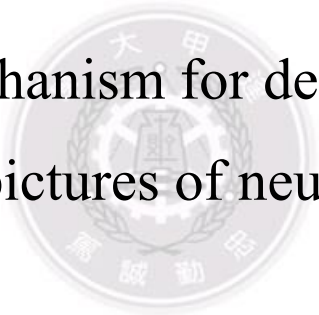


逢 甲 大 學  
通 訊 工 程 學 系  
碩 士 論 文

神經網路圖像檢測系統新物件的理解機制  
Reasoning mechanism for detecting new ob-  
jects in the pictures of neural networks



指導教授：林維崙博士

研 究 生：王紹賢

中 華 民 國 一 百 一 十 三 年 六 月

逢甲大學  
通訊工程學系碩士學位論文

中文題目:神經網路圖像檢測系統新物件的理解機制

英文題目: Reasoning mechanism for detecting new objects  
in the pictures of neural networks

☒ 學術論文

☐ 作品(藝術類)

☐ 書面報告(藝術類)

☐ 成就證明(應用科技類)

☐ 技術報告(應用科技類)

☐ 專業實務報告(專業實務類)

研究生: 王紹賢

經碩士學位考試合格特此證明

評審委員

陸清達

曾凡碩

林作育

指導教授

林作育

系主任

林作育

考試日期: 中華民國 113 年 6 月 27 日

## 逢甲大學研究生無違反學術倫理聲明書

### FCU Graduate Student Academic Ethics Statement

本人已完全瞭解學術倫理之定義與行為規範，謹此嚴正聲明，本人所呈繳之學位論文(包含作品、成就證明連同書面報告、技術報告或專業實務報告)

題目 (中文)： 神經網路圖像檢測系統新物件的理解機制，

如有抄襲、舞弊或違反著作權法等違反學術誠信與倫理之行為時，願自行承擔所有法律責任，以及概括承受一切後果，並無條件同意註銷本人之碩(博)士學位，絕無異議。

I am fully aware of and understand the University's regulations on plagiarism. And I declare herewith, that the thesis/ written report/ technical report/ professional practice report entitled "Reasoning mechanism for detecting new objects in the pictures of neural networks" is a presentation of my original work. In addition, I understand that any false claim or plagiarism in respect of this work will result in disciplinary action in accordance with University's regulations. I assume legal liability for this and completely agree to the withdrawal of the Master's/ Doctor's degree if any violation of academic ethics in the thesis is confirmed to be true.

聲明人 Student Signature : 王紹賢  
(請親筆書寫簽名 Signature)

學 號/ID.NO : M1131250

系 所 / Department : 通訊工程學系

日期 Date : 2024 / 6 / 27 (YYYY/MM/DD)

本人為 王紹賢 之指導教授，經檢視其學位論文內容，確實無抄襲或剽竊之行為。

To the best of my knowledge, this thesis has no plagiarism or violations of academic ethics.

指導教授(Academic Advisor) : 林作君 (親筆簽名 Signature)

聲明日期(Date) : 2024 年(Y) 6 月(M) 27 日(D)

\*本聲明書正本請與學位考試申請文件一併附上，影本請裝訂於紙本學位論文內。

The original statement should be included in your thesis defense application; the copy should be included in your thesis.

## 誌 謝

日月如梭，研究所的階段也逼近終點，但求學是不會結束的。國中生涯測試以及個人志向都偏向於商科的我，自從接觸到 Scratch 之後發現原來程式可以如此有趣，也慶幸自己在大學選擇通訊系繼續精進以及學習更多的程式語言以及通訊方面的知識。也感謝有那麼舒適的研究環境，以及伙伴們的互相交流，才得以走到研究所的最後階段。

首先我要感謝我的指導老師—林維崙副教授，自從大三加入老師的實驗室行列後，老師都會詢問我們新進的專題生有沒有什麼問題，進而給我們一些建議，也會整合我們出去比賽，以增進團隊合作精神以及程式開發的能力。到了大學畢業後，我決定繼續跟著老師的行列，在這兩年跟老師探討了更多 AI 方面的議題，也謝謝這兩年老師給我的建議還有討論，讓我對論文方向有更深的見解。

除了要感謝我的指導老師之外，也感謝求學階段的每一個老師，還有一起奮鬥的同學、朋友們，在求學階段也因為你們，才得以豐富我的求學生活。

最感謝的還是我的家人，因為會拿人生一大部份投資一張價格未定的股票，只有我的家人才會這麼做，我會盡我所能提升自我的價值，再回饋我的家人以及所有幫助過我的人，謝謝他們在人生成長階段的陪伴以及資助。最後再次感謝身邊所有的每一個人。

王紹賢 謹誌於

逢甲大學資訊電機學院

中華民國一百一十三年六月



## 摘要

本論文提出了模擬大腦區分新舊物件過程的架構，重點在於如何結合分類神經網路、檢測神經網路以及自注意力模型來模擬人類大腦的學習機制，其中包含大腦已經學習過的知識、與生俱來的常識能力、與如何處理未知物件的方法。研究主要目標是識別大腦已學會和未學會的圖像，並透過常識架構將未知物件加以定位與分類，以期提升系統的可解釋性和應用價值。

文中提出的框架以 YOLO、EfficientNetB5、Segment-Anything Model 等模型模擬大腦的不同功能區，並透過 Python 環境實施。透過網路蒐集的單物件樣本，本研究在模擬大腦學習和理解新物件方面呈現出可分辨新舊物件的良好性能。

**關鍵詞:**人類學習機制、神經網路

## Abstract

This paper proposes an architecture that simulates the brain's process of distinguishing old and new objects. The focus is on how to combine classification neural networks, detection neural networks, and self-attention models to simulate the learning mechanism of the human brain, which includes the knowledge that the brain has learned, and biological processes. Innate common sense ability and how to deal with unknown objects. The main goal of the research is to identify images that the brain has learned and unlearned, and to locate and classify unknown objects through a common sense framework, in order to improve the interpretability and application value of the system.

The framework proposed in this article uses models such as YOLO, EfficientNetB5, and Segment-Anything Model to simulate different functional areas of the brain, and is implemented through the Python environment. Through single object samples collected from the Internet, this study shows good performance in simulating the brain's learning and understanding of new objects in distinguishing old and new objects.

**Keywords:** human learning mechanism, neural network

# 目 錄

中文摘要.....	i
英文摘要.....	ii
目 錄.....	iii
圖 目 錄.....	v
表 目 錄.....	vi
第一章 緒 論.....	1
1.1 研究動機與目的 .....	1
1.2 論文架構 .....	4
第二章 文獻探討 .....	5
2.1 分類神經網路 .....	5
2.1.1 AlexNet.....	5
2.1.2 DenseNet.....	6
2.1.3 EfficientNet .....	7
2.2 檢測神經網路 .....	9
2.2.1 R-CNN.....	9
2.2.2 SSD.....	11
2.2.3 YOLO .....	12
2.3 自注意力模型 .....	16
2.3.1 Transformer.....	16
2.3.2 GPT.....	18
2.3.3 SAM.....	19
第三章 人類大腦學習架構探討 .....	21
3.1 大腦學習機制 .....	21
3.1.1 原有知識.....	22
3.1.2 識別與分類區.....	23
3.1.3 學習新知識區.....	23
3.2 神經網路架構.....	24
3.2.1 知識區.....	24
3.2.2 常識區.....	25
3.2.3 新知識之交叉驗證區.....	25
3.3 架構參數配置 .....	26
3.3.1 YOLOv9.....	26
3.3.2 YOLOv7.....	28
3.3.3 YOLOv7-tiny.....	29
3.3.4 EfficientNetB5.....	31
3.3.5 SAM.....	32

<b>第四章 資料準備和分配及模型架構實施 .....</b>	<b>35</b>
<b>4.1 圖像資料準備 .....</b>	<b>35</b>
<b>4.2 圖像資料分配 .....</b>	<b>35</b>
4.2.1 YOLOv7、YOLOv9 訓練資料集—知識區 .....	35
4.2.2 自定義模型檢測資料集—常識區 .....	36
4.2.3 新學習類別交叉驗證資料集—新類別之交叉驗證 .....	37
<b>4.3 模型架構實施 .....</b>	<b>37</b>
4.3.1 知識區實施 .....	38
4.3.2 常識區實施 .....	41
4.3.3 新知識之交叉驗證區實施 .....	47
<b>4.4 模型架構實施結果探討 .....</b>	<b>54</b>
4.4.1 知識區探討 .....	54
4.4.2 常識區探討 .....	55
4.4.3 新知識之交叉驗證區探討 .....	57
<b>第五章 結論與未來方向 .....</b>	<b>58</b>
<b>5.1 結論 .....</b>	<b>58</b>
<b>5.2 未來方向 .....</b>	<b>58</b>
<b>參考文獻 .....</b>	<b>60</b>





## 圖目錄

圖 2-1 AlexNet 模型示意圖[4] .....	6
圖 2-2 ResNet 模型架構圖[13] .....	6
圖 2-3 DenseNet 模型架構圖[5] .....	7
圖 2-4 DenseNet 模型示意圖[5] .....	7
圖 2-5 EfficientNet B0 模型架構圖 [3] .....	8
圖 2-6 Compound Scaling [3] .....	8
圖 2-7 EfficientNet 各版本指標比較[3] .....	9
圖 2-8 R-CNN 示意圖[7] .....	10
圖 2-9 VGG 層數及架構圖 [17] .....	11
圖 2-10 SSD 架構圖[10] .....	12
圖 2-11 YOLO 偵測示意圖[6] .....	13
圖 2-12 YOLO 偵測系統圖[6] .....	13
圖 2-13 E-ELAN 架構圖[18] .....	14
圖 2-14 PGI 架構圖[16] .....	15
圖 2-15 CSPNet 、 ELAN 、 GELAN 架構圖[20][19][21] .....	16
圖 2-16 RNN 架構[22] .....	17
圖 2-17 Transformer 架構[11] .....	17
圖 2-18 GPT-1 架構圖[12] .....	19
圖 2-19 ViT 模型架構[23] .....	20
圖 2-20 SAM 模型概述[8] .....	20
圖 3-1 大腦結構圖 .....	21
圖 3-2 大腦中儲存原有知識的區塊 .....	22
圖 3-3 大腦中處理物件識別之區塊 .....	23
圖 3-4 大腦中新知識形成之區塊 .....	23
圖 3-5 模擬大腦學習過程及理解之架構 .....	24
圖 3-6 模擬大腦學習結果驗證之架構 .....	26
圖 3-7 原圖與 SAM 對圖像偵測到的物件遮罩 .....	32
圖 4-1 知識區為 YOLOv7 訓練自定義模型之混淆矩陣 .....	38
圖 4-2 知識區為 YOLOv7 訓練自定義模型性能指標圖 .....	39
圖 4-3 知識區為 YOLOv9 訓練自定義模型之混淆矩陣 .....	39
圖 4-4 知識區為 YOLOv9 訓練自定義模型性能指標圖 .....	40
圖 4-5 切割框示意圖 .....	43
圖 4-6 餘弦相似度分類分佈 .....	44

圖 4-7 KL 散度分類分佈 .....	44
圖 4-8 知識區為 YOLOv7 自定義模型之常識區分類後訓練 YOLOv7-tiny 混淆矩陣 .....	49
圖 4-9 知識區為 YOLOv7 自定義模型之常識區分類後訓練 YOLOv7-tiny 之性能指標 .....	49
圖 4-10 知識區為 YOLOv7 自定義模型之常識區分類後訓練 EfficientnetB5 訓練走勢 .....	50
圖 4-11 知識區為 YOLOv9 自定義模型之常識區分類後訓練 YOLOv7-tiny 之混淆矩陣 .....	51
圖 4-12 知識區為 YOLOv9 自定義模型之常識區分類後訓練 YOLOv7-tiny 之性能指標 .....	51
圖 4-13 知識區為 YOLOv9 自定義模型之常識區分類後訓練 EfficientnetB5 訓練走勢 .....	52



## 表目錄

表 3-1 YOLOv9訓練之超參數 .....	26
表 3-2 YOLOv7訓練之超參數 .....	28
表 3-3 YOLOv7-tiny 訓練之超參數 .....	30
表 3-4 EfficientNetB5訓練之超參數 .....	31
表 3-5 SAM 生成遮罩之超參數 .....	33
表 4-1 YOLOv7、YOLOv9 訓練自定義模型使用資料 .....	35
表 4-2 YOLOv7、YOLOv9自定義模型檢測舊類別資料分配 .....	36
表 4-3 YOLOv7、YOLOv9 自定義模型檢測欲學習類別資料分配 .....	36
表 4-4 EfficientNetB5、YOLOv7-tiny 交叉驗證新學習類別資料分配 .....	37
表 4-5 EfficientNetB5、YOLOv7-tiny 交叉驗證未學習類別資料分配 .....	37
表 4-6 知識區為 YOLOv7 自定義模型檢測舊類別之效能 .....	40
表 4-7 知識區為 YOLOv9 自定義模型檢測舊類別之效能 .....	41
表 4-8 知識區為 YOLOv7 自定義模型檢測欲學習類別之效能 .....	41
表 4-9 知識區為 YOLOv9 自定義模型檢測欲學習類別之效能 .....	42
表 4-10 SAM 特徵裁剪參考指標 .....	42
表 4-11 YOLOv7 欲學習類別之原圖分類率 .....	45
表 4-12 YOLOv9 欲學習類別之原圖分類率 .....	45
表 4-13 YOLOv7 欲學習類別之 SAM 取最大遮罩分類率 .....	45
表 4-14 YOLOv9 欲學習類別之 SAM 取最大遮罩分類率 .....	46
表 4-15 YOLOv7 欲學習類別之 SAM 取最大預測 IoU 邊界框的分類率 .....	46
表 4-16 YOLOv9 欲學習類別之 SAM 取最大預測 IoU 邊界框的分類率 .....	46
表 4-17 YOLOv7 欲學習類別之 SAM 取最大預測 IoU 分割框的分類率 .....	47
表 4-18 YOLOv9 欲學習類別之 SAM 取最大預測 IoU 分割框的分類率 .....	47
表 4-19 知識區為 YOLOv7 自定義模型之常識區分類後訓練數據 .....	48
表 4-20 知識區為 YOLOv9 自定義模型之常識區分類後訓練數據 .....	48
表 4-21 知識區為 YOLOv7 自定義模型之常識區分類後訓練 EfficientnetB5 分類 報告 .....	50
表 4-22 知識區為 YOLOv9 自定義模型之常識區分類後訓練 EfficientnetB5 分類 報告 .....	52
表 4-23 知識區為 YOLOv7所衍生之常識區分類訓練後檢測新學習類別結果 ....	53
表 4-24 知識區為 YOLOv7所衍生之常識區分類訓練後檢測未學習類別結果 ....	53
表 4-25 知識區為 YOLOv9所衍生之常識區分類訓練後檢測新學習類別結果 ....	53
表 4-26 知識區為 YOLOv9所衍生之常識區分類訓練後檢測未學習類別結果 ....	54

表 4-27 知識區模型之舊類別檢測率比較 .....	54
表 4-28 知識區模型之新類別未檢測率比較 .....	54
表 4-29 知識區為 YOLOv7 自定義模型之常識區分類結果 .....	55
表 4-30 知識區為 YOLOv9 自定義模型之常識區分類結果 .....	56
表 4-31 知識區自定義模型之交叉驗證區訓練及測試集資料分配 .....	57
表 4-32 知識區自定義模型之交叉驗證區新學習類別檢測結果 .....	57
表 4-33 知識區自定義模型之交叉驗證區未學習類別檢測結果 .....	57



# 第一章 緒 論

## 1.1 研究動機與目的

深度神經網路是近期 AI 常用的應用，技術也隨著時間推移日新月異，從最久遠由 D. E. Rumelhart 提出的反向傳播演算法[1]，以及 G. E. Hinton 提出的深度信念網路(Deep belief network)[2]，奠定了深度神經網路的基礎，進而演化出各種不同類型的神經網路，例如：自注意力模型(Self-Attentive Model)、分類神經網路(Classification Neural Network)、檢測神經網路(Detection Neural Network) ... 等等。

分類神經網路中，M. tan 在2019年所提出的 Efficientnet[3]改善了先前的 AlexNet[4]、DenseNet[5]...等的參數、計算效率、模型泛化能力，以及快速訓練與推理...等，也是至今大眾常用的分類網路之一。

而檢測神經網路，J. Redmon 團隊在2015提出的 You Only Look Once (YOLO)[6]，跟早期的 Region-based Convolutional Neural Network (R-CNN)[7]不同的是 YOLO 在檢測與定位是通過單個網路模型一體成型的，而非 R-CNN 是檢測與定位是分開兩種階段進行的，所以對於需要處理大量圖像數據的應用，YOLO[6]耗時相比 R-CNN 是大幅減少的，且 YOLO 因經過全局損失函數最佳化模型，在圖像上進行目標檢測時，可以更準確的掌握目標的上下文訊息，所以在處理密集或小目標時，效能較 R-CNN 佳。

最後自注意力模型的概念是在2017年由 Z. Lin 所提出的，自注意力模型先前的架構是強調文字中不同單詞之間的依賴關係

及語意訊息，用於文字分類、機器翻譯...等，而 Meta AI 在2023年開發的 Segment Anything Model (SAM)[8]能把自注意力的機制套用在圖像上，針對圖像中不同區域生成語意分割的遮罩，進而更準確區分出圖像中的不同個體或區域。

隨著 AI 技術的發展，AI 系統在各個領域的應用越來越廣泛，如醫療診斷、金融決策、司法審判等。然而，傳統的 AI 系統，尤其是深度學習模型，通常被視為黑箱，其內部決策過程對於用戶而言難以理解。這種不可解釋性不僅限制了 AI 的應用範圍，還在一定程度上削弱了用戶對 AI 系統的信任。因此， Explainable AI (XAI)[9]應運而生。

XAI 於 2016 年由美國國防高等研究計畫署(DARPA)所提出，主要是讓使用者了解並相信機器學習演算法的過程和產出，XAI 包含了通過顯示模型輸出與輸入特徵之間的關係來幫助理解模型的決策過程的可視化技術、解釋模型...等。被提出應用於醫療[10]、金融[11]、自動駕駛[12]等等。在上述三種領域的 XAI 應用中應用領域中，都是在已知各領域的專有知識條件下去制定規則，例如腫瘤種類、理財方式、路障樣式等等這些已知的知識。如果遇到未知物件時，要如何定義該”未知物件”也是一大課題，在遇到未知物件時的處理規則尚未在 XAI 領域中有確切的討論議題，而在本論文中，我們會針對這一區塊進行實驗以及探討。

在本文中，我們將會架構分類神經網路、檢測神經網路跟自注意力模型去模擬人類大腦學習的運作流程，主要工作在於分辨哪些圖像是大腦本身就學會的，哪些圖像是大腦沒有學會的，而大腦沒有學會的圖像也能用大小、紋理、顏色、形狀...



等進一步分類，等某個分類被分到規定張數時，進而學習該分類。貫徹知之為知之，不知為不知且又能對不知道的類別分門別類。對於 XAI 而言，關於“理解”的問題尚缺乏確切的討論和深入研究。特別是在模擬大腦學習新物件的過程中，如何有效結合這些技術來模仿人類大腦的學習機制，仍然是一個未解決的挑戰。

本論文為這一挑戰探討對於已知與未知物件的處理原則下建構大腦學習新物件過程之架構，並探討實施結果，且架構都是以 Python 環境進行實施。



## 1.2 論文架構

本論文分為五個章節，第一章為緒論，內容描述研究的動機與目的，第二章為文獻回顧，內容介紹分類神經網路、檢測神經網路以及自注意力模型，第三章為大腦學習新物件過程之架構探討，針對模擬大腦學習新物件的機制去設計架構以及介紹如何運行架構，第四章為資料集準備以及利用不同模型實施架構之各部分並探討實施結果，且架構都是以 Python 環境進行實施，第五章為實驗結論與未來方向。



## 第二章 文獻探討

第二章探討目前常用分類神經網路、檢測神經網路以及自注意力模型，分別介紹分類神經網路 AlexNet、DenseNet[5]、EfficientNet 檢測神經網路 R-CNN、YOLO 與 SSD[10]，最後介紹自注意力模型 Transformer[11]、GPT[12]以及 SAM，而本論文採用 EfficientNet、YOLO 及 SAM 作為架構技術。

### 2.1 分類神經網路

分類神經網路是一種人工神經網路，被用於將輸入數據映射到特定類別或標籤的任務中。通常用於監督式學習，其中訓練數據包含了輸入與對應的標籤。分類神經網路的目標是從給定的輸入中學習到一個函數，這個函數能夠將輸入映射到正確的類別。

#### 2.1.1 AlexNet

AlexNet 是由 A. Krizhevsky、I. Sutskever 和 G. E. Hinton 於 2012 年開發的一個深度卷積神經網路（CNN），其主要目的是用於圖像分類任務。

在技術上 AlexNet 是第一個在大規模網路中使用 ReLU 的網路，ReLU 可以加速訓練過程，因為它比傳統的 sigmoid 和 tanh 函數更不容易出現梯度消失問題。為了減少過擬合，AlexNet 在全連接層中使用了 Dropout 技術。Dropout 會在訓練過程中隨機忽略一些神經元，使得模型更具泛化能力。

AlexNet 在 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 中的前五名錯誤率（Top-5 Error Rate）為

15.3%，這是一個非常顯著的改進，遠低於當時第二名的 26.2%。AlexNet 引領了深度學習的浪潮，對電腦視覺和整個人工智慧領域產生了必然的影響。

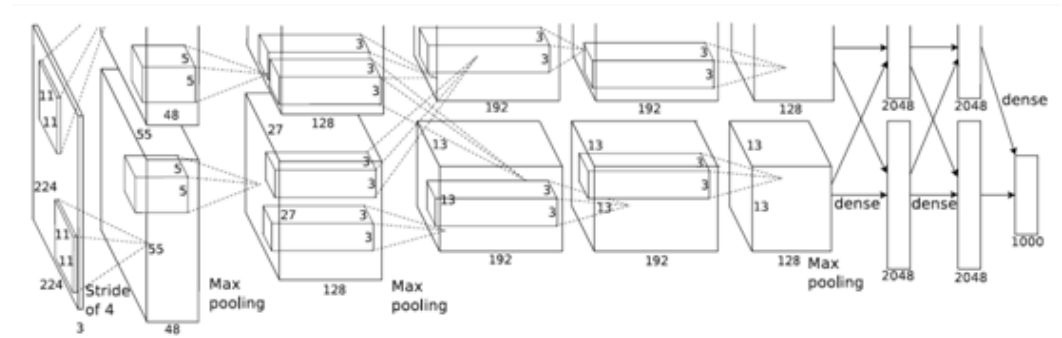


圖 2-1 AlexNet 模型示意圖[4]

### 2.1.2 DenseNet

由 G. Huang 等學者於 2017 年提出。其設計理念源自於 Residual Neural Network (ResNet) [13]，但相較於 ResNet，DenseNet 的每一層都與前面的所有層直接相連，形成了密集的连接結構。

在 DenseNet 中，每個層都直接接收來自前面所有層的特徵圖作為輸入，這種密集連接的設計使得網路的訊息傳遞更加高效，有助於緩解梯度消失問題，並能更充分地利用參數。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2 <sub>x</sub>	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3 <sub>x</sub>	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4 <sub>x</sub>	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5 <sub>x</sub>	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$

圖 2-2 ResNet 模型架構圖[13]

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	$112 \times 112$	$7 \times 7$ conv, stride 2			
Pooling	$56 \times 56$	$3 \times 3$ max pool, stride 2			
Dense Block (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	$56 \times 56$	$1 \times 1$ conv			
	$28 \times 28$	$2 \times 2$ average pool, stride 2			
Dense Block (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	$28 \times 28$	$1 \times 1$ conv			
	$14 \times 14$	$2 \times 2$ average pool, stride 2			
Dense Block (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	$14 \times 14$	$1 \times 1$ conv			
	$7 \times 7$	$2 \times 2$ average pool, stride 2			
Dense Block (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	$1 \times 1$	$7 \times 7$ global average pool			
		1000D fully-connected, softmax			

圖 2-3 DenseNet 模型架構圖[5]

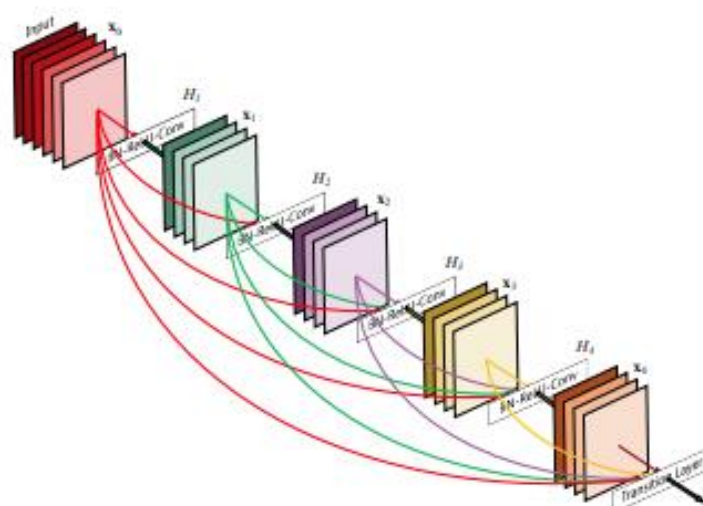


圖 2-4 DenseNet 模型示意圖[5]

## 2.1.3 EfficientNet

由 M. Tan 等學者於 2019 年提出的高效神經網路架構，作者採用了 Neural Architecture Search (NAS) 的方法架構 EfficientNet-B0，如圖 2-5，往後的 EfficientNet 版本基於 EfficientNet-B0 使用了 Compound Scaling [3] 融合了寬度、深度和解析度的縮放比例，作者使用縮放係數  $\phi$  將網路模型均勻地縮放，用於控制能使用多少資源在模型縮放上。例如：計算資源為  $2^\phi$  倍，則將網路深度、寬度、圖像解析度分別增加為  $\alpha^\phi$ 、 $\beta^\phi$ 、 $\gamma^\phi$  倍，其中  $\alpha$ 、 $\beta$ 、 $\gamma$  是進

行 small grid search 而決定的常數。Compound Scaling 可以策略將模型的參數量和計算量保持在可控的範圍內，也能依據給定的計算資源構建適當大小的模型，同時提高模型的效能，如圖 2-6 至圖 2-7。

EfficientNetB5 相比 EfficientNetB0 深度、寬度和解析度都有明顯擴展，如圖 2-7，使得 Compound Scaling 的使用更有彈性，模型精度更高以及泛化能力愈加穩定。

Stage $i$	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	$224 \times 224$	32	1
2	MBConv1, k3x3	$112 \times 112$	16	1
3	MBConv6, k3x3	$112 \times 112$	24	2
4	MBConv6, k5x5	$56 \times 56$	40	2
5	MBConv6, k3x3	$28 \times 28$	80	3
6	MBConv6, k5x5	$14 \times 14$	112	3
7	MBConv6, k5x5	$14 \times 14$	192	4
8	MBConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1

圖 2-5 EfficientNet B0 模型架構圖 [3]

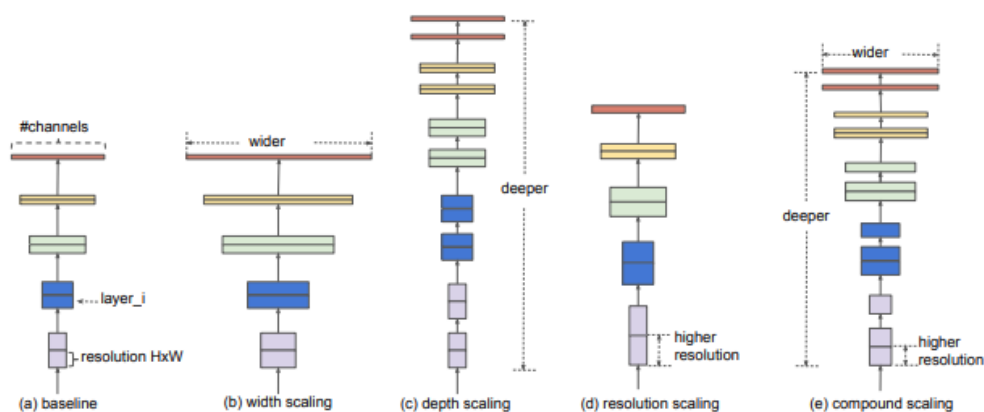


圖 2-6 Compound Scaling [3]



$$\text{Depth: } d = \alpha^0$$

$$\text{Width: } w = \beta^0$$

$$\text{Resolution: } r = \gamma^0$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

式 2-1 Compound Scaling 公式[3]

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
<b>EfficientNet-B0</b>	<b>77.1%</b>	<b>93.3%</b>	<b>5.3M</b>	<b>1x</b>	<b>0.39B</b>	<b>1x</b>
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
<b>EfficientNet-B1</b>	<b>79.1%</b>	<b>94.4%</b>	<b>7.8M</b>	<b>1x</b>	<b>0.70B</b>	<b>1x</b>
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
<b>EfficientNet-B2</b>	<b>80.1%</b>	<b>94.9%</b>	<b>9.2M</b>	<b>1x</b>	<b>1.0B</b>	<b>1x</b>
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
<b>EfficientNet-B3</b>	<b>81.6%</b>	<b>95.7%</b>	<b>12M</b>	<b>1x</b>	<b>1.8B</b>	<b>1x</b>
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
<b>EfficientNet-B4</b>	<b>82.9%</b>	<b>96.4%</b>	<b>19M</b>	<b>1x</b>	<b>4.2B</b>	<b>1x</b>
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
<b>EfficientNet-B5</b>	<b>83.6%</b>	<b>96.7%</b>	<b>30M</b>	<b>1x</b>	<b>9.9B</b>	<b>1x</b>
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
<b>EfficientNet-B6</b>	<b>84.0%</b>	<b>96.8%</b>	<b>43M</b>	<b>1x</b>	<b>19B</b>	<b>1x</b>
<b>EfficientNet-B7</b>	<b>84.3%</b>	<b>97.0%</b>	<b>66M</b>	<b>1x</b>	<b>37B</b>	<b>1x</b>
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

圖 2-7 EfficientNet 各版本指標比較[3]

## 2.2 檢測神經網路

檢測神經網路是一種用於檢測和識別輸入數據中物體、目標或事件的人工神經網路。與分類神經網路相比，檢測神經網路的目標是在輸入數據中標識出特定目標的位置和類別，而不僅僅是將整個輸入映射到單一類別。

### 2.2.1 R-CNN

R-CNN 在 2013 年被 R. Girshick 團隊所提出，是一種用於目標偵測的深度學習模型。

R-CNN 使用選擇性搜尋 (Selective Search) 等演算法從輸入影像中提取出若干候選區域，這些區域被認為可能包含感興趣的目標物體。這些候選區域被稱為「區域建議」。對於每一個區域建議，從輸入影像中截取出相應的區域，並將其調整為固定大小。然後，使用一個預先訓練的卷積神經網路來提取每個區域的特徵表示。對於每個區域提取的特徵，使用一個獨立的分類器來進行目標分類，判斷該區域內是否包含感興趣的目標，並將其歸類到相應的類別中，如圖 2-9。

R-CNN 還引入了一個邊界框回歸器，用於微調每個區域建議的邊界框位置，以更準確地擬合目標物體的位置。R-CNN 的主要優點是能夠準確地檢測出影像中的目標，並且具有很強的泛化能力。

然而，R-CNN 需要對每個候選區域都進行特徵提取和目標分類，導致運行速度較慢。由於需要為每個區域建議提取特徵，因此需要大量的記憶體來儲存這些特徵表示，導致記憶體消耗較大。R-CNN 的訓練過程需要獨立訓練每個區域的分類器，因此訓練時間較長且較為複雜。

R-CNN 後來也衍生出了 R-FCN[14]、Faster R-CNN[15] 及 Mask R-CNN[16] 等等.....進階模型以處理各種檢測任務中。

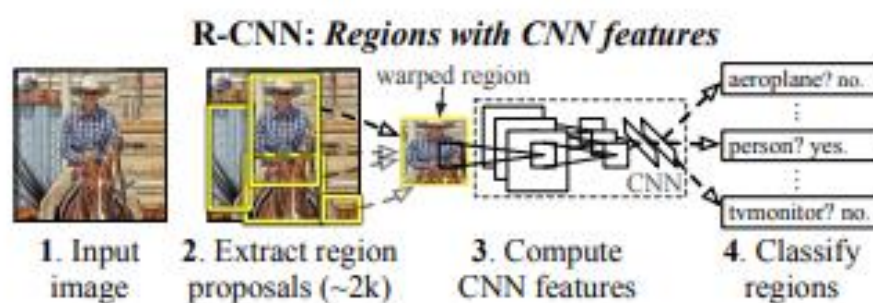


圖 2-8 R-CNN 示意圖[7]

## 2.2.2 SSD

Single Shot MultiBox Detector (SSD)，SSD 基於 Visual Geometry Group 16 (VGG16)[17]的架構下改造，是一種用於物體檢測的深度學習模型。主要特點是僅需一次前向傳播就能完成整個檢測過程，包括生成候選框和分類，也能利用不同層級的特徵圖來檢測不同大小的物體。這使得它在檢測大、中的物體時具有更高的靈活性和準確性。

缺點是小物體的特徵容易在深層卷積層中丟失，導致檢測效果不佳、因為多層特徵圖計算和預選框需要較高的計算資源所導致的資源問題。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

圖 2-9 VGG 層數及架構圖;(上) VGG 各版本網路層數 (下) VGG 各版本網路架構圖[17]

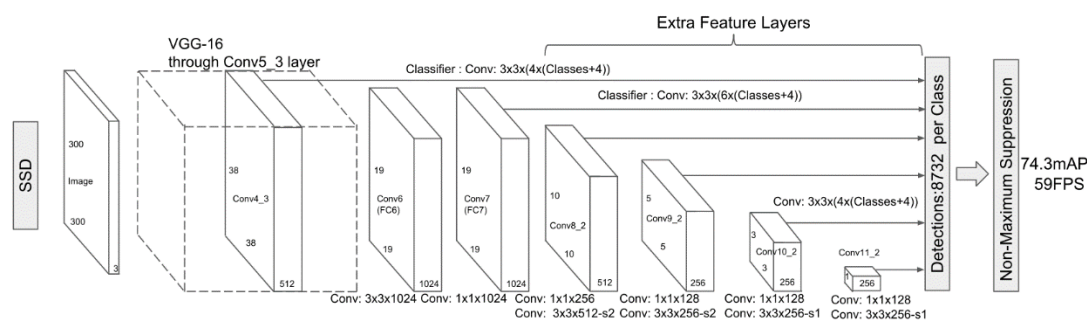


圖 2-10 SSD 架構圖[10]

## 2.2.3 YOLO

YOLO 初始版本由 Joseph Redmon 等人於 2016 年提出。它的設計理念是將物體檢測問題轉化為單一的回歸問題，直接從圖像預測邊界框和類別機率。

網路架構的部分使用了 24 個卷積層和 2 層全連接層，卷積層的部分用在提取圖像的特徵，全連接層用於預測圖像位置 and 是否有包含物體的信心指數及類別機率。

在使用 YOLOv1 時，會先將圖像大小改成 448\*448 的大小，並將圖片切成 7\*7 的網格，而每個網格會負責偵測兩個邊界框，每個邊界框都會輸出五個值：中心點(x,y)、長(w)、寬(h) 以及信心指數(conf)，而信心指數的依據是邊界框與真實數據(Ground Truth)之間的 IoU(Intersection over Union)，如圖 2-11。

而邊界框的生成不會只有一個，所以作者使用了 NMS (Non-Maximum Suppression)，先設定一個閾值，再將所有被預測的邊界框按照信心指數排序，後選擇信心指數最高的邊界框為輸出邊界框，如果該邊界框與其他邊界框的 IoU 大於閾值，代表兩個邊界框是高相關的，進而會把兩者之間信心指數較小的邊界框刪除；如果該邊界框與其他邊界框的 IoU 小於閾值，代表兩個邊



界框是低相關的，代表兩者所框到的高機率是不同物體，進而把兩個邊界框都留住，如圖 2-12。

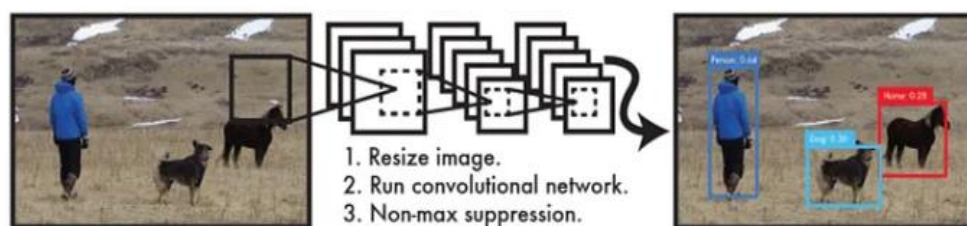


圖 2-11 YOLO 偵測示意圖[6]

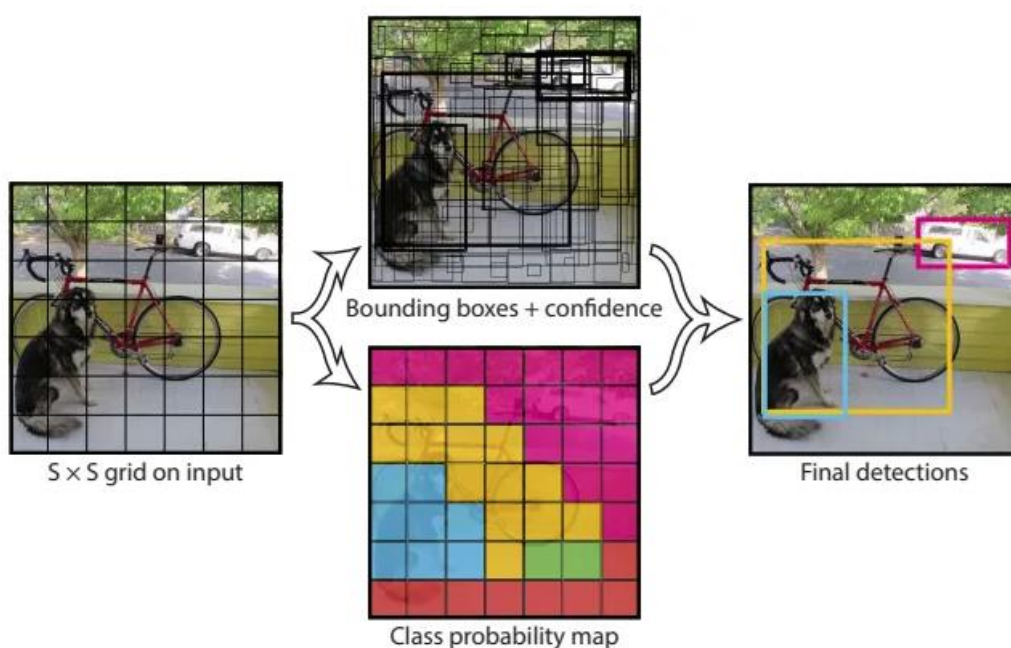


圖 2-12 YOLO 偵測系統圖[6]

YOLOv7 [18] 於 2022 年由 C. Y. Wang 的團隊提出，YOLOv7 構建了 E-ELAN [18] 為主要架構。

ELAN 透過有效聚合特徵、降低計算複雜度和增強特徵傳播來提高神經網路效能。E-ELAN [18] 基於 ELAN 架構採用技術來擴展、洗牌和合併特徵圖的基數。透過增加網路每個階段特徵的多樣性和豐富性，它可以實現更好的梯度流和特徵重複使用。這會提高學習效率和模型表現，如圖 2-13。

E-ELAN 的架構旨在維持高效率的梯度傳輸路徑。這確保了梯度在反向傳播期間可以更有效地流過網路這對於訓練深度網路至關重要。

E-ELAN 包括專為基於逐層的架構設計的複合擴展方法。這種方法可確保深度和寬度的縮放因子一起最佳化，從而在不同的硬體配置和應用場景中實現更好的整體效能。

且 YOLOv7-tiny[18]也在這時候問世，它減少了 YOLOv7 的模型架構，適合用於受限資源中，但這也降低了一些模型的準確率。

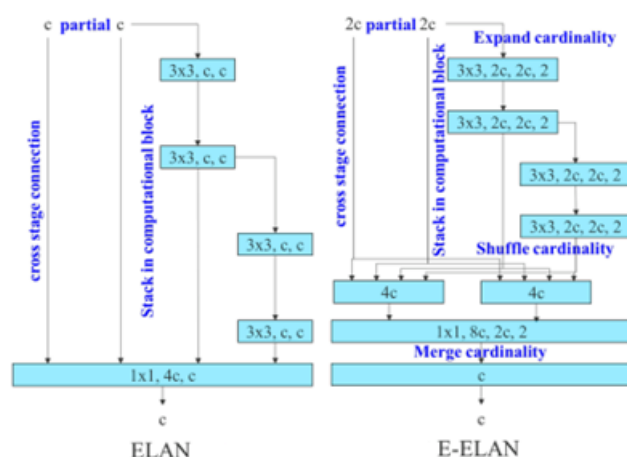


圖 2-13 E-ELAN 架構圖[18]

YOLOv9 [16] 則是在 2024 年被 C. Y. Wang 的團隊提出，在 YOLOv9 中使用了 Programmable Gradient Information (PGI) 以及 Generalized Efficient Layer Aggregation Network (GELAN) 處理深度學習網路中的損失問題。

PGI[16]中有三個主要構成要件，Main Branch 負責模型的主要訊息通道，用於推斷過程以及目標任務的計算；Auxiliary Reversible Branch 做為與 Main Branch 並行的輔助分支，可以透過輔



助分支的反向傳播更穩固生成的梯度，也能減少 Main Branch 對於目標任務計算的損失；Multi-level Auxiliary Information 透過在不同網路層次上引入額外的隱藏狀態，這些隱藏狀態有助於更好地保留在深度網路中可能丟失的細節和特徵，進而提高模型的準確性及效能。

GELAN[16]結合了 CSPNet[20]以及 ELAN[21]兩種神經網路，CSPNet[20]將特徵圖分成兩個部分，一部分進行提取特徵，另一部分與輸出相連，實現了訊息的有效傳遞及特徵重用，提高了模型的準確性及效率；ELAN 透過在計算區塊中堆疊來最佳化整個網路的梯度長度。ELAN 的目的在於解決深度模型在執行模型擴展時收斂逐漸惡化的問題。這個結構透過分析整個網路中每個層的最短梯度路徑和最長梯度路徑，設計了具有高效梯度傳播路徑的層聚合結構。在 ELAN 中，可以靈活設定計算區塊的堆疊次數，以在準確性和計算之間取得平衡。這種結合使得 GELAN 具有更好的特徵提取能力和更可靠的梯度訊息，有助於提高模型的性能和訓練效果。

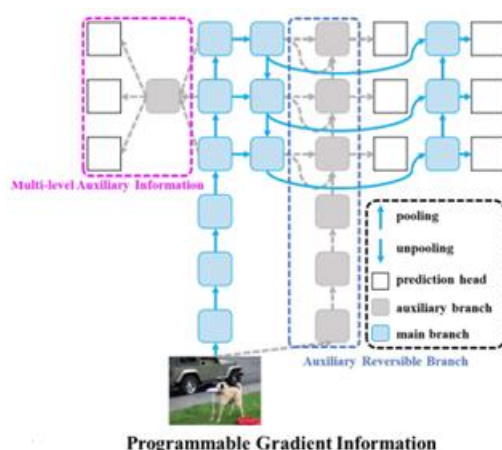


圖 2-14 PGI 架構圖[16]

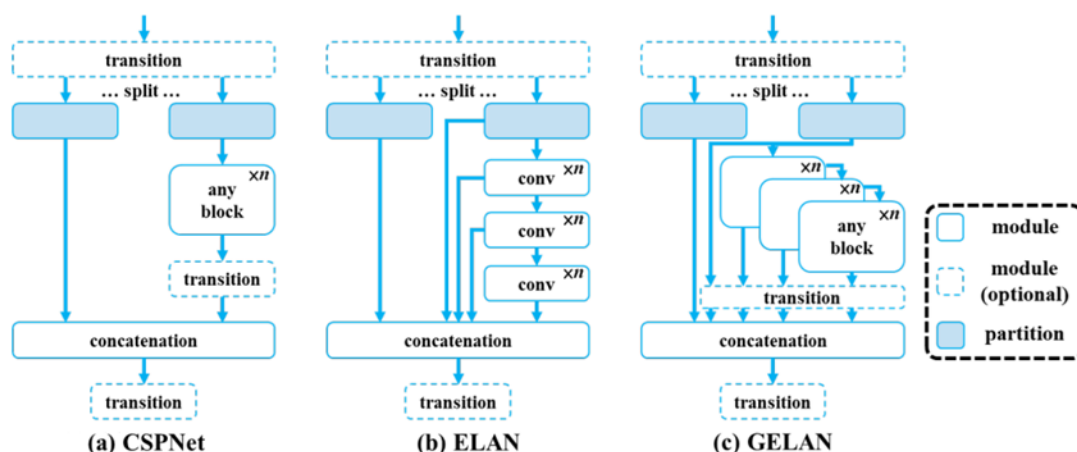


圖 2-15 CSPNet 、 ELAN 、 GELAN 架構圖[20][19][21]

## 2.3 自注意力模型

是一種用於處理序列資料的深度學習模型，最初被引入到自然語言處理領域中。它是透過計算輸入序列中各個元素之間的關聯性，來學習輸入序列的表示。自注意力模型的關鍵思想是對於輸入序列中的每個元素，它都可以與其他元素進行交互，並且根據它們之間的關聯性來調整自身的表示。

### 2.3.1 Transformer

Transformer[11]於 2017 年由 A. Vaswani 等人所提出，Transformer 不再像以往的神經網路使用 Recurrent neural network (RNN)[22]，而是以 Encoder 跟 Decoder 取代架構。

編碼器用於將輸入序列轉換為連續表示，而解碼器則用於根據這些表示生成輸出序列，並通過自注意力機制來捕捉序列中的依賴關係。這種編碼器-解碼器結構使 Transformer 模型能夠在各種序列轉換任務中取得出色的表現。

Transformer 下的自注意力機制使用了 Key(K)、Queries (Q) 與 Value(V) 進行運算。其中 Key 代表著標籤查詢中的訊息，以便查詢時可以對其關注；Queries 用於找出與 Key 相關的訊息向量；

Value 則是包含了 Queries 與 Key 相關的實際訊息。其算法分為兩個階段：計算注意力分數、加權總和,如圖 2-17。

通過計算 Q 和 K 的內積，將結果除以  $\sqrt{d_k}$  ( $d_k$  為 Key 的維度) 以縮放，再利用 Softmax 依照縮放的结果獲得注意力權重。得到注意力權重後乘上對應的 Value，獲得加權值，最後將每個訊息得到的加權值相加得到最後的注意力結果，以確保模型在計算注意力時將重點放在重要的部分，從而有效地捕捉序列中的關鍵訊息，如式 2-2。

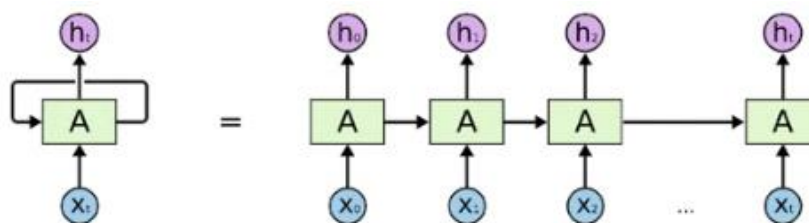


圖 2-16 RNN 架構[22]

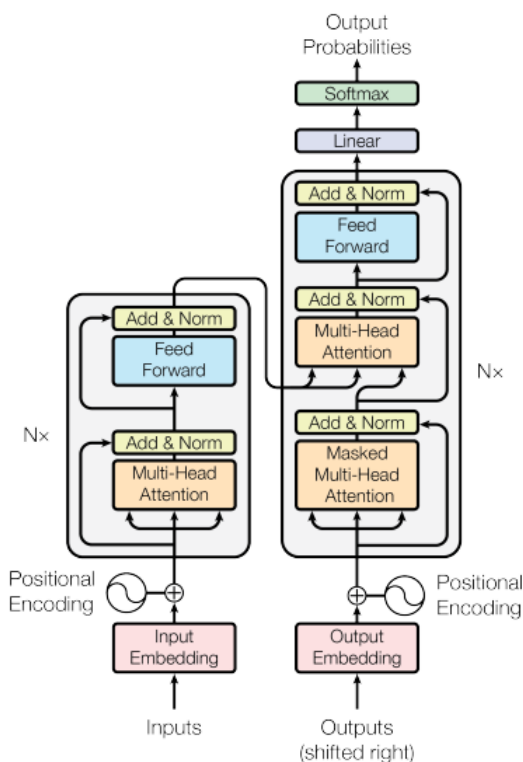


圖 2-17 Transformer 架構[11]

$$\text{Attention}(Q, K, v) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

式 2-2 自注意力機制算式[11]

### 2.3.2 GPT

GPT[12] 是基於 Transformer 下搭配監督式學習所建構的模型，在 2018 年被 A. Radford 等人所提出，參數量 1.17 億。GPT 模型可以運用在答覆問題、文字生成以及程式碼生成。

在架構中，GPT-1 使用了 12 個 Transformer 對於大規模的文字數據進行建模以及預訓練語言結構和語意關係。預訓練後經過 fine-tuning 去更適應特定的目標任務，例如判斷句與句之間的關係、文字分類...等，讓 GPT 在各種語言理解的分配任務上做得更加精確，如圖 2-18。

在 2019 年推出的 GPT-2，參數量 15 億，使用了 48 個 Transformer，在訓練的資訊擴大後，承接了 GPT-1 的 GPT-2 已經升級到可以跟使用者進行簡短對話以及生成故事。

在 2020 年推出的 GPT-3，參數量擴展到 1750 億，使用了 96 個 Transformer，比起前年推出的 GPT-2 更大幅提升了一個檔次，GPT-3 也是延續了前兩個版本的基礎，但能勝任的工作範圍更廣，包括了可以生成高品質的文章、撰寫程式碼、智能機器人...等。

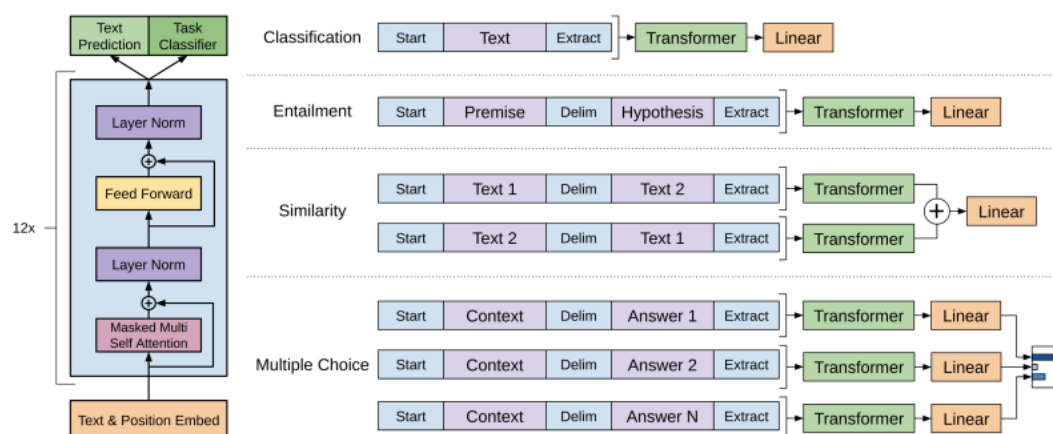


圖 2-18 GPT-1 架構圖[12]

### 2.3.3 SAM

在 2023 年，Facebook AI (Meta) 推出了 Segment Anything Model (SAM)，是一個基於提示的分割模型，旨在解決圖像分割任務。SAM 可以從單個前景點生成高品質的分割遮罩，並且展現出色的性能，包括邊緣檢測、對象提議生成、實例分割等任務。SAM 的訓練數據集包括公共分割數據集，並通過不斷重新訓練和優化模型，提高了標記速度和分割品質。SAM 的目標是為研究目的提供一個開放且易於使用的分割模型，並且已經釋出了 SA-1B 數據集供研究使用。

SAM 模型下有三個架構：圖像編碼器 (Image encoder)、提示編碼器 (Prompt encoder) 和輕量級遮罩解碼器 (Light weight mask decoder)，如圖 2-19。Image encoder 使用了 Masked Autoencoder (MAE) 預訓練的 Vision Transformer[23] (ViT)，並對其進行了最小程度的調整以處理高分辨率輸入；Prompt encoder 考慮了稀疏提示（點、框、標籤）和密集提示（遮罩）。它使用位置編碼，對點和框進行表示，並使用來自 Contrastive Language-image Pre-training (CLIP) 的標籤編碼器對自由標籤進行表示。密集提示（即遮罩）則使用卷積進行嵌入，並與圖像嵌入進行逐元素求



和；Light weight mask decoder 是一個輕量級組件，與 Prompt encoder 一起工作，從提示中生成遮罩。Light weight mask decoder 的技術包括使用卷積進行嵌入，並與圖像嵌入進行逐元素求和。這種設計使得 Light weight mask decoder 能夠快速且有效地生成分割遮罩。

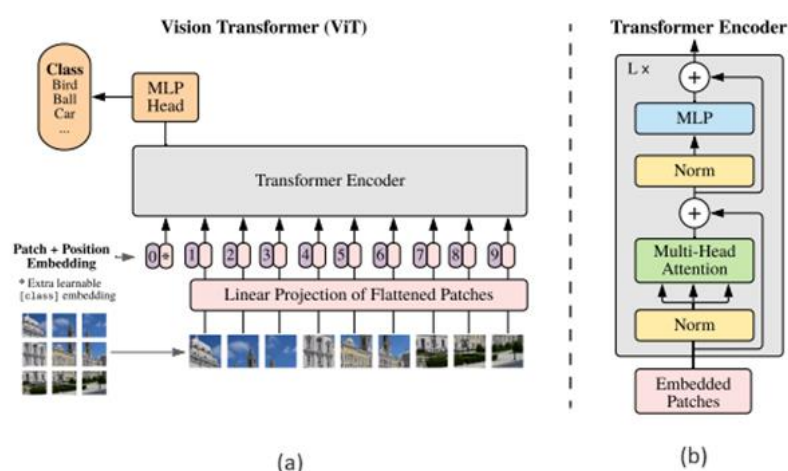


圖 2-19 ViT 模型架構[23]; (a) ViT 運作流程 (b) Transformer Encoder 內部架構

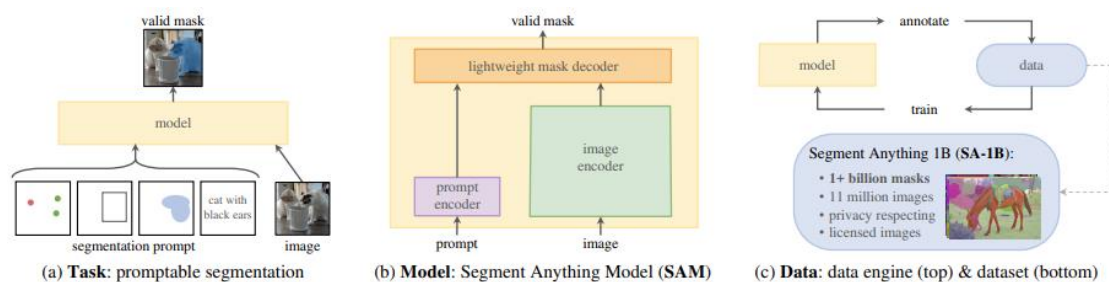


圖 2-20 SAM 模型概述[8]; (a) 可提示分割。(b) Segment Anything 模型 (SAM)。(c) 數據引擎 (上方) 和數據集 (下方)。



### 第三章 人類大腦學習架構探討

本章節會探討人類大腦在學習新物件中主要部位的功能，以及在設計架構上要如何結合檢測神經網路、分類神經網路以及自注意力機制，以模擬大腦學習理解的機制。

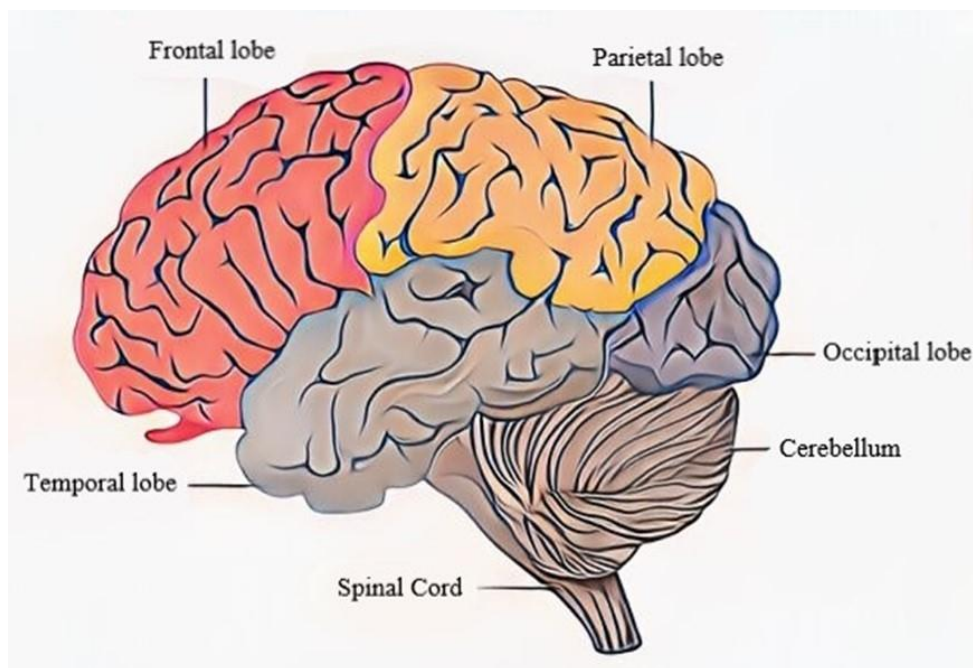


圖 3-1 大腦結構圖

#### 3.1 大腦學習機制

在大腦學習的過程，大腦的各部位都各司其職，在學習新物件時，又以枕葉及海馬體這兩個部位為主要的圖像學習要點，以下會介紹這兩個部位在學習物件時，對於物件的記憶、形體的處理流程。

### 3.1.1 原有知識

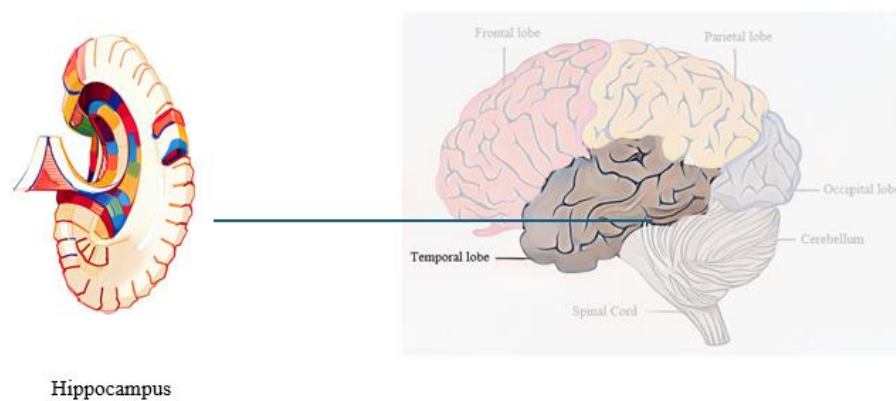


圖 3-2 大腦中儲存原有知識的區塊

顳葉(Temporal Lobe)的工作中包括了聽覺處理、語言理解以及情感處理之外，海馬體（Hippocampus）是位於大腦顳葉內的一個重要結構，形狀像一隻海馬，故而得名。海馬體在許多關鍵的認知和情感功能中發揮著重要作用，特別是與記憶和空間導航相關。也是人類存放語意長期記憶的地方。

### 3.1.2 識別與分類區

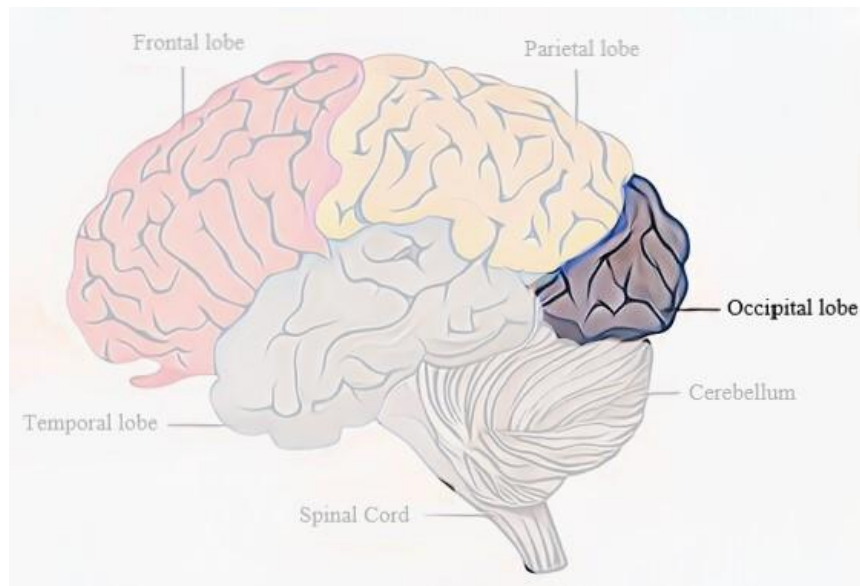


圖 3-3 大腦中處理物件識別之區塊

枕葉 (Occipital Lobe) 主要負責處理視覺的訊息，初級視覺皮質接收來自眼睛的視覺信號，進行基本的視覺處理和解釋。

### 3.1.3 學習新知識區

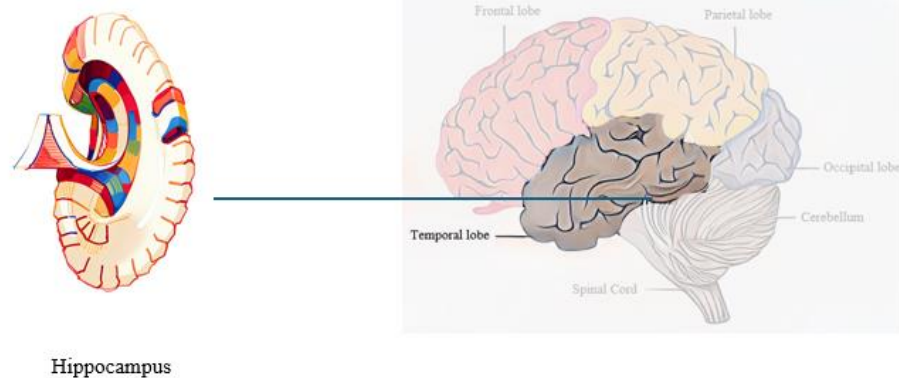


圖 3-4 大腦中新知識形成之區塊

海馬體在將短期記憶轉化為長期記憶的過程中起到至關重要的作用。當我們學習新的訊息或經歷新的事件時，海馬體幫助將這些訊息編碼並存儲在大腦的其他區域，如圖 4。

海馬體具有高度的可塑性，能夠在學習和記憶過程中發生結構和功能上的變化。這種可塑性對於學習新技能和適應新環境非常重要。

## 3.2 神經網路架構

以下對提出的模型架構進行解釋，主要分為存放原有知識的知識區、對於新物件在圖像中的位置判斷以及根據新物件特徵分類的常識區和學習新物件後驗證是否學習正確的新知識的交叉驗證區。架構如圖 3-5。

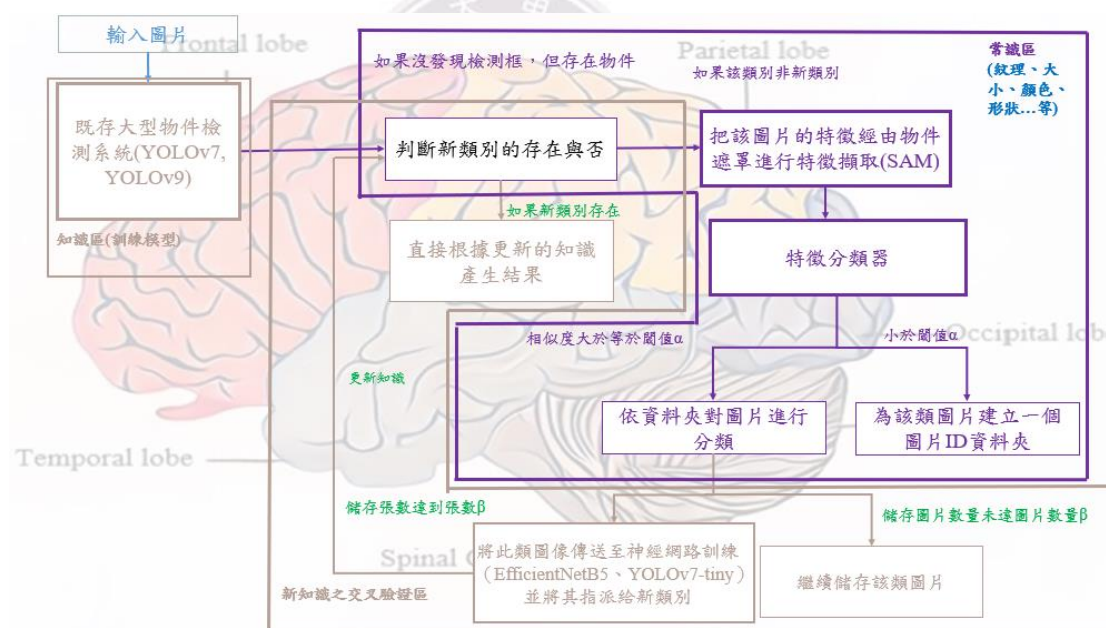


圖 3-5 模擬大腦學習過程及理解之架構

### 3.2.1 知識區

檢測神經網路 YOLOv9 模型對應大腦中的海馬體，存放著原有的記憶，可以知道該物件的類別以及物件在圖像中的位置

所在，在此架構會先建構 10 種類別的圖像的 YOLOv9 自定義模型當作 Pre-train model，在架構中將這 10 類歸類為舊類別。

### 3.2.2 常識區

如沒有被 YOLOv9 檢測到的圖像，會被歸類成未學習過的圖像，首先會判斷此新類別有沒有被學習過，如果沒有，會先經過圖像的紋理、大小、顏色和形狀...等對該圖像進行裁剪處理，這裡對應了枕葉。處理完之後按照特徵分類器將未學習過的圖像分門別類，這裡的處理表示模擬大腦除了知道學習過的類別有哪些，也會知道未學習過的又有哪幾類。分完類之後如果某些類未被檢測的張數到達閥值時，就對該類進行分類神經網路的訓練，並賦予一個新類別名稱且生成權重，且將此類別歸類為新類別，以便往後圖片輸入可以判斷新類別的存在與否，此步驟也對應了海馬體學習新知識的功能。

### 3.2.3 新知識之交叉驗證區

得到新類別後，會再以 YOLOv7-tiny 輕量級檢測神經網路訓練以及 efficientnetB5 訓練這些新類別，學習新類別的圖片資料集，之後會檢測學習新類別圖片和除了舊類別及新類別之外的圖片，來驗證新學到的類別是否能被交叉驗證正確，又能驗證除了舊類別及新類別之外的圖片是否能被辨識正確，如圖 3-6。



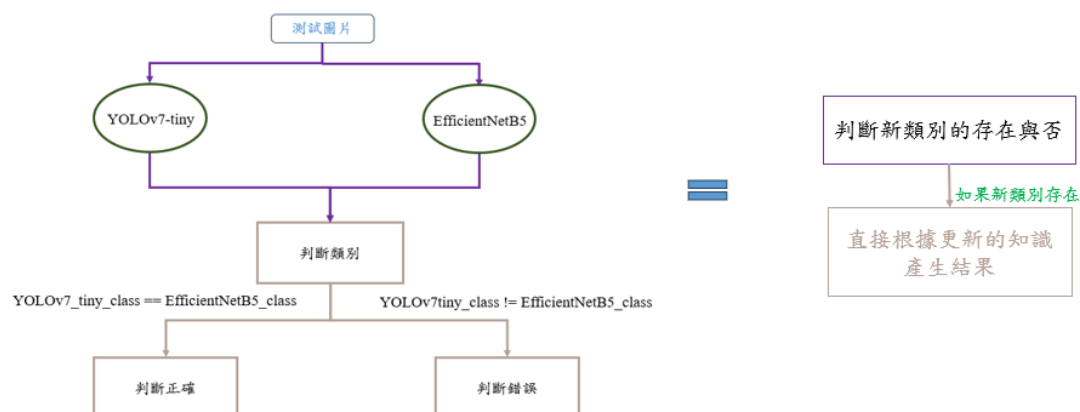


圖 3-6 模擬大腦學習結果驗證之架構

### 3.3 架構參數配置

以下對在模擬大腦學習新物件的機制架構中有使用到的神經網路，設定超參數以進行神經網路的建置。

#### 3.3.1 YOLOv9

YOLOv9 自定義模型參數配置，以  $\text{epochs} = 50$ ，權重為 YOLOv9 之原始權重 `yolov9.pt`，且 `data` 與 `cfg` 設定檔設定成自定義類別以及類別數，再搭配以下默認超參數進行建構自定義模型，如表 3-1。

表 3-1 YOLOv9 訓練之超參數

超參數	值	說明
<code>lr0</code>	0.01	初始學習率 (SGD=1E-2, Adam=1E-3)
<code>lrf</code>	0.01	最終 OneCycleLR 學習率 ( $\text{lr0} * \text{lrf}$ )
<code>momentum</code>	0.937	SGD 動量/Adam beta1
<code>weight_decay</code>	0.0005	優化器權重衰減 5e-4
<code>warmup_epochs</code>	3	熱身輪數 (可以是小數)
<code>warmup_momentum</code>	0.8	熱身初始動量



warmup_bias_lr	0.1	熱身初始偏差學習率
box	7.5	框損失增益
cls	0.5	分類損失增益
cls_pw	1	分類 BCELoss 正權重
obj	1	物體損失增益 (隨標籤縮放)
obj_pw	1	物體 BCELoss 正權重
iou_t	0.2	IoU 訓練閾值
anchor_t	5	錨點-多重閾值
# anchors	3	每個輸出層的錨點數 (0 表示忽略)
fl_gamma	0	focal loss gamma
hsv_h	0.015	圖像 HSV-色相 增強 (比例)
hsv_s	0.7	圖像 HSV-飽和度 增強 (比例)
hsv_v	0.4	圖像 HSV-明亮度 增強 (比例)
degrees	0	圖像旋轉 (+/- 度數)
translate	0.1	圖像平移 (+/- 比例)
scale	0.9	圖像縮放 (+/- 增益)
shear	0	圖像剪切 (+/- 度數)
perspective	0	圖像透視 (+/- 比例), 範圍 0-0.001
flipud	0	圖像上下翻轉 (機率)
fliplr	0.5	圖像左右翻轉 (機率)
mosaic	1	圖像馬賽克 (機率)
mixup	0.15	圖像混合 (機率)
copy_paste	0.3	分割 copy-paste (機率)

### 3.3.2 YOLOv7

YOLO7 自定義模型參數配置，以 epochs = 50，且權重為 YOLOv7 之原始權重 yolov7.pt，且 data 與 cfg 設定檔設定成自定義類別以及類別數，再搭配以下默認超參數進行建構自定義模型，如表 3-2。

表 3-2 YOLOv7 訓練之超參數

超參數	值	說明
lr0	0.01	初始學習率 (SGD=1E-2, Adam=1E-3)
lrf	0.01	最終 OneCycleLR 學習率 (lr0 * lrf)
momentum	0.937	SGD 動量/Adam beta1
weight_decay	0.0005	優化器權重衰減 5e-4
warmup_epochs	3	熱身輪數 (可以是小數)
warmup_momentum	0.8	熱身初始動量
warmup_bias_lr	0.1	熱身初始偏差學習率
box	0.05	框損失增益
cls	0.3	分類損失增益
cls_pw	1	分類 BCELoss 正權重
obj	0.7	物體損失增益 (隨標籤縮放)
obj_pw	1	物體 BCELoss 正權重
iou_t	0.2	IoU 訓練閾值
anchor_t	4	錨點-多重閾值
fl_gamma	0	focal loss gamma
hsv_h	0.015	圖像 HSV-色相 增強 (比例)
hsv_s	0.7	圖像 HSV-飽和度 增強 (比例)
hsv_v	0.4	圖像 HSV-明亮度 增強 (比例)

degrees	0	圖像旋轉 (+/- 度數)
translate	0.2	圖像平移 (+/- 比例)
scale	0.9	圖像縮放 (+/- 增益)
shear	0	圖像剪切 (+/- 度數)
perspective	0	圖像透視 (+/- 比例), 範圍 0-0.001
flipud	0	圖像上下翻轉 (機率)
fliplr	0.5	圖像左右翻轉 (機率)
mosaic	1	圖像馬賽克 (機率)
mixup	0.15	圖像混合 (機率)
copy_paste	0	分割 copy paste (機率)
paste_in	0.15	圖像 copy paste (機率), 訓練更快時設為 0
loss_ota	1	使用 ComputeLossOTA 訓練更快時設為 0

### 3.3.3 YOLOv7-tiny

YOLO7-tiny 自定義新類別模型參數配置，做為新類別之檢測神經網路，以 epochs = 100，且權重為 YOLOv7-tiny 之原始權重 yolov7-tiny.pt，且 data 與 cfg 設定檔設定成自定義新類別以及新類別數，再搭配以下默認超參數進行建構自定義新類別模型，如表 3-3。

表 3-3 YOLOv7-tiny 訓練之超參數

超參數	值	說明
lr0	0.01	初始學習率 (SGD=1E-2, Adam=1E-3)
lrf	0.01	最終 OneCycleLR 學習率 ( $lr0 * lrf$ )
momentum	0.937	SGD 動量/Adam beta1
weight_decay	0.0005	優化器權重衰減 $5e-4$
warmup_epochs	3	熱身輪數 (可以是小數)
warmup_momentum	0.8	熱身初始動量
warmup_bias_lr	0.1	熱身初始偏差學習率
box	0.05	框損失增益
cls	0.5	分類損失增益
cls_pw	1	分類 BCELoss 正權重
obj	1	物體損失增益 (隨像素縮放)
obj_pw	1	物體 BCELoss 正權重
iou_t	0.2	IoU 訓練閾值
anchor_t	4	錨點-多重閾值
fl_gamma	0	focal loss gamma
hsv_h	0.015	圖像 HSV-色相 增強 (比例)
hsv_s	0.7	圖像 HSV-飽和度 增強 (比例)
hsv_v	0.4	圖像 HSV-明度 增強 (比例)
degrees	0	圖像旋轉 (+/- 度數)
translate	0.1	圖像平移 (+/- 比例)
scale	0.5	圖像縮放 (+/- 增益)
shear	0	圖像剪切 (+/- 度數)
perspective	0	圖像透視 (+/- 比例), 範圍 0-0.001

flipud	0	圖像上下翻轉 (機率)
fliplr	0.5	圖像左右翻轉 (機率)
mosaic	1	圖像馬賽克 (機率)
mixup	0.5	圖像混合 (機率)
copy_paste	0	圖像 copy paste (機率)
paste_in	0.05	圖像 copy paste 機率, 訓練更快請設為 0
loss_ota	1	使用 ComputeLossOTA: 訓練更快請設為 0

### 3.3.4 EfficientNetB5

EfficientNetB5 做為新類別之分類神經網路，在新知識交叉驗證中，會與 YOLOv7-tiny 自定義新類別模型進行雙重驗證，判斷新類別是否辨識正確。訓練的超參數如表 3-4。

表 3-4 EfficientNetB5 訓練之超參數

超參數	值	說明
img_size	224	圖片的大小
image_threshold	200	選擇資料集來的圖片數量門檻
batch_size	32	每次訓練的批次大小
rotation_range	40	圖片旋轉的角度範圍
zoom_range	0.3	圖片縮放的範圍
width_shift_range	0.2	圖片水平移動的範圍
height_shift_range	0.2	圖片垂直移動的範圍



shear_range	0.2	圖片剪切變換的範圍
horizontal_flip	TRUE	圖片水平翻轉
fill_mode	'nearest'	填充模式
input_shape	(224, 224, 3)	輸入圖片的形狀
optimizer	Adam(lr=0.001)	優化器及其學習率
loss	tf.keras.losses.SparseCategoricalCrossentropy()	損失函數
monitor	'val_loss'	監控的指標
factor	0.2	學習率縮減因子
patience (ReduceLROnPlateau)	3	容忍步數
patience (EarlyStopping)	7	容忍步數
min_lr	1.00E-06	最小學習率
epochs	10	訓練的總迭代次數

### 3.3.5 SAM

根據 SAM 對於圖像中偵測到的每個物件進行遮罩，如圖 3-4，此架構所需要的圖像中的目標遮罩，也就是重點部分，取到目標遮罩範圍後依據此範圍對原圖裁剪，使用超參數如表 3-5。

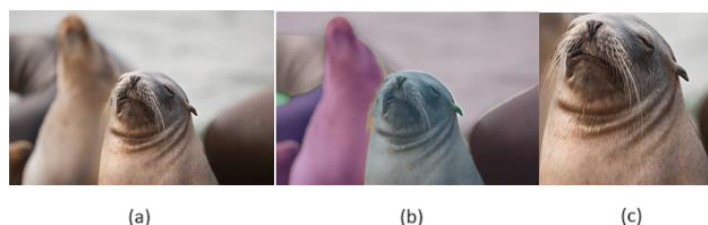


圖 3-7 原圖與 SAM 對圖像偵測到的物件遮罩；(a) 輸入原圖。(b) SAM 所畫出的所有遮罩。(c) 只取最重點遮罩並以原圖裁剪。

表 3-5 SAM 生成遮罩之超參數

超參數	值	說明
model	Sam	用於遮罩預測的 SAM 模型。
points_per_side	32	設定沿圖像一側抽樣的點數。
points_per_batch	64	設定模型同時運行的點數。
pred_iou_thresh	0.88	使用模型預測的遮罩質量進行過濾的閾值，範圍在 [0,1] 之間。
stability_score_thresh	0.95	使用在更改二值化模型遮罩預測的截止值時的遮罩穩定性進行過濾的閾值，範圍在 [0,1] 之間。
stability_score_offset	1	計算穩定性分數時偏移截止值的量。
box_nms_thresh	0.7	非最大抑制（NMS）用於過濾重複遮罩的框 IoU 截止值。
crop_n_layers	0	如果 >0，則會在圖像的裁剪部分上重新運行遮罩預測。
crop_nms_thresh	0.7	非最大抑制（NMS）用於過濾不同裁剪之間重複遮罩的框 IoU 截止值。
crop_overlap_ratio	512 / 1500	設定裁剪重疊的程度。

crop_n_points_downscale_factor	1	在第 n 層抽樣的一側的點數 按 crop_n_points_downscale_factor**n 縮放。
point_grids	None	用於抽樣的點的顯式網格列表，標準化到 [0,1] 之間。
min_mask_region_area	0	如果 >0，將應用後處理以移除面積小於 min_mask_region_area 的遮罩中的斷開區域和孔洞。需要 opencv。
output_mode	"binary_mask"	遮罩返回的形式。可以是 'binary_mask', 'uncompressed_rle' 或 'coco_rle'。

## 第四章 資料準備和分配及模型架構實施

此章節說明資料集來源，資料集張數，以及圖片張數對於每個階段的分配。

### 4.1 圖像資料準備

圖像來自於 pixapay、unsplash...等免費圖庫網站。下載了 12500 張總共 20 類圖片作為模型的數據及測試。

### 4.2 圖像資料分配

在模擬大腦學習過程中對新物件的理解機制，針對第三章第二節的三種階段，分配圖片的應用張數。

#### 4.2.1 YOLOv7、YOLOv9訓練資料集—知識區

設定原有的知識為 10 類圖片，每一類有 500 張圖片，如表 4-1，將每一類的張數以 8:2 的比例分成 train data 以及 test data，以建構自定義 10 類類別模型。

表 4-1 YOLOv7、YOLOv9 訓練自定義模型使用資料

類別	張數
Baseball	500
Bee	500
Chair	500
Cup	500
Glasses	500
Mask	500
Rabbit	500
Screen	500
T-shirt	500
Strawberry	500

### 4.2.2 自定義模型檢測資料集—常識區

在此會將檢測資料集分為兩種不同學習階段來分類。分別為舊類別：YOLOv7、YOLOv9 自定義模型訓練的 10 個類別，如表 4-2；欲學習類別：沒有在舊類別範圍，會被 EfficientNetB5 學習，還有被 YOLOv7-tiny 訓練的類別，如表 4-3。

表 4-2 YOLOv7、YOLOv9 自定義模型檢測舊類別資料分配

舊類別	張數
Baseball	400
Bee	400
Chair	400
Cup	400
Glasses	400
Mask	400
Rabbit	400
Screen	400
T-shirt	400
Strawberry	400

表 4-3 YOLOv7、YOLOv9 自定義模型檢測欲學習類別資料分配

欲學習類別	張數
Backpack	500
Chow	500
Pencil	500
Sealion	500
Sheep	500



### 4.2.3 新學習類別交叉驗證資料集—新類別之交叉驗證

在此會將檢測資料集分為兩種不同學習階段來分類。分別為新類別：欲學習類別中被 EfficientNetB5 及 YOLOv7-tiny 訓練過之類別，如表 4-4；未學習類別：沒有在舊類別範圍以及欲學習類別之圖片，代表大腦是在訓練出新類別後是第一次見到的類別，如表 4-5。

表 4-4 EfficientNetB5、YOLOv7-tiny 交叉驗證新學習類別資料分配

新類別	張數
Backpack	100
Chow	100
Pencil	100
Sealion	100
Sheep	100

表 4-5 EfficientNetB5、YOLOv7-tiny 交叉驗證未學習類別資料分配

未學習類別	張數
Bat	100
Clock	100
Guitar	100
Hedgehog	100
Tabby	100

## 4.3 模型架構實施

在此會依照第四章第一節的圖片分配，再依以下不同的分類神經網路和檢測神經網路去實施模型之架構。

### 4.3.1 知識區實施

YOLOv7: 依照每類 500 張，每類 400 張為 Train data，100 張為 Test data。在圖 4-1 可以觀察到最低的合理之 prediction 為 0.98，但在訓練成果各項指標的起伏是大的，如圖 4-2。

YOLOv9: 依照每類 500 張，每類 400 張為 Train data，100 張為 Test data。在圖 4-3 可以觀察平均表現比 YOLOv7 好。在平均精度 mean Average Precision (mAP) 上也有不錯的表現，如圖 4-4。

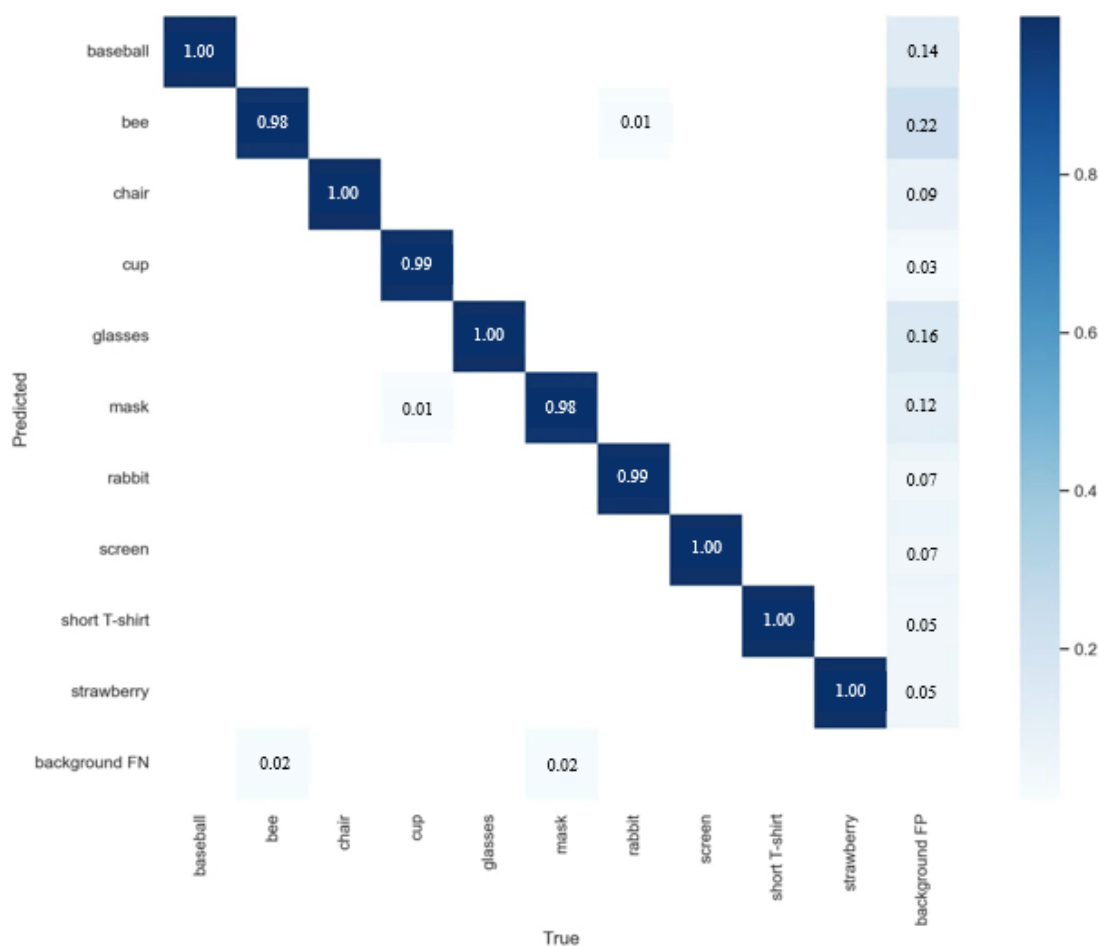


圖 4-1 知識區為 YOLOv7 訓練自定義模型之混淆矩陣

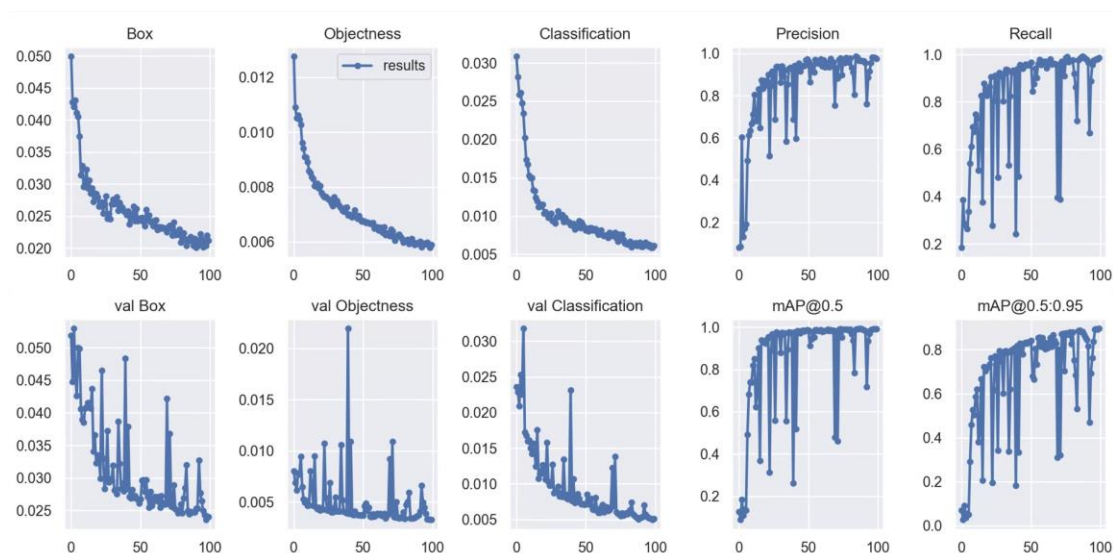


圖 4-2 知識區為 YOLOv7 訓練自定義模型性能指標圖

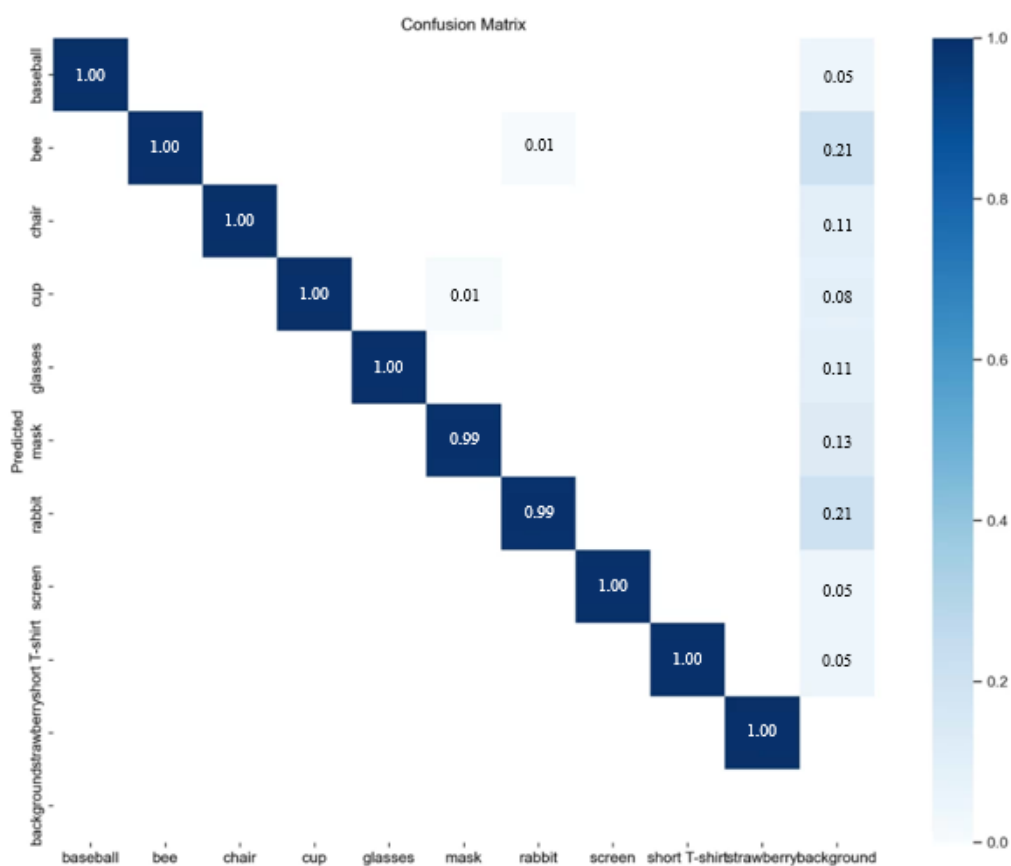


圖 4-3 知識區為 YOLOv9 訓練自定義模型之混淆矩陣

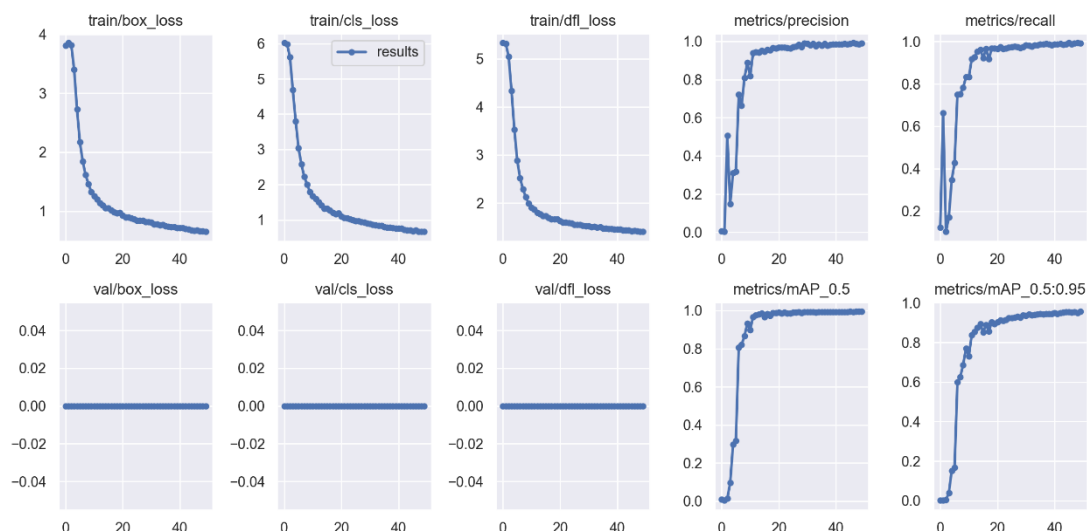


圖 4-4 知識區為 YOLOv9 訓練自定義模型性能指標圖

以檢測信心閾值 0.8 當作檢測的門檻下，YOLOv7 在檢測舊類別的檢測率比 YOLOv9 低了 12%，其中又以 Bee 類最低，與 YOLOv9 最低的 Mask 類相比還低了 33%，如表 4-6 和表 4-7。

表 4-6 知識區為 YOLOv7 自定義模型檢測舊類別之效能

舊類別種類	成功檢測數	總張數	檢測率
Baseball	375	400	0.9375
Bee	234	400	<b>0.5850</b>
Chair	353	400	0.8825
Cup	380	400	0.9500
Glasses	385	400	0.9625
Mask	257	400	0.6425
Rabbit	288	400	0.7200
Screen	386	400	0.9650
T-shirt	391	400	0.9975
Strawberry	346	400	0.9775
Total	3395	4000	0.8488

表 4-7 知識區為 YOLOv9 自定義模型檢測舊類別之效能

舊類別種類	成功檢測數	總張數	檢測率
Baseball	397	400	0.9925
Bee	371	400	0.9275
Chair	396	400	0.9900
Cup	394	400	0.9850
Glasses	396	400	0.9900
Mask	368	400	<b>0.9200</b>
Rabbit	387	400	0.9675
Screen	397	400	0.9925
T-shirt	399	400	0.9975
Strawberry	400	400	1.0000
Total	3905	4000	0.9763

### 4.3.2 常識區實施

與知識區相反的是，在檢測欲學習類別方面，知識區為 YOLOv7 自定義模型在未被檢測率比知識區為 YOLOv9 自定義模型高出 24%，如表 4-8 和表 4-9。

表 4-8 知識區為 YOLOv7 自定義模型檢測欲學習類別之效能

欲學習類別	未被檢測張數	張數	未被檢測率
Backpack	472	500	0.9440
Chow	456	500	0.9120
Pencil	488	500	0.9760
Sealion	470	500	0.9400
Sheep	471	500	0.9420
Total	2357	2500	0.9428



表 4-9 知識區為 YOLOv9 自定義模型檢測欲學習類別之效能

欲學習類別	未被檢測張數	張數	未被檢測率
Backpack	313	500	0.6260
Chow	301	500	0.6020
Pencil	467	500	0.9340
Sealion	353	500	0.7060
Sheep	314	500	0.6280
Total	1748	2500	0.6992

在知識區的過濾下，會對欲學習類別進行 SAM 的特徵裁剪，裁剪後再對這些特徵裁剪完的圖片進行餘弦分類及 KL 散度分類的比較。SAM 的特徵裁剪會分別依據四種指標裁剪如表 4-10，並進行以下指標裁剪下的分類率比較。

表 4-10 SAM 特徵裁剪參考指標

指標	說明
area	遮罩的像素面積
bbox	遮罩的邊界框，格式為 XYWH
predicted_iou	模型對遮罩質量的預測
crop_box	用於生成此遮罩的圖像裁剪框，格式為 XYWH

在以下指標中，本實驗利用 SAM 圖像最大遮罩、最大預測 IoU 的邊界框及最大預測 IoU 的切割框這三種方式擷取圖像特徵。

1. 最大遮罩 (Biggest mask)：物件在 SAM 圖像處理中遮罩像素面積為最大的。

2. 最大預測 IoU 的邊界框(MaxIoU bbox)：圖像中 IoU 最大值所對應的邊界框，且格式依照最小 x、y 點、寬、高。如圖 4-5 的 B、C、D 框。
3. 最大預測 IoU 的切割框(MaxIoU C\_bbox)：圖像中 IoU 最大值所對應的切割框，在模型中切割框被定義為有包覆其他比它小的邊界框之感興趣區域。且格式依照最小 x、y 點、寬、高，如圖 4-5 中的 A 框。

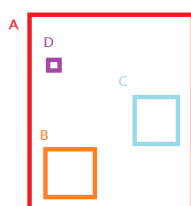


圖 4-5 切割框示意圖

在分類部分，額外分別使用了 EfficientNetB5 搭配餘弦相似度前處理，如圖 4-6，餘弦相似度算法將 A、B 為兩個不同圖片的特徵向量，將 A、B 點積後除以 A、B 特徵向量距離得到兩個特徵向量的相似程度，如式 4-1。以及 EfficientNetB5 搭配 KL 散度前處理進行比較，如圖 4-7，KL 散度將特徵向量轉換成機率分佈  $P(i)$ 、 $Q(i)$ ，其中  $P(i)$  為真實分佈， $Q(i)$  為預測分佈，對  $P$  取  $\log$  避免因為經過轉換成機率分佈所造成的數值下溢問題，如式 4-2。最後根據 imagenet1000 類做分類，可以發現餘弦相似度的方式比 KL 散度的方式讓分類結果更密集，且分類較正確，以下分類都是以 EfficientNetB5 搭配餘弦相似度前處理做為本實驗分類的方法。分類情況如表 4-11 至表 4-18，可以觀察到普遍來說是以知識區為 YOLOv9 自定義模型的分類率比較高，但是以分類總張數來說，都是以知識區 YOLOv7 自定義模型的分類率較多。

$$\text{Cosine\_similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|}$$

式 4-1 餘弦相似度算法

$$D_{KL}(P||Q) = \sum_i P(i) \log \left( \frac{P(i)}{Q(i)} \right)$$

式 4-2 KL 散度算法

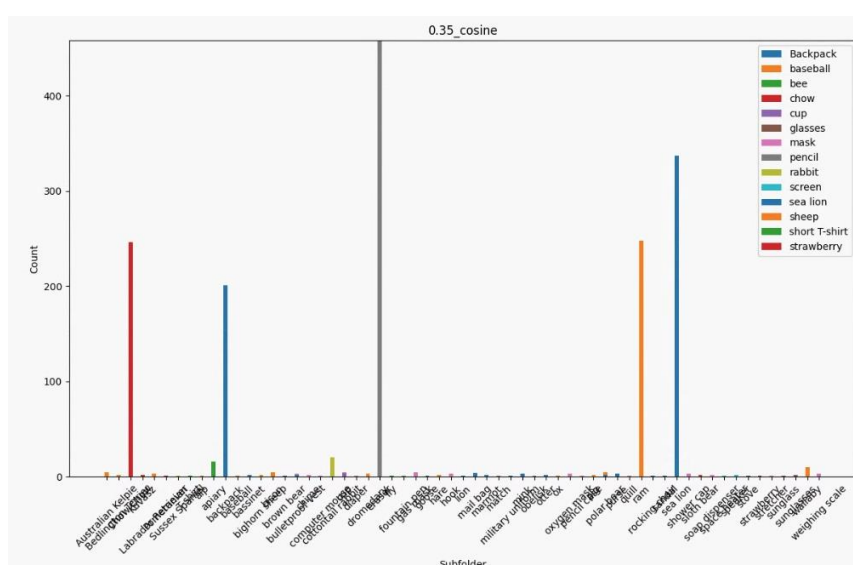


圖 4-6 餘弦相似度分類分佈

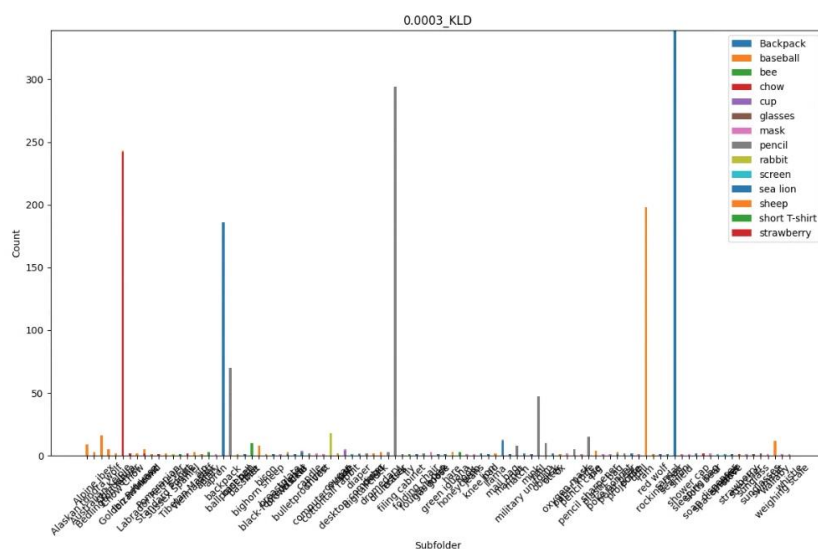


圖 4-7 KL 散度分類分佈

表 4-11 YOLOv7 欲學習類別之原圖分類率

欲學習類別	未被檢測張數	被歸為新類別張數(原圖)	分類率
Backpack	472	450	0.9534
Chow	456	441	0.9671
Pencil	488	484	0.9918
Sealion	470	459	0.9766
Sheep	471	399	0.8471
Total	2357	2233	0.9474

表 4-12 YOLOv9 欲學習類別之原圖分類率

欲學習類別	未被檢測張數	被歸為新類別張數(原圖)	分類率
Backpack	313	289	0.9233
Chow	301	291	0.9668
Pencil	467	464	0.9936
Sealion	353	346	0.9802
Sheep	314	280	0.8917
Total	1748	1670	0.9554

表 4-13 YOLOv7 欲學習類別之 SAM 取最大遮罩分類率

欲學習類別	未被檢測張數	被歸為新類別張數(Biggest mask)	分類率
Backpack	472	444	0.9405
Chow	456	437	0.9583
Pencil	488	484	0.9918
Sealion	470	454	0.9660
Sheep	471	377	0.8004
Total	2357	2196	0.9317

表 4-14 YOLOv9 欲學習類別之 SAM 取最大遮罩分類率

欲學習類別	未被檢測張數	被歸為新類別張數(Biggest mask)	分類率
Backpack	313	281	0.8978
Chow	301	273	0.9070
Pencil	467	464	0.9936
Sealion	353	341	0.9660
Sheep	314	244	0.7771
Total	1748	1603	0.9170

表 4-15 YOLOv7 欲學習類別之 SAM 取最大預測 IoU 邊界框的分類率

欲學習類別	未被檢測張數	被歸為新類別張數(MaxIoU bbox)	分類率
Backpack	472	417	0.8835
Chow	456	434	0.9518
Pencil	488	473	0.9693
Sealion	470	408	0.8681
Sheep	471	316	0.6709
Total	2357	2048	0.8689

表 4-16 YOLOv9 欲學習類別之 SAM 取最大預測 IoU 邊界框的分類率

欲學習類別	未被檢測張數	被歸為新類別張數(MaxIoU bbox)	分類率
Backpack	313	291	0.9297
Chow	301	286	0.9502
Pencil	467	456	0.9764
Sealion	353	295	0.8357
Sheep	314	224	0.7134
Total	1748	1552	0.8878

表 4-17 YOLOv7 欲學習類別之 SAM 取最大預測 IoU 分割框的分類率

欲學習類別	未被檢測張數	被歸為新類別張數(MaxIoU C_bbox)	分類率
Backpack	472	451	0.9555
Chow	456	442	0.9693
Pencil	488	484	0.9918
Sealion	470	461	0.9809
Sheep	471	390	0.8280
Total	2357	2228	0.9453

表 4-18 YOLOv9 欲學習類別之 SAM 取最大預測 IoU 分割框的分類率

欲學習類別	未被檢測張數	被歸為新類別張數(MaxIoU C_bbox)	分類率
Backpack	313	288	0.9201
Chow	301	292	0.9701
Pencil	467	464	0.9936
Sealion	353	346	0.9802
Sheep	314	278	0.8854
Total	1748	1668	0.9542

### 4.3.3 新知識之交叉驗證區實施

欲學習類別在經過常識區分類後，分別用 YOLOv7-tiny 以及 EfficientNetB5 學習這些資料，再以 YOLOv7-tiny 的學習結果與 EfficientNetB5 的學習結果進行交叉驗證。在此實驗中，會分別使用知識區為 YOLOv7 自定義模型之常識區以最大預測 IoU 的切割框分類的圖片再分成 8:2 的比例，以分成 YOLOv7-tiny 與 EfficientnetB5 的訓練集和測試集，如表 4-19 和 4-20，其訓練曲線及性能評估，如圖 4-8 至圖 4-13 及表 4-21 至表 4-22，可以發現在 YOLOv7-tiny 的訓練方面，知識區為 YOLOv7 自定義模型



以及知識區為 YOLOv9 自定義模型的表現是差不多的，在 EfficientNetB5 的訓練成果可以發現在 Backpack、Chow 類都是表現較差的，但是準確率還是有 90% 以上。

表 4-19 知識區為 YOLOv7 自定義模型之常識區分類後訓練數據

新學習類別	常識區分類圖片數	Train data	Test_data
Backpack	451	362	89
Chow	442	348	94
Pencil	484	388	96
Sealion	461	371	90
Sheep	390	313	77
Total	2228	1782	446

表 4-20 知識區為 YOLOv9 自定義模型之常識區分類後訓練數據

新學習類別	常識區分類圖片數	Train data	Test_data
Backpack	288	230	58
Chow	292	233	59
Pencil	464	371	93
Sealion	346	276	70
Sheep	278	222	56
Total	1668	1332	336



圖 4-8 知識區為 YOLOv7 自定義模型之常識區分類後訓練 YOLOv7-tiny 混淆矩陣

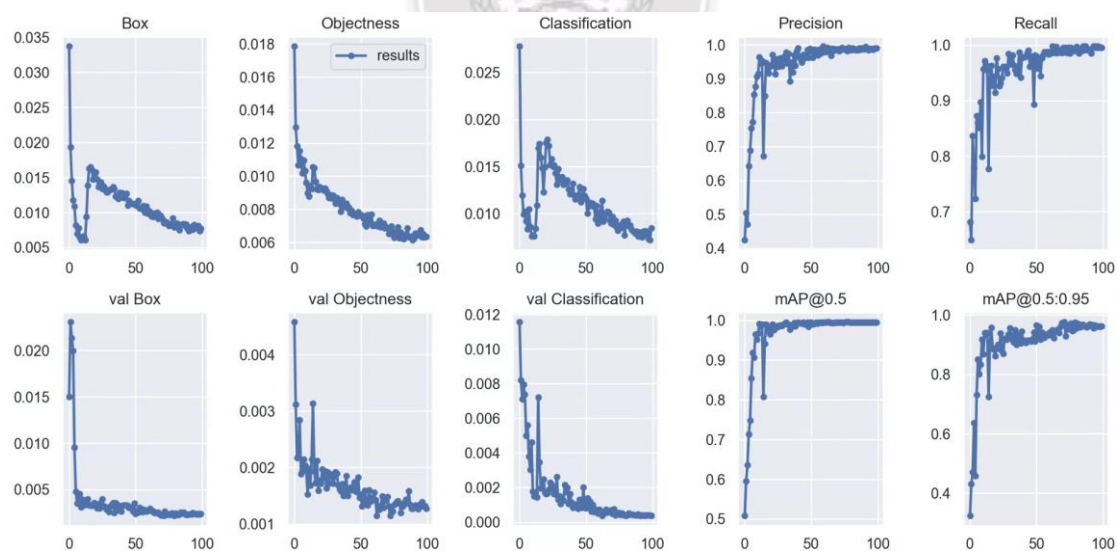


圖 4-9 知識區為 YOLOv7 自定義模型之常識區分類後訓練 YOLOv7-tiny 之性能指標

表 4-21 知識區為 YOLOv7 自定義模型之常識區分類後訓練 EfficientnetB5 分類報告

Class	Precision	Recall	F1-score	Support
class_1	0.989	1	0.995	92
class_2	1	0.989	0.994	89
class_3	1	1	1	97
class_4	1	1	1	93
class_5	1	1	1	78
accuracy	None	None	0.998	449
macro avg	0.998	0.998	0.998	449
weighted avg	0.998	0.998	0.998	449

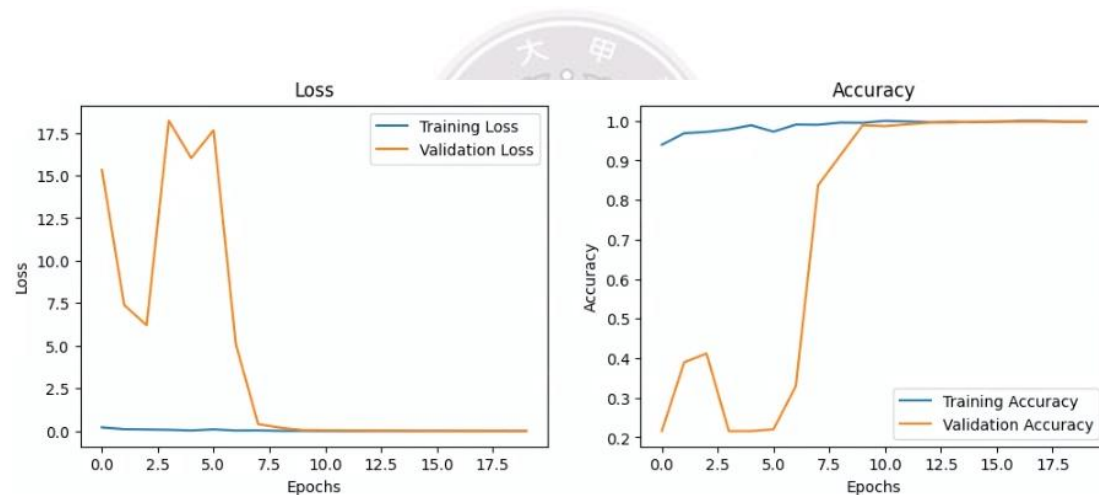


圖 4-10 知識區為 YOLOv7 自定義模型之常識區分類後訓練 EfficientnetB5 訓練走勢

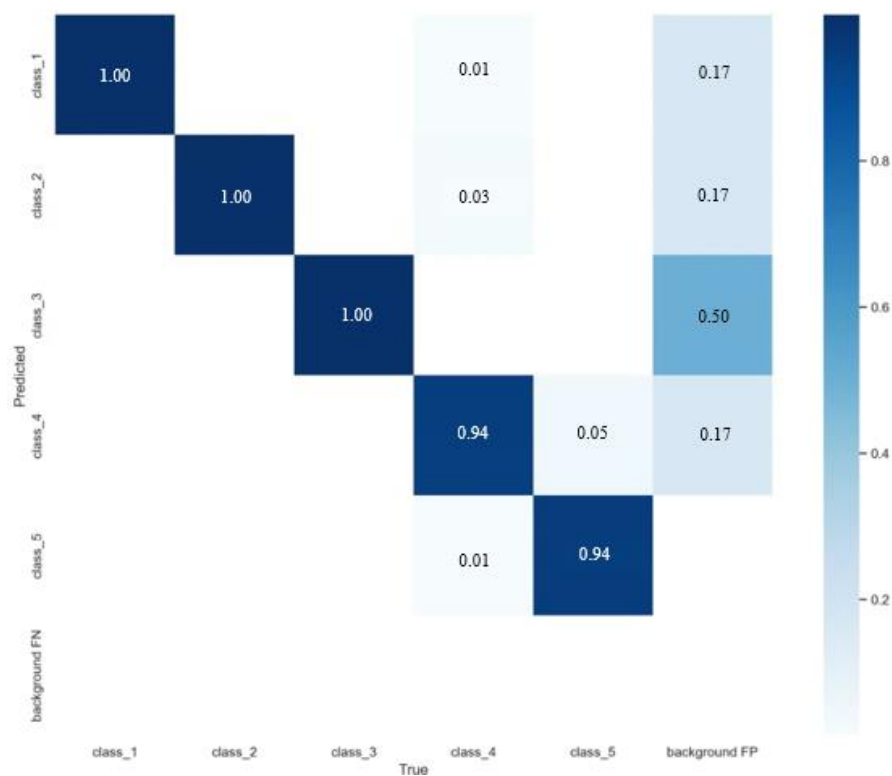


圖 4-11 知識區為 YOLOv9 自定義模型之常識區分類後訓練

YOLOv7-tiny 之混淆矩陣

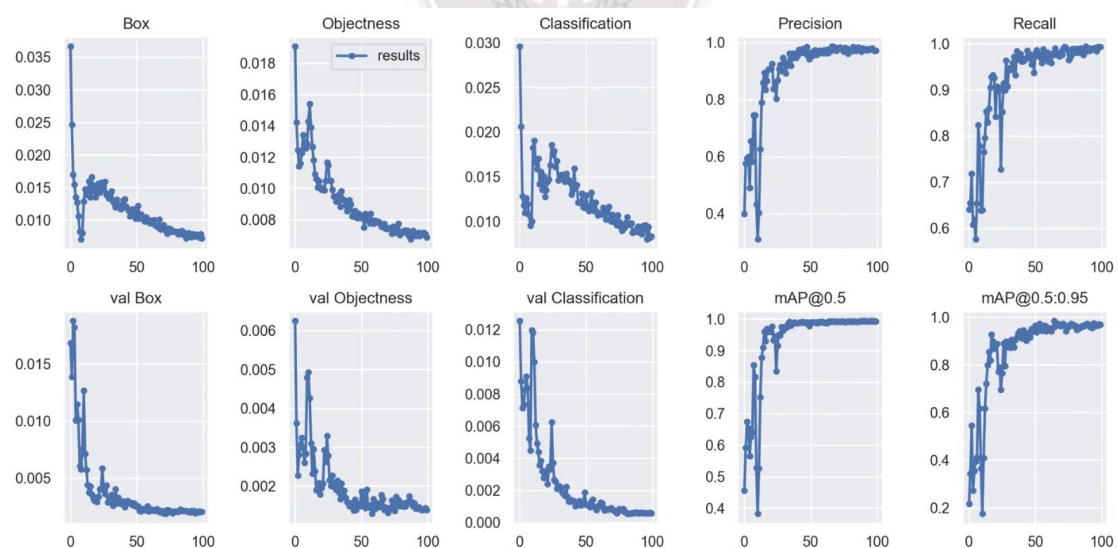


圖 4-12 知識區為 YOLOv9 自定義模型之常識區分類後訓練

YOLOv7-tiny 之性能指標

表 4-22 知識區為 YOLOv9 自定義模型之常識區分類後訓練 EfficientnetB5

## 分類報告

Class	Precision	Recall	F1-score	Support
class_1	0.983	1	0.991	58
class_2	1	0.983	0.991	59
class_3	1	1	1	93
class_4	1	1	1	70
class_5	1	1	1	56
accuracy	None	None	0.997	336
macro avg	0.997	0.997	0.997	336
weighted avg	0.997	0.997	0.997	336

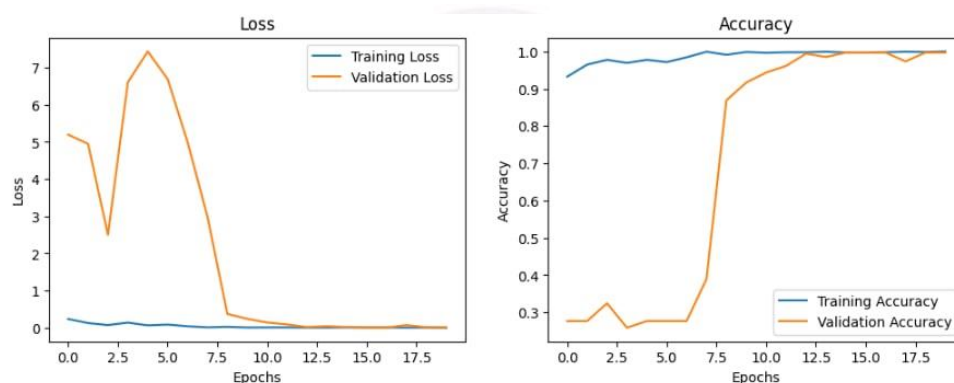


圖 4-13 知識區為 YOLOv9 自定義模型之常識區分類後訓練 EfficientnetB5 訓練走勢

以知識區為 YOLOv7 自定義模型以及知識區為 YOLOv9 自定義模型建構完 YOLOv7-tiny 與 EfficientNetB5 的自定義模型，再以 YOLOv7-tiny 與 EfficientNetB5 的自定義模型對於新學習類別和未學習類別進行交叉驗證如表 4-23 至表 4-26。可以觀察到對於未學習類別來說，因為鉛筆跟球棒都是屬於長條型的相似特徵，所以在球棒方面有嚴重的誤檢情形。交叉驗證時，

YOLOv7-tiny 之信心閾值設為 0.8，EfficientB5 之信心閾值設為 0.5。

表 4-23 知識區為 YOLOv7 所衍生之常識區分類訓練後檢測新學習類別結果

新學習類別	測試張數	被檢測為新類別張數	檢測率
Backpack	100	91	0.91
Chow	100	93	0.93
Pencil	100	78	0.78
Sealion	100	89	0.89
Sheep	100	53	0.53
Total	500	404	0.808

表 4-24 知識區為 YOLOv7 所衍生之常識區分類訓練後檢測未學習類別結果

未學習類別	測試張數	未被檢測為新類別張數	未被檢測率
Bat	100	21	0.21
Guitar	100	75	0.75
Clock	100	91	0.91
Hedgehog	100	98	0.98
Tabby	100	83	0.83
Total	500	369	0.758

表 4-25 知識區為 YOLOv9 所衍生之常識區分類訓練後檢測新學習類別結果

新學習類別	測試張數	被檢測為新類別張數	檢測率
Backpack	100	86	0.86
Chow	100	55	0.55
Pencil	100	86	0.86
Sealion	100	62	0.62
Sheep	100	46	0.46
Total	500	335	0.67



表 4-26 知識區為 YOLOv9 所衍生之常識區分類訓練後檢測未學習類別結果

未學習類別	測試張數	未被檢測為新類別張數	未被檢測率
Bat	100	12	0.12
Guitar	100	98	0.98
Clock	100	96	0.96
Hedgehog	100	100	1.00
Tabby	100	90	0.90
Total	500	396	0.792

## 4.4 模型架構實施結果探討

本節對於第四章第三節的各個區域的實驗結果進行分析及探討。

### 4.4.1 知識區探討

首先在知識區分析本實驗實施 YOLOv7 與 YOLOv9 之差異：使用 YOLOv7 的自定義模型可以發現，檢測舊類別的物件是比 YOLOv9 少很多的，如表 4-27，但在檢測預學習類別時，YOLOv7 未檢測的舊類別物件是比 YOLOv9 還要多的，如表 4-28，這呈現了模型的兩種極端。

表 4-27 知識區模型之舊類別檢測率比較

模型	總張數	成功檢測數	檢測率
YOLOv7 自定義模型	4000	3395	0.8488
YOLOv9 自定義模型	4000	3905	<b>0.9763</b>

表 4-28 知識區模型之新類別未檢測率比較

模型	總張數	未被檢測數	未被檢測率
YOLOv7 自定義模型	2500	2357	<b>0.9428</b>
YOLOv9 自定義模型	2500	1748	0.6992

#### 4.4.2 常識區探討

在常識區中討論 SAM 三種裁剪方式的差異：

1. 最大遮罩：在大部分圖像的重點部位占圖像的50%以上的情況下，有可能直接裁剪圖像的重點部位，但也可能裁剪到背景，因為背景大多數不像切割框一樣內部還有包覆其他邊界框。
2. 取最大 IoU 值的邊界框：因為重點部位可能會被模型判定成本應該是重點部位的局部範圍。例如重點部位應該是狗，但模型判定成是狗的鼻子。
3. 取最大 IoU 值的切割框：在模型中切割框被定義為有包覆其他比它小的邊界框之感興趣區域。因為圖像中的重點部位通常會由好幾個組件組成，取最大 IoU 值的切割框有高機率獲取圖像的重點部位。

根據以上分析經由 SAM 三種指標分別裁剪後與原圖比較的分類準確性：原圖 > 取最大 IoU 值的切割框 > 取最大遮罩 > 取最大 IoU 值的邊界框。不管知識區為 YOLOv7 自定義模型還是 YOLOv9 自定義模型，都是如此，如表 4-29 和 4-30。

表 4-29 知識區為 YOLOv7 自定義模型之常識區分類結果

裁剪方式	未被檢測張數	被分類張數	分類率
原圖	2357	2233	<b>0.9476</b>
最大遮罩	2357	2196	0.9317
最大 IoU 之邊界框	2357	2048	0.8689
最大 IoU 之切割框	2357	2228	0.9453

表 4-30 知識區為 YOLOv9 自定義模型之常識區分類結果

裁剪方式	未被檢測張數	被分類張數	分類率
原圖	1748	1670	<b>0.9554</b>
最大遮罩	1748	1603	0.9170
最大 IoU 之邊界框	1748	1552	0.8878
最大 IoU 之切割框	1748	1668	0.9542

再來列出 KL 散度搭配 EfficientNet 分類的分佈較餘弦搭配 EfficientNet 分類較分散的可能原因：

1. EfficientNet 較適合的可能原因：EfficientNet 模型提取的特徵向量通常是高維度的實數向量，且這些特徵向量都是未正規化的，其中每個維度都對應於圖像在某個特定特徵方向上的強度或權重。在這種情況下，使用餘弦相似度作為相似性度量方法是合適的，因為餘弦相似度專注於向量之間的方向而不是長度，適合用於比較高維度特徵向量的相似性。因此，EfficientNet 模型提取的特徵向量與餘弦相似度的性質相匹配，使得餘弦相似度在與 EfficientNet 結合進行圖像分類時表現出色。
2. KL 散度較不適合的可能原因：第一點有提到 EfficientNet 提取的特徵向量都是未正規化的，但是 KL 散度需要做到正規化的步驟，也就是透過 softmax 將特徵向量轉成機率分佈，再計算成 KL 散度，複雜的計算當然也會將一些資訊流失掉，所以在 EfficientNet 的特徵向量變成 KL 散度會沒有辦法百分之百還原進而比餘弦相似度的做法還要來得不準確。

### 4.4.3 新知識之交叉驗證區探討

從知識區自定義模型，檢測完後經由常識區分類的圖像數量會影響到後續新知識交叉驗證區的建構模型結果，可以觀察到知識區為 YOLOv7 的自定義模型時，在新學習類別之檢測率高於知識區為 YOLOv9 的自定義模型的，而在未學習類別之未被檢測率低於知識區為 YOLOv9 的自定義模型，是因為在訓練 YOLO 的過程中，張數越多就會學到更多的細節特徵而導致誤檢。到如表 4-31 至表 4-33。

表 4-31 知識區自定義模型之交叉驗證區訓練及測試集資料分配

知識區	Train data	Test_data
YOLOv7 自定義模型	<b>1782</b>	<b>446</b>
YOLOv9 自定義模型	1332	336

表 4-32 知識區自定義模型之交叉驗證區新學習類別檢測結果

知識區	新學習類別測試張數	檢測張數	檢測率
YOLOv7 自定義模型	500	404	<b>0.808</b>
YOLOv9 自定義模型	500	335	0.670

表 4-33 知識區自定義模型之交叉驗證區未學習類別檢測結果

知識區	未學習類別測試張數	未檢測張數	未被檢測率
YOLOv7 自定義模型	500	369	0.738
YOLOv9 自定義模型	500	396	<b>0.792</b>

## 第五章 結論與未來方向

本論文使用 YOLO 進行新舊知識的學習以及搭配 SAM 及 EfficientNetB5對於圖像中的未知物件進行重點裁剪處理及將裁剪結果進行未知物件的分類。根據第三章的實驗結果，本論文比較了在不同版本的 YOLO 以及搭配 EfficientNetB5並使用不同 SAM 指標進行特徵處理的效果，並對其進行了深入分析。結果表明，所採用的模型能夠有效區分已知與未知物件，並且對未知物件進行分類的預期目標得以實現。

### 5.1 結論

在建構模擬大腦學習新物件的架構中，我們測試了2500張新學習類別的圖片，結果顯示，其中最多有89%的新學習類別圖片被正確分類並訓練成新類別。這表明模型在新類別學習上的表現相當出色。在測試新類別的500張圖片時，模型的最高正確檢測率達到80%，進一步證明了其在實際應用中的穩定性和可靠性。

### 5.2 未來方向

未來的發展方向可以集中在以下幾個方面：首先，進一步模擬人類大腦的學習機制，以提升 XAI 的性能，例如自動挑選對於實驗樣本最佳的檢測神經網路與分類神經網路。其次，可以實現網路模型的自動調參和結果解釋，使得該架構能夠成為一個具備自我學習能力的模擬人類大腦的架構，實現永無止境的學習與進步。

這不僅促進了技術的發展，也與可持續發展目標（SDGs）中的目標相呼應，如 SDG 9：產業、創新和基礎設施，深度學習

和人工智慧技術的應用能夠提高工業自動化和生產效率。例如，智能檢測系統可以優化生產流程，減少資源浪費，並提高產品質量，從而實現更高效、更環保的工業生產方式。





## 參考文獻

- [1] R. J. Williams, G. E. Hinton, and D. E. Rumelhart, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533-536, Oct. 1986.
- [2] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527-1554, Aug. 2006.
- [3] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” *arXiv preprint arXiv:1905.11946*, 2019.
- [4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS'12)*, vol. 1, pp. 1097–1105, 2012.
- [5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Honolulu, HI, USA, 2017, pp. 2261-2269, Jul. 2017.
- [6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *arXiv preprint arXiv:1506.02640*, 2015.
- [7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *arXiv preprint arXiv:1311.2524*, 2014.
- [8] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W. Y. Lo, P. Dollár, and R. Girshick, “Segment anything,” *arXiv preprint arXiv:2304.02643*, 2023.
- [9] Defense Advanced Research Projects Agency (DARPA), “Explainable artificial intelligence (XAI),” Retrieved from <https://www.darpa.mil/program/explainable-artificial-intelligence>.
- [10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *arXiv preprint arXiv:1512.02325*, 2015.

- [11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” arXiv:1706.03762, 2017.
- [12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” OpenAI, 2018.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv preprint arXiv:1512.03385, 2015.
- [14] J. Dai, Y. Li, K. He, J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks,” arXiv preprint arXiv:1605.06409, 2016.
- [15] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” arXiv preprint arXiv:1506.01497, 2016.
- [16] K. He, G. Gkioxari, P. Dollár, R. Girshick, “Mask R-CNN,” arXiv preprint arXiv:1703.06870, 2017.
- [17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2015.
- [18] C. Y. Wang, A. Bochkovskiy, and H. Y. Mark Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” arXiv preprint arXiv:2207.02696, 2022.
- [19] C. Y. Wang, I. H. Yeh, and H. Y. Mark Liao, “YOLOv9: Learning what you want to learn using programmable gradient information,” arXiv preprint arXiv:2402.13616, Feb. 2024.
- [20] C. Y. Wang, H. Y. Mark 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, 2019.
- [21] C. Y. Wang, H. Y. Mark Liao, and I. H. Yeh, “Designing network design strategies through gradient path analysis,” arXiv preprint arXiv:2211.04800, 2022.
- [22] O. Christopher, “Understanding LSTM networks,” Retrieved from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>, Aug. 2015.

[23] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterhiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv preprint arXiv:2010.11929, 2020.

