但大部分的人看不懂手語，所以與喑啞人士之間的溝通也較不方便，常常造成喑啞人士形成自己的群體，不與其他生理功能正常的人接觸。而我們並不希望這樣的關係持續下去，我們希望能讓喑啞人士融入社會，讓「溝通不便」不再是造成喑啞人士被社會隔離在外的原因。

我們的構想是希望能有一套可以將手語辨識並用於溝通的系統，我們第一個想到的方法就是手勢辨識，利用即時辨識就能透過鏡頭與程式將手語轉換為語音，能立刻了解他們所想表達的，而這樣的系統甚至可以應用在無法用聲音或電訊傳播的時候(例如潛水人員)所以我們的研究目的是：

(一) 利用人工智慧訓練電腦判讀手語
(二) 將手語動作即時翻譯成注音圖像和聲音
(三) 將翻譯的注音圖像拼出發音

二、 研究方法或過程

表一、研究設備及器材		
硬體		軟體
桌上型電腦		Microsoft Windows 10 64-bit
Processor (CPU)	AMD Ryzen 7 3700X 8-Core Processor	LabelImg
Motherboard	Gigabyte X570 AORUS ELITE WIFI	Yolo v3/v4
Memory (RAM)	32768 MB	Python 3.7.9
SSD	ADATA XPG SX8200 PRO 512G	tensorflow-gpu 1.14.0
Graphic Card (GPU)	NVIDIA GeForce RTX 2070 SUPER	kearas 2.2.4
WebCam	Logitech WebCam C310	OpenCV 4.4.0
筆電		Visual Studio 2019
Processor (CPU)	Intel® Core™ i7-8565U processor 1.8GHzquad-core with Turbo Boost	Cuda 10.0
Graphic Card (GPU)	NVIDIA® GeForce® GTX 1050 Max-Q2GB GDDR5 VRAM	cuDNN 8.0.3
Memory (RAM)	16GB 2400MHz DDR4 onboard	Anaconda(64-Bit)
SSD	512GB PCIe® 3.0 x2 SSD	DocFetcher

(一)、 國語注音口手語介紹

手語就如同我們口語一樣，受到不同的文化習俗和生活習慣的影響，法國系統，中國系統，日本系統為世界上三大手語系統。台灣與韓國的手語是屬於日本手語系統，目前台灣手語大部分和日本手語相同，另一部分是台灣的本土手語，一小部分是和中國、美國…等混合手語，但日本手語在台灣還是佔有優勢的地位。

台灣手語分成很多種類，自然手語、中文式手語、文字手語、國語口手語。自然手語是由表音、表意、表字組合而成，一字使用上通常有好幾種意思，文字簡單、視覺上生動傳

神，加上「文字手語」的補強，適合手語同步翻譯使用；中文式手語的產生主要是從小到大習慣使用國語文法的影響，在使用時常會口手語一起表達，為年輕一代『會口語的聽障者』普遍應用；文字手語主要是求一字一手語，是目前是啟聰學校手語教學上主要的工具，也是自然手語的好幫手；而國語口手語又稱音標式手語，分成了口語和注音兩個部分，特點是手口並用，手語的部分是利用手勢變化，和左右手搭配，表示國語中的三十七個注音符號、四聲及標點符號，總共約有五十五個手勢，如此用手語來表示注音，功能是希望能增強聽障學生用語音思考的能力，多練習發音和認字以提昇語文能力，在台灣已經推展多年，國小學生可能會使用，但後因溝通方式較慢，不方便使用。

本次研究主要辨識為國語口手語，主要原因是注音手語的動作固定，對於 AI 人工智慧影像辨識成功率較高，需要計算的資料量較少(只有 37 個手勢)，比較容易訓練。

(二)、 Yolo 介紹

1.Yolo 的優點

在進行物件偵測的作法上，通常都是先用一些手段將特徵物件選出之後，再進行物件辨識，這樣的作法被稱作 two stage learning。然而 two stage learning 的最大問題即是當選出的物件數過多時，硬體必須耗費大量資源與時間運算，當這樣的功能運作在低硬體支援的平台上，幾乎無法做到我們訴求的即時運算，故並非一個經濟的做法。不過從 Yolo 的架構我們能發現，物件的偵測和辨識是一體的，也就是偵測跟辨別能夠一次到位，這正是業界和社會正需求的科技之一，「快、狠、準」極為 One stage Learning 的特性。這樣的特色在 Google 於 2015 年 12 月所提出的 Single Shot Detector (SSD)文件摘要也有提及到，其寫道：「We present a method for detecting objects in images using a single deep neural network.」，只要用單一神經網路就能完成圖片中的物件偵測。這樣的模式具有快速辨別的特性，雖然精確度沒有 two stage learning 來的好，不過因為大致上可接受，也較被廣泛採用在行動裝置上。

2.Yolov4 的優點

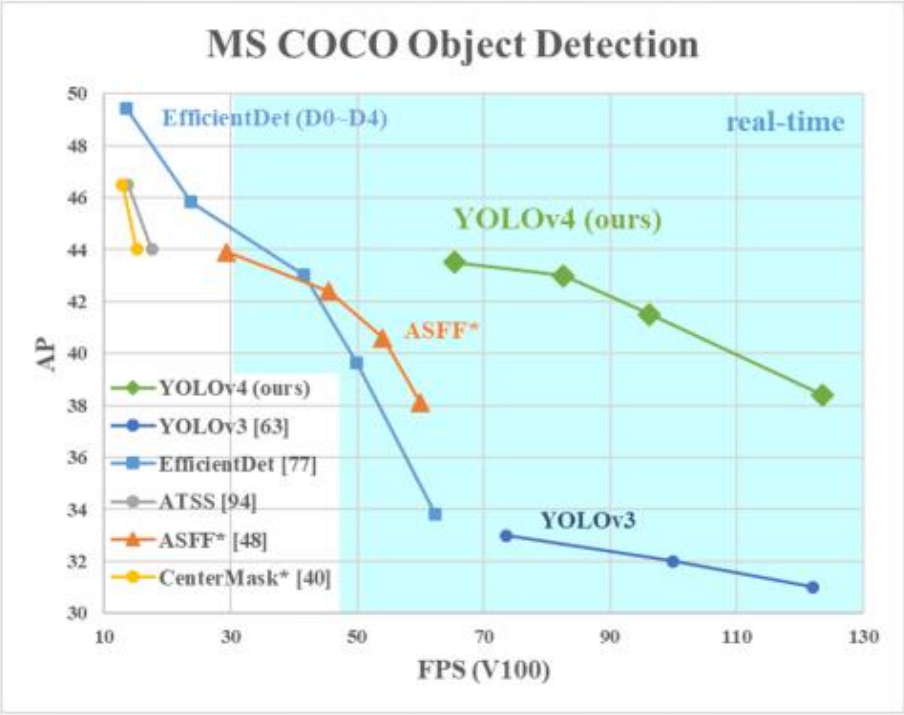


圖 1　Yolov4 與較先進的物件偵測元件之 AP 與 FPS 比較

在 Yolov4 的原作者論文當中，提供了一張與 Yolov3、EfficientDet……等技術的相比 MS COCO 數據集偵測結果比較圖，並以 AP(Av　erage Precision)、FPS 作為參照指標。從中可以發現其 Yolov4 的 AP 與 FPS 皆進步將近 1 成。表示在準確度、實時處理能力上，Yolov4 著實略勝一籌。而且 Yolov4 強調讓一般人只要用 1080Ti 的 GPU 就能夠訓練人工智慧、應用人工智慧，這也是 Yolov4 相較於 Yolov3 所被賦予的意義與價值。在 Yolov4 的論文中也提到了所謂「五大改進、二十多項最先進的目標檢測技巧」，這都是 Yolov4 之所以強大的原因。不過 Yolov4 的進步與強大是作者在網路結構上的調整才獲得的成就。

3.物件偵測常用到的技術：

(1)卷積運算：

用一個遮罩遮住圖片的某個部份，並對圖片的數值以及遮罩的數值做點對點的乘積，再將所有相承的數值相加在一起，從左上開始移動遮罩，一直移動到右下，即完成圖片的卷積運算。

而執行卷積運算做主要的目的是為了擷取圖片的特徵，而且能讓要推估的權重數量減少，使運算更加輕鬆

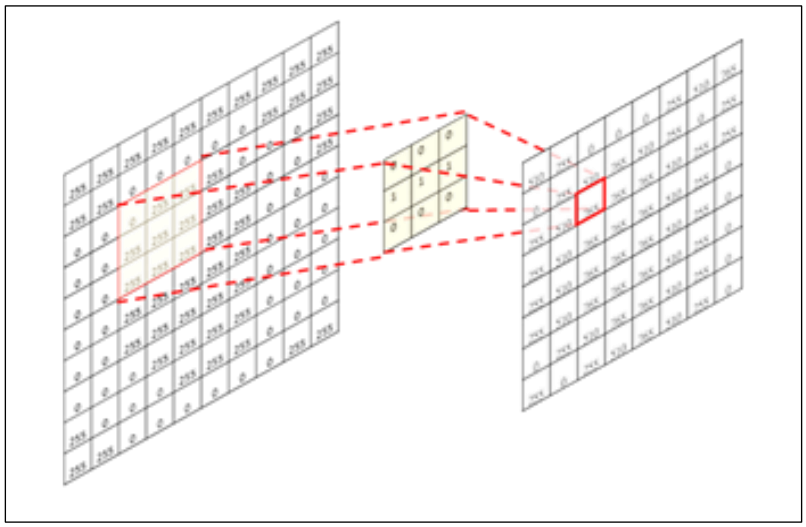


圖 2、卷積運算

(2)Dropout：

避免模型過度凝和，所以刻意關閉一些神經元，避免神經元之間太多依賴，當資料過度凝和，可能造成模型只在訓練資料有好表現，而對其他資料的判斷並不好。

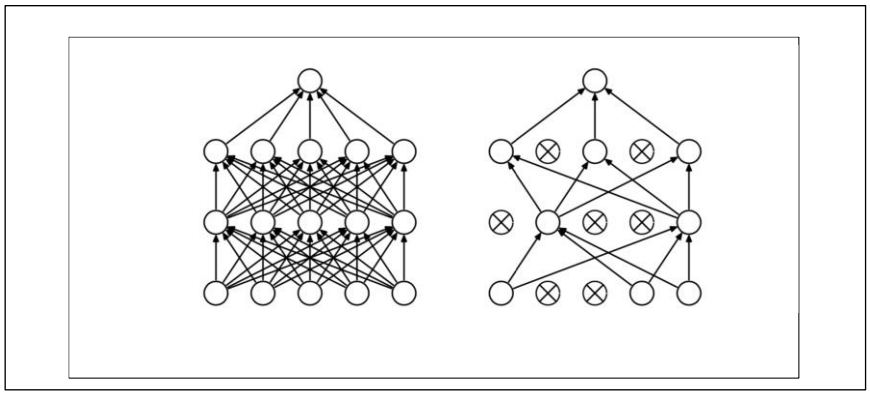


圖 3、Dropout

4.Yolov4 基本概念、架構

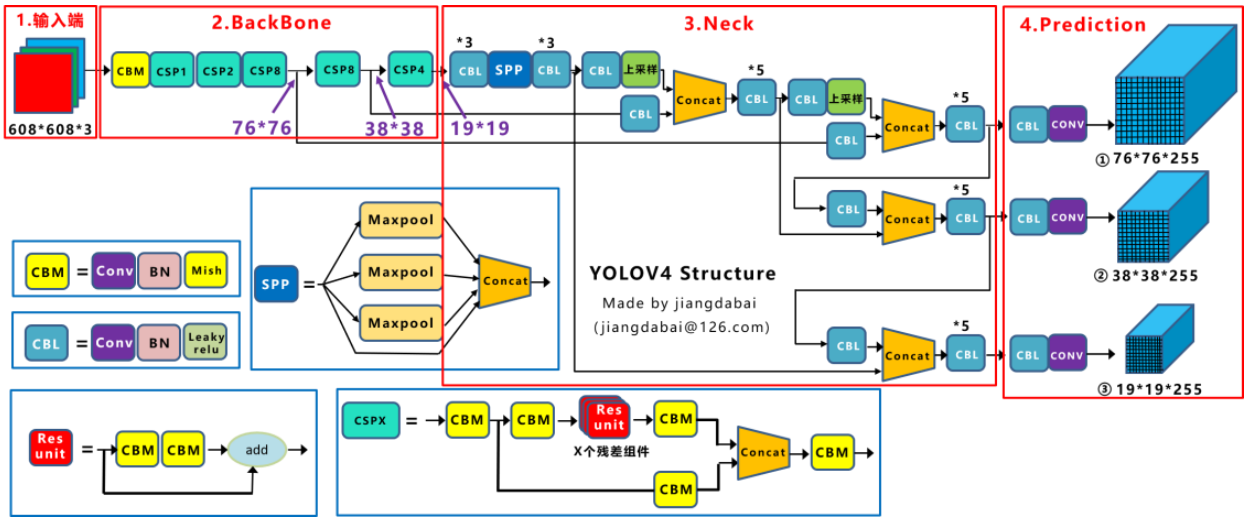


圖 4、YoloV4 網路基本架構圖

5.Yolov4 運算技術介紹與 Yolov3 及 Yolov4 的比較

Yolo 在偵測物件的一連動作串中有三個動作極為重要，這三個動作也能作為 yolo 的精神象徵，這三個動作分別是 (1) 調整圖片大小 (2) 執行卷積神經網路 (3) 依據模型的信心程度得到偵測結果。

這三個重要的步驟都會在以下的 yolo 介紹中都會講到，而除了這三個步驟之外，我們將從輸入到輸出將 Yolo 分成四個部分(Input、Backbone、Neck、Prediction)來介紹在 Yolo 中使用到的重要技術。

而在這次的研究中我們曾經嘗試以 yolov3 作為訓練所用之技術，但發生了影片檔案無法辨識、準確度有待商榷……等問題，針對此現象我們也整理了相關資訊以比較 Yolov3 與 Yolov4 之差別。將一併在接下來 Yolov4 使用技術介紹中提到 Yolov3 與 Yolov4 的差異。

(1) Input 區塊

(a) Mosaic 數據增強

會先將所輸入的訓練資料隨機裁切、排列，增加小目標的訓練量(因圖片縮小後，原來標註的目標也會隨之變小)，並一次把多張圖片拼湊成一張圖片 ，除了減少訓練的圖片總量

外，也能避免因卷積運算導致特徵擷取不完全、模型過度適應大尺寸特徵導致小目標辨識準確率下降……等問題，這樣的技術既能增加模型的準確程度和適應能力，也能降低硬體的負擔。是一個在 Yolov4 才加進來的一種新技術，讓 Yolov4 在辨識上可以比 Yolov3 更精準。

(2) Backbone 區塊

Backbone 是 yolov4 和其他偵測器不同的一個地方，他將各種在辨識中新的方法結合起來，包含了 CSPdarknet53、Mish 函數以及 Dropblock 等，是 Yolov3 與 Yolov4 差異較大的一層。

(a) CSPdarknet53:

這層的主要作用是將圖片一層一層的濃縮，產出該圖片的特徵圖。而這樣做的優點是因為圖片的檔案大小縮小，有效降低 ram 的空間，因為 CSPdarknet53 的特徵擷取能力很強，所以能同時能消耗的硬體資源減少，但仍然保持準確度，就很像在傳輸大量資料時而壓縮檔案。

而相較於 Yolov3 的 darknet53 架構，Yolov4 新增了一個 CSP 模組。而 CSPDarknet53 的概念則是來自於更早提出的 CSPNet。CSPNet 作者在當時認為，計算量過大可歸咎於梯度訊息多次重覆而導致，故 CSP 模組能夠將基礎層已經抓出的特徵(如：點、線、邊框…等資訊)先拉開成為兩部分，並在最後再串聯(Concatenate)合併。

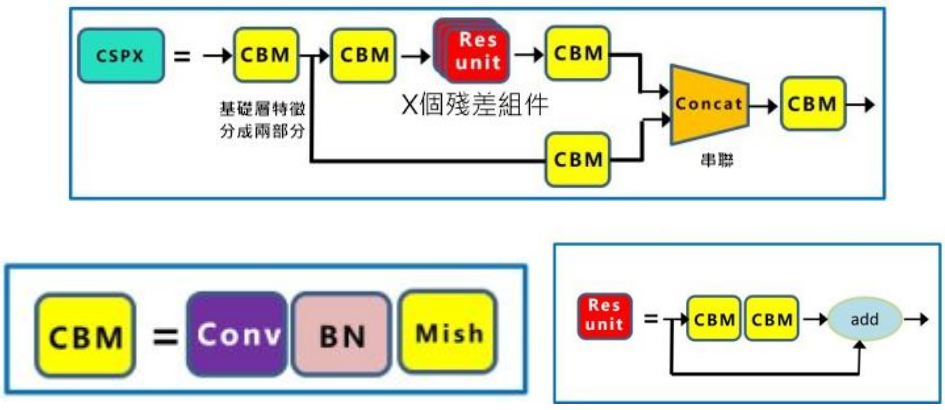


圖 5　CSP 結構可視圖

（ConV：卷積運算(convolutional)，BN：批次標準化(Batch normalization)）

CSPx 可說是構成 CSPDarknet53 的零件，裡面包含了卷積層與 x 個殘差單元，透過卷積運算能夠大幅減少運算量，以 CSPDarknet53 而言，裡面有著 5 個 CSP 的模組，又每一個卷積核之遮罩大小為 3*3，步長為 2，故以一個 1024*1024 的圖片為例，特徵圖變化從 1024→512→256→128→64→32，最後僅需要處理 32*32 大小的圖片即可，這樣不但能降低計算上的困難和所耗費的記憶體，同時也能增強模型的學習能力並有輕量、準確的特性。

(b) Mish 函數

在 backbone 層中以 mish 作為激活函數，激活函數的目的是將原本縮小的特徵相片轉乘另一個形式傳送到下一層，就像是打開 zip 檔解壓縮一樣。

在 Yolov3 中所使用的激活函數並不是 Mish，而是 Leaky_ReLU。Leaky_ReLU 的前身為 ReLU 函數，但 ReLU 函数會出現輸入負值即發生梯度消失問題，故 Leaky_ReLU 改善了這樣的問題。而 Mish 函數的特性為其函數相對平滑，較容易保存較小的負值，穩定了梯度的變化量，同時也不會發生因提度消失，神經元無法更新的問題，故也較適合需要深入的神經網路，而過往實驗也證明 Mish 函數的準確率比 ReLU 函數準確率來的高。不過 Neck 層的部分仍維持採用 Leaky_relu 作為激化函數。

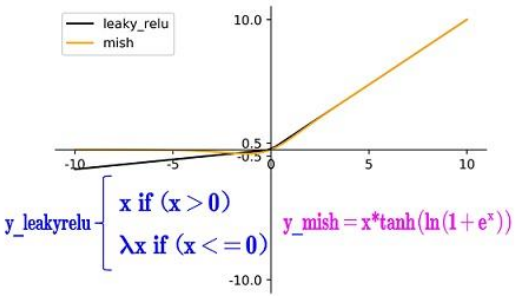


圖 6 Leaky_relu 與 Mish 函數比較圖

(c) Dropblock

隨機刪除網路中的神經元，避免資料過度凝和，功能跟 dropout 類似，但損失的方式與 dropout 的傳統方式有些許的不同。

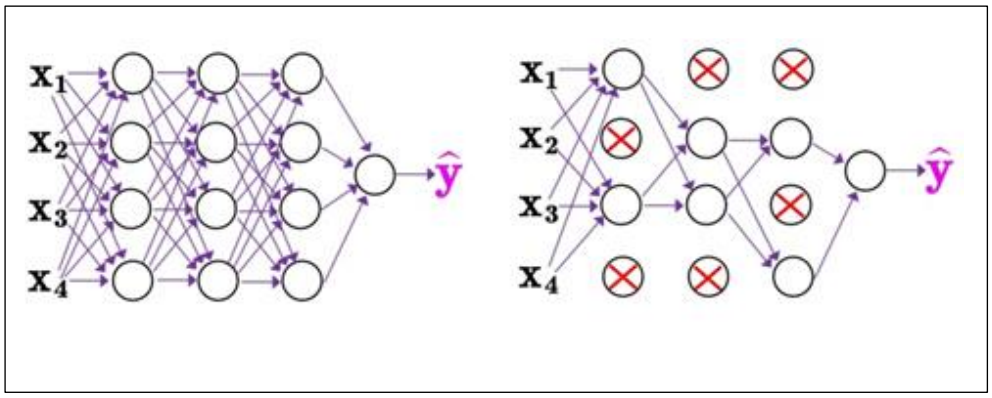


圖 7　Dropblock 概念圖

(3) Neck 區塊

Neck 是一個在 Backbone 以及輸出之間一個不可或缺的角色，他將一張圖中的多個特徵擷取(例如:擷取物件邊緣)，為模型的判斷增加可用資料，提高模型判斷準確度。在這層中，與 Yolov3 相差最多的就是由 FPN 架構改為 SPP+PAN 架構，是 Yolov4 重大的創新與突破。

(a) SPP 模組

在模型中使用 k={1*1,5*5,9*9,13*13}的池化層擷取圖片特徵，較原本 k*k 的池化方式有效率地增加了特徵擷取範圍並保留更多特徵

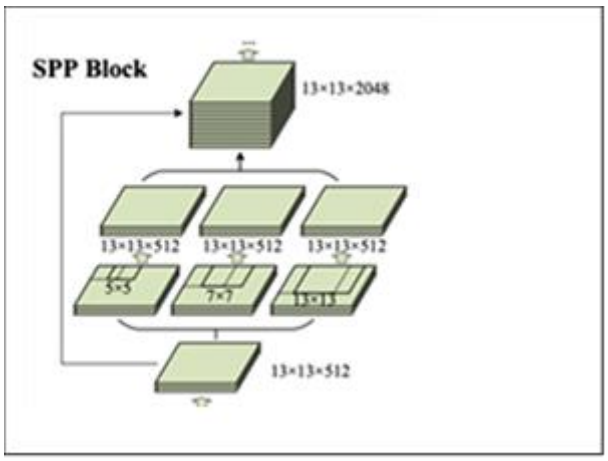


圖 8　SPP 模組概念圖

(b) FPN+PAN

在 Yolov3 的架構下，原始的 Neck 層僅有 FPN 架構，但 Yolov4 捨棄了 FPN，進而轉用 FPN+PAN 之組合。從圖中可以發現 FPN 是以上採樣的方式來得到預測的特徵圖。FPN 主要在傳達強烈的「語意」特徵，也可以說他是在傳達偏「形容詞」這類的東西。

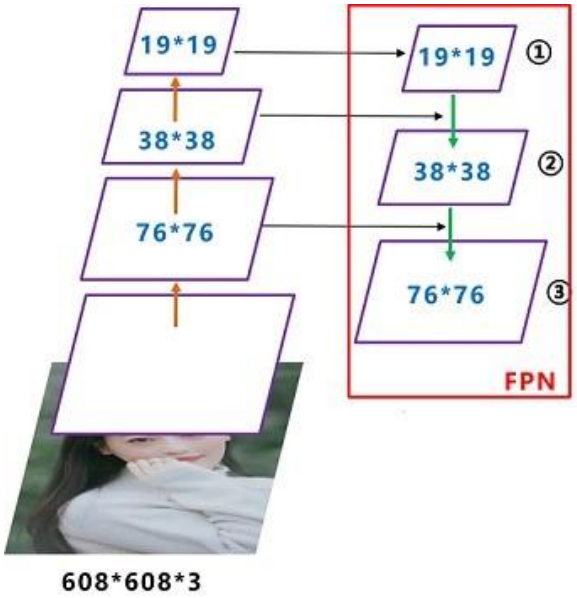


圖 9　FPN 架構

然而 Yolov4 的 FPN 層後面又加上了底向上的特徵金字塔，在這個過程之間也包含了兩個 PAN 結構。不同於 FPN 的是 PAN 主要傳遞強烈的「定位」特徵，透過這兩者的整合，幾乎就得到一個特徵的辨識依據了，這也是 Yolov4 在特徵抓取的能力較強的關鍵之一。

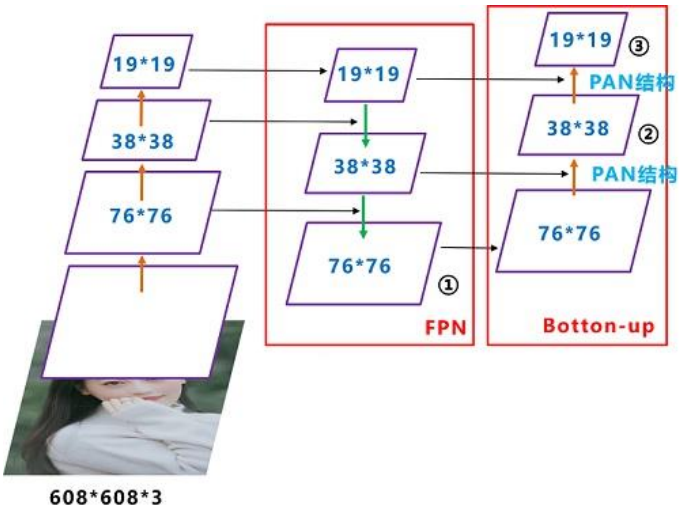


圖 10 FPN+PAN 架構

另一個不同的地方在於 Yolov3 會將最小的特徵圖與經過 FPN 過程中後產出的特徵圖都用來進行預測。以上圖為例，Yolov4 因為有 PAN 架構，則是直接使用經由 FPN 產生的 76*76 的特徵圖搭配 PAN 完成預測，並直接輸出預測後的 38*38 和 19*19 特徵圖，省去要過給其他神經網路處理的流程。

(4)Prediction 區塊

對要輸出的內容作出篩選、合併，讓辨識的成果能有最好的展現。

(a) CIOU_Loss

在取辨識物件交集時同時考慮物件重疊面積、中心點距離、長寬比的一種損失函數，這個函數讓 Yolov4 在預測框範圍標示上的速度與精準度有大幅提升。

$$\text{CIOU\_Loss} = 1 - \text{CIOU} = 1 - (\text{IOU} - \frac{\text{Distance\_2}^2}{\text{Distance\_C}^2} - \frac{v^2}{(1-IOU)+v})$$

(b) DIOU_nms

相較於普通的 nms 這邊的 DIOU_nms 函數可以判斷更多重疊部分，在 Yolov4 中最多可以一次辨識 98 個物件，是 Yolov3 所望塵莫及的，而這個函數也為辨識帶來更好的效果。

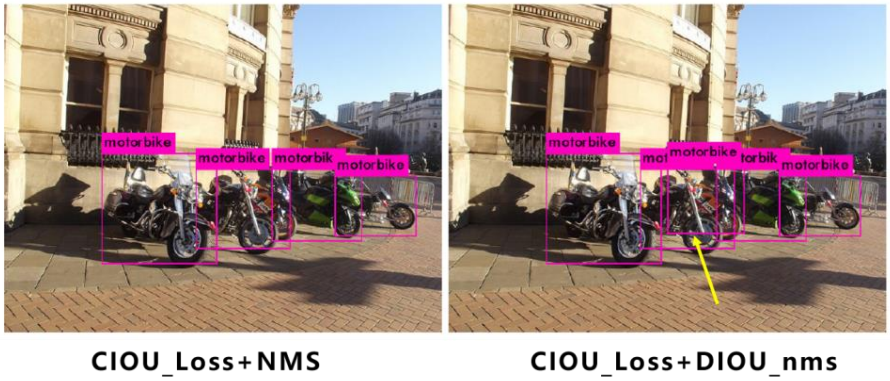


圖 11 NMS 與 DIOU_nms 比較圖

(三)、 拍攝注音手語資料圖片、建立 Yolov3 訓練環境進行訓練

1. 拍攝 37 個注音的手語手勢：

我們以國語口手語音標手勢的辨識及拼音作為這次研究的目標成果，因為網路上的資料不多，所以我們自行準備許多的訓練資料。一開始我們是用拍照來收集資料，但因為照片資料搜集速度慢，所以我們用錄製影片的方式收集資料，錄製完之後再用程式將影片以每 30 個 frame 截一張照片，最後總共得到一萬三千多張照片。

注音符號	英文代號	手語手勢	注音符號	英文代號	手語手勢	注音符號	英文代號	手語手勢
ㄅ	a		ㄒ	n		ㄜ	ab	
ㄆ	b		ㄓ	o		ㄝ	ac	
ㄇ	c		ㄔ	p		ㄞ	ad	
ㄈ	d		ㄕ	q		ㄟ	ae	
ㄉ	e		ㄖ	r		ㄠ	af	
ㄊ	f		ㄗ	s		ㄡ	ag	
ㄋ	g		ㄘ	t		ㄢ	ah	
ㄌ	h		ㄙ	u		ㄣ	ai	
ㄍ	i		ㄧ	v		ㄤ	ai	
ㄎ	j		ㄨ	w		ㄥ	ak	
ㄏ	k		ㄩ	x		ㄦ	al	
ㄐ	l		ㄚ	y				
ㄑ	m		ㄛ	z				

圖 12 注音手勢與代號(自製圖)

2.建立相關資料夾：labels、 images

3. 開始 label 相片

我們使用 LabelImg Label 軟體標註目標手勢。剛執行時畫面是空的，按「Open Dir」、「Change Save Dir」選擇剛剛建立的 images 以及 labels 資料夾，接下來便可從下窗格中選擇要 label 的相片，「按下 Create RectBox」便可開始 label。資料夾下應會分別有相同數目的 image 檔及 xml 的 label 檔。準備好 labelImg 標記好的圖片以及 xml 檔，將 JPG 圖片放入 JPEGImages 資料夾中，將 xml 檔放入 Annotations 資料夾中。

4.轉換 VOC labels 為 YOLO 格式

YOLO 所需要的 label 格式不同於我們所熟知、ImageNet 使用的 PASCAL VOC xml，而是採用 text 文字檔，第一欄為 class 的 ID，其它皆以物件框相對於整張圖片的比例來呈現：

YOLO Format

[category number] [object center in X] [object center in Y] [object width in X] [object width in Y]

1 0 0.6784060846560847 0.33630952380952384 0.2767857142857143 0.2837301587301587

2 0 0.35383597883597884 0.6729497354497355 0.26455026455026454 0.26785714285714285

3 0 0.23792989417989419 0.3169642857142857 0.22453703703703703 0.18617724867724866

5. 建立設定檔 cfg 資料夾

(1)新增 cfg 資料夾，我們將在此資料夾下放置 YOLO 設定檔。obj.names、obj.data、train.txt、test.txt、yolov3.cfg or yolov3-tiny.cfg

(2)obj.names：此檔內容為 label classes 的列表，裡面包含 37 注音代號和控制用的 hand 和 face 共 39 classes。

(3)obj.data：定義 label 數目以及各個設定檔及 weights 目錄的 path，YOLO 訓練及預測時皆會讀取。

(4)train.txt：所有 JPEGImages 檔案名稱列表中的 80%（或其它比例，可視需求變更），訓練時 YOLO 會依次讀取該檔內容取出相片進行訓練.。

(5)test.txt：所有 images 檔案名稱列表中的 20%（或其它比例，可視需求變更），訓練時 YOLO 會依次讀取該檔內容取出相片進行 validation.。

(6)yolov3.cfg：YOLO 模型設定檔，從 Darknet 安裝目錄下的 cfg 資料夾找到需要的 YOLO cfg 檔，複製到本 cfg 資料夾

(7)修改 yolo 模型的 cfg 檔

yolo-obj.cfg

#17 行 增加參數：

- add flip=0 讓左右對稱視為不同形狀

6. 下載 darknet 預設權重 yolov3.weights 開始訓練

7. 檢視訓練成果

(四)、 注音規則與程式

1. 注音拼音規則

所有的注音符號分為三大類：聲符（21 個）、介符（3 個）和韻符（13 個），總共有 37 個，將這 37 個符號有規則的組合並加上聲調，就是我們習以為常的注音拼音。

聲符						介符	韻符			
ㄅ	ㄉ	ㄍ	ㄐ	ㄓ	ㄗ	ㄧ	ㄚ	ㄞ	ㄢ	ㄦ
ㄆ	ㄊ	ㄎ	ㄑ	ㄔ	ㄘ	ㄨ	ㄛ	ㄟ	ㄣ	
ㄇ	ㄋ	ㄏ	ㄒ	ㄕ	ㄙ	ㄩ	ㄜ	ㄠ	ㄤ	
ㄈ	ㄌ			ㄖ			ㄝ	ㄡ	ㄥ	

圖 13 注音符號表

(1) 聲符

聲符作為注音拼音開頭的符號，而在聲符中只有ㄓ、ㄔ、ㄕ、ㄖ、ㄗ、ㄘ、ㄙ可以單獨存在，其他的皆無法單獨存在。

(2) 介符

介符可以單獨成字，能當開頭、結尾的符號，可與韻符結合成為結合韻。

(3) 韻符

韻符可作為注音拼音中的最後一個字，而每個韻符都可以單獨成字。

(4) 結合韻

由介符和韻符組成，能以結合韻直接存在，也能和聲符一起搭配拼音。

(a) 與ㄧ結合的結合韻

「ㄧ」只能和ㄚ、ㄛ、ㄝ、ㄞ、ㄠ、ㄡ、ㄢ、ㄣ、ㄤ、ㄥ等韻符結合，成為結合韻，而ㄧㄛ結合的結合韻只能單獨使用。

(b) 與ㄨ結合的結合韻

「ㄨ」只能和ㄚ、ㄛ、ㄞ、ㄟ、ㄢ、ㄣ、ㄤ、ㄥ結合，且拼音後的結合韻不與ㄅ、ㄆ、ㄇ、ㄈ、ㄐ、ㄑ、ㄒ結合。

(c) 與ㄩ結合的結合韻

「ㄩ」只和ㄝ、ㄢ、ㄣ、ㄥ結合為結合韻，而聲符中只有ㄋ、ㄌ、ㄐ、ㄑ、ㄒ可與結合後的結合韻結合。

在我們的手語辨識系統中，我們要讓判斷出來的注音符號透過程式自動進行拼音，讓我們的輸出可以不要只是注音符號，而是一個拼好的字，所以我們參考了注音的拼音規則，而將拼音的過程做成了一個判斷流程圖。

但因為程式的問題，所以沒辦法直接用注音符號訓練 yolo，所以要用其他可以判斷的符號代替，我們後來選擇使用英文字母的 a ～ z 以及 ab ～ al 代表注音符號的ㄅ～ㄦ，再加上避免資料判斷錯誤的 face 還有作為開始及結束的 hand 總共有 39 個 label 名稱。

2. 拼音判斷流程

根據前面所述的注音拼音規則，我們可以找出一個較有系統的拼音規則，可以統整出如流程圖中的判斷程序。在每個字之間，我們用 hand 手勢分隔每個字的拼音，當資料進來的時候，我們會先判斷手勢的姿態是聲符、介符或韻符。

圖 14　擷取拼音

(1) 拼音前判斷

(a) 當判斷為聲符時，再判斷是否可單獨成字，如果能單獨成字就需特別標記，在拼音時特別注意。

(b) 當資料進來是介符(ㄧㄨㄩ)時，因為介符可以單獨成字、作為結合韻、也可只和聲符做結合，變化很多，所以留到要拼音時才做更多的判斷。

(c) 當資料進來的是韻符(ㄚㄛㄜㄝㄞㄟㄠㄡㄢㄣㄤㄥㄦ)時，都可以單獨成字和拼音，也留到拼音時再做判斷。

(2) 拼音時判斷

透過兩個 hand 之間符號的個數，我們可以簡單地將注音拼音分成三大類：一個符號、兩個符號、三個符號，再分成這三大類之後，就能更有效率地拼音。

(a) 一個符號

當資料只有一個符號時，而這些符號必定是能單獨成字的，所以我們只需要將判斷出的符號直接輸出即可。

(b) 兩個符號

資料有兩個符號時可能的情況有下列三種：聲符+介符、聲符+韻符、韻符+介符，其中聲符+介符與聲符+韻符沒有特別的規範，可以直接輸出。而當韻符和介符組合時(結合韻)就有較多的規範，需要依照上段中提到的規則拼音。

(c) 三個符號

當有三個符號時，只能是聲符+介符+韻符(聲符+結合韻)，因為其中必定有結合韻，所以也需要參考結合韻的規則。

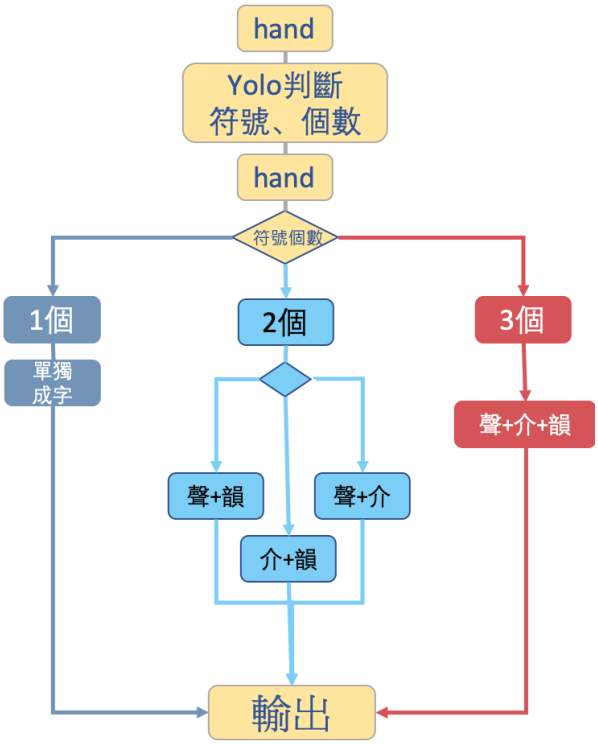


圖 15 程式判讀流程圖

三、 研究結果與討論

(一)、 研究結果

1. 訓練結果:

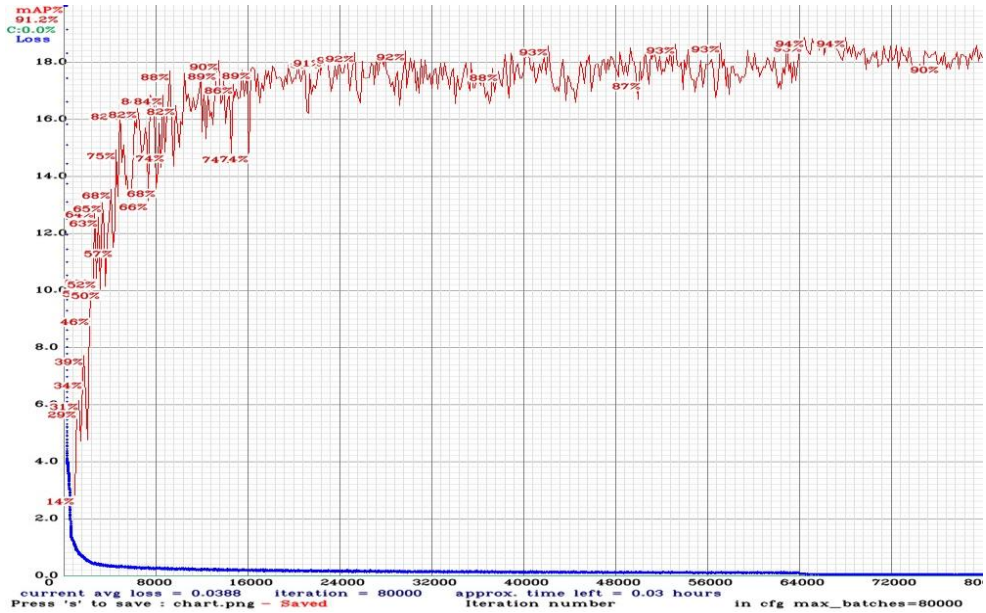


圖 16 mAP 與 Loss 關係圖

2. 照片辨識研究成果：

使用 Yolov3 或 Yolov4 辨識靜態圖片時，不管是訓練或非訓練的資料都表顯得十分良好。

圖 17

3. 影片辨識研究成果：

利用網路上的手語教學影片測試 37 個注音有 7 個(ㄇ，ㄔ，ㄖ，ㄝ，ㄡ，ㄧ，ㄩ)辨識不太出來，辨識率：81.08 %

https://youtu.be/6G9Knhv4f_A

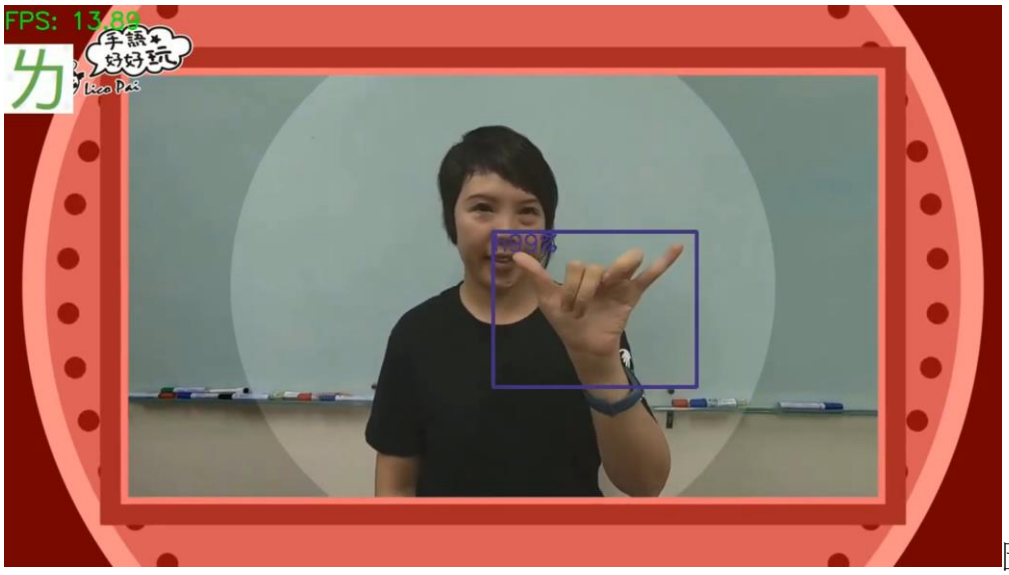


圖 18

4. Real time (即時辨識) 研究成果：

除了 n(ㄋ:58.8%)，t(ㄊ:46.8%)，w(ㄨ:32.2%)，x(ㄐ:40.7%)， z(ㄗ:44.5%)，ag(ㄡ:24.9%)大部分注音的辨識率都高於 60%。其中以”ㄡ”的辨識度最低，”ㄏ”的辨識度最高(99.7%)

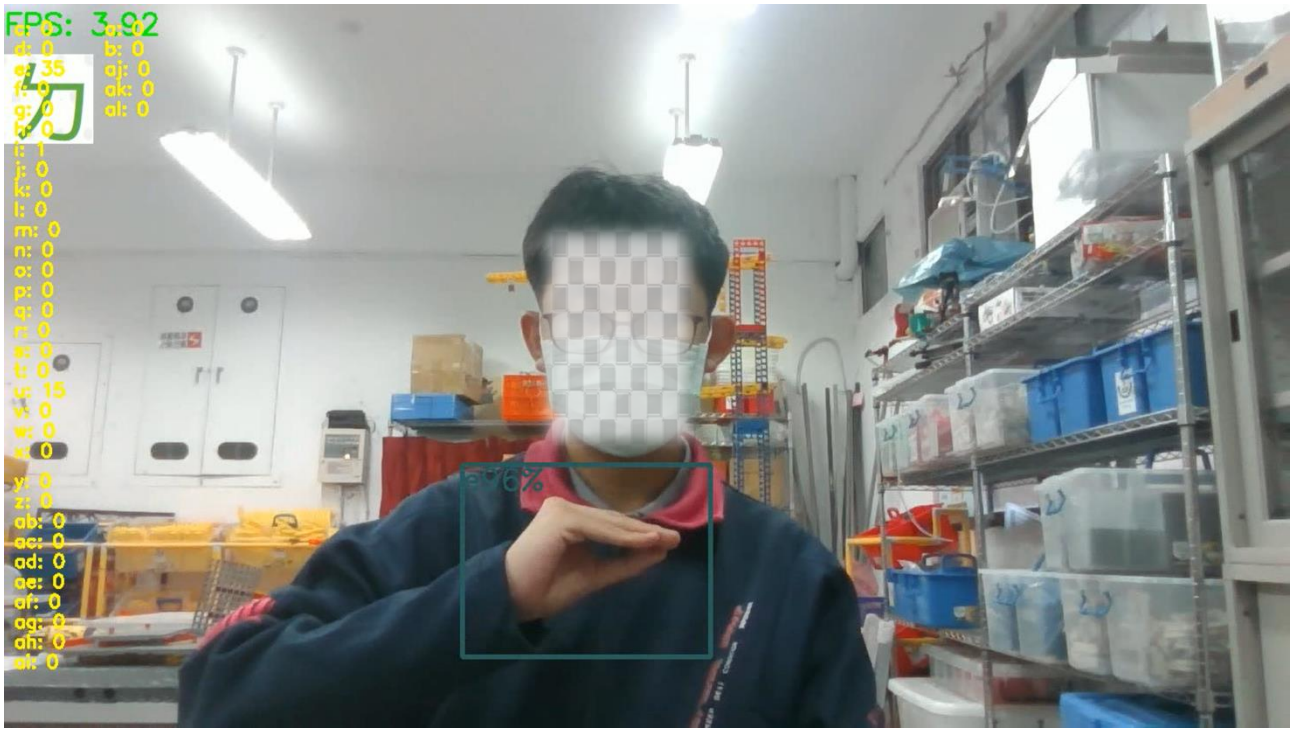


圖 19

Confussion Matrix																																			Precision	
a	488	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	1	0	0		94.4%	a
b	0	373	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	1	90	0		79.0%	b
c	4	6	261	0	0	0	55	0	0	0	0	0	0	0	0	0	0	3	53	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	68.1%	c
d	0	0	0	385	0	0	4	0	0	0	0	4	0	0	0	1	0	0	0	0	0	0	2	0	0	3	0	43	0	0	0	0	0	1	0		86.9%	d
e	0	0	0	0	494	0	0	0	0	0	0	14	0	0	1	0	0	12	0	2	1	0	0	0	0	8	5	0	100	0	0	0	0	0	1		77.4%	e
f	0	0	0	0	0	288	1	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	12	55	0	0	0	0	0	0	0	0	0	0		76.6%	f
g	0	0	9	0	0	0	313	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	0	4	0	0	0	0	0	1	0	6	3	0	0	2	92.3%	g
h	0	1	7	0	0	0	0	266	0	1	0	4	0	0	0	0	0	13	8	1	0	0	0	30	0	0	0	0	0	0	0	0	0	2	0	0	1	2	79.6%	h
i	1	0	0	3	0	0	0	0	238	0	0	0	0	0	11	0	25	19	1	0	0	0	1	0	0	0	0	0	0	0	0	4	58	3	0	0	2	65.4%	i
j	6	0	0	0	0	0	0	1	350	0	0	0	0	0	0	15	24	2	3	34	0	0	1	0	0	0	0	5	48	0	0	0	0	10	0		70.1%	j
k	0	0	0	0	0	0	0	0	1	0	581	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		99.7%	k
l	1	0	0	0	0	0	0	0	2	0	0	416	0	0	1	0	6	48	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	74		69.9%	l
m	3	0	0	0	0	0	0	1	0	0	0	0	358	0	0	0	0	0	0	0	0	10	0	0	0	0	25	0	0	0	0	6	1	2	113	0		69.0%	m
n	0	98	0	0	0	0	0	0	0	0	107	0	0	298	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0		58.8%	n
o	2	0	0	0	0	0	0	0	0	0	0	0	0	0	534	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0		99.4%	o
p	0	0	3	13	0	0	3	0	0	0	0	0	0	0	0	251	0	8	0	10	83	0	0	0	0	2	1	0	0	0	0	4	0	4	0	0		65.7%	p
q	5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	376	15	12	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	91.7%	q
r	22	0	0	0	0	0	0	0	2	0	0	12	0	0	1	0	0	446	0	2	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	7		86.3%	r
s	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	299	0	0	0	0	0	0	0	2	0	0	0	0	0	37	0	0	0		72.0%	s
t	0	0	0	0	37	0	0	2	0	0	1	0	0	0	74	0	0	6	0	225	0	0	0	24	0	0	0	0	0	0	0	0	0	112	0	0		46.8%	t
u	0	0	0	0	27	0	114	3	0	0	0	0	0	0	0	20	0	0	0	13	308	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0		63.1%	u
v	5	57	0	0	0	0	4	0	0	0	0	0	0	0	12	6	0	0	0	0	6	212	0	0	0	0	0	0	8	0	0	0	0	0	21	0	2	64.0%	v
w	141	0	0	0	0	0	0	0	0	4	0	0	0	0	10	0	0	176	11	0	18	0	180	0	1	0	0	0	0	1	0	0	0	0	0	0		33.2%	w
x	0	0	0	0	0	0	5	40	11	0	0	14	0	0	26	0	0	25	0	132	0	0	0	0	188	0	0	0	0	3	0	0	0	0	0	18	0		40.7%	x
y	1	0	0	0	0	53	2	0	0	0	0	0	0	0	1	0	3	5	1	8	0	0	0	9	248	0	0	0	2	0	0	0	0	4	0	0		73.6%	y
z	0	0	5	0	23	24	0	0	0	0	0	0	0	0	0	0	0	1	0	0	5	0	0	0	0	174	138	18	2	0	0	1	0	0	0	0		44.5%	z
ab	0	0	0	4	63	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	359	9	0	0	0	0	0	0	0	0		81.0%	ab
ac	0	0	0	0	45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0	12	190	0	2	0	0	4	0	0	0		61.5%	ac
ad	0	0	0	0	0	0	1	0	0	10	9	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	496	1	9	0	0	0	0	16		91.0%	ad
ae	0	0	0	0	30	4	0	0	0	12	0	7	0	0	0	0	0	0	3	0	19	0	0	0	0	0	0	0	0	360	0	0	1	2	0	2		81.8%	ae
af	12	0	0	0	19	0	0	1	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	422	0	17	0	0	9	79		75.0%	af
ag	0	42	0	0	1	61	0	0	0	46	0	0	0	0	0	0	7	1	0	0	6	17	15	0	0	0	0	0	0	0	0	78	3	36	0	0	0	3	24.9%	ag
ah	0	0	0	2	0	58	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	2	0	9	0	349	36	3	0	0		75.5%	ah
ai	0	0	0	0	0	19	19	3	9	0	6	0	0	0	0	0	0	1	10	14	0	0	0	15	0	0	0	0	0	0	0	0	0	295	0	0	0		75.4%	ai
aj	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	6	0	0	0	0	0	0	0	0	0	4	0	0	0	48	0	414	0	0		86.4%	aj
ak	0	0	0	1	0	0	0	5	0	0	0	0	35	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	2	380	1		85.0%	ak
al	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	47	2	7	0	1	0	1	0	467		87.6%	al
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	ab	ac	ad	ae	af	ag	ah	ai	aj	ak	al	N/A	n=10	
Prediction																																								

	>=40
	GT
	NA

圖 20 即時辨識混淆矩陣(n=10)

5. 拼音判斷研究成果：

拼音的辨識並不太成功，原因是因為判讀的資訊跳動非常快，還有就是有些符號辨識度不高，所以辨識成功組成一個正確的字成功機率不高。

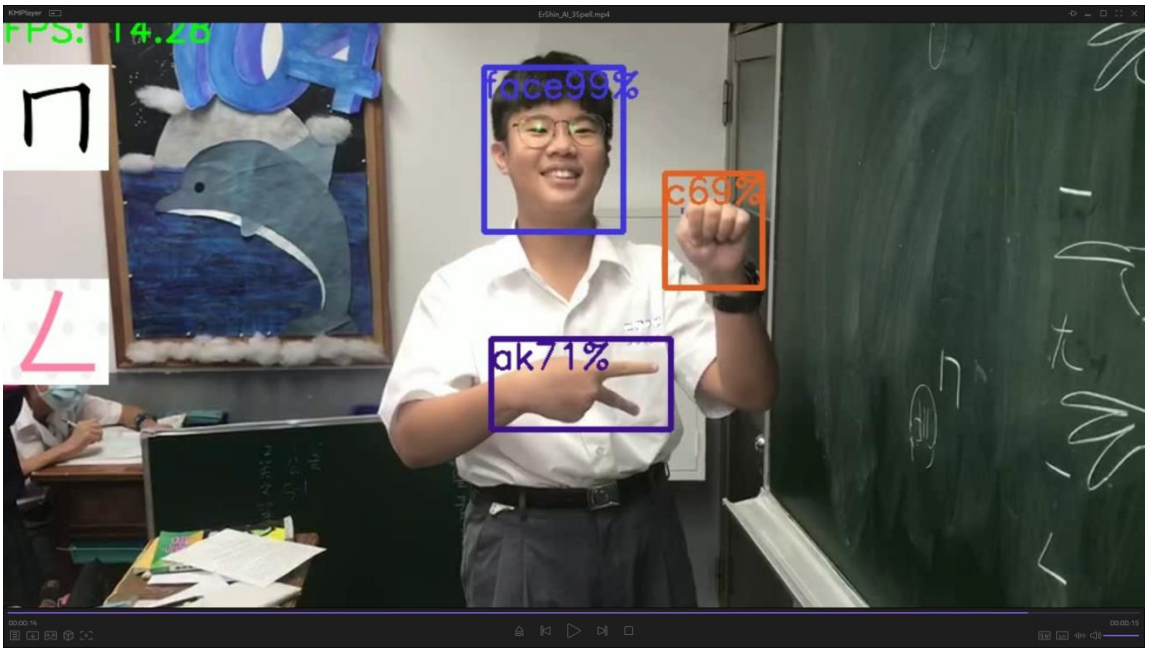


圖 21

6. 連結語音研究成果：

原本判讀影像時 FPS 都在 14 以上，但加上發音後 FPS 有時瞬間降到 1 以下，音質雖然沒變，但反應時間加長，延遲反應很明顯。

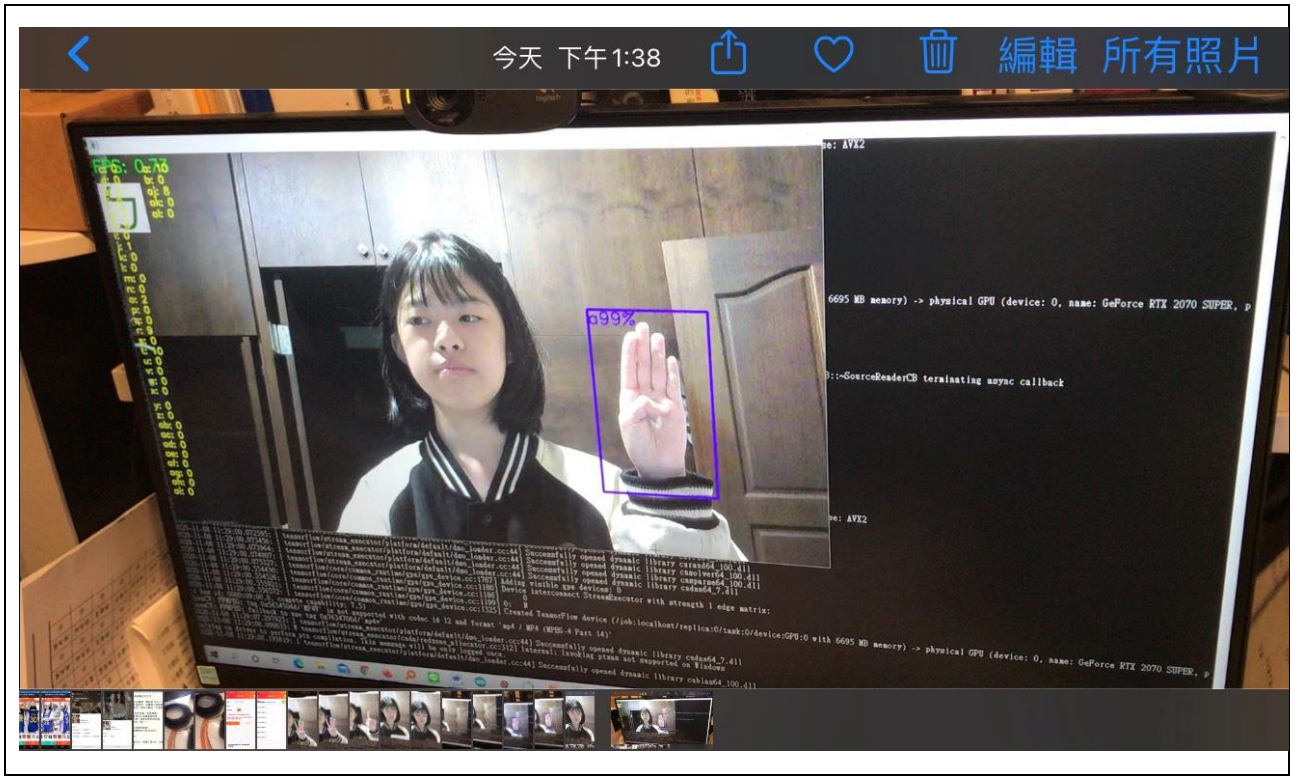


圖 22

https://youtu.be/8zxmHpaDOcg

(二)、 討論

在這次訓練 Yolo 的過程中，我們發現了幾個較大的問題和障礙：

1. Yolov3 無法辨識影片檔案，僅能處裡圖片檔。

在起初我們看中 Yolov3 在訓練時間相對較短的優勢下而採用了此技術作為訓練基礎， 不過當我們將影片檔案輸入進行測試時，卻發現 Yolov3 並沒有辦法跟上影片速度辨識，導致實際上是沒辦法得到結果的。也因此我們看中 Yolov4 其每秒近 60fps 的辨

識速度優勢，而改成訓練 Yolov4，但訓練過程相對不順利許多，包含環境建置上的錯誤和標籤順序錯誤。環境無法建置或建置錯誤。

2. 在設定程式和相關文檔時，我們誤將 classes.txt 檔案和程式的類別順序誤植，導致最後測試時類別顛倒，還好最後有發現問題出在設定，並非原始檔案錯誤。

3.標籤錯誤問題

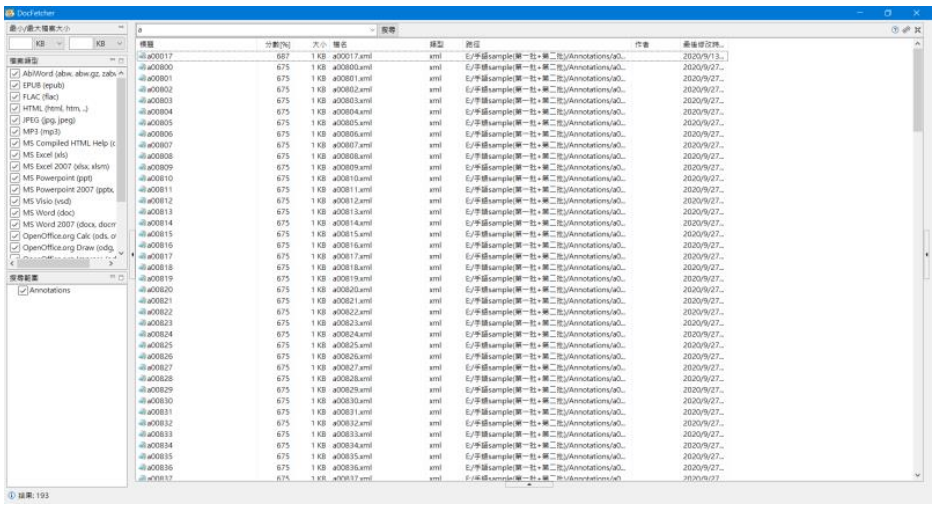
在這次的研究中，資料標籤作業皆由作者三人協力完成，我們在短短兩周內尋找願意協助拍攝人員和標籤工作，導致標籤產生了不少疏失，最後也在訓練上產生了影響。事後我們以「DocFetcher」程式進行批次搜尋，也成功解決相關問題。

4. darknet 檔案讀取問題

因為 darknet 無法讀取 xml 檔，所以我們需要將原本用 labelImg 標示的 xml 檔透過 xml to txt

的程式進行轉檔。

圖 23 利用 DocFetcher 找錯誤標籤



5.左右手辨識問題

原本的程式會將左右手比相同動作視為同一個，但注音口手語左右手比的符號是不同的，所以要加一行 「--add flip=0」指令進 cfg 中，避免程式自動利用鏡像擴增資料。

6. 辨識結果名稱相反

因為在 yolo 的 classes.txt 以及 darknet 中標籤名稱順序輸入不同，如果資料順序錯誤，就

會發生將臉（face）辨認成手（hand）的狀況，需要將其順序改成一樣。

7.訓練時遇到的其他問題：

遇到的問題	解決辦法
1. Resource Exhausted	GPU 的效能不夠支持訓練，所以我們將程式中原本設 32 的 batch size 調成 4(batch size 只能調 2 的平方)，在調整之後就可以繼續訓練，也沒有在顯示這樣的錯誤了。
2. 辨識程式正常運行，但沒辦法偵測手勢	--model,--classes 的參數沒指定模型
3. 臉部正常辨識，手部無法辨識	可能是資料不平衡導致，因為我們每筆資料都有將 face 標示出來，可能讓 face 太多，而其他資料相較起來太少，所以我們將 face 的一部分刪除，讓資料可以較平衡。
4. 訓練週期錯誤	基本是 classes 數 2000 倍，若<6000 就改成 6000
5. Hand、face 辨識相反	Object.name 和 classes.txt 中 label 名稱順序不同導致的狀況，一樣的順序才能顯示出對的 label
6. Epoch 00100: early stopping	正常訊息

而在訓練過程中，我們面對了訓練時間的大輻增長，一開始僅跑少數標籤分類就花了近八小時，導致我們一度擔心若全部訓練是不是會花上近兩周的時間，不過在中間我們發現 Yolov4 每 1000 次訓練就會存下權重檔，而且可以從中斷的地方繼續再訓練，所以就算程式意外終止還是可以繼續執行訓練。

四、 結論與應用

（一）、 結論

1. Yolov 模型辨識速度非常快，確實可以在外加顯卡(GPU)的情況下做到即時的影像辨識。Yolov3 圖片辨識很精準，但影片辨識速度就不如 Yolov4 快，在本實驗的配備下就無法使用。
2. 如果加上語音，電腦速度明顯變慢有嚴重的延遲現象，很顯然這是電腦資源不足，如果可以提升電腦設備或是改變演算法應該可以解決這個問題。所以開發一個即時的手

語翻譯工具應該是指日可待的。

（二）、 應用

1. 未來希望可以開發成電腦 App，方便使用者攜帶和使用。
2. 可以應用在無法發聲的環境例如潛水作業、外太空的環境或是軍事行動等等需要靜音執行任務的工作場合。

五、 參考文獻

（一）、 論文書籍

林大貴(2017)。**TensorFlow+Keras 深度學習人工智慧實務應用。**
出版商:博碩文化股份有限公司

李立宗(2017)。**科班出身的 AI 人必修課：OpenCV 影像處理 使用 python。**
出版商:深智數位股份有限公司

蘇宇楨(2019)。運用深度學習辨識手語手勢之創新研究。世新大學資訊管理學系

Alexey Bochkovskiy,Chien-Yao Wang＆Hong-Yuan Mark Liao.(2020), “**YOLOv4: Optimal Speed and Accuracy of Object Detection.**”
Retrieved from https://arxiv.org/abs/2004.10934

（二）、 網路資源

國語口手語音標手勢, **常用手語辭典**,
引用自 https://signlanguage.moe.edu.tw/basis/detail/a45f6a67-ac3e-4214-bd25-f44acfd7fae7

YOLOv4, **Jonathan Hui**,
引用自 https://jonathan-hui.medium.com/yolov4-c9901eaa8e61

卷積神經網路— 卷積運算、池化運算,**Tommy Huang**,
引用自 https://medium.com/@chih.sheng.huang821/卷積神經網路-convolutional-neural-network-cnn-卷積運算-池化運算-856330c2b703

深度學習系列: 什麼是 AP/mAP?, **Tommy Huang**,
引用自 https://medium.com/@chih.sheng.huang821/深度學習系列-什麼是 ap-map-aaf089920848

深入淺出 Yolo 系列之 Yolov3&Yolov4&Yolov5 核心基礎知識完整講解, 江大白,

引用自 https://zhuanlan.zhihu.com/p/143747206
YOLOv4 中的 Mish 激活函数, Miracle R,
引用自 https://blog.csdn.net/moxibingdao/article/details/108289489
YOLO object detection with OpenCV。
引用自 https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/
Sigmiod、Tanh、Relu、Mish...等激活函數的那點事，江大白，
引用自 https://zhuanlan.zhihu.com/p/139696588
YOLO V4 — 網絡結構解析，江大白，
引用自 https://zhuanlan.zhihu.com/p/150127712

（三）、 圖片來源：

圖 1：https://arxiv.org/pdf/2004.10934.pdf
圖 2：https://miro.medium.com/max/1200/1*iHJoSKL4MbsmEe8Cwt_CbQ.png
圖 3：https://ithelp.ithome.com.tw/articles/10209518
圖 4：https://zhuanlan.zhihu.com/p/143747206
圖 5：https://zhuanlan.zhihu.com/p/143747206
圖 6：https://www.programmersought.com/article/90494548653/
圖 7：https://zhuanlan.zhihu.com/p/143747206
圖 8：https://zhuanlan.zhihu.com/p/143747206
圖 9：https://zhuanlan.zhihu.com/p/143747206
圖 10：https://zhuanlan.zhihu.com/p/143747206
圖 11：https://zhuanlan.zhihu.com/p/143747206

【評語】190027

本研究主題清楚且聚焦，且可用科學方法檢驗研究成果。建議實驗需以實際慣用手語進行辨識為宜。