

國立宜蘭大學電子工程學系

碩士論文

Department of Electronic Engineering

National Ilan University

Master Thesis

深度學習應用於自駕車交通標誌辨識

Recognition of Traffic Signs by Deep Learning  
for Self-Driving Vehicles

研究生：唐嘉宏

Graduate Student : Jia-Hong Tang

指導教授：游竹 博士

Advisor : Chu Yu, Ph. D.

中華民國 110 年 7 月

July 2021

國立宜蘭大學碩士學位論文  
指導教授推薦函

電子工程學系碩士班 唐嘉宏君所提之

論文（題目）： 深度學習應用於自駕車交通標誌辨識

Recognition of Traffic Signs by Deep Learning for Self-Driving Vehicles

係由本人指導撰述，同意提付審查。

指導教授 游竹 (簽章)

系所主管 邱建文 (簽章)

中華民國 110 年 04 月 27 日

國立宜蘭大學碩士學位論文  
口試委員會審定書

電子工程學系碩士班唐嘉宏君所提之

論文（題目）：深度學習應用於自駕車交通標誌辨識

Recognition of Traffic Signs by Deep Learning for  
Self-Driving Vehicles

經本委員會審議，認定符合碩士資格標準。

學位考試委員

夏至賢 嚴茂旭 游竹

指導教授

游竹

中華民國 110 年 6 月 25 日

## 中文摘要

近年來交通事故的頻繁發生，已經成為現代人不可忽視的一項重大問題，而隨著深度學習的蓬勃發展，許多大廠也發展出了智慧自動駕駛車以及駕駛員輔助系統，技術等級持續推升中。對於是否能夠正確的辨識出道路所出現的不同情況，如車輛之間的距離、車子周圍是否有行人以及路旁兩側不同種類的交通標誌等，這些都是自動駕駛車及輔助系統中，非常重要的安全措施。除了識別的性能之外，對於是否能夠做到即時檢測成為重點，本論文以交通標誌中的限速標誌為例，強化辨識能力，以提供自動駕駛車更準確安全駕駛的輔助系統。我們以目前公認辨識能力與運算速度皆最佳的 YOLOv4 深度學習演算法為基礎，採用的方法是先使用 YOLOv4 進行限速標誌的初步檢測，再透過一級新增的卷積神經網路分類器，執行進一步細部分類。經實驗結果顯示，使用原始 YOLOv4 進行檢測限速標誌 30~110 km/h，平均 mAP 為 0.73；在同一組影像資料下，採用的方法平均 mAP 達 0.95，大幅提升 30%，顯著提升交通限速標誌正確率。最後，基於使用 Geforce RTX 2080 Ti 的 GPU 顯示卡，檢測一張圖像大小為 640×640，整體執行時間約為 26 毫秒，新增分類器僅增加約 2 毫秒的執行時間，平均 1 秒鐘能夠檢測約 38 張圖像，符合即時交通標誌檢測的要求。

關鍵詞：深度學習、智慧自動駕駛車、YOLOv4、卷積神經網路

## **Abstract**

In recent years, traffic accident has become a major problem for traffic jam. With the vigorous development of deep learning, many large manufacturers have also developed smart autonomous vehicles and driver-assistance systems. The level of the technology is also continuously promoted. Being able to correctly identify different situations on the road, such as the distance between vehicles, pedestrians around the vehicle, and various types of traffic signs, are the important tasks in autonomous vehicles and driver-assistance systems. Moreover, in addition to the performance of recognition, real-time detection is also essential for self-driving vehicles. In this Thesis, the speed limit sign in traffic signs was chosen as the target for application in self-driving vehicles. We adopted YOLOv4 as the algorithm of detection of speed limit signs, and then add an additional convolutional neural network (CNN) for refining the results. Compared with the original YOLOv4, this work can improve the accuracy rate and avoid identifying wrong traffic signs. The mAP of the original YOLOv4 is 0.73, while the one of this work is 0.95. Our work obtains a significant increase of 30% for accuracy rate. Finally, for an image of  $640 \times 640$ , the total execution time is about 26 ms using an accelerator of Geforce RTX 2080 Ti GPU. The additional CNN only increases about 2 ms. The average detection time of this work is about 38 images/second, which meets the requirement of real-time detection.

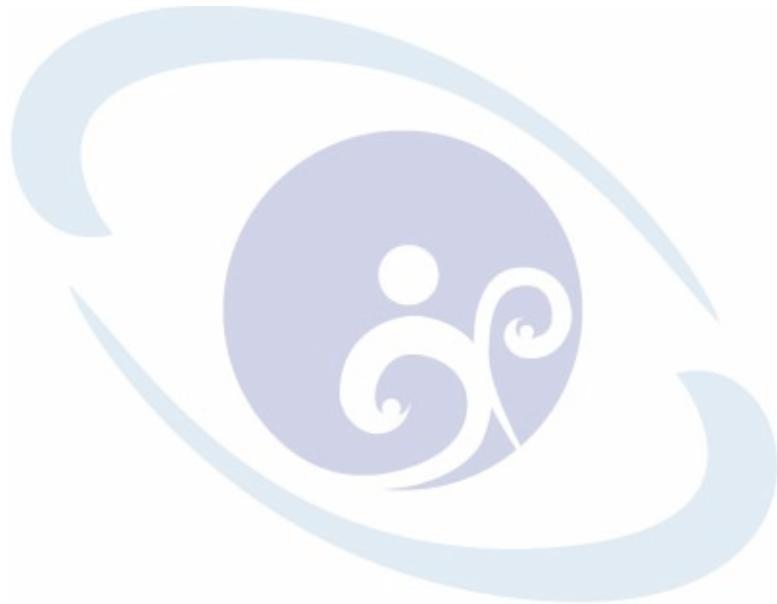
**Keyword:** Deep Learning, Autonomous Vehicles, YOLOv4, Convolutional Neural Network

## 目錄

中文摘要 .....	I
Abstract .....	II
目錄 .....	III
圖目錄 .....	VI
表目錄 .....	IX
第 1 章 緒論 .....	1
1.1 背景 .....	1
1.2 研究動機 .....	2
1.3 論文架構 .....	3
第 2 章 研究方法 .....	4
2.1 簡介 .....	4
2.2 類神經網路 .....	4
2.2.1 卷積神經網路 .....	5
2.2.2 卷積層 .....	6
2.2.3 池化層 .....	7
2.2.4 Dropout 層 .....	7
2.2.5 Softmax 層 .....	8
2.2.6 激勵函數 .....	8
2.2.7 批次正規化(Batch Normalization, BN) .....	9
2.2.8 反向傳遞演算法(Backpropagation) .....	10
2.2.9 卷積神經網路測試 .....	11
2.3 目標檢測器 .....	12
2.3.1 兩段式目標檢測器 .....	14
2.3.1.1 Mask R-CNN .....	16
2.3.1.2 RoI 池化(Pooling) .....	17

2.3.1.3 RoI 對齊(Align).....	18
2.3.1.4 全卷積網絡.....	18
2.3.2 單段式目標檢測器 .....	19
2.3.2.1 YOLOv4.....	20
2.3.2.2 YOLOv4 架構.....	21
2.3.2.3 跨階段局部網路[8].....	22
2.3.2.4 路徑聚合網路[14] .....	23
2.3.2.5 空間注意力模塊(Spatial Attention Module Block, SAM-block)[15] .....	24
2.3.2.6 數據增強技術.....	25
2.3.2.7 DropBlock 正則化(Regularization) [10].....	26
2.3.2.8 Mish 激勵函數 .....	26
2.3.2.9 損失函數的重疊度問題[13].....	28
2.3.2.10 DIoU-NMS [12] .....	30
2.3.2.11 交叉微批次正規化(Cross mini-Batch Normalization, CmBN) .....	31
2.3.2.13 消除格點敏感度(Eliminate Grid Sensitivity).....	32
2.3.2.14 多個錨點用於一個目標框.....	32
2.3.2.15 餘弦退火排程器(Cosine Annealing Scheduler) ..	32
2.3.2.16 各種演算法的比較實驗結果.....	32
2.3.3 Mask R-CNN 與 YOLOv4 檢測結果.....	33
第 3 章 標誌檢測實現.....	35
3.1 簡介 .....	35
3.2 資料收集 .....	35

3.3 網路訓練 .....	39
3.3.1 YOLOv4 訓練 .....	39
3.3.2 卷積神經網路訓練 .....	42
第 4 章 實驗結果與比較 .....	43
4.1 混淆矩陣 .....	43
4.2 評估 .....	44
4.3 檢測結果 .....	44
第 5 章 結論 .....	51
參考文獻 .....	52



## 圖目錄

圖 2-1、神經元示意圖 .....	4
圖 2-2、神經網路架構 .....	5
圖 2-3、卷積運算示意圖 .....	6
圖 2-4、池化計算示意圖 .....	7
圖 2-5、Dropout 示意圖 .....	8
圖 2-6、Softmax 示意圖 .....	8
圖 2-7、激勵函數圖形 .....	9
圖 2-8、初始網路架構 .....	10
圖 2-9、卷積神經網路 .....	11
圖 2-10、目標物檢測圖層三步驟[4] .....	13
圖 2-11、通用型目標物檢測演進[5] .....	13
圖 2-12、對目標物體檢測不同的處理方式[17] .....	14
圖 2-13、R-CNN 演算法處理方式[16] .....	14
圖 2-14、R-CNN 系列的 mAP 效能比較[4] .....	16
圖 2-15、Mask R-CNN 的實例分割[6] .....	16
圖 2-16、MaskR-CNN 的實例分割[6] .....	17
圖 2-17、RoI 池化 .....	18
圖 2-18、RoI 對齊 [24] .....	18
圖 2-19、以卷積層取代全連接層[7] .....	19
圖 2-20、YOLO 檢測系統[23] .....	19
圖 2-21、YOLO 模型結構[23] .....	20
圖 2-22、YOLOv4 架構圖 .....	22

圖 2-23、CSPNet 骨幹.....	22
圖 2-24、CSPNet 簡化結構.....	23
圖 2-25、用於 YOLOv4 的路徑聚合網路 .....	23
圖 2-26、空間注意力模塊.....	24
圖 2-27、用於 YOLOv4 的空間注意力模塊 .....	24
圖 2-28、CutMix 示意圖 .....	25
圖 2-29、馬賽克示意圖 .....	26
圖 2-30、DropBlock 正則化[10] .....	26
圖 2-31、Mish 激勵函數 .....	27
圖 2-32、IoU 示意圖 .....	28
圖 2-33、GIoU loss 的計算方式 .....	29
圖 2-34、DIoU loss 的計算方式 .....	29
圖 2-35、BN、CBN 與 CmBN[12].....	31
圖 2-36、各種演算法的比較結果[12].....	33
圖 2-37、Mask R-CNN 檢測結果 .....	34
圖 2-38、YOLOv4 檢測結果.....	34
圖 3-1、影像辨識流程圖.....	35
圖 3-2、Google 地圖截圖影像.....	36
圖 3-3、交通標誌資料集影像.....	36
圖 3-4、Mosaic 影像 .....	37
圖 3-5、圖像標註.....	37
圖 3-6、標誌裁切後影像.....	38
圖 3-7、YOLOv4 訓練圖.....	39

圖 3-8、YOLOv4 測試(遠).....	40
圖 3-9、YOLOv4 測試(中).....	41
圖 3-10、YOLOv4 測試(近).....	41
圖 3-11、分類網路訓練損失曲線圖 .....	42
圖 3-12、測試集部分預測結果.....	42
圖 4-1、單獨使用 YOLOv4 的混淆矩陣 .....	43
圖 4-2、YOLOv4+CNN 的混淆矩陣.....	43
圖 4-3、限速標誌檢測(遠).....	45
圖 4-4、限速標誌檢測(中).....	46
圖 4-5、限速標誌檢測(近).....	47
圖 4-6、YOLOv4+CNN 的錯誤檢測.....	49
圖 4-7、單獨使用 YOLOv4 的錯誤檢測.....	50

## 表目錄

表 2-1、驗證以及測試集的 Top-1 準確率 .....	12
表 2-2、各種激勵函數的表現[25].....	27
表 3-1、YOLO 標註格式.....	38
表 4-1、兩種方法 AP 與 mAP .....	44



# 第1章 緒論

## 1.1 背景

人工智能在 1956 年時的達特茅斯會議中被正式被提出，距今已經有 60 幾年的歷史了，目的是希望機器的行為最終能夠像是人類一樣能夠自主行動且思考。而在這 60 幾年的發展當中，也先後迎來了三次的熱潮與兩次的低潮。

第一次熱潮出現在 1956 至 1974 年，又被稱為「古典人工智能」，在這些年裡出現了許多成功的人工智慧程序與新的研究方向，如搜索式推理。搜索式推理指的是機器為達成某一個目的，一步一步的去進行嘗試，就如同走迷宮一樣，若是遇到了死路就重新找尋另一條路繼續嘗試直至找到迷宮的出口。然而當時的人工智慧能處理一些較簡單的問題，如拼圖、象棋等簡單的遊戲，在面對現實生活中更複雜的問題時，無法獲得有效的解決方法，而隨著時間的推移，人工智能漸漸地淡出人們的視線中，於是迎來了第一次的低潮。

第二次熱潮出現在 1980 年代，人工智能開始走向了專業化，知識處理成為了研究熱點，發展出了能夠應用在某些特定的專業領域的程序，像是能夠應用在醫學上的醫療診斷系統，這種利用已知的專業知識並從中推算出邏輯規則，使問題能夠得以解決的程序又稱作專家系統。然而，1987 年到 1993 年 Apple 和 IBM 所生產的個人電腦，其效能遠超於所使用的 Symbolics 和 Lisp 等機器，再加上專家系統的維護費用居高不下且難以升級，政府經費開始下降，於是迎來第二次的低潮。

第三次浪潮出現在 2010 年代，隨著網際網路的快速發展，大量的資料流通在網路上以及計算機效能的突破，使得機器學習逐漸興起。機器學習是人工智能的一個分支，大致上可分成兩種方式，分別是監督式學習與無監督式學習，監督式學習指的是將事先以人為方式準備好的訓練資料來讓機器進行學習，例如給機器好幾張貓與狗的圖片，並且告訴機器哪些是貓那些是狗讓機器去學習。而無監督式學習與之相反，事前並不會告訴機器

這些圖片的類別，而是希望機器能夠從中學習貓狗的特徵並且對圖片做出正確的判斷。而近幾年受到人們關注的深度學習則是機器學習中的其中一個分支。

深度學習(Deep Learning)以人工神經網路為架構，與機器學習不同的是，它能夠在資料中找尋出特徵並且學習，不再以人為的方式進行特徵萃取。目前常用的數種深度學習架構，如深度神經網路、卷積神經網路(Convolutional Neural Network, CNN)和深度置信網路和迴圈神經網路已被應用在電腦視覺、語音辨識、自然語言處理、音訊辨識與生物資訊學等領域，並取得了極好的效果，目前也廣泛應用在各個方面上。以卷積神經網路為例，Wenbo Lan 等人基於 YOLOv2 進行改進，添加了三個直通層以提取網路淺層更多細粒度的行人特徵，並在行人檢測中取得了良好的效果[1]。Changxing Ding 與 Dacheng Tao 透過人為方式對影像進行模糊處理來豐富卷積神經網路的訓練數據，以加強對模糊影像的辨識，提出了 TBE-CNN 的新型卷積神經網路架構，有效地提取整體臉部影像，改善劇烈的姿勢變化和遮擋的困擾[2]。Khairul Anuar Ishak 等人提出了一種實時檢測系統，透過顏色分割以及形狀辨識來找到限速標誌的位置，之後藉由多層前饋神經網絡來檢測限速的數字，達到分類的目的[3]。

## 1.2 研究動機

隨著人類文明的發展，交通工具已經漸漸成為人類生活中不可或缺的一環，直到現在幾乎是每一個家庭都會有一輛以上的汽車，但是隨之而來的問題是交通事故的增加，據行政院主計處統計，108 年道路交通事故為 34.1 萬件，較 107 年增 6.6%，而 109 年 1-8 月道路交通事故 23.4 萬件，較 108 年同期年增 5.1%，可見交通事故的發生正逐年增長中。

有鑑於此，如何預防以及減少交通事故的發生越來越受到大家的重視，因此以深度學習為主的自動駕駛車與駕駛員輔助系統也應運而生，其中如何辨識到交通標誌，並且了解其中含意，成為了主要技術發展目標，交通標誌的存在可提供駕駛員在駕駛時所需要的相關資訊，如禁止通行、迴轉

限制以及道路限速等，透過深度學習的影像辨識技術，以及特定感測器來收集交通標誌的訊息，並且讓機器自動做出決策，再根據交通標誌上的訊息，即時主動地做出反應又或是提醒駕駛員，讓駕駛員能夠專心的在前方的路況上，藉此以達到大幅降低交通事故的發生，以及保障用路人的生命財產。

依據內政部警政署取締違規分析統計，歷年來最大宗的違規事件為超速，若是使用深度學習的影像辨識技術，則可以即時辨識出交通標誌上的限速限制，並且語音提醒駕駛，以免發生事故或是違規事件。但一般深度學習的目標檢測網路，比較適合通用物件類別檢測，如行人、車子等等差異性較大的類別，若是用於交通標誌等外觀接近的目標，除了檢測外尚須辨識差異，例如限速 60 與限速 80 可能會被認為一樣，所以不單是套用時下最佳的演算法而已，仍需調整演算法，並針對各種標誌做正確判定。

### 1.3 論文架構

本文架構總共分成 5 章，第 1 章是緒論，敘述本論文中的相關背景資料與研究動機、人工智慧的發展與在不同領域上的應用。第 2 章為本論文的研究方法，介紹類神經網路的架構以及目標檢測器的發展。第 3 章介紹本論文所採用的檢測方法的各個步驟。第 4 章為本論文所採用方法的實驗結果，並與原始 YOLOv4 演算法所做的比較。最後在第 5 章作結論以及探討未來的研究。

## 第2章 研究方法

### 2.1 簡介

本章介紹與本研究所採用的相關方法與其技術，在 2.2 節介紹深度學習中的類神經網路以及卷積神經網路，2.3 節介紹目標檢測器以及 YOLO 系列與 R-CNN 系列的發展。

### 2.2 類神經網路

類神經網路 (Artificial Neural Network, ANN[15]) 又稱人工神經網路，是一種參考生物腦中的神經網路結構與功能的運算模型，在神經網路中通常由許多層所構成，裡面有著大量的人工神經元，能夠對外來的輸入給出對應的輸出。不同層之間的神經元彼此互相連接形成一個複雜的網路，並且能夠藉由適當地加深網路層數來提高模型學習能力使其能夠處理複雜程度較高的問題，最後透過預先輸入的大筆訓練樣本讓網路學習目標特徵來微調模型參數，得到讓電腦能夠像人一樣具有簡單的決定與判斷能力的模型。在類神經網路架構中，基本分為輸入層、隱藏層和輸出層。首先由輸入層接收資料，之後與隱藏層中的神經元權重經過計算得到一個值，最後經過激勵函數(Activation Function)得到一個純量結果，計算方式如式(2-1)所示。

$$o = f(\sum_{k=1}^n i_k * w_k + b) \quad (2-1)$$

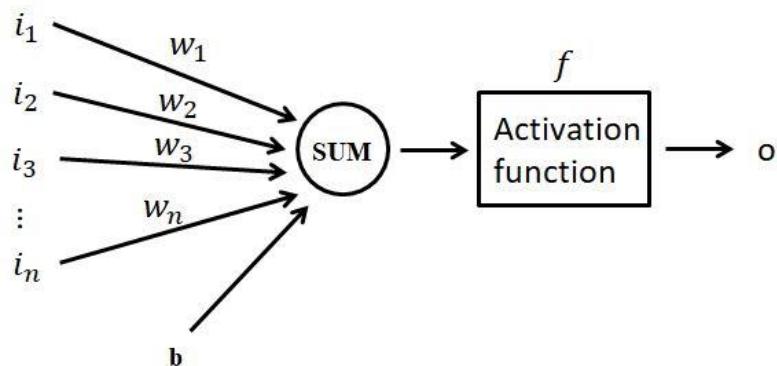


圖 2-1、神經元示意圖

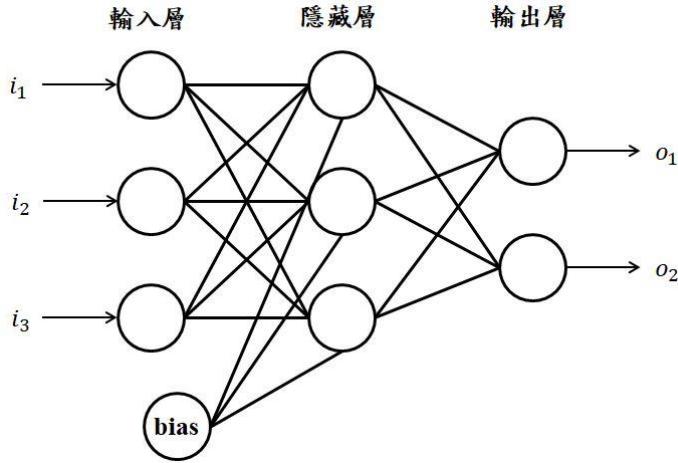


圖 2-2、神經網路架構

- 輸入層：為網路的起始層，主要功能是接收資料的輸入並將其傳遞至隱藏層，此層不會經過任何運算。
- 隱藏層：介於輸入層與輸出層之間，可以是一或以上的層數，層數越多越能夠處理較複雜的問題，但過多層數也可能會導致模型過擬合(Overfitting)。過擬合的意思是當你的模型學習到了訓練數據的特性，但卻不是你實際關心的目標物，一般構建機器學習系統的人，最擔心出現此問題。
- 輸出層：為網路的最後一層，接收來自最後一層隱藏層的輸出，用以判斷最後的結果。

而在訓練的過程中，網路也會透過如倒傳遞演算法(Backpropagation)，不斷地修正隱藏層當中的每個神經元之間的權重值，使神經網路在學習的過程中，讓所得到的預測值與實際值之間的誤差，慢慢變小使網路能夠獲得更好的性能。

### 2.2.1 卷積神經網路

卷積神經網路是一種前饋式神經網路，通常由數個卷積層和頂端的全連接層所組成，同時也包括對應的權重和池化層。與類神經網路中的神經元不同的是，它可以反應出一部份涵蓋範圍內的結果，使網路能夠利用輸入資料的二維結構，所以對於圖像與語音辨識方面的議題，能夠得到更好

的結果。下面小章節將介紹卷積層、池化層、激勵函數，以及相關常用的技術。

### 2.2.2 卷積層

卷積層主要是透過滑動不同大小的卷積核(Kernel Map)，在輸入圖像上進行卷積運算形成特徵圖，在每一個滑動的位置上，卷積核與輸入圖像之間會執行一個點對點的對應乘積，並加總這些數值，以便將感受野(Receptive Field)內的資訊投影到特徵圖(Feature Map)中一個點像素。而滑動的過程可稱為步幅(Stride)，步幅能夠控制特徵圖尺寸，步幅越大，特徵圖尺寸越小。

如圖 2-3 所示，將卷積核大小設為  $3 \times 3$ ，步幅大小設為 1，經過卷積後的特徵圖大小變為  $3 \times 3$ 。首先從圖左上角開始，依照對應的卷積核大小進行卷積，執行完一次卷積後，依照步幅大小繼續向右移動，直到卷積核完全掃過整張圖像。

1	0	1	1	0
1	1	0	1	0
1	0	0	1	1
0	0	0	0	1
0	1	1	1	1

\*

0	1	0
1	1	1
0	1	0

=

2		

1	0	1	1	0
1	1	0	1	0
1	0	0	1	1
0	0	0	0	1
0	1	1	1	1

\*

0	1	0
1	1	1
0	1	0

=

2	3	

1	0	1	1	0
1	1	0	1	0
1	0	0	1	1
0	0	0	0	1
0	1	1	1	1

\*

0	1	0
1	1	1
0	1	0

=

2	3	3

⋮

1	0	1	1	0
1	1	0	1	0
1	0	0	1	1
0	0	0	0	1
0	1	1	1	1

\*

0	1	0
1	1	1
0	1	0

=

2	3	3
2	1	3
1	1	3

圖 2-3、卷積運算示意圖

### 2.2.3 池化層

池化層的主要概念是對特徵圖進行下採樣(Downsampling)，相當於縮放，由於一張圖像當中有著大量的特徵，對於目標來說某些特徵可能是無用的或是重複的，且每個特徵間的相對位置遠比某特徵的準確位置還要來的重要，一般來說，會在網路中的卷積層之間周期性地加入池化層進行特徵刪減，不僅能夠減少網路的計算量，也可以保持特徵圖的不變性，常用的池化方式有最大池化(Max Pooling)與平均池化(Average Pooling)。

如同卷積層，池化也有一個核(Kernel)，也是在輸入圖像上進行滑動運算，但與卷積層不同的是在滑動的過程中，核不會重複覆蓋同一區域，並且依照不同的計算方式，分為取最大值得最大池化以及與平均值的平均池化。圖 2-4 是最大池化與平均池化的示意圖。

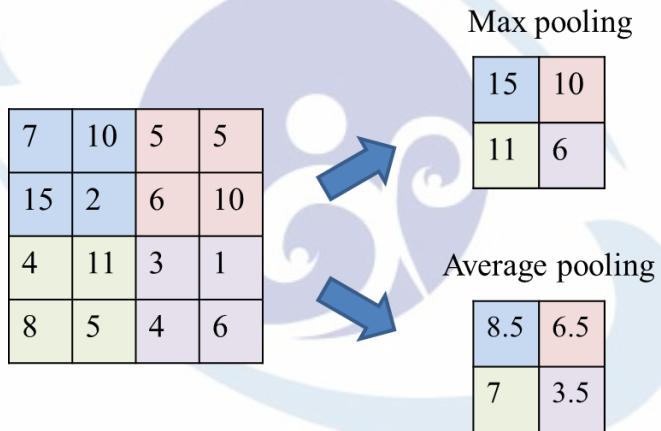
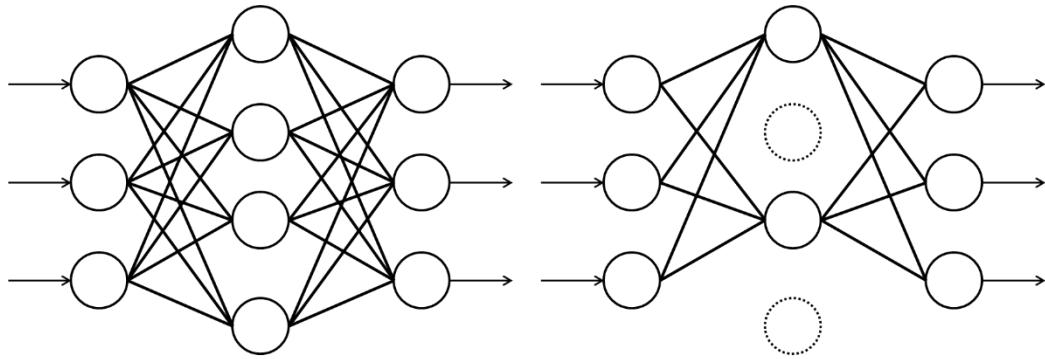


圖 2-4、池化計算示意圖

### 2.2.4 Dropout 層

當使用較小的訓練資料集來訓練神經網路時，訓練出來的模型很容易產生過擬合的現象，意思是該模型對於訓練資料集的數據擬合程度非常高，能夠得到較小的損失，預測準確率高；但是改換到訓練集以外的資料上時，所獲的損失就會比較大，導致預測準確率低下。Dropout 的方法可以在網路進行訓練時隨機丟棄部分的神經元，如圖 2-5 所示，並且在更新參數時忽略這些被丟棄的神經元，使模型盡可能的不會太偏向於某些區域的特徵，減少模型過擬合的現象。



一般的神經網路

經過Dropout後的神經網路

圖 2-5、Dropout 示意圖

### 2.2.5 Softmax 層

Softmax 層通常會放在類神經網路的最後一層將多分類的輸出值轉換成相對機率，將前一層的所有節點的輸出都經過指數函數運算後，個別的輸出作為分子，加總作為分母，形成數個介於 0 到 1 之間的值，並且所有的值總和為 1，依照最高的值作為分類的結果。

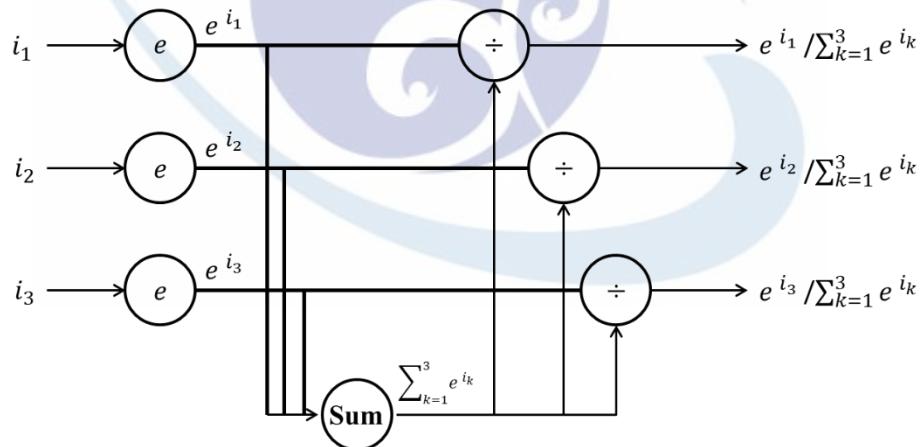


圖 2-6、Softmax 示意圖

### 2.2.6 激勵函數

在神經元中，輸入的值經過權重計算加總後會經過一個激勵函數，其作用可以讓神經網路具有非線性，沒有激勵函數的神經網絡只是線性回歸模型，其輸出與輸入為單一的線性組合，權重和偏差也只會進行簡單的線

性變換，雖然計算簡單，對於解決複雜問題的能力是有限的，從數據中學習複雜特徵圖的能力也較弱。目前比較常使用的激勵函數為 Relu 函數，由式(2-2)定義，它解決了 Sigmoid 函數在正區間時會有的梯度消失(Gradient Vanishing)的問題，但缺點是在負區間的梯度永遠為零，導致部分的神經元將永遠不會被啟動。為了解決這個問題，後來延伸出了 Leaky Relu 函數，由式(2-3)所定義，在負區間的部分有了不為零的梯度。圖 2-7 為兩種激勵函數的圖形。

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{if } x \leq 0 \end{cases} \quad (2-2)$$

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \lambda x, & \text{if } x \leq 0 \end{cases} \quad (2-3)$$

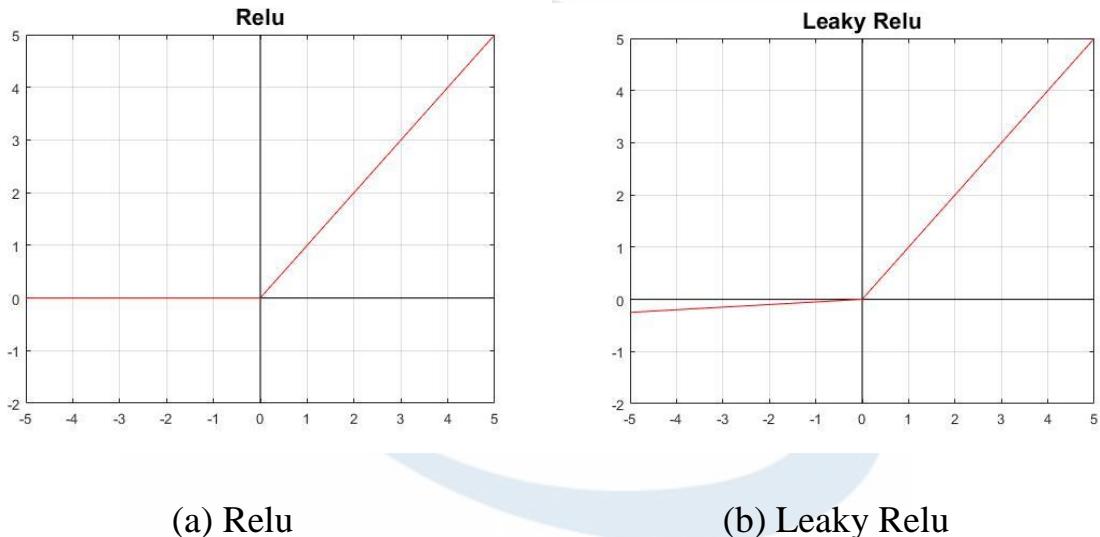


圖 2-7、激勵函數圖形

### 2.2.7 批次正規化(Batch Normalization, BN)

批次正規化是一種通過重新居中和重新縮放對輸入圖層進行正規化，以使類神經網絡更快更穩定的方法。批次指的是訓練數據中的一部分數據，也就是批數據。批次正規化則是指將這些批數據進行正規化處理得到常態分佈，也就是平均值為 0、標準差為 1。透過批次正規化能夠確保輸入值經過激勵函數時不會掉進邊界區，能夠有效的解決梯度消失的問題。

## 2.2.8 反向傳遞演算法(Backpropagation)

反向傳遞演算法在深度學習中是相當重要的演算法，用於更新現有權重值以達到損失函數最小化的目的，一開始網路會進行前向傳遞，經過權重計算與激勵函數後得到預測值，之後計算預測值與實際值之間的損失，透過反向傳遞來修正權重值，進一步的降低損失，使網路往好的方向去發展。

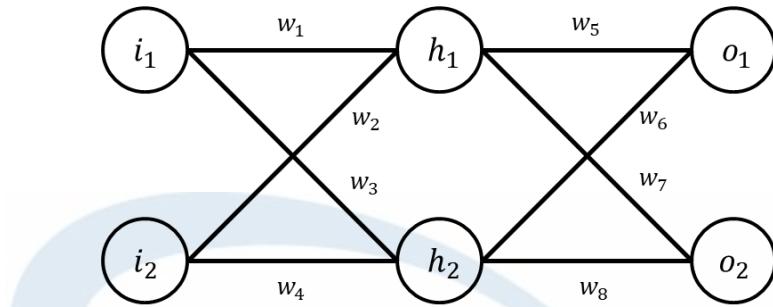


圖 2-8、初始網路架構

圖 2-8 為一個初始網路的架構，包含了輸入層、隱藏層與輸出層，輸入值  $input_i$  經過權重加成後得到會  $net_h$ ，經過激勵函數  $f$  後得到隱藏層的輸出  $out_h$ ：

$$out_{h1} = f(net_{h1}) = f(input_{i1} * w_1 + input_{i2} * w_2) \quad (2-4)$$

$$out_{h2} = f(net_{h2}) = f(input_{i1} * w_3 + input_{i2} * w_4) \quad (2-5)$$

以同樣的方式計算出網路輸出層的預測值  $Predict_o$ ：

$$Predict_{o1} = f(net_{o1}) = f(out_{h1} * w_5 + out_{h2} * w_6) \quad (2-6)$$

$$Predict_{o2} = f(net_{o2}) = f(out_{h1} * w_7 + out_{h2} * w_8) \quad (2-7)$$

計算出預測值與實際值  $target_o$  之間的總誤差  $E_{total}$ ，使用誤差平方和：

$$E_{total} = E_{o1} + E_{o2} = \frac{1}{2}(target_{o1} - Predict_{o1})^2 + \frac{1}{2}(target_{o2} - Predict_{o2})^2 \quad (2-8)$$

以權重  $w_5$  為例，更新後的權重值  $new_{w_5}$  如式(2-9)定義，其中  $\eta$  為學習率，並且透過鏈鎖法則可以計算出  $\frac{\partial E_{total}}{\partial w_5}$ ，如式(2-10)所示：

$$new_{w_5} = w_5 - \eta \frac{\partial E_{total}}{\partial w_5} \quad (2-9)$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial Predict_{o1}} * \frac{\partial Predict_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5} \quad (2-10)$$

分別求解三個偏微分，假設此時的激勵函數為 Sigmoid 函數：

$$\frac{\partial E_{total}}{\partial Predict_{o1}} = -(target_{o1} - Predict_{o1}) \quad (2-11)$$

$$\frac{\partial Predict_{o1}}{\partial net_{o1}} = Predict_{o1}(1 - Predict_{o1}) \quad (2-12)$$

$$\frac{\partial net_{o1}}{\partial w_5} = out_{h1} \quad (2-13)$$

## 2.2.9 卷積神經網路測試

綜合上述對卷積神經網路的介紹，本論文對於標誌分類的部分所設計的卷積神經網路如圖 2-9 所示，CBL 為卷積層，使用了 Leaky relu 作為激勵函數並且採用批次正規化 BN，該網路是由數個卷積層與 Dropout 層組成，並且使用最大池化層來對特徵圖進行下採樣，最後使用 Softmax 層轉換成相對機率對目標進行分類。

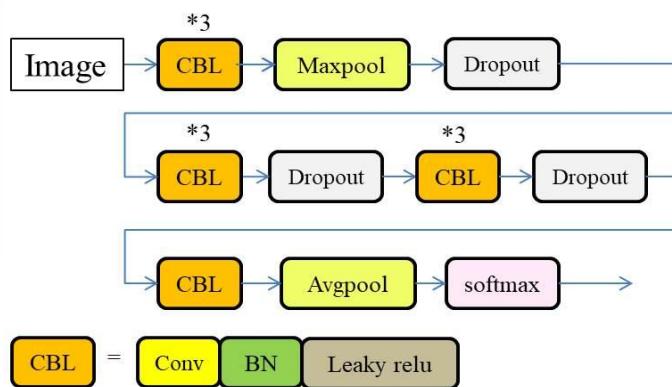


圖 2-9、卷積神經網路

為了驗證設計的網路是否可行，從德國交通標誌識別數據集 (The German Traffic Sign Recognition Benchmark, GTSRB) 中對於限速標誌進行測試，該資料集中有七種限速牌，刪除少於 1000 筆資料的標誌後，對於剩下的六種限速牌，每種類 1000 個樣本，共有 6000 張圖像作為訓練集，且每個種類 300 個驗證集，共 1800 張圖像作為驗證集，測試集的數量同為驗證

集，結果如表 2-1 所示，所有種類的 Top-1 準確率皆有 90% 以上，驗證集平均為 96.4%，並且檢測一張圖像的時間約為 2 毫秒。

表 2-1、驗證以及測試集的 Top-1 準確率

限速	Valid Top-1	Test Top-1
30	97.3%	98.7%
50	92%	91.3%
60	96.3%	98.7%
70	97%	97.7%
80	96%	96%
100	99.7%	100%
平均	96.4%	97%

### 2.3 目標檢測器

近年來，深度學習模型逐漸取代傳統電腦視覺(Computer Vision)方法，成為目標物檢測(Object Detection)領域的主流演算法。有別於傳統「目標檢測」方法步驟微區域選擇、提取特徵、分類回歸，新式目標物檢測法則改採「深度學習」技術，並分成三個步驟，即分類(Classification)、檢測(Detection)與分割(Segmentation)，如圖 2-10 所示。分類關心的是整體，得到的結果是對於整張圖像的內容描述，而檢測則較注重於目標物體，要求獲得目標的同時也能夠得到類別與位置的資訊。相較於分類，檢測給出的是對圖像前景和背景的理解，我們需要從背景中分離出感興趣的目標，並確定這一目標的描述（類別和位置），因而檢測模型的輸出是一個清單，列表的每一項使用一個資料組給出檢出目標的類別和位置，一般常用矩形檢測框的座標表示之；分割包括語義分割(Semantic Segmentation)和實例分割(Instance Segmentation)，前者是對前背景分離的拓展，將影像中的畫素分類為有意義的目標類別，如天空、道路或公共汽車；後者是檢測任務的拓展，要求描述出目標物的輪廓（相比檢測框更為精細）。分割是對圖像的圖元素描述，

它賦予每個圖元素類別意義，適用於理解要求較高的場景，如自駕駛中對道路和非道路的分割。

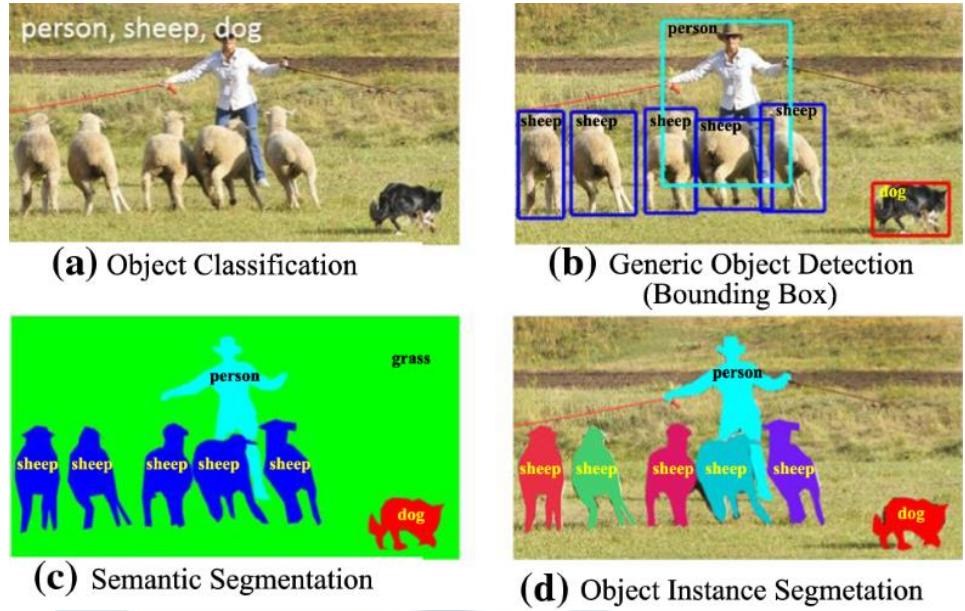


圖 2-10、目標物檢測圖層三步驟[4]

圖 2-11 為通用型目標物檢測演進。「目標物檢測」演算法大家族主要劃分為兩大派系，一個是 R-CNN (Regions with CNN features) 系列演算法為代表的兩段式目標檢測器，另一個則是以 YOLO、SSD 演算法為代表的單段式目標檢測器。

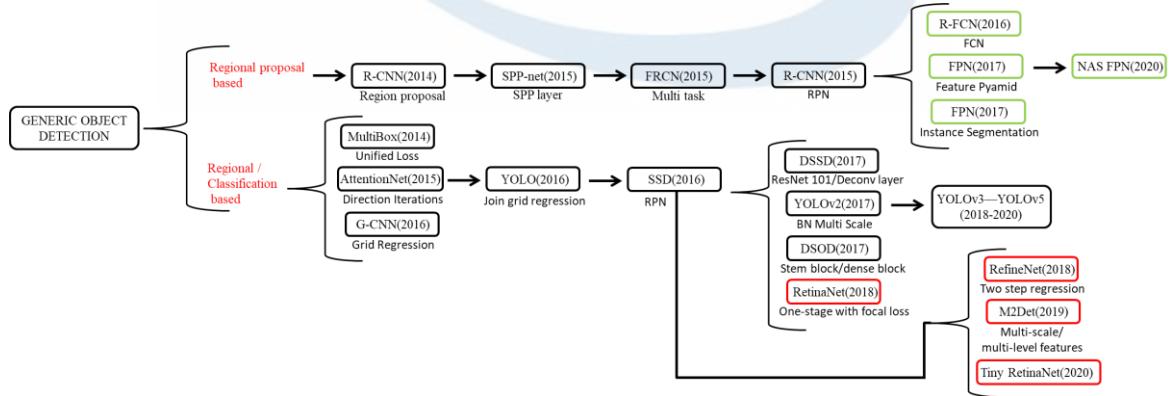


圖 2-11、通用型目標物檢測演進[5]

兩段式處理分為生成「候選區域(Region Proposal) CNN 萃取特徵」，再接著進行「分類器分類並修正位置」；單段式處理直接對預測的目標物體進

行回歸分類。這兩種不同處理方式，如圖 2-12 所示。接下來，我們將針對前述這兩類分別探討其特性，並選擇本論文採用的處理方法。

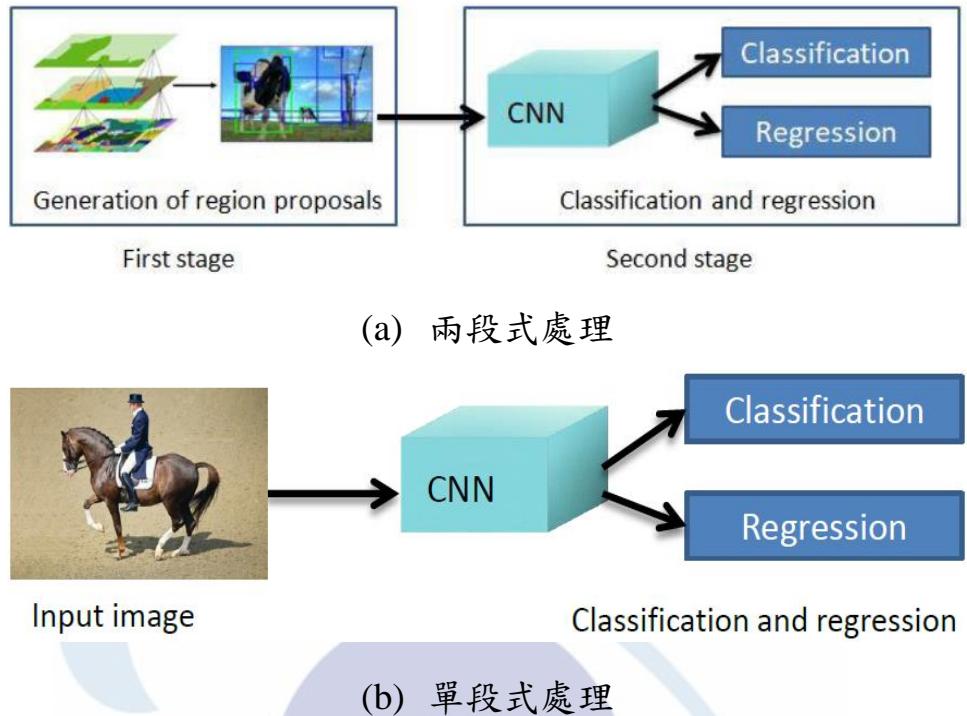


圖 2-12、對目標物體檢測不同的處理方式[17]

### 2.3.1 兩段式目標檢測器

兩段式目標檢測器演算法，最早在 2014 年提出第一個使用 CNN 版本的 R-CNN[16]，R-CNN 演算法是在輸入圖像中創建多個邊界框，並檢查這些邊框中是否含有目標物體。R-CNN 採用選擇性搜尋測試圖像以萃取出候選區域，而後將這些候選區域分別裁剪下來，縮放後送入 CNN 做計算並進行分類，最後每個區域會輸出一個標籤，用以決定是否有欲識別的目標物，其處理過程如圖 2-13 所示。

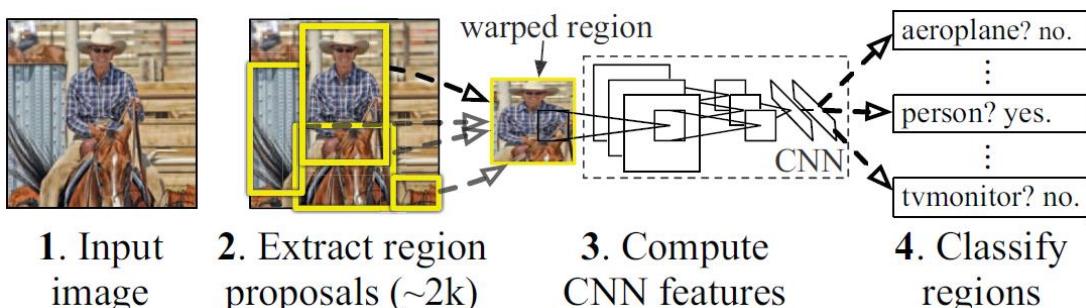


圖 2-13、R-CNN 演算法處理方式[16]

上述的 R-CNN 演算法存在運算太慢的缺點，因此原作者之一 R. Girshick 再提出一個改進的版本稱為 Fast R-CNN 演算法[18]，它基本上仍是 R-CNN 演算法，不過新增滑動窗捲積，且每張影像上只運行一次 CNN，故大幅降低運算量。此外，Fast R-CNN 演算法將原有結構改成並行運算，同時進行分類及對邊界框回歸，此也降低整體演算法的運算時間。然而 Fast R-CNN 演算法得到候選區域的步驟仍然非常緩慢，引發後續發展出 Faster R-CNN 演算法，結果比 Fast R-CNN 算法快得多。

前面的方法中存在兩個問題，一個是生成大量無效區域將造成運算上的浪費，但生成區域不足則會發生漏檢問題；二是生成候選區域的演算法是在 CPU 上執行，訓練則在 GPU 上面，跨結構交互必定會有損效率。因此 Faster R-CNN 演算法提出了一個候選區域網路(Region Proposal Network)的概念，它取代選擇性搜尋提高運算速度，它也利用神經網路自己學習去生成候選區域。這種候選區域網路同時解決了上述的兩個問題，神經網路可以學到更加高層、語義、抽象的特徵，生成的候選區域的可靠程度大大提高，且與原有網路一起預測，大幅減少參數量和預測時間。

提出 Faster R-CNN 演算法的共同作者提出另一個想法，在 Faster R-CNN 演算法之後加上一個圖像的遮罩(Mask)資訊，就成為知名的 Mask R-CNN 演算法。它不僅可以做「目標物檢測」還可以同時做「語義分割」，將兩個電腦視覺基本任務融入一個框架中。Mask R-CNN 演算法算是 Faster R-CNN 的橫向擴充，由原來的兩個任務（分類+回歸）變為了三個任務（分類+回歸+分割）。

圖 2-14 顯示前述 R-CNN 系列的 mAP 效能比較，IoU 參數是指預測框與目標框的面積交並比，由圖中清楚看到，Mask R-CNN 演算法的 mAP 最高，即辨識率平均值最高。

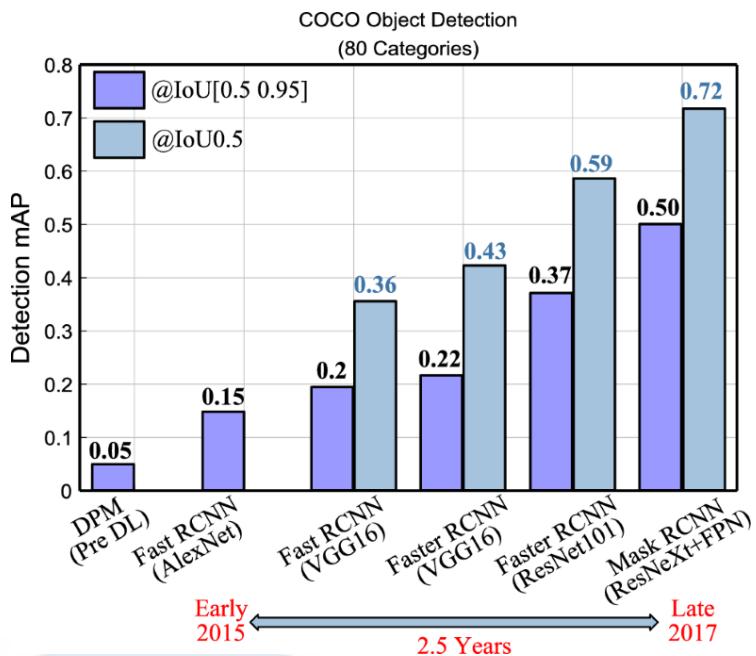


圖 2-14、R-CNN 系列的 mAP 效能比較[4]

### 2.3.1.1 Mask R-CNN

Mask R-CNN 演算法[6]是 ICCV-2017 的最佳論文，它能夠在一個網路中，同時對目標物進行檢測並且做到實例分割。實例分割和語義分割的差異，在於實例分割需要將屬於同一類的不同實例用不同的顏色標明，然而語義將屬於同一類的不同實例，全用一種顏色標示。例如圖 2-15 中最後得到的就是實例分割的結果，如果是語義分割，那麼所有人用一種顏色標識即可。該演算法在單 GPU 上的運行速度大約是 5 FPS，並且在微軟 COCO 資料集的三個挑戰賽：instance segmentation、bounding-box object detection、person keypoint detection 中的效能都要優於現有的演算法，包括在微軟 COCO 2016 比賽中的冠軍演算法。

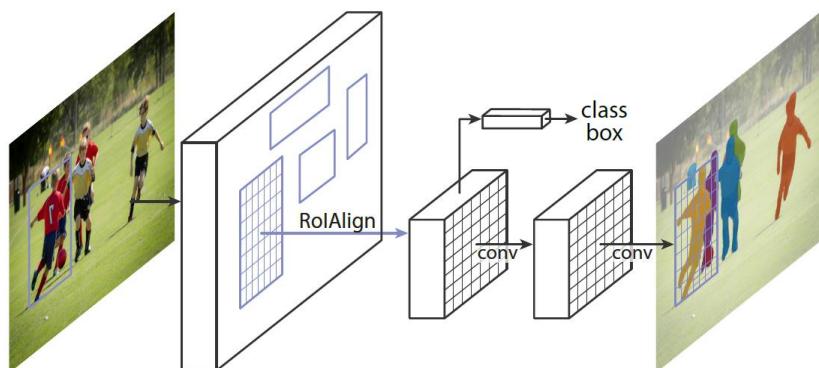


圖 2-15、Mask R-CNN 的實例分割[6]

如前述 Mask R-CNN 可以經由 Faster R-CNN 加以擴展而得到，如圖 2-15 所示。在 Faster R-CNN 中，對於每個關注區(Region of Interest, RoI)主要有兩個輸出，一個輸出是分類結果，也就是預測框的標籤；另一個輸出是回歸結果，也就是預測框的座標位移。而 Mask R-CNN 則是添加了第三個輸出，稱為目標物遮罩(Object Mask)，也就說對每個關注區都輸出一個遮罩，該支路是通過如圖 2-16 中兩個卷積層的全卷積網路(FCN) [7]做實現。並且將 Faster R-CNN 中所使用的 RoI Pooling 取代為 RoI Align，減少特徵圖中的像素偏差，有利於提高實例分割的準確度。圖 2-16 顯示較詳細的運作示意圖。

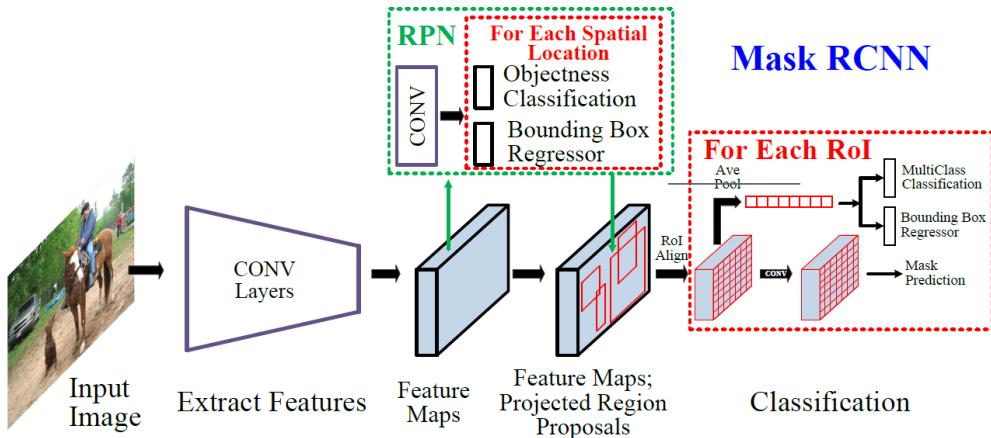


圖 2-16、MaskR-CNN 的實例分割[6]

### 2.3.1.2 RoI 池化(Pooling)

RoI 池化為 Faster R-CNN 原有的方法，由於輸入的圖像經網路後縮放，其檢測到的目標邊界框(Bounding Box)可能不會落在座標點上，在經過 RoI 池化時會將這些小數點座標量化成整數，導致部分特徵丟失。如圖 2-17 所示，紅色框為檢測目標邊界框，橘色與綠色的部分為量化後所得到的對應的特徵圖，而藍色部分則是對於邊界框來說所丟失的特徵，對於需要精確到像素點的目標分割是個嚴重的問題。

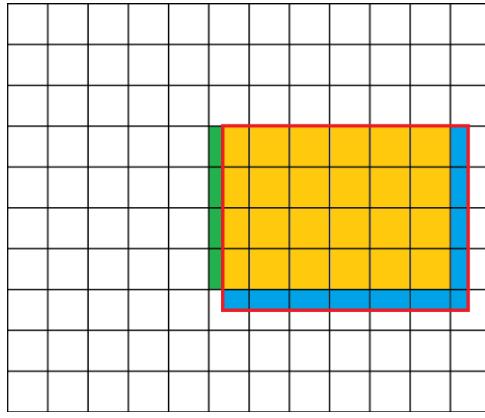


圖 2-17、RoI 池化

### 2.3.1.3 RoI 對齊(Align)

不同於 RoI 池化中的量化操作，在 RoI 對齊所使用的是雙線性插值演算法，充分的利用了原圖中虛擬點四周的四個真實存在的畫素值來共同決定目標圖中的一個畫素值，如圖 2-18，將黑色邊界框切割成  $2 \times 2$  大小，並且將每個小框細分成 4 個網格(Grid)，在每個網格中以鄰近的 4 個像素值執行雙線性插值，最後進行簡單的最大池化即可得到結果。

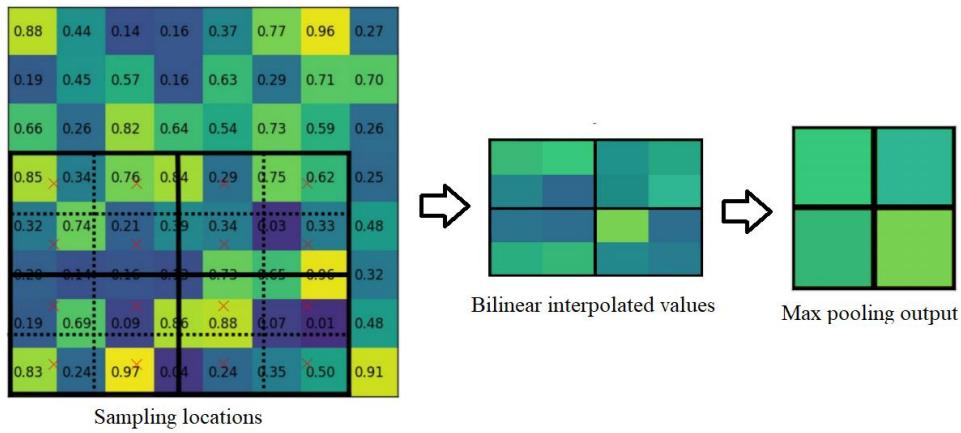


圖 2-18、RoI 對齊 [24]

### 2.3.1.4 全卷積網路

如前述所說，Mask R-CNN 為 Faster R-CNN 加上全卷積網路所形成的，它能夠對圖像進行像素級的分類，使圖像可以實現實例分割。全卷積網路與一般的 CNN 分類網路不同，它刪除了全連接層以及 softmax 層並且全部以卷積層來取代，使網路能夠接收任何尺寸的輸入圖像，之後對最後一個

卷積層的特徵圖進行上採樣，使圖像變成與輸入時的尺寸相同，從而可以對每個像素產生一個預測，產生出的特徵圖又稱作熱度圖(Heat Map)如圖 2-19 所示，保留了原始圖像中每個像素點的空間訊息，進而實現實例分割。

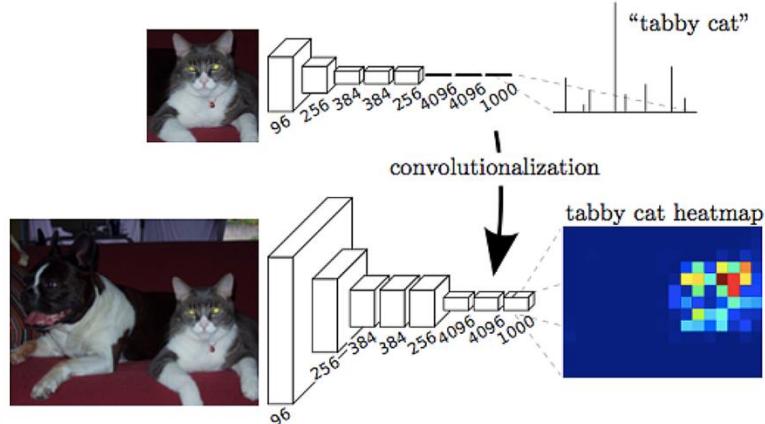


圖 2-19、以卷積層取代全連接層[7]

### 2.3.2 單段式目標檢測器

單段式目標檢測器最廣為人知的是在 2015 年由 R. Joseph 等人所提出 YOLOv1[23]，它採用一個單獨的 CNN 模型實現端到端的目標檢測，整個過程如圖 2-20 所示，首先將圖片調整成符合網路大小的輸入，之後送入 CNN 中進行檢測與分類，最後會經過非極大值抑制(Non-Maximum Suppression, NMS)刪除重複的預測框得到最終預測的結果。

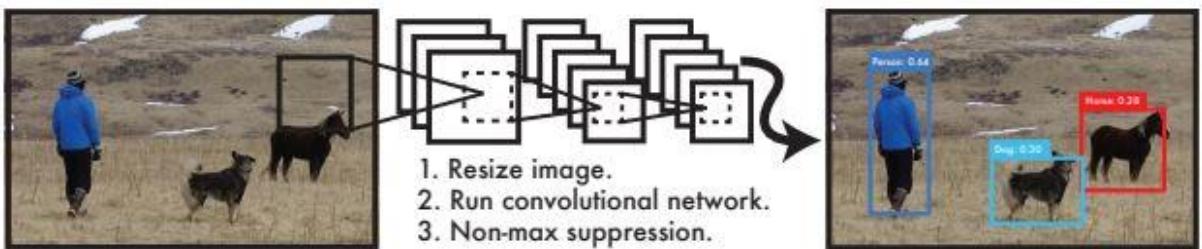


圖 2-20、YOLO 檢測系統[23]

預測框的具體生成方式如圖 2-21 所示，首先將輸入圖像劃分為  $S \times S$  個網格(Grid)，並且檢查待檢測目標的中心是否落入某個網格，則該網格負責檢測該目標。同時，這些網格會產生數個邊界框，其中每個邊界框對應五個輸出，分別是相對於網格單元邊界的中心座標  $x$  與  $y$ ，邊界框的寬高與整

張圖像寬高的比例  $w$  與  $h$  與其對應的置信度(Confidence)  $c$ ，之後經由非極大值抑制選出置信度最高的框作為預測結果。

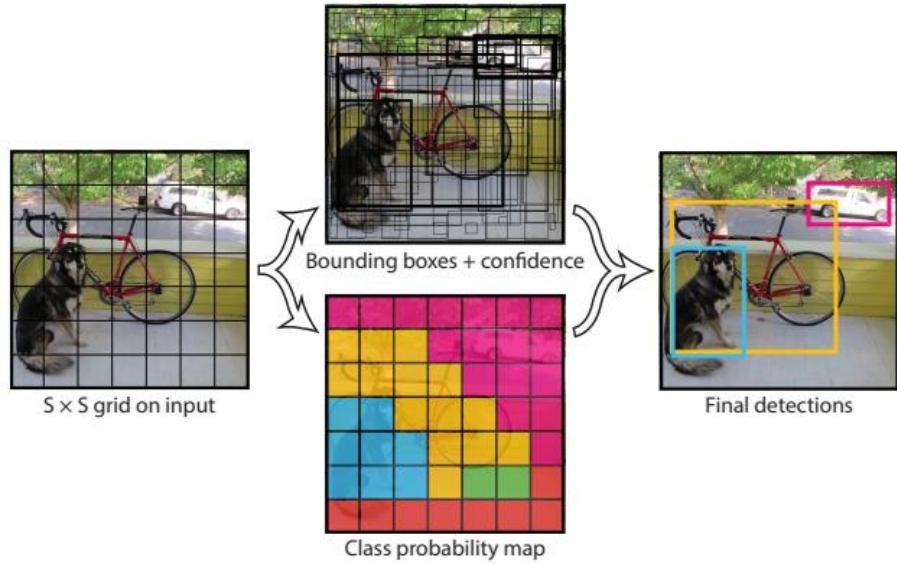


圖 2-21、YOLO 模型結構[23]

在 YOLOv1 之後也陸續出現了 YOLOv2、YOLOv3 以及 YOLOv4，在 YOLOv2 主要是對於 YOLOv1 的加強以及缺點進行改良，使用了新的骨架網路 Darknet-19，Darknet-19 移除了全連接層改採用卷積層的方式使網路能夠接受不同大小的輸入，使用批次正規化(Batch Normalization, BN)取代 Dropout 層使網路 mAP 提升 2.4%，採用多尺度輸入訓練策略讓網路學習各種不同大小的圖像，增加網路的穩健性，借鑒了錨框(Anchor Boxes)來預測邊界框，每個錨框擁有不同的分類概率值，改善兩個檢測物體靠太近時被非極大值抑制刪除的情形等。在 YOLOv3 中加入了特徵金字塔網路(Feature Pyramid Networks, FPN)進行多尺度預測，用以檢測不同大小的物體，以及引入殘差網路(ResNet)架構加深原有的骨架網路，稱為 Darknet-53。最後是近年提出的 YOLOv4，融合了最近許多的新技術，使 YOLO 網路表現得更加出色。

### 2.3.2.1 YOLOv4

YOLOv4 是由中研院廖弘源特聘研究員及王建堯博士研究員團隊與俄羅斯學者博科夫斯基(Alexey Bochkovskiy)所共同開發，他們在 YOLOv3 基礎上做了許多方面改善，並在保證速度下，尚能有效提高檢測準確度，同

時降低硬體資源。因此在 2020 年 9 月 28 日的 MSCOCO 目標物檢測競賽中一舉榮獲世界排名第一。YOLOv4 的開發，是從 2017 年科技部提出「業界出題、學界解題」AI 計畫，由企業提出問題需求，再由學界提供技術來解決，是一個非常成功的典範。它主要的貢獻有以下三點：

- 一、提出了一個高效強大的目標物檢測模型，使用 Nvidia RTX 1080Ti 或 2080Ti 的 GPU 即能進行訓練，並且又快速準確。
- 二、驗證目前幾個最先進的方法，了解對檢測器的影響。
- 三、修改了最先進的方法，包括 PAN, SAM, CBN 等，讓檢測器更有效率，並支持單一 GPU 即可完成訓練。

### 2.3.2.2 YOLOv4 架構

YOLOv4 模型的架構如圖 2-22 所示，是由以下三部分所組成：

#### 一、 Backbone: CSPDarknet53

CSPDarknet53 作為 YOLOv4 的骨幹網路(Backbone)，是由 YOLOv3 的骨幹網路 Darknet53 與跨階段局部網路(Cross Stage Partial Network, CSPNet)結合而成，其優點是能夠增加網路的學習能力，使網路能夠獲取更豐富的梯度融合訊息(Gradient Combinations)用以最大化梯度多元性。

#### 二、 Neck: SPP+PAN

為了能夠擴大感受野及融合不同尺度特徵圖的訊息，YOLOv4 在骨幹網路後加入空間金字塔池化層(Spatial Pyramid Pooling, SPP)與路徑聚合網路(Path Aggregation Network, PAN)。前者用於強化感受野作用，使最重要的特徵區分出來；後者則取代 YOLOv3 的 FPN，在不同尺度的特徵圖中進行參數聚合(Parameter Aggregation)。

#### 三、 Head: 與 YOLOv3 使用同樣的作法。

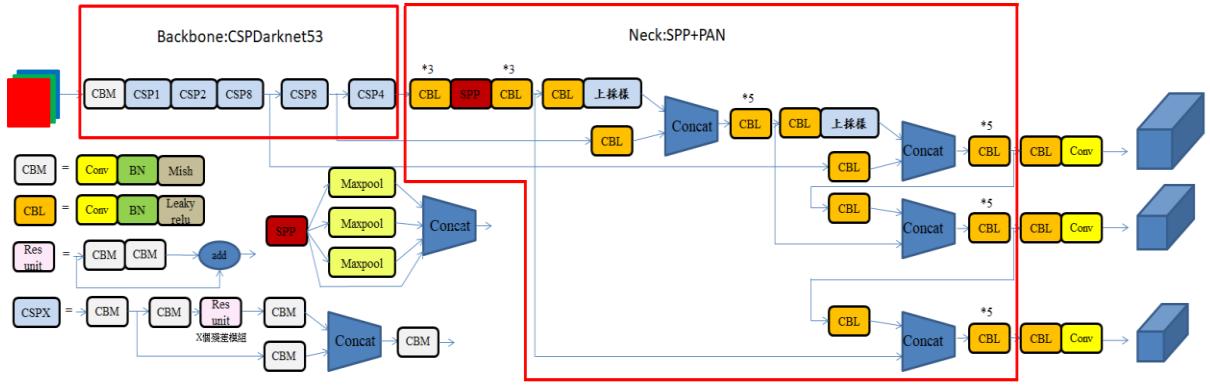


圖 2-22、YOLOv4 架構圖

### 2.3.2.3 跨階段局部網路[8]

為 YOLOv4 中所提出的一種新型的骨幹，圖 2-23 以 DenseNet 為例，圖(a)為原始架構，圖(b)為加入 CSPNet 後的 CSPDenseNet。CSPNet 為 YOLOv4 最具創新的部分，主要目的是使網路能夠獲取更豐富的梯度融合訊息，並且降低計算量。它先將前一層的輸出特徵圖劃分為兩部分，一部分不進行運算直接傳遞，另一部分經過密集塊(Dense Block)與過渡層(Transition)，最後再將兩個部分經由過渡層進行合併。而在 YOLOv4 中的 CSPDarknet53 則是使用殘差塊構成。

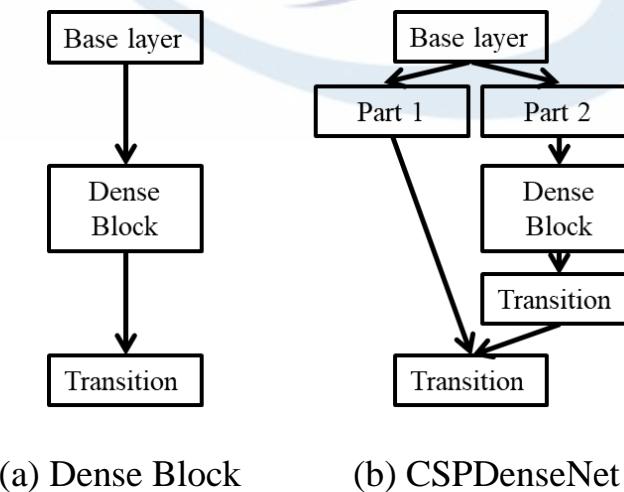


圖 2-23、CSPNet 骨幹

圖 2-24 為 CSPNet 反向傳遞學習過程。其重點在於局部過渡層以及資訊的分流，局部過渡層能夠在反向傳遞時透過截斷梯度流的方式來防止網

路學習重複的梯度訊息，藉此能夠最大化梯度多元性。另外，由於資料分流的關係，部份結果直接進行傳遞並且未進行分析計算，使得整體計算量減少，能夠加快網路整體的速度。

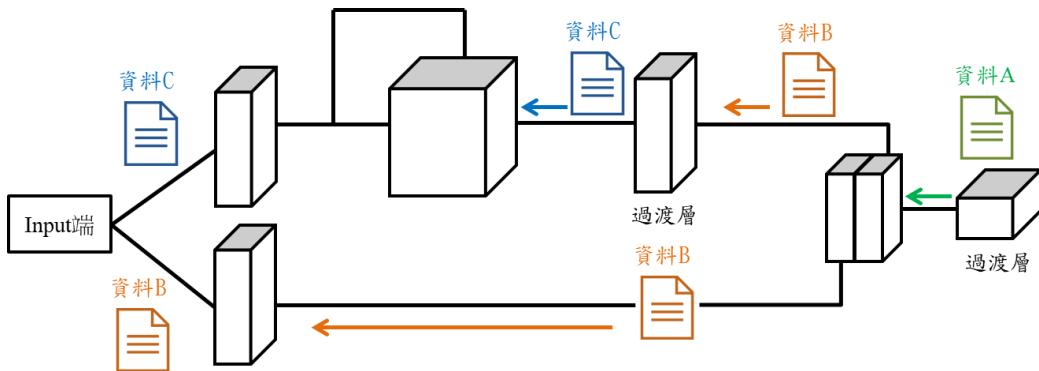


圖 2-24、CSPNet 簡化結構

#### 2.3.2.4 路徑聚合網路[14]

PANet 是一種基於實例分割框架，它是快速、簡單且非常有效的，通過自底部往上的路徑，使用低層精確的定位信號作為增強整個特徵層次結構，並對所有層的特徵進行池化，用以縮短了低層與頂層之間的資訊路徑距離，且使用增強路徑來豐富每一層的特徵，它能夠準確地儲存空間資訊，有助於正確定位像素點形成遮罩。在原本的 PANet 網絡中，兩個特徵圖的結合是用相加的，然而在 YOLOv4 中則改為合併(Route)操作，如圖 2-25 所示，從而提高預測的準確性。

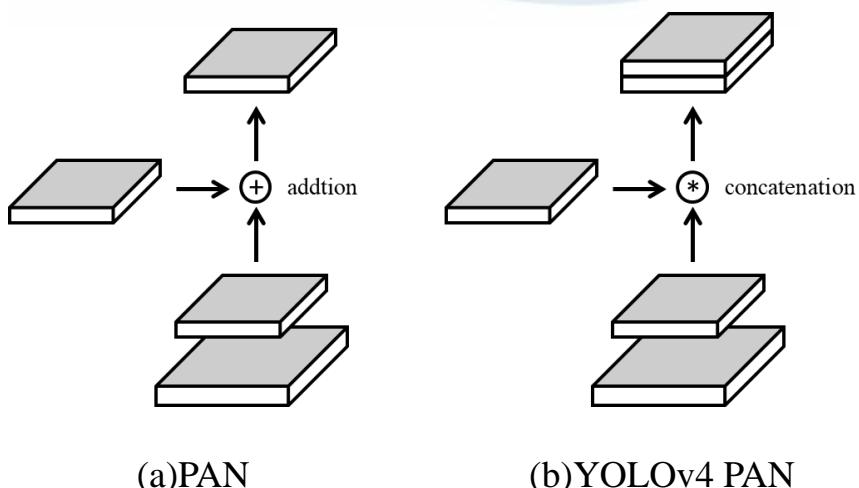


圖 2-25、用於 YOLOv4 的路徑聚合網路

### 2.3.2.5 空間注意力模塊(Spatial Attention Module Block, SAM-block)[15]

空間注意力模塊源自於卷積方塊注意力模塊(Convolutional Block Attention Module, CBAM) 中兩種注意力機制其中的一種。先對不同特徵圖上相同位置的像素值，進行最大池化與平均池化處理，接著將這兩個特徵圖進行合併，再利用一個  $7 \times 7$  的卷積進行運算後，經過 Sigmoid 函數得到權重系數  $\mathbf{M}_s$ ，最後將這個權重系數與原來的特徵  $\mathbf{F}'$  相乘，就可以得到縮放後的新特徵，如圖 2-26 所示。

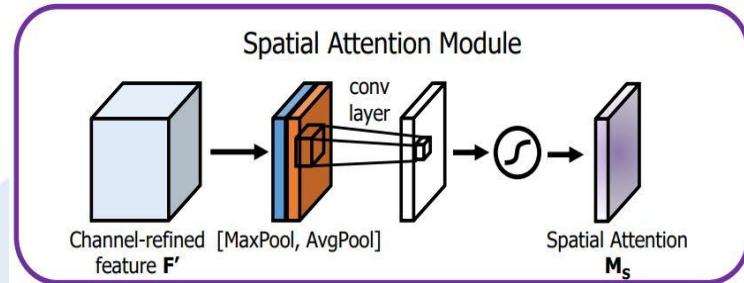


圖 2-26、空間注意力模塊

然而在 YOLOv4 中的方式有別於 SAM，它捨棄最大池化與平均池化處理，而是直接送進 CNN(卷積核為  $7 \times 7$ )操作，如圖 2-27 所示。它的好處就是不會有池化這樣將特徵圖大小縮小的動作，從而可能會丟失任何資訊，所以它屬於點(Pointwise)的注意方式，取代以往的空間注意力。

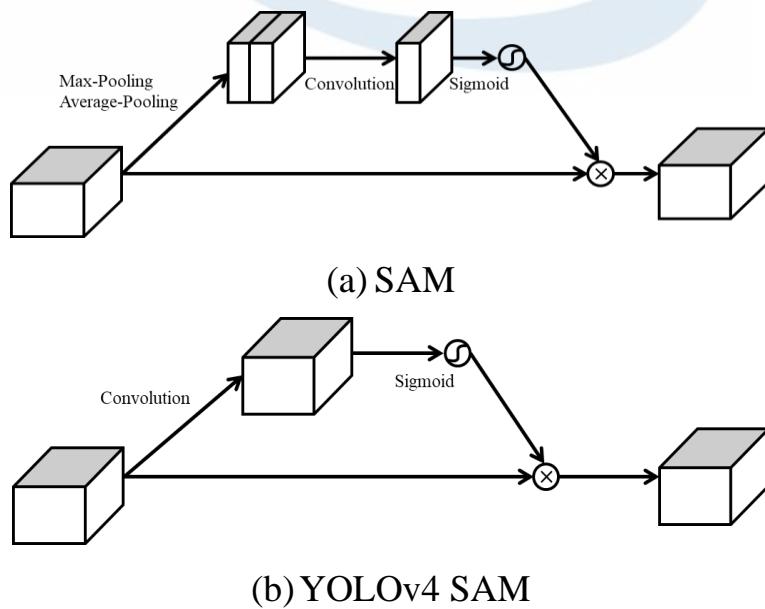


圖 2-27、用於 YOLOv4 的空間注意力模塊

### 2.3.2.6 數據增強技術

在對網路進行訓練時，經常會碰到訓練資料不足的情況，對此也衍生出了許多種的數據增強技術，透過翻轉、色彩調整與剪裁等方式產生新的訓練圖片，而在 YOLOv4 中使用了一種新的數據增強技術 Mosaic，它的想法是源自於 CutMix。

#### 1. CutMix 技術[9]

CutMix 的方法是用兩張圖像進行拼接，將原始圖像的一部分區域刪除但是不填入 0 像素，而是將訓練集中其他圖像的區域像素隨機填充到原始圖像上，得到一張全新圖像，且分類的結果按一定的比例分配，如圖 2-28 所示。

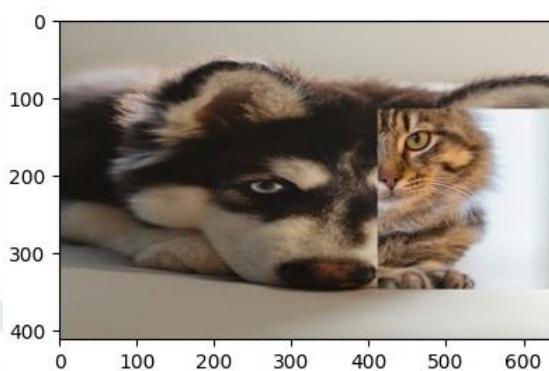


圖 2-28、CutMix 示意圖

#### 2. 馬賽克(Mosaic)

馬賽克為一種新的數據增強技術，它參考前述 CutMix 方法，由原來的 2 張訓練圖像混合拼接改成 4 個，如圖 2-29 所示，相當於一次傳入四張圖像進行學習，並且因隨機縮放增加了很多小目標，讓模型更穩健，此外，由於一次看入四張圖片相當於降低 mini-batch 對網路的影響，因此能夠在單 GPU 上進行訓練。



圖 2-29、馬賽克示意圖

### 2.3.2.7 DropBlock 正則化(Regularization) [10]

類似於 Dropout 技術，但不同於 Dropout 採刪除獨立的隨機神經元，它是從特徵圖中刪除連續區域，這將使得網路著重於學習某些特徵，以實現正確分類與避免產生過擬合現象。如圖 2-30 所示，圖(a)為原始圖像，圖(b)與圖(c)分別為 Dropout 與 DropBlock。

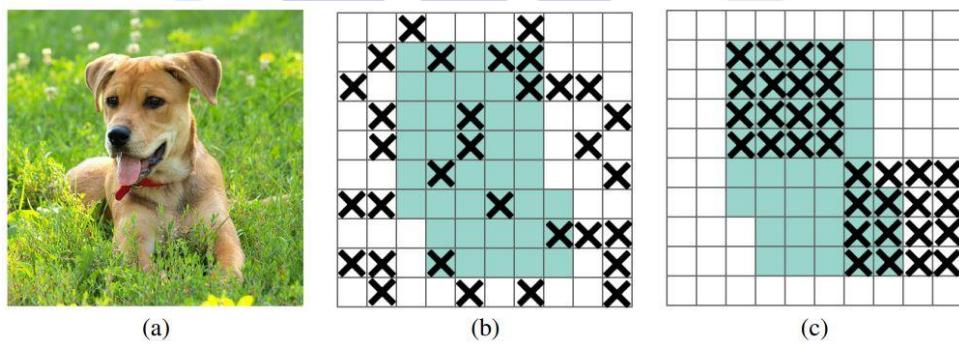


圖 2-30、DropBlock 正則化[10]

### 2.3.2.8 Mish 激勵函數

Mish 是光滑的非單調激勵函數，如圖 2-31 所示。公式為(2-14)，其中式(2-15)屬一個 softmax 激勵函數。Mish 與 ReLU 激勵函數相比，Mish 激勵函數的梯度更佳平滑，且在負值時允許含較小的負梯度，此能穩定網路梯度流，因此具更好的泛化能力。

$$f(x) = x * \tanh(\varsigma(x)) \quad (2-14)$$

$$\varsigma(x) = \ln(1 + e^x) \quad (2-15)$$

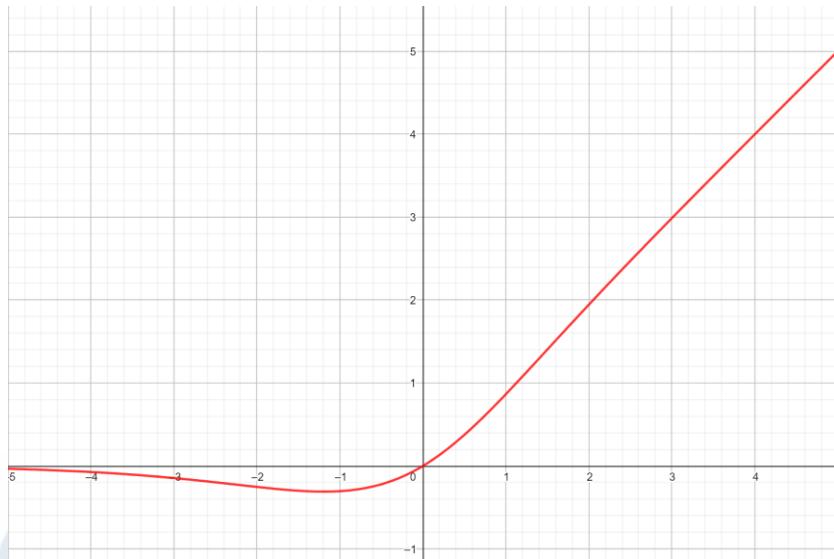


圖 2-31、Mish 激勵函數

另外，在骨幹網路中常用的激勵函數有 ReLU, Swish, Mish，如表 2-2 顯示了三種不同的激勵函數分別在不同網路中的表現，其中使用 Mish 函數可稍微提高精度。

表 2-2、各種激勵函數的表現[25]

Model	Mish	Swish	ReLU
ResNet v2-20	<b>92.02%</b>	91.61%	91.71%
WRN 10-2	<b>86.83%</b>	86.56%	84.56%
SimpleNet	<b>91.70%</b>	91.44%	91.16%
Xception Net	<b>88.73%</b>	88.56%	88.38%
Capsule Net	<b>83.15%</b>	82.48%	82.19%
Inception ResNet v2	<b>85.21%</b>	84.96%	82.22%

### 2.3.2.9 損失函數的重疊度問題[13]

#### 1. IoU (Intersection over Union)損失:

IoU 又稱為交並比，是目標檢測判斷兩個框的接近程度時經常使用的指標，其計算方式為兩個矩形框的交集面積除上聯集面積，如圖 2-32 所示，在計算 IoU 損失時，IoU 的定義如式(2-16)所示，其中  $B$  與  $B^{gt}$  分別代表預測框以及目標框。IoU 有個缺點，當預測框和目標框完全不相交時，IoU 就會為 0，因此 IoU 無法反應兩個框間的接近程度，且損失函數為不可導，無法計算其梯度，因此難以最佳化兩個框不相交的情況。

$$IoU = \frac{|B \cap B^{gt}|}{|B \cup B^{gt}|} \quad (2-16)$$

$$L_{IoU} = 1 - IoU \quad (2-17)$$

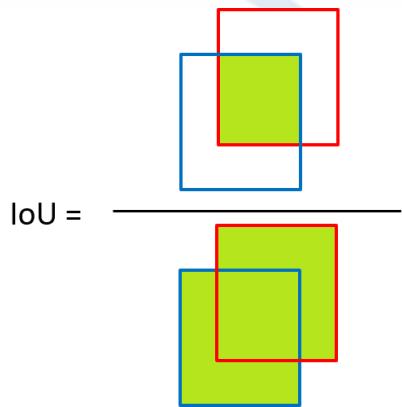


圖 2-32、IoU 示意圖

#### 2. GIoU(Generalized Intersection over Union)損失:

為了解決 IoU 損失中的兩框完全不相交的問題，GIoU 損失在後面加了一個懲罰項，如式(2-18)所示，其中  $C$  為包圍預測框  $B$  與目標框  $B^{gt}$  的最小矩形框，與 IoU 只關注重疊區域不同，GIoU 除了前述的重疊區域，還關注其他的非重疊區域，也就更能反映兩者框間的重疊度，如此即可比 IOU 更能反應兩個框的接近程度。

$$L_{GIoU} = 1 - IoU + \frac{|C - B \cup B^{gt}|}{|C|} \quad (2-18)$$

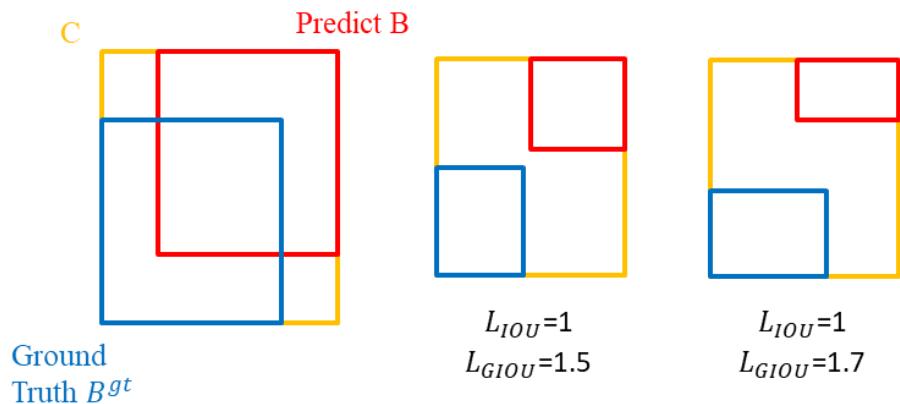


圖 2-33、GIoU loss 的計算方式

### 3. DIoU(Distance Intersection over Union)損失:

若預測框剛好落在目標框的內部，此時 GIoU 損失與 IoU 損失的結果是一樣的，無法區分兩個框之間的相對位置，如圖 2-34 所示。而在 DIoU 損失中除了考慮到重疊面積外，也考慮到兩個框的中心點距離，更改了懲罰項用於最小化的中心點距離，由式(2-19)定義，其中  $c$  為最小包圍框的對角線長， $\rho$  為預測框  $B$  與目標框  $B^{gt}$  中心點之間的距離。此外，由於 DIoU 損失可以直接最小化兩個框中心點的距離，比起 GIoU 損失是盡量減少最小包圍框的面積，能夠有更快的收斂速度。

$$L_{DIOU} = 1 - IoU + \frac{\rho^2(B, B^{gt})}{c^2} \quad (2-19)$$

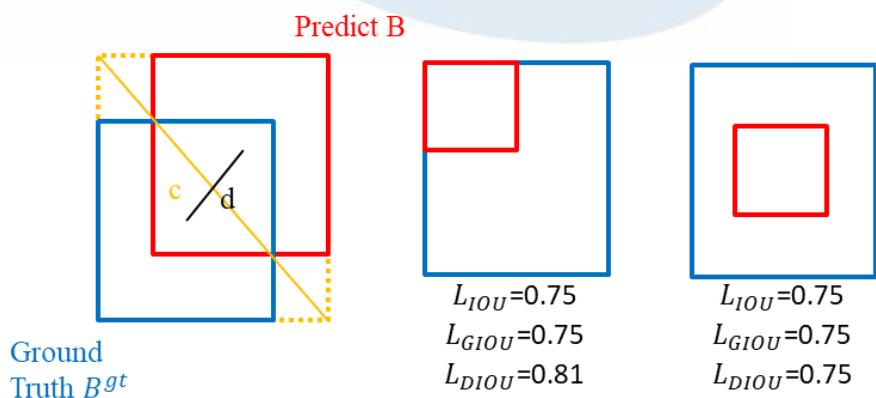


圖 2-34、DIoU loss 的計算方式

#### 4. CIoU(Complete Intersection over Union)損失:

CIoU 損失被 YOLOv4 所採用，在 DIoU 損失的基礎上另外考慮了重疊面積、中心點距離與長寬比這三個重要的幾何因素，因此在 DIoU 的基礎上多增加了一個懲罰項 $av$ ，成為 CIoU 損失，如式(2-20)所示。它將長寬比的因素考慮進去， $v$ 是衡量長寬比的相似性， $a$ 是權重函數。

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(B, B^{gt})}{c^2} + av \quad (2-20)$$

$$\alpha = \frac{v}{(1-IoU)+v} \quad (2-21)$$

$$v = \frac{4}{\pi^2} (\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h})^2 \quad (2-22)$$

CIoU 損失的梯度計算類似於 DIoU 損失，但有考慮 $v$ 的梯度。在長寬在 $[0,1]$ 間， $w^2 + h^2$ 的值通常會很小，這會導致梯度爆炸，因此實現 $1/(w^2 + h^2)$ 時將被設為 1。

$$\frac{\partial v}{\partial w} = \frac{8}{\pi^2} (\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h})^2 \times \frac{h}{w^2+h^2} \quad (2-23)$$

$$\frac{\partial v}{\partial h} = -\frac{8}{\pi^2} (\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h})^2 \times \frac{w}{w^2+h^2} \quad (2-24)$$

##### 2.3.2.10 DIoU-NMS [12]

在一般的非極大值抑制中，預測框中置信度最高的框會和其他預測框依次算出對應的 IoU 值，並將超過門檻值的預測框過濾掉。但在實際情況下，當兩個不同的待檢測物體靠得太近時，由於兩者間的 IoU 值比較大，在經過 NMS 算法後，可能會將其中一個刪除，導致遺漏檢測的狀況發生。DIoU-NMS 不僅僅考慮 IoU 值，在刪除重複的預測框時，還考慮兩個框的中心點之間的距離。對於有最大置信度的預測框 $M$ 以及重疊框 $B_i$ ，如果兩個框之間 IoU 比較大，但是他們之間的距離比較遠，會認為是不同物體的檢測框，因而不會被過濾掉，讓模型能更加穩健地處理不同卻重疊的目標。置信度由式(2-25)定義，其中 $\varepsilon$ 為非極大值抑制門檻值。

$$s_i = \begin{cases} s_i, & IoU - R_{DIoU}(M, B_i) < \varepsilon, \\ 0, & IoU - R_{DIoU}(M, B_i) \geq \varepsilon, \end{cases} \quad (2-25)$$

### 2.3.2.11 交叉微批次正規化(Cross mini-Batch Normalization, CmBN)

傳統的批次正規化是對每一個 Mini-Batch 進行正規化，透過統一分散的數據，能夠有效地趨緩梯度消失的問題，但當批次大小(Batch Size)太小時，會導致抽樣不均勻造成正規化出問題，使網路性能降低。為了解決前述問題，交叉批次正規化 CBN [11]採用對當前及前幾次 Mini-Batch 的結果進行正規化，透過該方法能夠變相的擴大批次大小使網路性能得以改進。YOLOv4 新創的交叉微批次正規化 CmBN 則是基於 CBN 再做修正，它在 Mini-Batch 間不做更新計算，而是在一個批次做完後才去更新網路參數。前述三種正規化如圖 2-35 所示。

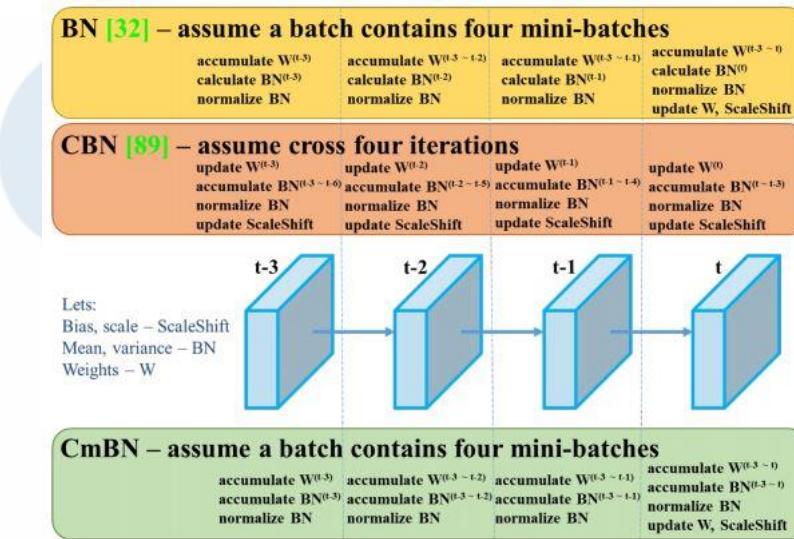


圖 2-35、BN、CBN 與 CmBN[12]

### 2.3.2.12 自對抗訓練(Self-Adversarial Training, SAT)

自對抗訓練是 YOLOv4 開發者提出的一種新的數據增強方法，能夠加強網路性能且降低過擬合，也就是別太專注在某些特徵上。分為兩個階段進行，在第一階段類神經網絡會先進行前向傳遞，接著進行反向傳遞時改變圖像，以這種方式使網絡對其自身執行對抗攻擊，從而改變原始圖像以產生對圖像上沒有待測物體的假象；第二階段以正常方式來檢測此被攻擊過後的圖像上，執行目標物檢測任務。

### 2.3.2.13 消除格點敏感度(Eliminate Grid Sensitivity)

當待測物的移動路線是連續時，待測物必然會出現在格點邊緣，即當待測物的中心座標落在格點(grid)邊緣時，就很難被預測為目標物中心，Sigmoid 在格點邊緣時導數接近 0，這時容易發生梯度消失，導致很難學習到準確的結果。因此在 Sigmoid 函數前乘上一個大於 1 的值，並且考慮到不同的格點尺寸對於邊界效應的敏感度，採用  $(1 + a) \times \text{Sigmoid} - (0.5 \times a)$ ，其中當格點解析度越高  $a$  值就越高。

### 2.3.2.14 多個錨點用於一個目標框

YOLOv4 採用多個錨點(Anchor)去負責一個目標框。對於目標框來說，只要  $\text{IoU}(\text{GT}, \text{anchor}_i) > \text{IoU}$  門檻值，就讓  $\text{anchor}_i$  去負責目標框。

### 2.3.2.15 餘弦退火排程器(Cosine Annealing Scheduler)

學習率的數值要越來越小，模型較能收斂，但數值過小也可能導致陷入區域最小值問題。YOLOv4 利用餘弦函數來調整學習率，一開始學習率會先慢慢下降，然後中途再加速下降，最後再次地緩慢下降，以此找到全域的最小值。

### 2.3.2.16 各種演算法的比較實驗結果

以 MS COCO 資料集作為基礎且在不同的 GPU 上與其他最先進的物體檢測器獲得的結果進行比較，如圖 2-36 所示。YOLOv4 無論是在檢測速度或者是準確度上，都比其他的檢測器還要來的快且精準。

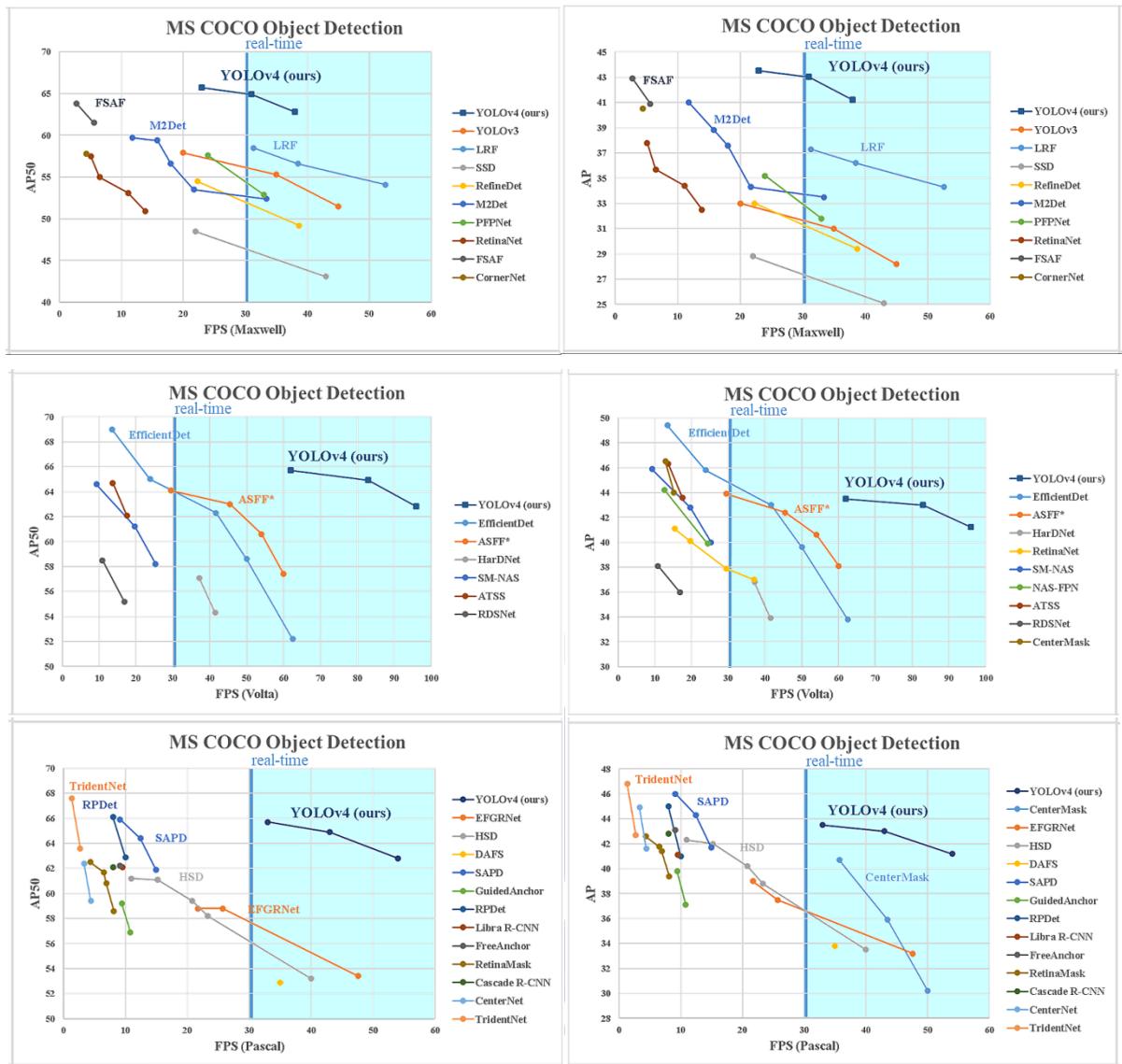


圖 2-36、各種演算法的比較結果[12]

### 2.3.3 Mask R-CNN 與 YOLOv4 檢測結果

Mask R-CNN 與 YOLOv4 的檢測結果分別如圖 2-37 與圖 2-38 所示，兩種方法皆可以成功的檢測出目標位置以及類別，使用 Mask R-CNN 時多了一項實例分割，對於辨識限速標誌來說目標的位置並不是那麼主要，重要的是能夠正確的檢測出類別即可。Mask R-CNN 檢測一張圖像所需要的時間約為 5.6 秒，YOLOv4 只需要約 0.024 秒，相較於 Mask R-CNN，YOLOv4 更適合運用在實時檢測的場合中。

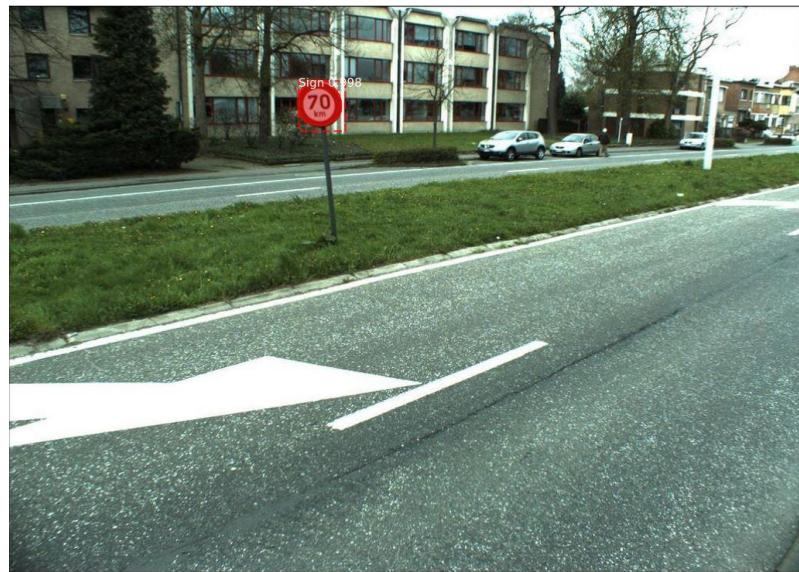


圖 2-37、Mask R-CNN 檢測結果



圖 2-38、YOLOv4 檢測結果

## 第3章 標誌檢測實現

### 3.1 簡介

本篇論文將針對交通標誌當中的限速標誌進行辨識，我們將從網路上所收集下來包含限速標誌的圖片進行處理後，分別對 YOLOv4 以及卷積神經網路進行訓練，得到訓練後的權重用以對限速標誌進行檢測與分類。檢測的流程如圖 3-1 所示，輸入一張包含限速標誌的圖像後，首先會經過 YOLOv4 初步檢測出目標位於圖像的位置，之後將該位置的目標預測框內的圖像經過調整大小後放入卷積神經網路中進行近一步的分類，最後輸出檢測的結果。

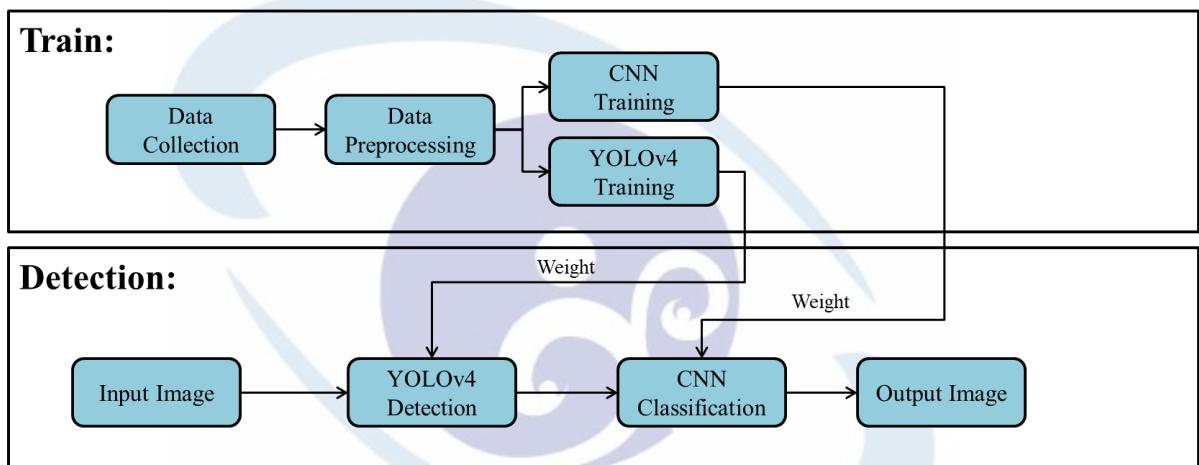


圖 3-1、影像辨識流程圖

### 3.2 資料收集

為了要訓練類神經網路以及 YOLOv4，首先必須要準備訓練用的資料。由於訓練時需要大量的資料，為了方便起見，本論文從網路上各種不同的交通標誌資料集以及 Google 地圖中篩選出含有限速標誌的圖像，並且為了增加訓練集的多樣性，使用了前述所提到的數據增強技術。



圖 3-2、Google 地圖截圖影像



圖 3-3、交通標誌資料集影像



圖 3-4、Mosaic 影像

使用開源軟體 LabelImg 對圖像進行標註，如圖 3-5 所示，並將製作好的標註資料與來源圖像放在一起，製作訓練以及驗證的資料集。標註後的資料如表 3-1 所示。

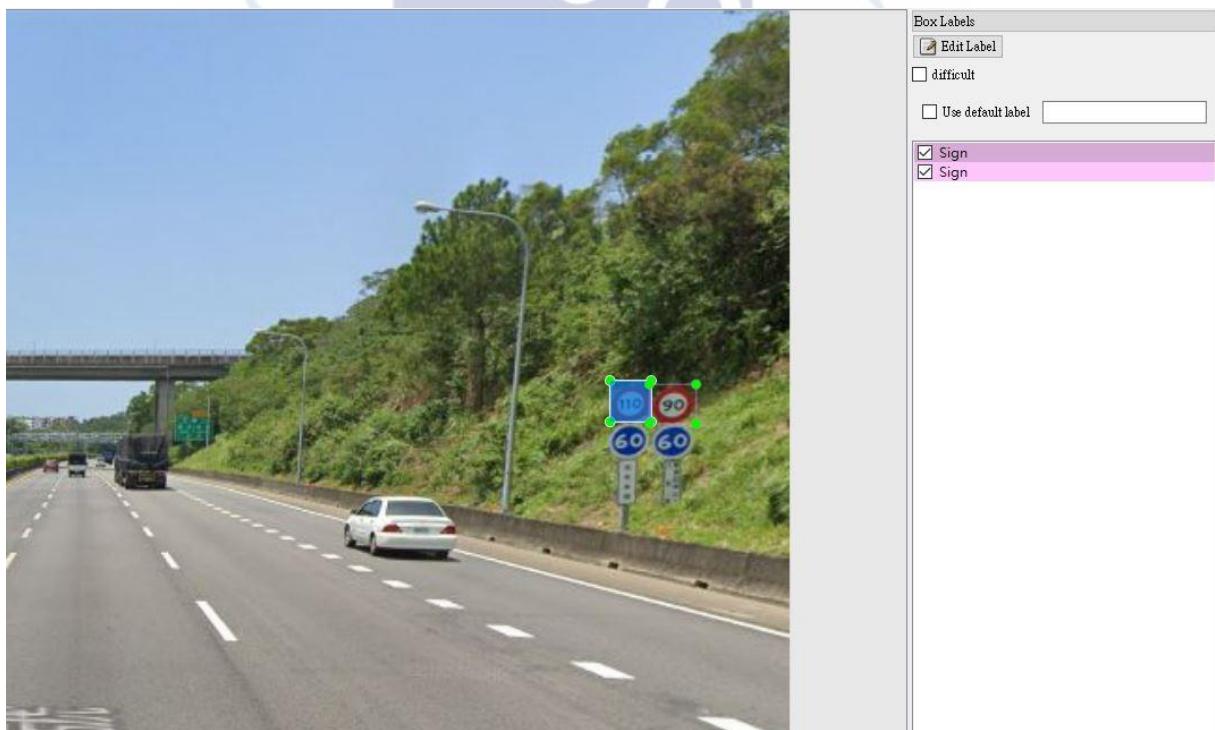


圖 3-5、圖像標註

表 3-1、YOLO 標註格式

類別	預測框中心 x 座標 佔原圖比例	預測框中心 y 座標 佔原圖比例	預測框寬度 w 佔原圖比例	預測框高度 h 佔原圖比例
0	0.871017	0.355932	0.033384	0.037288
0	0.905918	0.356780	0.036419	0.035593

將收集到的資料分成訓練集以及驗證集兩個，在訓練集中含有限速 30 至 110 等九種標誌各 1000 張，而驗證集則為每種標誌各 300 張，並且由於訓練卷積神經網路所需要的資料為標誌整體，所以必須將原始圖像上包含標誌的部分進行剪裁。將 YOLO 標註格式轉換成座標點形式，公式如式(3-1)至式(3-4)所示，得到預測框的左上角點座標( $box_{xmin}, box_{ymin}$ )以及右下角點座標( $box_{xmax}, box_{ymax}$ )。

$$box_{xmin} = w_{img} * \left( x - \left( \frac{w}{2.0} \right) \right) \quad (3-1)$$

$$box_{ymin} = h_{img} * \left( y - \left( \frac{h}{2.0} \right) \right) \quad (3-2)$$

$$box_{xmax} = w_{img} * \left( x + \left( \frac{w}{2.0} \right) \right) \quad (3-3)$$

$$box_{ymax} = h_{img} * \left( y + \left( \frac{h}{2.0} \right) \right) \quad (3-4)$$



圖 3-6、標誌裁切後影像

最後為了要驗證否能夠辨識台灣的限速標誌，另外收集了與訓練及驗證不同的另外每種類各 200 張圖像作為測試集，測試集當中的所有標誌皆為台灣的限速標誌，並以同樣的方法製作卷積神經網路測試集以及 YOLOv4 測試集。

### 3.3 網路訓練

#### 3.3.1 YOLOv4 訓練

將 9000 筆資料放進網路當中進行訓練，訓練過程中使用小批次訓練。網路輸入大小為 640×640，在訓練的過程中 YOLOv4 在每 10 次疊代中會隨機改變輸入的大小，讓網路能夠適應對不同尺寸的目標進行檢測。圖 3-7 為 YOLOv4 訓練的損失以及驗證集 mAP，紅色折線表示驗證集的 mAP，而藍色點為每次疊代的損失值，隨著訓練次數的增加，驗證集 mAP 會越來越高且損失值越來越小。

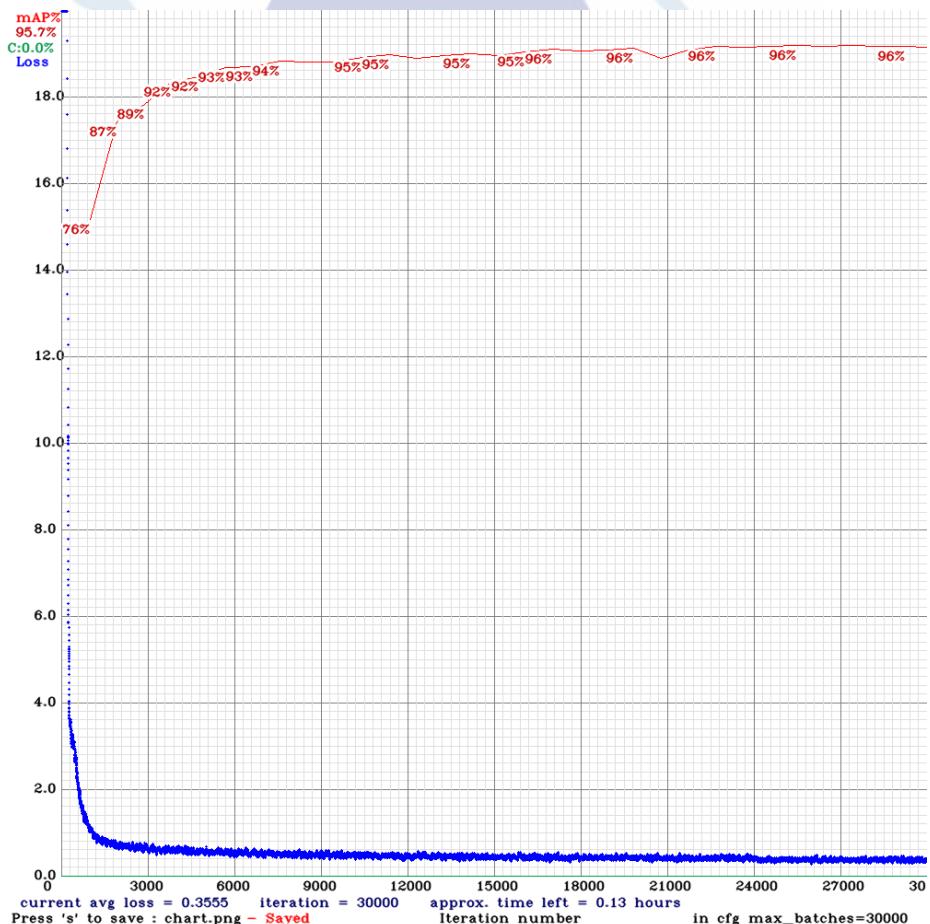


圖 3-7、YOLOv4 訓練圖

圖 3-7 中找出驗證集 mAP 趨於平緩的地方如第 15000 次疊代附近，選取該權重進行測試，結果如圖 3-8 到圖 3-10 所示，從三張圖中可以看到該網路對於遠、中與近的目標能夠正確的檢測到，並且不會被最低限速的標誌影響到。



圖 3-8、YOLOv4 測試(遠)



圖 3-9、YOLOv4 測試(中)



圖 3-10、YOLOv4 測試(近)

### 3.3.2 卷積神經網路訓練

將 9000 筆剪裁過後的圖像放入網路當中進行訓練，輸入網路前會將所有圖像進行調整，以符合網路的輸入大小，訓練方式使用小批次訓練，在每一次疊代(Iteration)中訓練一小批資料(Batch)，並且根據這一批資料來更新現有參數，如此可以使得梯度下降較為穩定，也能夠加快整個訓練的過程。訓練的損失曲線如圖 3-11 所示，隨著訓練次數的增加，其損失值會越來越小。圖 3-12 為該網路的預測結果，類別後的數字為置信度，越接近 1 表示其結果置信度越高。

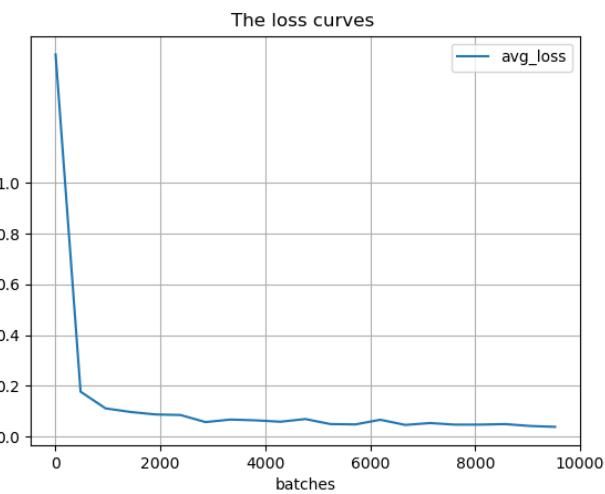


圖 3-11、分類網路訓練損失曲線圖



圖 3-12、測試集部分預測結果

## 第4章 實驗結果與比較

在第 4 章中，會將本論文所採用的方法與單獨使用 YOLOv4 進行檢測做比較，單獨 YOLOv4 訓練的方式如同前述方法，使用同一個資料集進行訓練與驗證，將檢測一個類別改為九個類別，並且訓練資料集有 1000 筆，驗證資料集有 200 筆。

### 4.1 混淆矩陣

使用測試集分別對兩種方法做出混淆矩陣，如圖 4-1 與圖 4-2 所示，對角線表示的是正確的預測數量，其餘的則是錯誤的預測，同樣的訓練資料集，使用 YOLOv4+CNN 的正確預測數量要比單獨使用 YOLOv4 還要來的多，並且在錯誤預測中，對於把較低限速斷成較高限速的情形，也就是下半三角形的部分，比較常發生在單獨使用 YOLOv4，這可能會導致駕駛在行車的途中發生危險。

預測\標籤	30	40	50	60	70	80	90	100	110
30	145	0	45	5	3	30	9	0	0
40	0	163	0	5	3	0	0	3	0
50	11	0	106	2	8	3	0	0	0
60	1	17	2	162	0	29	0	0	0
70	2	1	0	1	136	0	2	0	0
80	8	2	15	13	8	102	3	10	0
90	6	0	18	1	2	28	166	9	0
100	3	0	2	5	5	0	16	159	117
110	0	13	0	0	0	0	1	6	83

圖 4-1、單獨使用 YOLOv4 的混淆矩陣

預測\標籤	30	40	50	60	70	80	90	100	110
30	197	7	18	0	1	2	1	0	0
40	0	200	0	0	0	0	0	0	0
50	0	0	173	1	1	0	0	0	0
60	0	0	0	194	0	0	0	0	0
70	0	0	0	0	192	0	0	0	0
80	0	0	0	0	0	194	2	0	0
90	1	0	4	0	2	0	191	0	0
100	0	0	0	0	0	2	0	198	1
110	0	0	0	1	3	0	0	0	192

圖 4-2、YOLOv4+CNN 的混淆矩陣

## 4.2 評估

為了評估模型的好壞，本論文採用 mAP(Mean Average Precision)，它是一個主流的評價指標，目的是用來判斷該目標檢測模型的表現是否良好。在影像辨識領域的相關演算法中，經常會提到精確率(Precision)和召回率(Recall)，通常是希望他們的分數越高越好，但一般情況下卻不是這樣的，如一堆圖片中只檢測出一個結果，那麼精確率就會是 100%但是召回率卻很低，而全部都檢測為同一個結果的話，召回率就會很高但精確率很低，因此通常會繪製精確率與召回率的曲線圖，而 AP 就是將曲線圖下的面積，mAP 則是所有類別的 AP 平均值，數值越趨近於 1，代表該模型有著更良好的表現。

兩種方法下，每個限速標誌的 AP 以及 mAP 如表 4-1 所示。實驗結果表明，與原版 YOLOv4 相比，所採用的方法 mAP 達 0.95，大幅提升 30%，顯著提升交通限速標誌正確率。

表 4-1、兩種方法 AP 與 mAP

限速	AP	
	YOLOv4	This Work
30	0.61	0.98
40	0.94	0.95
50	0.74	0.85
60	0.86	0.96
70	0.82	0.96
80	0.60	0.96
90	0.85	0.96
100	0.55	0.99
110	0.64	0.95
mAP	0.73	0.95

## 4.3 檢測結果

圖 4-3 至圖 4-5 是本論文以及單獨使用 YOLOv4 兩種方法的檢測結果，分別為限速 70 的遠、中與近距離圖像。在中與近距離時，兩種方法都能成

功的檢測並分類出限速 70，但在遠距離的部分 YOLOv4 將限速 70 錯誤地檢測成 100，對於實際運用的場景可能會發生危險。



(a) YOLOv4



(b) YOLOv4+CNN

圖 4-3、限速標誌檢測(遠)



(a) YOLOv4



(b) YOLOv4+CNN

圖 4-4、限速標誌檢測(中)



(a) YOLOv4



(b) YOLOv4+CNN

圖 4-5、限速標誌檢測(近)

圖 4-6 是所採用方法的錯誤檢測的部分結果，通常大多數的錯誤出現在當標誌離車子過遠的時候，由於待檢測圖像中的限速牌位置離得太遠，使得上面的數字特徵較模糊時，雖然能夠檢測出位置但是在分類階段則會因為不明顯的特徵而導致分類的失敗。



(a) 限速 50 檢測成 30



(b) 限速 80 檢測成 30



(c) 限速 90 檢測成 80

圖 4-6、YOLOv4+CNN 的錯誤檢測

圖 4-7 是單獨使用 YOLOv4 檢測的部分錯誤結果，在大部分的情況下，YOLOv4 無論限速牌距離的遠近，皆有可能發生檢測錯誤的情形。



(a) 限速 50 檢測成 30



(b) 限速 60 檢測成 80



(c) 限速 110 檢測成 100

圖 4-7、單獨使用 YOLOv4 的錯誤檢測

## 第5章 結論

隨著目標檢測的日益進步，越來越多的有效且快速演算法被提了出來，檢測精度較高的兩段式目標檢測器與精度稍差但較為快速單段式目標檢測器成為了研究的主題，根據前述的比較結果，單段式目標檢測器 YOLOv4 對於實時檢測的應用上較為合適。雖然 YOLOv4 對於物件檢測的表現非常優異，但由於欲檢測的目標可能以各種型態出現，當資料集不足或相似度過高時，常會出現檢測沒問題，但辨識成另一個分類的錯誤。例如交通標誌限速 50 檢測成限速 80 的嚴重錯誤。為解決此問題，我們採用一個 YOLOv4 加強版，即採用 YOLOv4 及串接一級簡易的卷積神經網路分類器，YOLOv4 負責從圖像中初步檢測出限速標誌的位置，並且將預測框範圍內的標誌放進卷積神經網路內進行再細分類，以得到較準確的最後結果。

本論文使用 mAP 指標將採用的方法與原版 YOLOv4 進行比較來評估方法的可行性，結果顯示，在使用相同的訓練集與測試集進行訓練及測試，本論文的方法在 mAP 的分數上優於原版 YOLOv4，且總執行時間比起原版 YOLOv4 只增加約 2 毫秒，但卻能有效提升辨識率。實驗結果顯示，與原版 YOLOv4 相比，而我們所採用的方法 mAP 達到了 0.95，大幅提升 30%，顯著提升交通限速標誌正確率。

本論文所採用的方法目前僅針對限速標誌進行試驗，而現實中的交通標誌數量及種類眾多，未來也可以藉由加入更多不同種類的交通標誌進行訓練，讓整個檢測網路的實用性能有所提高，以利能實際應用於智慧自駕車。

## 參考文獻

- [1] W. Lan, J. Dang, Y. Wang, and S. Wang, “Pedestrian Detection Based on YOLO Network Model,” *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, Changchun, China, 2018, pp. 1547–1551.
- [2] C. Ding and D. Tao, “Trunk-Branch Ensemble Convolutional Neural Networks for Video-Based Face Recognition,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 1002–1014, Apr. 2018.
- [3] K. A. Ishak, M. M. Sani, N. M. Tahir, S. A. Samad, and A. Hussain, “A Speed limit Sign Recognition System Using Artificial Neural Network,” *2006 4th Student Conference on Research and Development*, Shah Alam, Malaysia, Jun. 2006, pp. 127–131.
- [4] Li Liu,Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen, “Deep Learning for Generic Object Detection: A Survey,” *International Journal of Computer Vision*, pp. 1–58, Oct. 2019.
- [5] L. Aziz, M. S. B. Haji Salam, U. U. Sheikh, and S. Ayub, “Exploring Deep Learning-Based Architecture, Strategies, Applications and Current Trends in Generic Object Detection: A Comprehensive Review,” in *IEEE Access*, vol. 8, pp. 170461–170495, Sep. 2020.
- [6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 2980–2988.
- [7] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *arXiv preprint*, arXiv:1411.4038, Mar. 2015.
- [8] C.-Y. Wang, H.-Y. M. Liao, I-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A New Backbone that can Enhance Learning Capability of CNN,” *arXiv preprint*, arXiv:1911.11929, Nov. 2019.
- [9] S. Yun, D. Han, S. J. Oh, and S. Chun, “CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features,” *arXiv preprint*, arXiv:1905.04899, Aug. 2019.
- [10]G. Ghiasi, T.-Y. Lin, and Q. V. Le, “DropBlock: A regularization method for convolutional networks,” *arXiv preprint*, arXiv:1810.12890, Oct. 2018.
- [11]Z. Yao, Y. Cao, S. Zheng, G. Huang, and S. Lin, “Cross-Iteration Batch Normalization,” *arXiv preprint*, arXiv:2002.05712, Feb. 2020.

- [12]A. Bochkovskiy, C. Y. Wang, H. Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint*, arXiv:2004.10934, Apr. 2020.
- [13]Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren, “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,” *arXiv preprint*, arXiv:1911.08287, Nov. 2019.
- [14]Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia, “Path Aggregation Network for Instance Segmentation,” *arXiv preprint*, arXiv:1803.01534, Sep. 2018.
- [15]Sanghyun Woo, Jongchan Park, J.-Y. Lee, and In So Kweon, “CBAM: convolutional block attention module,” *arXiv preprint*, arXiv:1807.06521, Jul. 2018.
- [16]Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Oct. 2014, pp. 580–587.
- [17]F. Sultana, A. Sufian, and P. Dutta, “A Review of Object Detection Models based on Convolutional Neural Network,” *2nd International Conference on Communication, Devices and Computing*, Oct. 2019.
- [18]R. Girshick, “Fast R-CNN,” arXiv:1504.08083, Sep. 2015.
- [19]S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, “Detection of traffic signs in real-world images: The German traffic sign detection benchmark,” *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–8.
- [20]M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, “Traffic sign recognition — How far are we from the solution?,” *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp. 1–8.
- [21]D. Tabernik and D. Skočaj, “Deep Learning for Large-Scale Traffic-Sign Detection and Recognition,” in *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1427–1440, Apr. 2020.
- [22]S. Šegvić et al., “A computer vision assisted geoinformation inventory for traffic infrastructure,” *13th International IEEE Conference on Intelligent Transportation Systems*, Sep. 2010, pp. 66-73.
- [23]J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788.
- [24]Nikhil Sardana, (2018, Jan 10). Networks (1st ed.) [Online]. Available: <https://tjmachinelearning.com/lectures/1718/instance/>

[25]Diganta Misra, “Mish: A Self Regularized Non-Monotonic Activation Function,” *arXiv preprint*, arXiv: 1908.08681, Aug. 2019.

