朝陽科技大學
資訊管理系

碩士論文

結合 YOLO 與 U-net 在行人偵測的應用研究

Application research of combining YOLO and U-net in pedestrian detection

指導教授：陳榮靜 博士

研究生：吳芷苓

中華民國 112 年 7 月

朝陽科技大學資訊管理系

Department of Information Management

Chaoyang University of Technology

碩士論文

Thesis for the Degree of Master

結合 YOLO 與 U-net 在行人偵測的應用研究

Application research of combining YOLO and U-net in pedestrian detection

指導教授：陳榮靜 博士 (Rung-Ching Chen)

研究生：吳芷苓 (Chih-Ling Wu)

中華民國 112 年 7 月

July, 2023

摘要

台灣時常會發生交通事故，主要原因包括不太適宜的道路設計、刑罰不足，以及少數民眾的素質較低有關，例如，汽機車常常無視紅燈、不禮讓行人、違規停車等，這些問題都是台灣人每天面臨的交通挑戰。隨著科技進步，深度學習技術可以對行人偵測問題進行複雜的運算，做出準確的決策，它在語音識別、農業和生物醫學等領域得到廣泛應用，此外在交通方面也開始結合深度學習技術與行人偵測的研究。

在本文中，我們研究了當前基於神經網路的深度學習偵測技術，提出了一種結合 YOLOv5 與 U-net 的行人偵測方法，第一階段為使用 YOLOv5 模型對行人資料集進行訓練和測試，然後我們將會獲得經過 YOLOv5 模型處理後的行人區域 Bounding Box，這樣一來就可以確認行人在圖像中的精確位置，再使用行人圖像裁切程式碼，就能得到多張經過裁切處理的行人圖像；第二階段則是使用 U-net 模型對行人的裁切圖像進行訓練與測試。結合深度學習的行人偵測方法可以更容易地偵測出行人，未來的研究將基於現有的方法進行改善和擴展，透過深度學習技術的應用，希望能有效解決台灣交通事故的問題。

關鍵詞：行人偵測、語義分割、U-net、YOLO、深度學習

Abstract

Traffic accidents often occur in Taiwan. The main reasons include inappropriate road design, insufficient penalties, and the low quality of the minority people. For example, cars and motorcycles often ignore red lights, fail to yield to pedestrians, and park illegally. Transportation challenges Taiwanese face every day. With the advancement of science and technology, deep learning technology can perform complex calculations on pedestrian detection problems and make accurate decisions. It has found widespread applications in fields such as speech recognition, agriculture, and biomedical research, and has also started to be integrated into traffic-related studies concerning pedestrian detection.

In this paper, we study the current deep learning detection technology based on neural network, and propose a pedestrian detection method combining YOLOv5 and U-net. The first stage is to use the YOLOv5 model to train and test the pedestrian data set. Then we will get the Bounding Box of the pedestrian area processed by the YOLOv5 model, so that we can confirm the precise position of the pedestrian in the image, and then use the pedestrian image cropping code to get multiple cropped images images of pedestrians; The second stage is to use the U-net model to train and test the cropped images of pedestrians. The pedestrian detection method combined with deep learning can detect pedestrians more easily. Future research will improve and expand based on existing methods. Through the application of deep learning technology, it is hoped that the problem of traffic accidents in Taiwan can be effectively solved.

Keywords： Pedestrian Detection, Semantic Segmentation, U-net, YOLO, Deep Learning

誌謝

在完成這份論文的過程中，我想向許多人表達我的真誠感謝。首先，我要感謝我的指導教授，感謝您的專業的指導和悉心的付出，從一開始的論文選題、資料蒐集到完成撰寫論文的過程，您的支持和鼓勵對我在這個研究領域的成長至關重要。其次，我要感謝我的家人和朋友們，感謝你們在我學習的過程中給予的支持和鼓勵，你們的理解和鼓勵是我堅持下去的動力。同時，我還要感謝實驗室的詠存、稟濬、峻瑋和其他同學們，感謝你們在這段期間中所提供的任何幫助，你們的共同努力讓這個研究成為可能。最後，我再一次真誠地向曾經幫助過我的老師、同學和家人表示感謝。

目錄

圖目錄

表目錄

第一章 緒論

在第一章節之中將逐步說明本研究的研究背景、研究動機、研究目的，與論文架構。

1.1 研究背景

台灣從過去到現在一直有道路工程設計不安全的問題 [1]，例如行人穿越道的設計上，路口轉角處的退縮距離不足；夜晚的路燈無法完整照亮斑馬線；車輛的轉彎半徑過大，這些不良設計的設施通常會造成駕駛員不易從視野裡看見被死角擋住的行人，容易導致交通事故的發生。道路設施的設計不良是無法滿足行人需求的主要原因，而且這些不良設計的設施也不容易在短時間內得到改善，這些問題都是需要有專業的工程師和相關專家的參與，做出深入的規劃和審慎的設計才能進行有效的改進，以確保道路設計符合行人和駕駛員在安全性與適用性方面的需求。

除了上述容易造成交通事故發生的原因之外，台灣還存在少數素質不佳的駕駛員，例如台灣報考駕照的費用便宜且門檻低，且與其他國家相比，台灣報考駕照的難度更低、更好考、駕照沒有鑑別度。因此，也更需要積極推動交通安全宣傳和教育，提高大眾對交通安全的意識，減少交通事故的發生。根據 CNN Travel [3]報導指出，在台灣行人在穿越馬路時需要格外的謹慎與小心，因為有些駕駛員都不會停車禮讓行人，極不尊重行人的路權，且更指出台灣的駕駛員會在人行道違規停車的狀況也很常發生。基於上述提到的多項原因，且根據交通部資料如圖 1，台灣歷年來的死亡事故總件數呈現不斷上升的趨勢 [2]，可以認定台灣道路安全確實是 CNN Travel 報導評論的「行人交通地獄」，亦顯示台灣的交通安全規定有必要接軌國際標準、加強落實交通安全與維護行人權益。

近年來，深度學習在各式各樣的圖像切割任務中展現出巨大的潛能，舉

例來說行人偵測 (Pedestrian Detection)目前已成為電腦圖像辨識視覺研究領域的其中一個重要技術，尤其是在汽車、監控、機器人和交通等應用方面中是至關重要的，行人偵測常見的應用領域有：人工智慧先進駕駛輔助系統(AI Advanced Driver Assistance Systems, AIADAS)、智慧型視訊監控系統、智慧型機器人等。行人偵測是應用於車輛駕駛輔助系統的關鍵技術，它能快速且及時的進行安全預警以及避開障礙物，能減少或避免碰撞事故的發生，消除人為失誤或外在因素所造成的碰撞，大大減少交通事故的發生，以及事故的死傷人數。

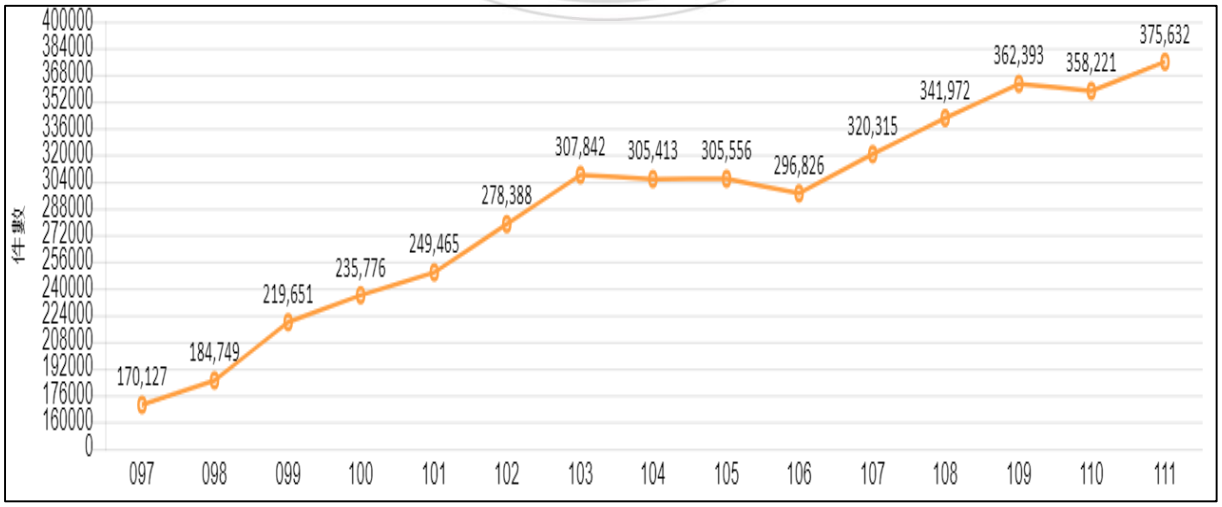

圖 1 全國歷年來死亡事故總件數 [2]

1.2 研究動機

由於台灣目前的道路設計不良狀況不容易在短時間內得到改善，而且這些不良的道路設計還會間接地影響駕駛員，使其難以從視野裡看見被死角擋住的行人，進而容易導致交通事故的發生。因此對於前述之情況，深度學習技術成為一項有助於解決問題的工具，特別是針對道路上的行人進行辨識，可以協助駕駛員判斷可能發生的狀況，進而降低嚴重事故的發生機率。因此，本篇論文針對街景、校園以及馬路等不同場景中的行人偵測問題，提出了一種結合 YOLOv5 以及 U-net 為基礎的行人偵測方法。

1.3 研究目的

在本篇研究中，我們基於多個網路上較為常見的公開行人偵測資料集，來蒐集、擴充行人在街景、校園以及馬路等多個場景中的行為動作圖像，並且透過 YOLOv5 以及 U-net 圖像識別與深度學習方法，來實作出一套在偵測不同場景圖像中行人的方法。

1.4 論文架構

本篇論文將在第一章介紹研究背景、動機與目的；第二章整理相關文獻；第三章為本次研究所使用的研究方法，探討模型的建立以及模型的訓練流程；第四章則是針對本次研究的實驗結果進行分析與討論；最後將在第五章做結論與未來研究展望。

第二章 文獻探討

2.1 類神經網路與卷積神經網路

類神經網路(Neural Network, NN)，可被稱為人工神經網路(Artifical Neural Network, ANN)或神經網路，類神經網路是模仿大腦神經運作的機器學習方式，被普遍用於辨識、決策、控制、預測等工作，如圖 2 所示，神經網路由多個神經元節點互相連結而成，由最左邊的輸入層和最右邊的輸出層，以及中間的隱藏層而組成，每一個節點或神經元皆會連接到另一個節點或神經元，每一個節點都有獨立的權重，神經元會經過隱藏層，並且會受其影響，而它主要功能為傳遞與處理資料，而具有多個隱藏層的神經網路則被稱為深度神經網路 (Deep Neural Network, DNN)。

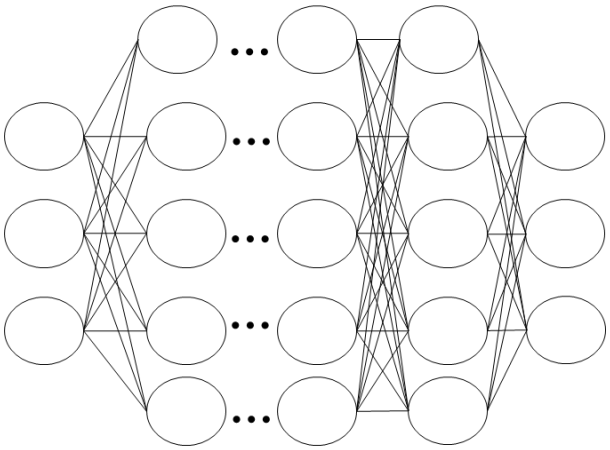

圖 2 深度神經網路 (DNN) 的結構圖

卷積神經網路 (Convolutional Neural Network, CNN)，相較於深度神經網路多了卷積層 (Convolutional Layer) 以及池化層 (Pooling Layer)，最後再以全連接層 (Fully-Connected) 做輸出。上述提到的卷積層以及池化層是負責做圖像的特徵擷取任務，而最後的全連接層則是負責分類的任務。卷積神經網路能被廣泛運用在圖像分類上 (Image Classification)，CNN 較為經典的模型有： LeNet、 AlexNet、VGG、ResNet 等。

2.2 YOLO

目前深度學習的物件偵測方法主要分為兩類：Two Stage 的物件偵測方法，和 One Stage 的物件偵測方法。Two Stage 的物件偵測方法是會先提取候選區域 (Region Proposals)找出物件進行定位，再根據選出的物件去做分類，最後得到的結果，精確度較高、速度相對於 One Stage 慢，Two Stage 算法的代表為 RCNN 系列。YOLO 是目前較為 常用的典型 One Stage 目標偵測網路，One Stage 的算法則是直接通過主幹網路給出類別和位置訊息，這樣的算法速度更快，但是整體辨識精確度相較 Two Stage 目標偵測網路還要略低。

You Only Look Once (YOLO) 最早是由 Joseph Redmon 等人 [4]提出的一種物件偵測演算法。YOLO 的網路架構是參考 GoogleNet，YOLO 最初的網路架構有 24 個卷積層 (Convolutional Layer) 和 2 個全連結層 (Fully Connected Layer)，其他層全部採用 Leaky Relu 激活函數，與 GoogleNet 不同的地方在於 Joseph Redmon [4]提出的卷積層主要使用 1 × 1 的卷積做 Channel Reduction，接著再使用 3 × 3 的卷積，YOLOv1 模型的整體架構如圖 3。圖 4 為 YOLO 的三個重要目標偵測整體流程圖，三個流程的簡要介紹如下：

流程一：將輸入的圖像 Resize 成 448 x 448，並分割成 7 x 7 的網格；

流程二：對圖像執行卷積神經網路；

流程三：通過 NMS (Non-Maximum Suppression)和閾值 (Threshold) 過濾信心程度 (Confidence) 較低的 Bounding Boxes 來得到預測結果。

圖 5 為說明整體 YOLO 模型偵測物件和分類的過程：將輸入的圖像切割成 7 x 7 的網格 (Grid Cell)，接著每個網格會負責偵測 2 個 Bounding Boxes 的預測，以及輸出屬於各個類別 (Class) 的機率預測，最後會進行一遍 NMS 來消除一些重疊的 Bounding Box、信心程度較小的 Bounding Box

，輸出剩下的 Bounding Box 就是選出來的物件結果。

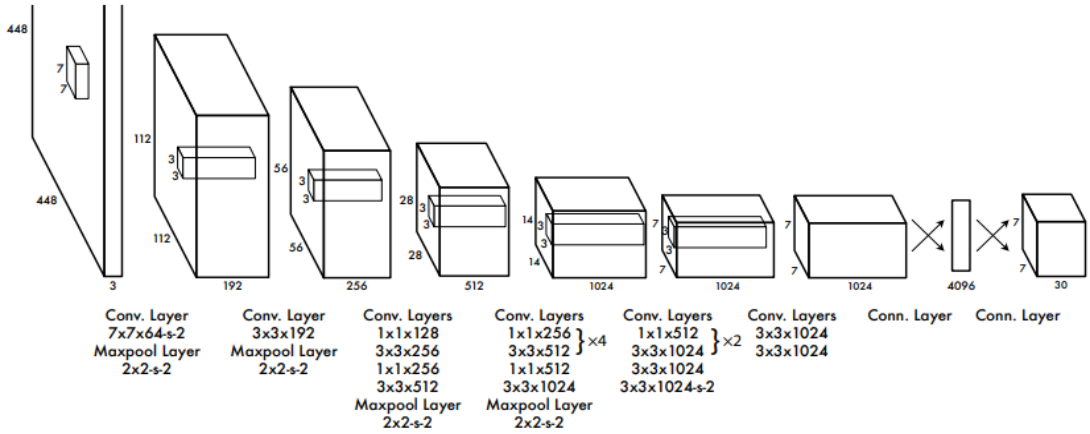

圖 3 YOLOv1 模型的架構圖 [4]

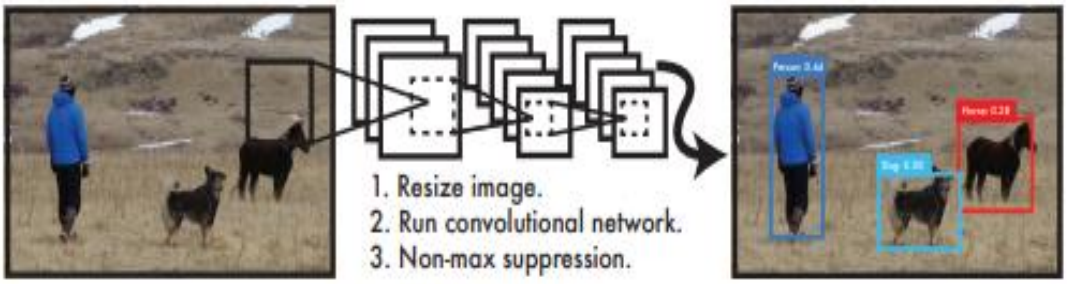

圖 4 YOLO 偵測系統的三個流程圖 [4]

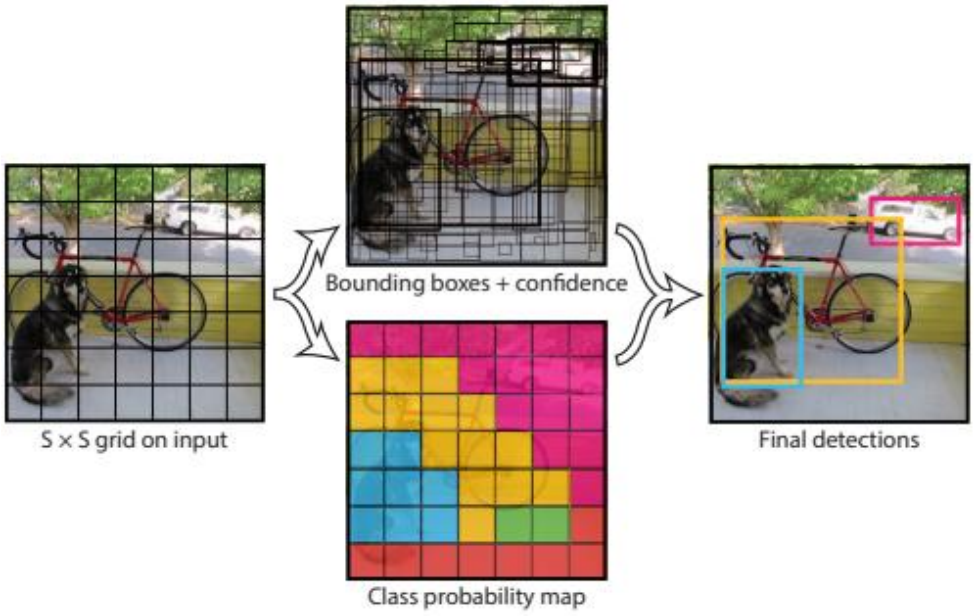

圖 5 整體 YOLO 模型偵測物件和分類的過程圖 [4]

YOLOv5 官方代碼給出的目標偵測的檔案為 Yaml 格式文件，如圖 6，與原本的 YOLOv3、YOLOv4 中呈現的 Cfg 格式不同，YOLOv5 網路模型中一共給出了 YOLOv5s、YOLOv5m、YOLOv5l、YOLOv5x、YOLOv5n 等五種不同的變體網路結構，根據圖 7 作者 [5]提供的 YOLOv5 系列演算法性能測試圖，可以觀察出 YOLOv5s、YOLOv5n 是網路深度最小，速度最快，準確度 (mAP)也最小。YOLOv5m、YOLOv5l、YOLOv5x，在此基礎上，不斷加深加寬網路，準確度 (mAP)也不斷提升，但是目標偵測速度的消耗也在不斷增加。YOLOv5 的模型架構圖 [6]如圖 8 所示分為四個部分，輸入端、Backbone、Neck、Prediction，以下將詳細介紹 YOLOv5 架構的四個部分：

(1) 輸入端

YOLOv5 的輸入端採用了和 YOLOv4 一樣的 Mosaic 資料增強方法，會隨機讀取四張圖像，分別對四張圖像進行隨機縮放、隨機裁剪、隨機排列、色域變化等方法進行拼接。Mosaic 資料增強方法的優點在於：增加資料集的豐富度，和減少訓練模型對 GPU 的負擔。此外，YOLOv5 在輸入端用到的方法與 YOLOv4 不一樣的地方則有：自適應圖像縮放 (統一圖像尺寸)、自適應錨框 (Anchor Size) 的計算。

(2) Backbone

由圖 8 的 YOLOv5 架構圖 [6]可以看到，Backbone 共有四個關鍵 module： Focus module、CBL (conv+BN+Leaky relu) module、Cross Stage Partial (CSP) module 和 Spatial Pyramind Pooling (SPP) module。

在 Backbone 中，輸入的圖像會先通過一個 Focus Module，此階段會先對圖像進行使用切片操作如圖 9，4 × 4 × 3 的圖像經過切片後變成 2

× 2 × 12 的 Feature Map ，舉例而言以 YOLOv5s 的結構為例，若輸入圖像的解析度為 640 × 640 × 3 輸入 Focus 結構，進行切片操作後會先變成 320 × 320 × 12 的 Feature Map，再經過一次 32 個卷積核的捲積操作，最後變成 320 × 320 × 32 的 Feature Map。

Focus Module 後面接著的是 CBL Module，CBL 是一個基本的捲積 module，由 Conv2D、 BatchNormal 和 LeakyRELU 組成。

CBL Module 後面接著的 Module 則是 CSP Module，它能提取 Feature Map 的特徵，提供我們想要的豐富訊息，與其他深度 CNN 相比，CSP 結構減少了優化過程中梯度訊息的重複出現。

SPP Module，能將任意大小的特徵圖轉換成固定大小的特徵向量。

(3) Neck

在 Neck 階段，YOLOv5 採用了 FPN+PAN (Perceptual Adversarial Network) 的結構組合，YOLOv5 與 YOLOv4 的不同之處在於，YOLOv4 使用的是普通的捲積運算，而 YOLOv5 使用了全新設計的 CSP 結構，增強了網路對特徵融合的能力。

(4) Prediction

YOLOv4 是使用 CIOU Loss 作為目標 Bounding Box 的損失函數，而 YOLOv5 中則是採用了 GIOU Loss 做為 Bounding Box 的損失函數。在目標偵測的後處理流程中，針對很多目標框的篩選，通常需要非最大抑制 (Nonmaximum Suppression, NMS) 的操作，YOLOv4 在 DIOU Loss 的基礎上採用 DIOU NMS 的方式，而在 YOLOv5 中仍然採用加權 NMS 的方式，此方法能提高偵測被遮擋目標物件的能力。

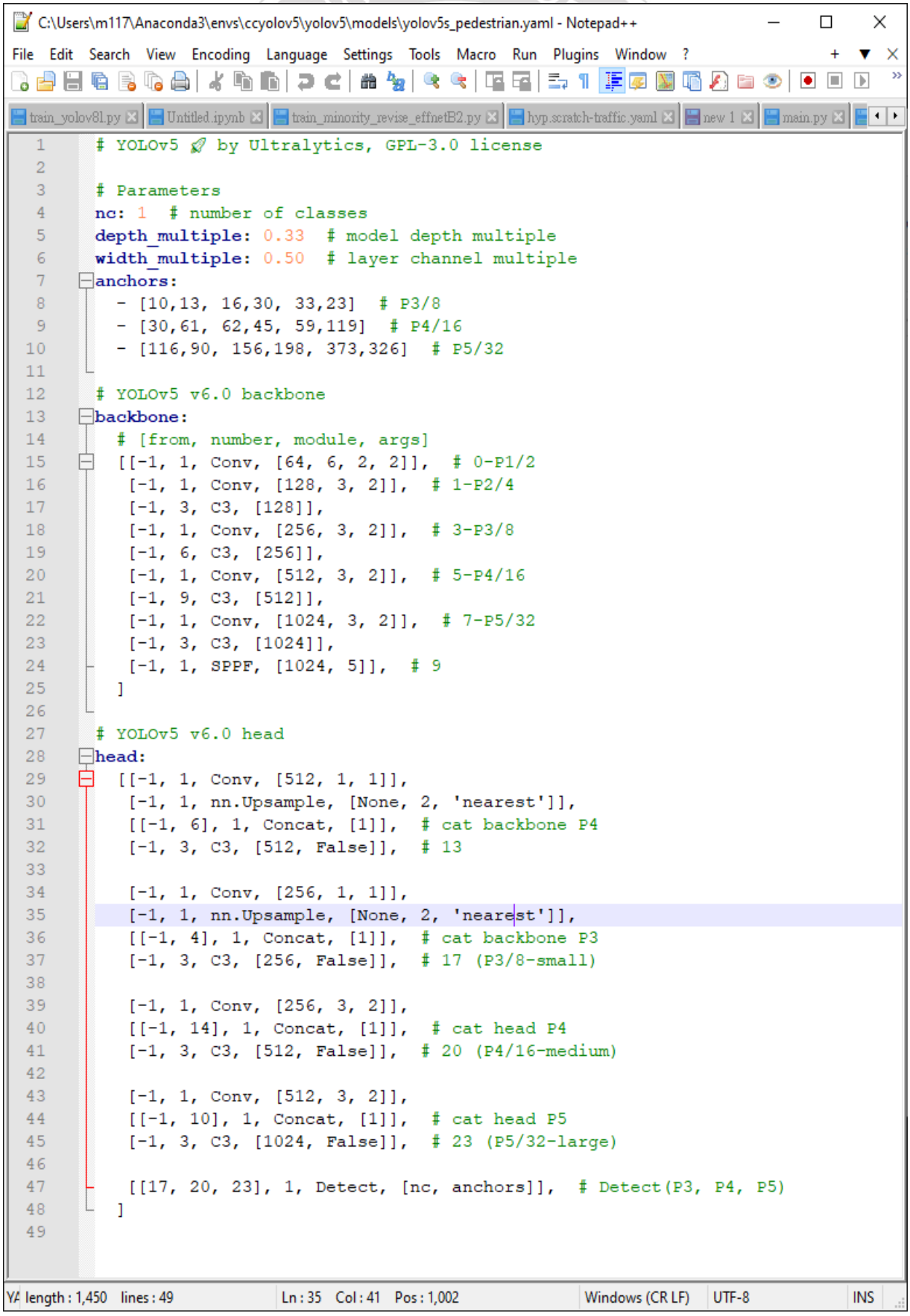

圖 6 YOLOv5s 的目標偵測 yaml 文件

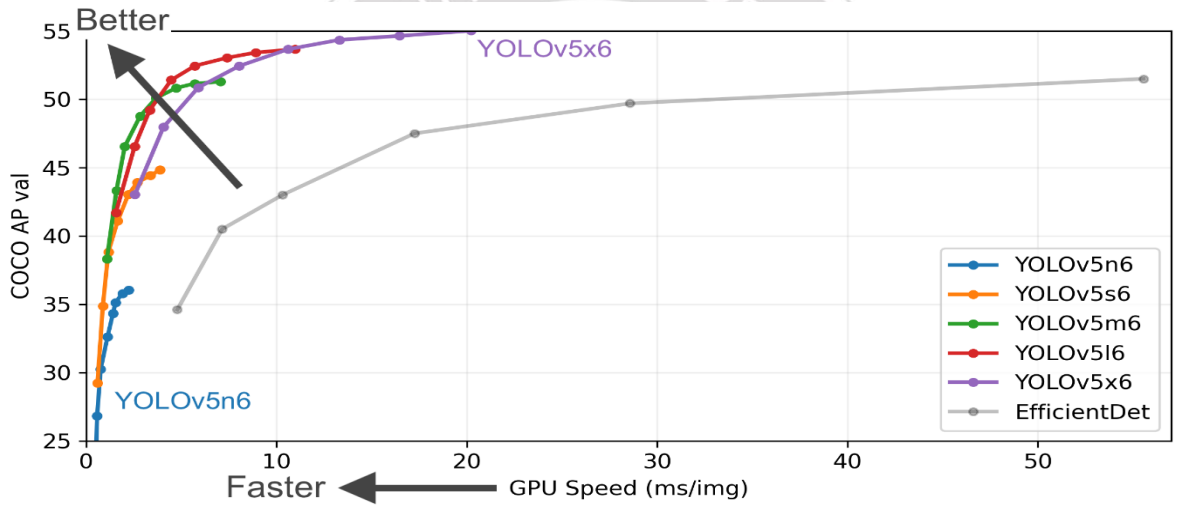

圖 7 YOLOv5 系列演算法性能測試圖 [5]

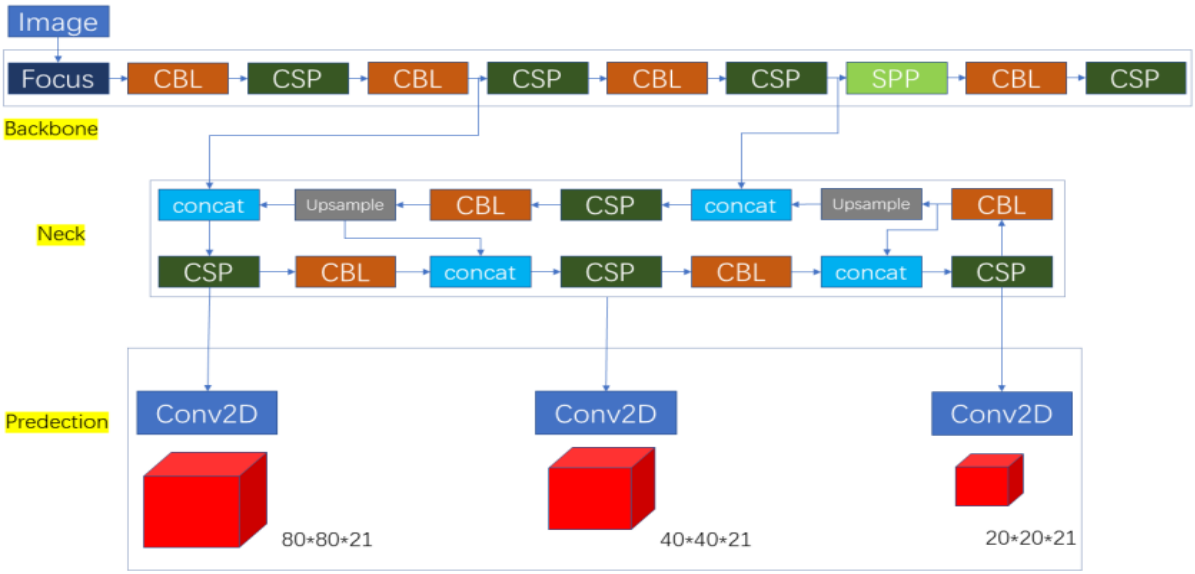

圖 8 YOLOv5 模型架構圖 [6]

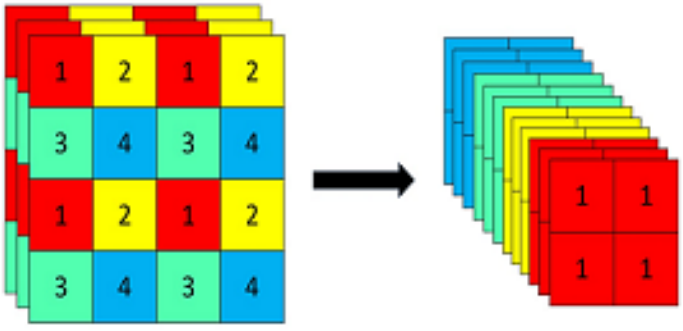

圖 9 YOLOv5 切片操作圖

2.3 U-Net 網路結構

圖像分割 (Image Segmentation)是一種針對像素進行偵測的分類，圖像分割常被廣泛應用於人物肖像、自動駕駛、生物醫學、智慧畜牧等領域中，常見的圖像分割為語義分割 (Semantic Segmentation)，以及實例分割 (Instance Segmentation) 兩方面，具體的圖像分割範例 [7]可以參考圖 10。

(1) 語義分割即演算法會針對 2D 圖像中的所有像素 (Pixel) 進行分類。

(2) 實例分割是指針對圖像中的每一個像素進行分類，即使是相同的類別也會分割成不同物件。

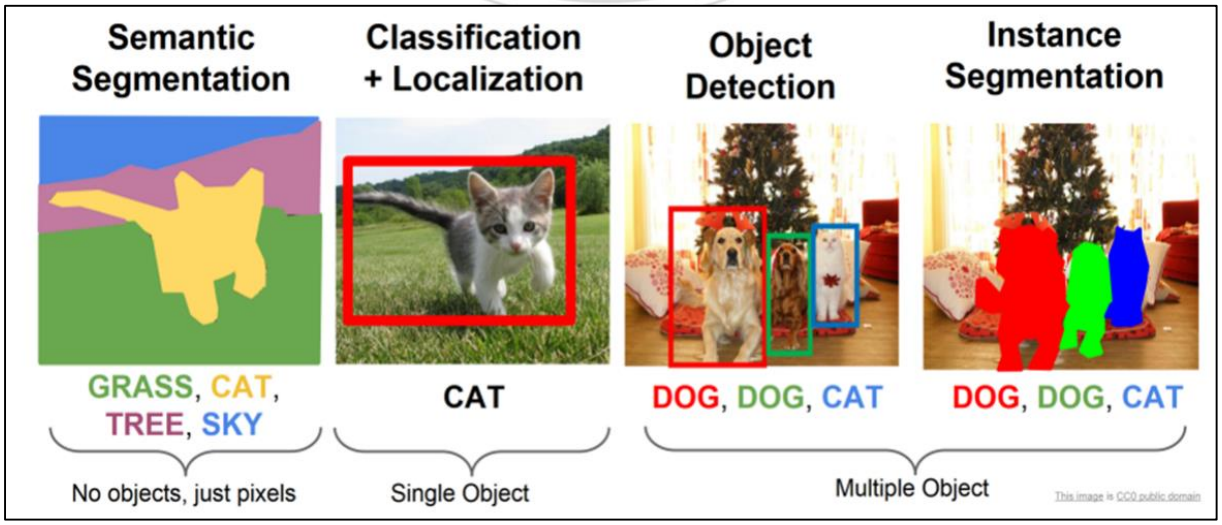

圖 10 圖像分割範例圖 [7]

U-Net 深度學習網路可以實現上述圖像分割介紹中的語義分割任務。U-Net 是由傳統的卷積神經網路 (CNN) 演變而來的，且 U-Net 是全卷積神經網路 (Fully Convolutional Network, FCN) 的其中一種代表模型，於 2015 年首次被用於處理生物醫學圖像 [8]。U-Net 的結構流程圖如圖 11，網路的輸入是一張 572 x 572 的圖像；藍色箭頭表示 3 x 3 卷積，用於特徵提取；紅色箭頭表示最大池化 (Max Pooling)，用於降低維度；綠色箭頭表示上取樣 (Up sampling)，用於恢復維度；灰色箭頭表示跳躍連接 (Skip-Connection)，用於特徵融合；青色箭頭表示 1 x 1 卷積，用於輸出結果。

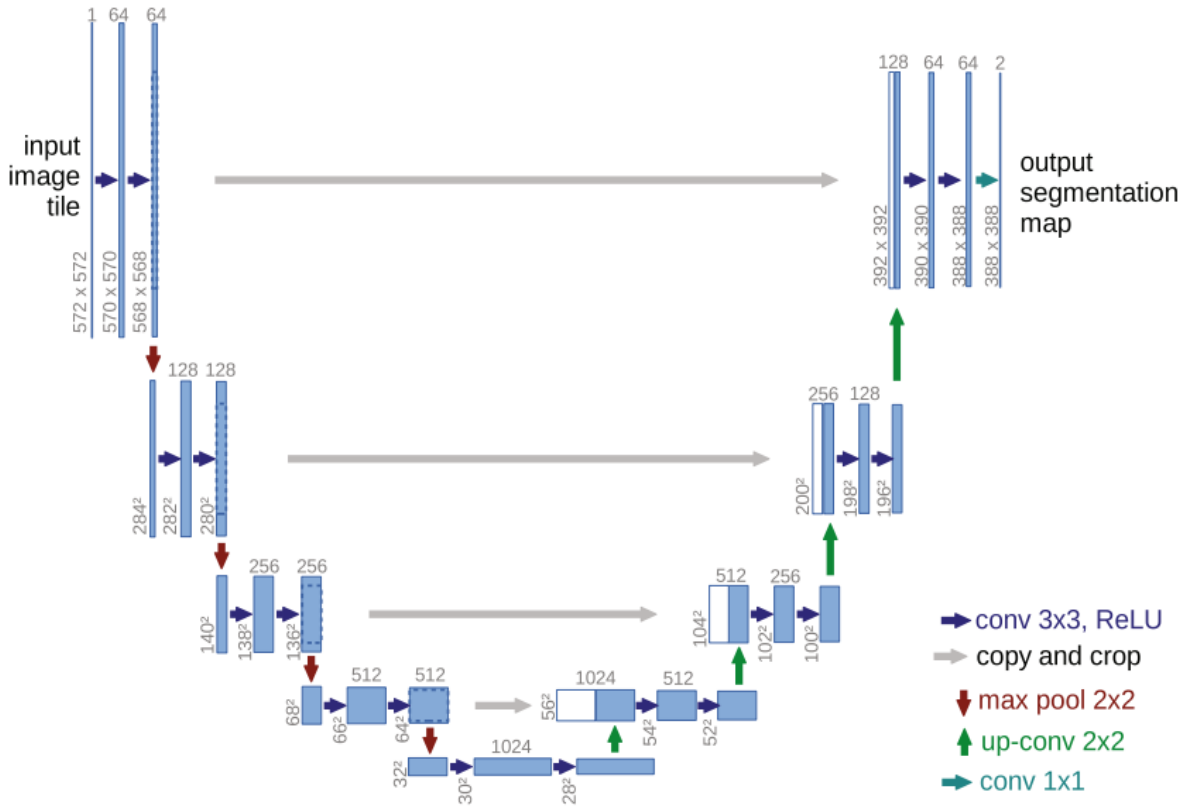

圖 11 U-Net 的結構流程圖 [8]

U-Net 是由 Encoder-Decoder 組成的結構，編碼 (Encoder) 的部分負責提取特徵，而解碼 (Decoder) 的部分則是要將特徵圖 (Feature Map) 恢復原始解析度。U-net 為一個 U 型對稱網路結構，U-net 的每一層都會做 2 個 3 x 3 的卷積運算來提取特徵，並且運用最大池化層以及上取樣操作來做降低維度以及恢復維度，而 U-net 在上取樣時，還會參考在 Encoder 的特徵圖，與 Decoder 的特徵圖做跳躍連接用於特徵融合，也就是說會增加描述圖像的特徵，進而提升輸出結果的準確率，最後一層則會做一個 1 x 1 的卷積運算，用於將 64 個特徵向量映射到所需的分類數，來得到最終的預測結果。U-net 模型的損失函數選擇採用交叉熵損失函式 (Binary cross entropy, BCE)，一般常被用於二元分類的問題上。

2.4 行人偵測相關文獻回顧

近年來深度神經網路在圖像分割，以及電腦輔助診斷領域中取得了重大的成就，深度學習技術能在圖像分割領域中發揮有力和準確的辨識作用。本章節將針對學者近年來的行人偵測工作進行文獻回顧。

在霧霾天氣時拍攝的行人圖像的可見度是很差的，會嚴重限制了當前行人偵測方法的有效性。G. Li 等學者 [9]提出了三種基於 Yolo 的霧霾天氣行人偵測深度學習方法，他們使用了 Yolo 的概念來偵測行人，使用深度和逐點卷積 (Depthwise And Pointwise Convolution) 代替標準的卷積來降低計算成本，還使用 Priori Boxes 使網路更加精確，此外，提出了一種新的加權組合層 (Weighted Combination Layer) 來提高偵測精度性能。研究結果表明，提出的方法在 HazePerson 資料集的偵測結果實現了 86.6% 的平均準確率。

L. Chen 等學者 [10]指出車輛和行人偵測是自動駕駛中的關鍵任務之一，在文章中分析了幾種主流的目標偵測架構，以及幾種典型的特徵提取器，通過使用 KITTI 資料集進行大量實驗，多個測試模型的實驗結果比較表明，Faster R-CNN ResNet 50 實現了最好的準確性 (AP = 58%)、(116.17 ms) 執行時間結果。ResNet 的性能更好的原因在於，ResNet 採用 Skip-Connection 可以避免梯度消失，且 ResNet 通常適用於較難的任務，比如有大量圖像或物體的資料集，因此在複雜的資料集中的偵測任務比較能取得較為優異的性能表現。

B. Han 等學者 [11]提到現有的方法，對於距離鏡頭相對較遠的行人偵測存在不足。他們在文章中，提出了一種用於小規模行人偵測的新型深度小規模感知網路 (SSN)，實驗結果表明在 VIP 行人資料集上 AP 實現了 90.51%。此方法在小規模行人偵測中取得了良好的偵測性能。

W. Wang 等學者 [12]提出了一些基於偵測器、中心和尺度預測 (CSP) 的自適應方法，針對行人偵測器原有的 CSP 進行改進，在 CityPersons 資料集上進行了實驗，研究結果與其他現有技術的比較，取得了良好的成績，此方法明顯提高了 CSP 的性能。

Z. Shao 等學者 [13]提到無人機是一種靈活的移動平台，使用無人機進行行人偵測是一個很有前途的選擇。因此，他們提出了一種輕量級行人偵測網路來準確偵測行人，偵測網路是由 PeleeNet、多尺度空間注意力模塊和偵測層組成的，為了要探索像人頭這樣的小尺寸物體的特徵，他們通過反捲積和連接融合了三個尺度(19 × 19、38 × 38、76 × 76)的特徵圖。最終實驗結果表明，此方法達到了 92.22% 的 AP 和 76 FPS (Frames Per Second)。

上述的論文都基於深度學習方法，提出的行人偵測研究都有良好的回饋，故本篇論文將使用深度學習的方式進行實驗。

第三章 研究方法

3.1 研究架構

本研究提出的架構如摘要所述，研究第一階段為使用 YOLOv5 模型對蒐集好的行人圖像資料集進行訓練、偵測與測試，第二階段則是使用 U-net 模型，讀取經過裁切處理的行人圖像資料集進行訓練與測試。本文提出的研究流程圖如圖 12。

圖 12 研究流程圖

(1) YOLOv5 模型訓練

在此一階段中的行人資料集的要求為：Img 和 Yolo 標記格式文件如圖 13。使用 YOLOv5 模型對蒐集好的行人資料集進行訓練、偵測與測試

。YOLOv5 模型的優勢為相較於其他較低版本的 YOLO 在於，YOLOv5 可以使用 “save-txt” 參數，將訓練結果的偵測框 (Bounding Box) 圖像結果保存成為對應圖像名稱的 Txt 偵測框 (Bounding Box)文件，該文件可以在 ./runs/detect/exp/labels 資料夾中找到，YOLOv5 模型訓練後得到的偵測框文件如圖 14。

(2) 將 YOLOv5 的偵測結果進行裁切處理

YOLOv5 模型訓練後將得到的行人偵測框 Txt 文件，將原始的行人圖像(圖 15 左圖)，以及事先經過圖像標記的行人圖像(圖 15 右圖)進行裁切處理。

(3) U-net 模型訓練

使用 U-net 模型對事先做好裁切處理的行人圖像進行訓練及測試。

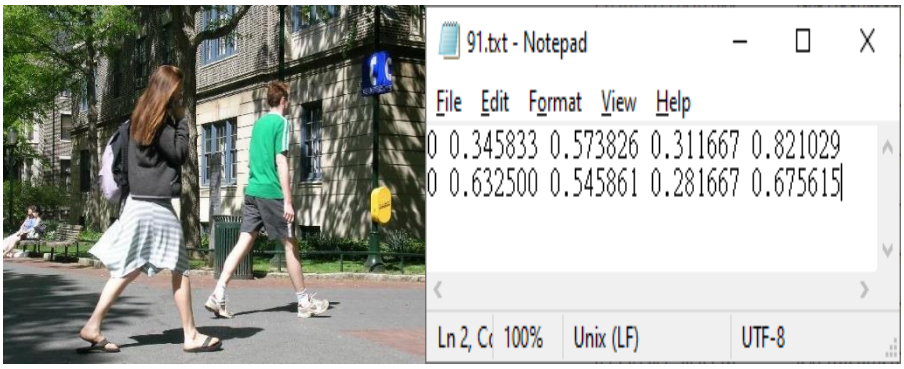

圖 13 YOLOv5 行人資料集

圖 14 YOLOv5 模型訓練後得到的偵測框文件

圖 15 U-net 行人資料集

3.2 研究環境架設

在進行 YOLOv5 以及 U-net 訓練之前，要預先進行開發環境的安裝架設，在研究中我們將會需要用到 Anaconda、Cuda、Cudnn、Python、PyTorch 等套件。本實驗的硬體設備為 Intel® Core™ i7-8700 @ 3.20 GHz CPU 與 32 GB 實體記憶體，運行於 Windows 10 作業系統。

3.3 YOLOv5 訓練前處理

(1) 資料集準備：

在本次實驗終將利用 Labelimg 給資料集來打上標記，Labelimg 是一款開放原始碼的資料標記工具，可以將資料集標記為三種格式：Pascal VOC 標記格式，保存為 Xml 文件；Yolo 標記格式，保存為 Txt 文件；COCO 標記格式，保存為 Json 格式。在 Anaconda Prompt 輸入 *pip install labelimg* 來安裝 Labelimg。Labelimg 標記軟體如圖 16 所示，在本研究中會將資料集的標記資料儲存為 Yolo 標記格式，做為 YOLOv5 模型訓練的標記資料。表 1 將針對 Labelimg 的常用快捷鍵做簡要的說明：

圖 16 Labelimg 標記軟體

表 1 Labelimg 的常用快捷鍵

Ctrl + U :	打開圖像文件夾
Ctrl + R :	更改結果保存位置
W :	開始畫框
Ctrl +S :	保存標記圖像
D :	下一張圖像
A :	上一張圖像
Del :	刪除畫的框
Ctrl + :	圖像放大
Ctrl - :	圖像縮小

(2) 安裝虛擬環境：

先透過 Anaconda Prompt 建立一個 YOLOv5 專用的 Python 虛擬環境，接著載入剛才建立好的虛擬環境，使用 Git clone 下載 YOLOv5 的原始

碼、安裝 Pytorch、下載權重文件。YOLOv5 的安裝需求如圖 17，可以使用 *pip install -r requirements.txt* 指令來安裝 YOLOv5 模型需求的各種套件。

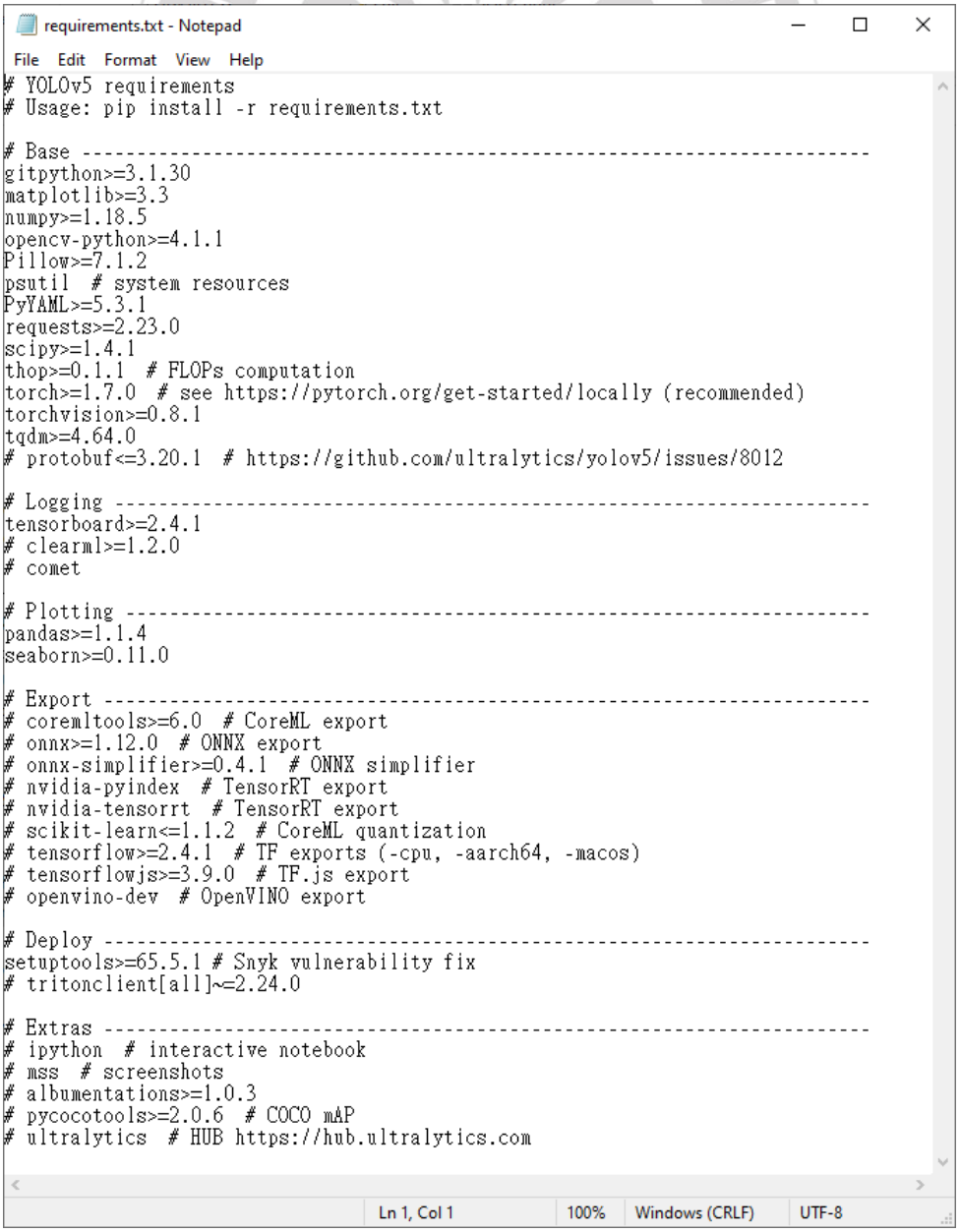

圖 17 YOLOv5 的安裝需求

(3) 製作行人資料集：

將事先在網路上下載好的行人圖像資料集放入 ./VOCData 之下的 /images/train/與/images/val/ 文件夾；再將與圖像檔案名稱相同的 YOLO 格式的標記資訊的文字檔，放入 ./VOCData 之下的 /labels/train 與/labels/val 文件夾。

(4) 修改 Data 文件夾之下的 Yaml 文件：

從 yolov5/data 文件夾中複製一份 voc.yaml 文件，並將檔案名稱修改為 pedestrian.yaml，文件內容如圖 18 所示，其中 train 表示訓練集圖像的資料夾路徑位置，val 表示測試集圖像的資料夾路徑位置，nc 表示類別數量，names 表示類別名字。

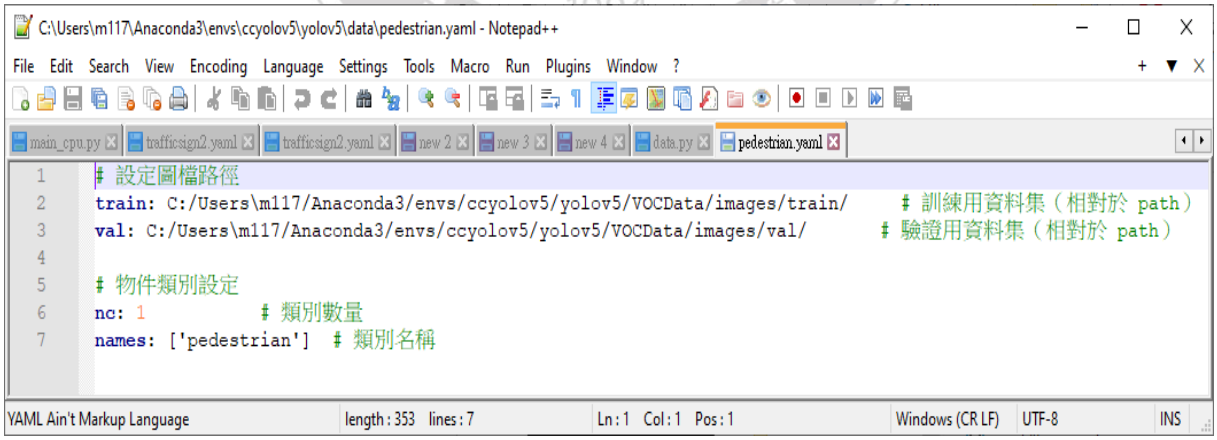

圖 18 pedestrian.yaml 文件

(5) 修改 Models 文件夾之下的 Yaml 文件：

從 ./models 目錄下選擇一個模型的配置文件，複製一份 Yaml 文件並且修改 *.yaml 文件中的 nc 參數。以 yolov5s 舉例，複製一份 yolov5s.yaml 檔案，將其重新命名為 yolov5s_pedestrian.yaml，修改其類別個數，文件內容如圖 7 所示。

(6) 訓練 YOLOv5 模型：

在 YOLOv5 模型的訓練階段中，會先在 Anaconda Prompt 載入 YOLOv5 專用的 Python 虛擬環境，接著輸入 *python train.py* 指令對 YOLOv5 進行訓練，訓練好的模型會被保存在 YOLOv5 目錄底下的 ./runs/train/exp/weights/。

(7) 模型預測：

輸入 *python detect.py--save-txt* 指令，將預測後的結果自動保存到 ./runs/detect/exp/labels 路徑下。

(8) 模型測試：

輸入 *python val.py* 指令，將測試的結果自動保存到 ./runs/val/exp 路徑下。

3.4 U-net 訓練前處理

(1) 資料集準備：

在進行 U-net 訓練之前，需要先進行資料集的準備，需要先在 Anaconda Prompt 輸入 *pip install labelme==3.16.7* 安裝 Labelme 標記軟體，Labelme 標記軟體如圖 19 所示。在 Labelme 標記完圖像之後，會得到一個圖像名稱的 Json 格式，如圖 20。

根據 U-net 模型的資料集需求，將 Json 檔案轉換成含有類別顏色標記的可視化圖像。只要完成以下兩者中的其中一個動作，即可以生成一個包含原始圖像、標記圖像、帶標記的可視化圖像、記錄了標記名稱的文字檔的資料夾，如圖 21。(1) 轉換一張標記圖像：Anaconda Powershell 輸入 *labelme_json_to_dataset 資料夾路徑 <文件名>.json* 指令。(2) 轉換多張標記圖像： Anaconda Powershell 輸入 *cd 資料夾路徑*，再輸入 *json_to_mask.bat*。json_to_mask.bat 檔案，檔案內容如圖 22 所示。表 2 將針對 Labelme 的常用快捷鍵，做簡要的說明。

圖 19 Labelme 標記軟體

12.json - Notepad

File Edit Format View Help


```
{
  "version": "3.16.7",
  "flags": {},
  "shapes": [
    {
      "label": "pedestrian",
      "line_color": null,
      "fill_color": null,
      "points": [
        [
          153.88888888888889,
          112.6984126984127
        ],
        [
          155.0793650793651,
          106.74603174603175
        ],
        [
          149.12698412698413,
          95.63492063492063
        ],
        [
          152.6984126984127,
          82.14285714285714
        ],
        [
          164.60317460317746,
          77.77777777777777
        ],
        [
          175.71428571428572,
          80.55555555555556
        ],
        [
          183.65079365079364,
          86.9047619047619
        ],
        [
          186.82539682539684,
          96.82539682539682
        ],
```


Ln 1, Col 1 100% Windows (CRLF) UTF-8

圖 20 Labelme 的 Json 標記格式

圖 21 執行 labelme_json_to_dataset 生成標記資料夾

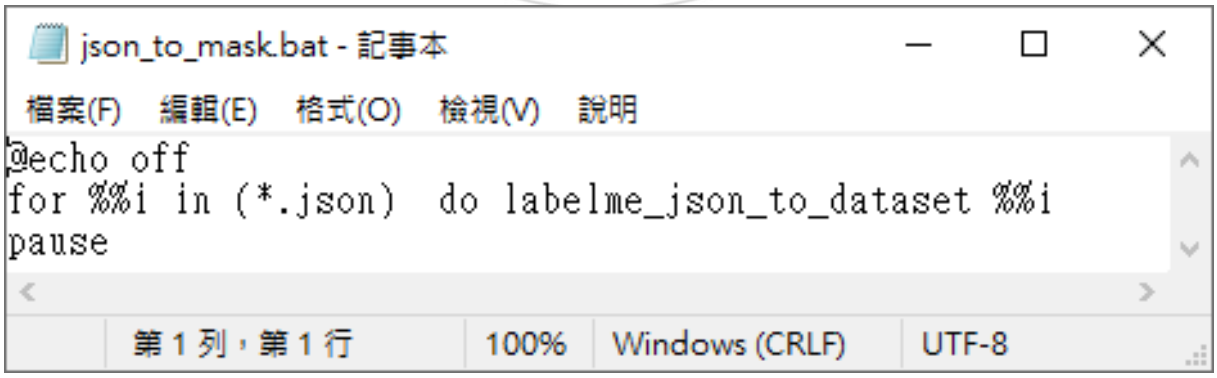

圖 22 json_to_mask.bat

表 2 Labelme 的常用快捷鍵

Ctrl + E :	選中標記框時，按下可打開編輯視窗
Ctrl + N :	創建多邊形標記框
D :	下一張圖像
A :	上一張圖像
Delete	刪除標記框
Ctrl + :	圖像放大
Ctrl - :	圖像縮小
Ctrl + F :	還原圖像尺寸

以 YOLOv5 模型偵測得到的偵測框 Txt 文件為基準，來執行行人裁切程式碼，裁切並且儲存行人的目標框圖像，將最終裁切出來的行人切割圖像，以及行人標記切割圖像作為 U-Net 模型訓練與測試的資料集。圖 23 將分別顯示 24 位元深度的行人切割圖像，與 8 位元深度的行人標記切割圖像。在得到行人切割圖像之後，會進行圖像事前處理，將 24 位元深度的圖像進行 8 位元灰階顏色處理，圖像灰階處理對比圖如圖 24 所示。

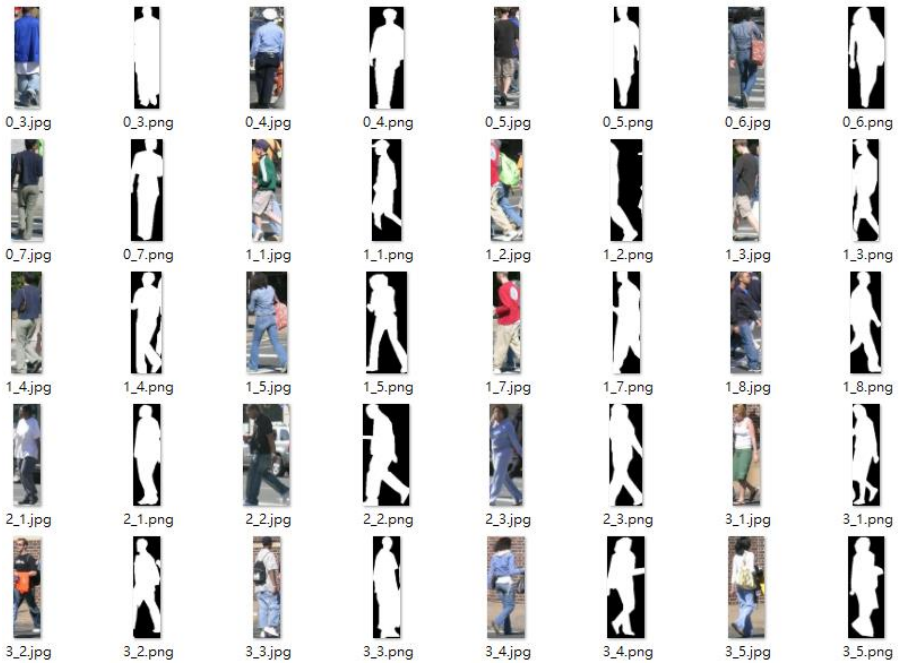

圖 23 行人切割圖像

圖 24 圖像灰階處理對比圖

(2) U-net 環境建立以及參數設定：

在進行 U-Net 模型的訓練之前，為了使自己的 Python 深度學習環境不和其他 Python 套件產生衝突，需要先將 Python 的虛擬環境建立起來，因此在這個階段中要先透過 Anaconda Prompt 輸入指令 *conda create --name python=3.9*，接著使用 *conda activate name* 切換到剛才建立好的這個虛擬環境中。在 U-Net 訓練的過程中可以使用 Tensorflow GPU 進行訓練加速，但是需要先透過輸入 *nvidia-smi* 指令，查詢目前電腦顯卡驅動版本的訊息，例如圖 25，將會顯示目前電腦 CUDA 的版本為 11.8，接下來即可以依照目前的 CUDA 版本安裝相對應版本的 tensorflow-gpu，Tensorflow GPU 具體的安裝資訊可參考圖 26 [14]。這樣一來在模型訓練的過程中才比較不會發生錯誤訊息，tensorflow-gpu 的安裝指令為 *pip install tensorflow-gpu==2.10.0*，在 U-net 的訓練以及測試程式碼之中加入以下代碼： *os.environ ["CUDA_VISIBLE_DEVICES"] = "0,1"*，來控制 GPU0 和 GPU1 的使用。

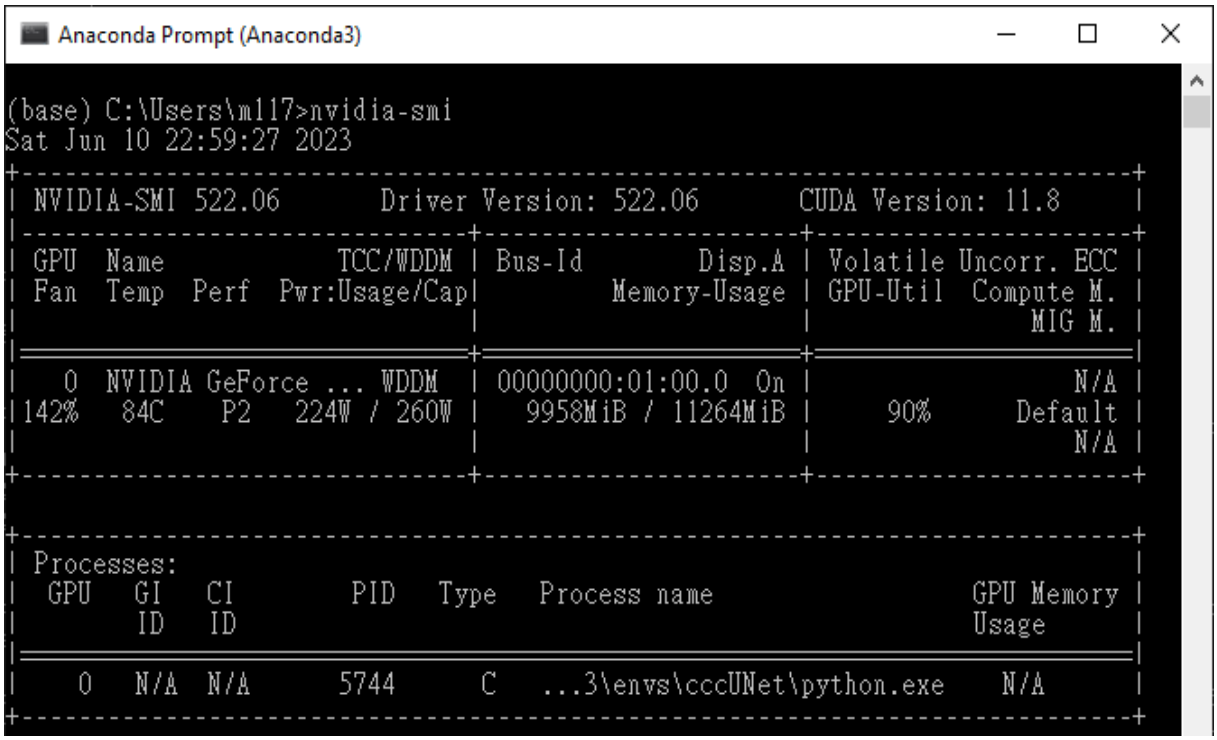

圖 25 nvidia-smi 指令輸出內容

GPU

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.13.0	3.8-3.11	Clang 16.0.0	Bazel 5.3.0	8.6	11.8
tensorflow-2.12.0	3.8-3.11	GCC 9.3.1	Bazel 5.3.0	8.6	11.8
tensorflow-2.11.0	3.7-3.10	GCC 9.3.1	Bazel 5.3.0	8.1	11.2
tensorflow-2.10.0	3.7-3.10	GCC 9.3.1	Bazel 5.1.1	8.1	11.2
tensorflow-2.9.0	3.7-3.10	GCC 9.3.1	Bazel 5.0.0	8.1	11.2
tensorflow-2.8.0	3.7-3.10	GCC 7.3.1	Bazel 4.2.1	8.1	11.2
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1
tensorflow-2.1.0	2.7, 3.5-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0

圖 26 Tensorflow GPU 的安裝資訊 [14]

建立好虛擬環境並且安裝好 tensorflow-gpu 之後，接著就可以載入虛擬環境，使用 Git clone 下載 U-net 的原始碼，並且安裝 U-net 訓練所需要的各種 Python 套件，例如： Keras、Scikit-Image、Matplotlib 等，再將準備好的行人資料集放入 U-Net 的訓練資料夾中。在 Anaconda Prompt 輸入 *python main.py* ，即可開始 U-net 模型的訓練與預測，預測結果將會自動保存到 ./ test_predict / 文件夾。表 3 為 U-Net 模型的參數設定。U-net 模型架構詳細說明如表 4。

表 3 U-Net 參數設定

	YOLOV5s 串接 U-Net	U-Net
Input	2612	654
Output	628	157

Environment	Windows 10
Bit Depth	8
Input Size	256 x 256
Batch Size	8 2
Optimizer	Adam
Loss Function	Binary Crossentropy
Reduce Learning Rate	Reducelronplateau, Earlystopping
Learning Rate	1e-4
Metrics	Accuracy, Dice_Coef, Precision, Recall, F1Score

表 4 U-Net 模型架構表

	Convlayer No.	Filters	Size	Activation Function
Encoder	Conv 1 Dropout (Conv1)	64	3 x 3	Relu
	Conv 2 Dropout (Conv2)	128	3 x 3	Relu
	Conv 3 Dropout (Conv3)	256	3 x 3	Relu
	Conv 4 Dropout (Conv4)	512	3 x 3	Relu
	Conv 5 Dropout (Conv5)	1024	3 x 3	Relu
Decoder	Up 6 Concatenate (Conv 4, Up 6) Dropout (Conv6)	512	3 x 3	Relu
	Up 7 Concatenate (Conv 3, Up 7) Dropout (Conv7)	256	3 x 3	Relu
	Up 8 Concatenate (Conv 2, Up 8) Dropout (Conv8)	128	3 x 3	Relu
	Up 9 Concatenate (Conv1, Up 9) Dropout (Conv9)	64	3 x 3	Relu
	Conv 10	1	1 x 1	Sigmoid

3.5 實驗評估及說明

本小節將逐步介紹在本研究中 YOLOv5 模型與 U-net 模型應用到的實驗評估效能指標。混淆矩陣 (Confusion Matrix) 是一種用來評估模型好壞的常見方法，可以用來加以計算準確率 (Accuracy)、精確率 (Precision)、召回率 (Recall)、AP (Average Precision)、mAP (mean Average Precision)、F1 Score 等多個衡量指標，這也是 YOLO 與 U-net 模型中常用的評估指標。

表 5 為混淆矩陣表格，以下將列出排列組合起來的四種可能情況，分別是 True Positive (TP)、True Negative (TN)、False Positive (FP)和 False Negative (FN)。舉例來說，當辨識模組要預測的對象為動物時，輸入的圖像有物件為動物，而模組有使用偵測框畫出動物，被歸類為 TP；如果沒有任何物件出現動物，模組也沒有使用任何偵測框，被歸類 FN；如果沒有任何物件出現動物，但模組有畫出偵測框，被歸類為 FP；如果有出現動物，但沒畫出偵測框，則被歸類為 TN。

表 5 混淆矩陣 (Confusion Matrix)

	Actual	Positive	Negative
Predict	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

精確率 (Precision)

是代表預測為真的情況下，有多少個為預測準確樣本的百分比。精確率公式如 (1) 所定義：

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

召回率 (Recall)

是代表預測為真的情況下，有多少百分比為被正確判斷出來的樣本。召

回率公式如 (2) 所定義：

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

F1 Score

精確率與召回率的調和平均數。F1 Score 公式如 (3) 所定義：

$$F1\ Score = \frac{2\ X\ Precision\ X\ Recall}{Precision + Recall} \tag{3}$$

mAP (mean Average Precision)

是計算每個類別的精度並取平均值，k 代表的模型的類別，n 代表在該類別下的數量。mAP 公式如 (4) 所定義：

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP\kappa \tag{4}$$

準確率 (Accuracy)

準確率，如公式 (5) 所定義，分母為所有情形之加總，分子為判斷正確的情況，表示為預測正確的圖像像素之百分比，是最基本的評估指標。

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

聯合交集(Intersection over Union, IoU)

IoU 為一種計算不同圖像相互重疊(overlap)，或交集比例的標準。圖像重疊區域除以聯及區域將會得出交集分數，重疊的比例愈高代表預測結果愈準確。IoU 的值為 0 表示預測結果與標準答案完全不一致；IoU 的值為 1 則表示預測結果完全吻合標準答案。IoU 公式如 (6) 所定義：

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} = \frac{TP}{TP + FP + FN} \tag{6}$$

Dice 係數 (Dice Coefficient)

Dice 係數與聯合交集 (IoU) 兩個指標是非常相似的，當 Dice 是 0.961951 時，對應的 IOU 是 0.926691 [15]，根據此資料得出結論使用 Dice 係數可以得到一個比聯合交集更高的指標分數，因此，本研究採用 Dice 係數來衡量兩個集合之間相似性。Dice 係數如圖 27 所示是一種用於計算兩個樣本相似度的度量函式，Dice 係數範圍介於 0 到 1 之間，Dice 係數越高代表樣本的重疊範圍越大，即表示預測結果與實際樣本越相近；Dice 係數越低，則表示樣本的重疊範圍越小，代表預測結果與實際樣本之間的差異越大。Dice Coefficient 公式中的 X 為預測的結果樣本，Y 為實際的樣本，Dice Coefficient 公式計算如 (7) 所示：

$$Dice(X,Y) = \frac{2\,|X \cap Y|}{|X|+|Y|} \tag{7}$$

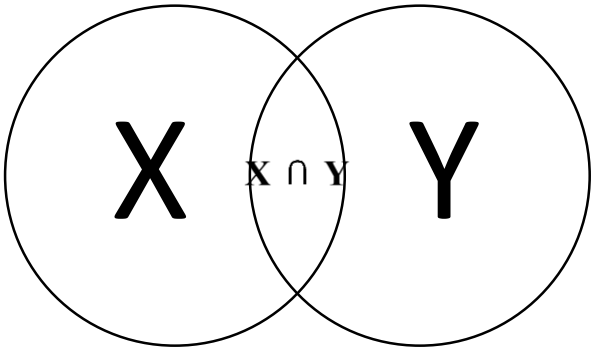

圖 27 Dice Coefficient 公式示意圖

第四章 實驗結果與討論

4.1 資料集

本研究的行人偵測圖像的研究資料集資料來源於網路上公開下載的四個 Dataset，分別為 Penn-Fudan Database [16]、ETH Pedestrian Dataset [17]、CUHK Occlusion Dataset [18]，和 Human Detection Dataset [19]。以下將簡單介紹上述提到的四種常用於行人偵測研究的圖像資料集：

(1) Penn-Fudan Database： 一個用於行人偵測的圖像資料庫，該資料集包含 170 張圖像和 345 個已標記的行人，它取自於校園和城市街道周圍的場景。

(2) ETH Pedestrian Dataset： 一個用於行人偵測的資料集，它包含三個影片剪輯中的 1,804 張圖像，資料集是從安裝在汽車上的 AVT Marlins F033C 立體裝置捕獲而來的，解析度為 640 x 480，幀率為 13–14FPS。

(3) CUHK Occlusion Dataset： 包含 361 張沒有行人的圖像，以及 558 張含有行人的圖像，資料集主要來源於 cctv 鏡頭錄像的圖像。

(4) Human Detection Dataset： 包含來自 Caltech [20]、ETHZ [21]、TUD-Brussels [22]、INRIA [23]、Caviar [24]的資料集和 Dataset 提供者收集的圖像的 1063 張被遮擋的行人圖像。

4.2 YOLOv5 的實驗結果

本研究利用 YOLOv5 模型進行行人識別，圖 28 為節錄 train.py 模型訓練的參數設定；圖 29 為 節錄 detect.py 模型偵測的參數設定；圖 30 為節錄 val.py 模型測試的參數設定，表 6 將針對 YOLOv5 模型的訓練、預測、測試參數做簡要的說明：

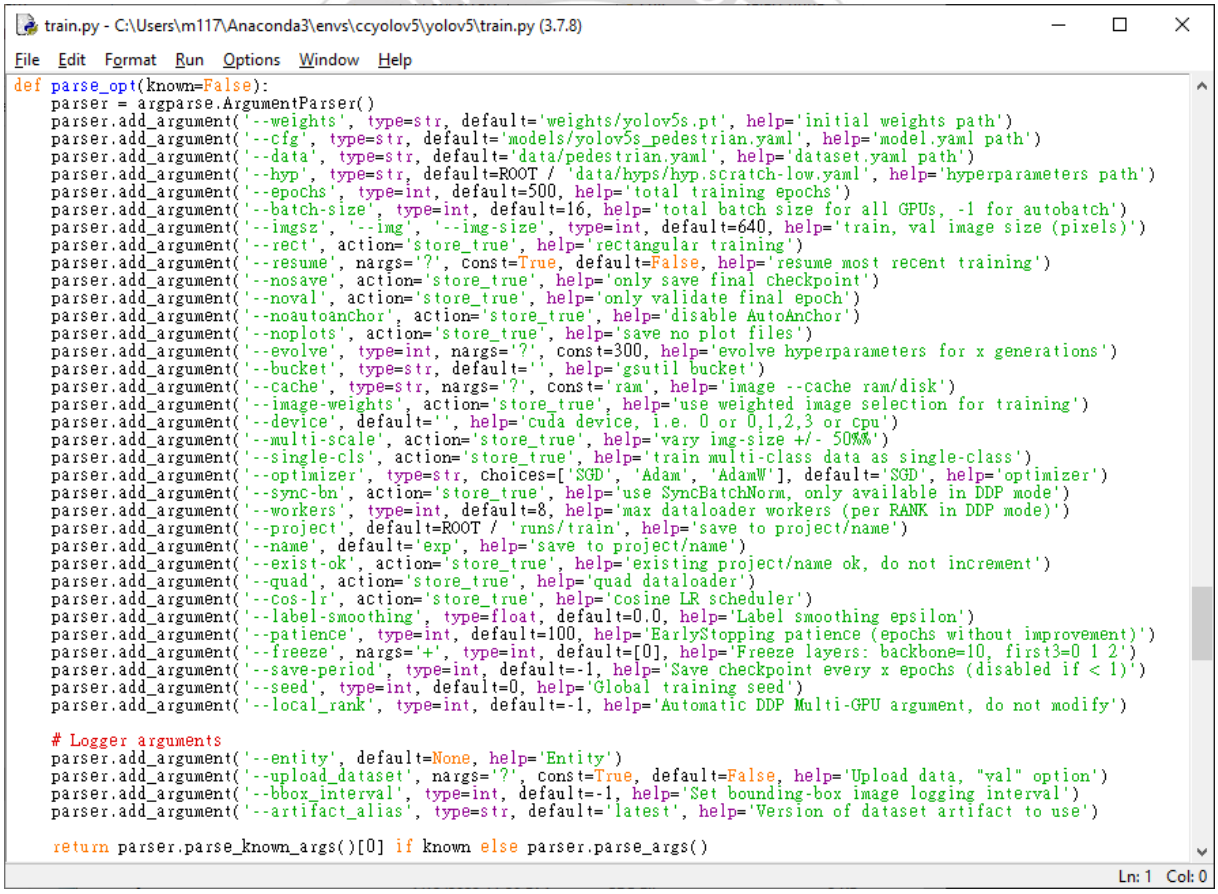

圖 28 節錄 train.py 模型訓練的參數設定

圖 29 節錄 detect.py 模型偵測的參數設定

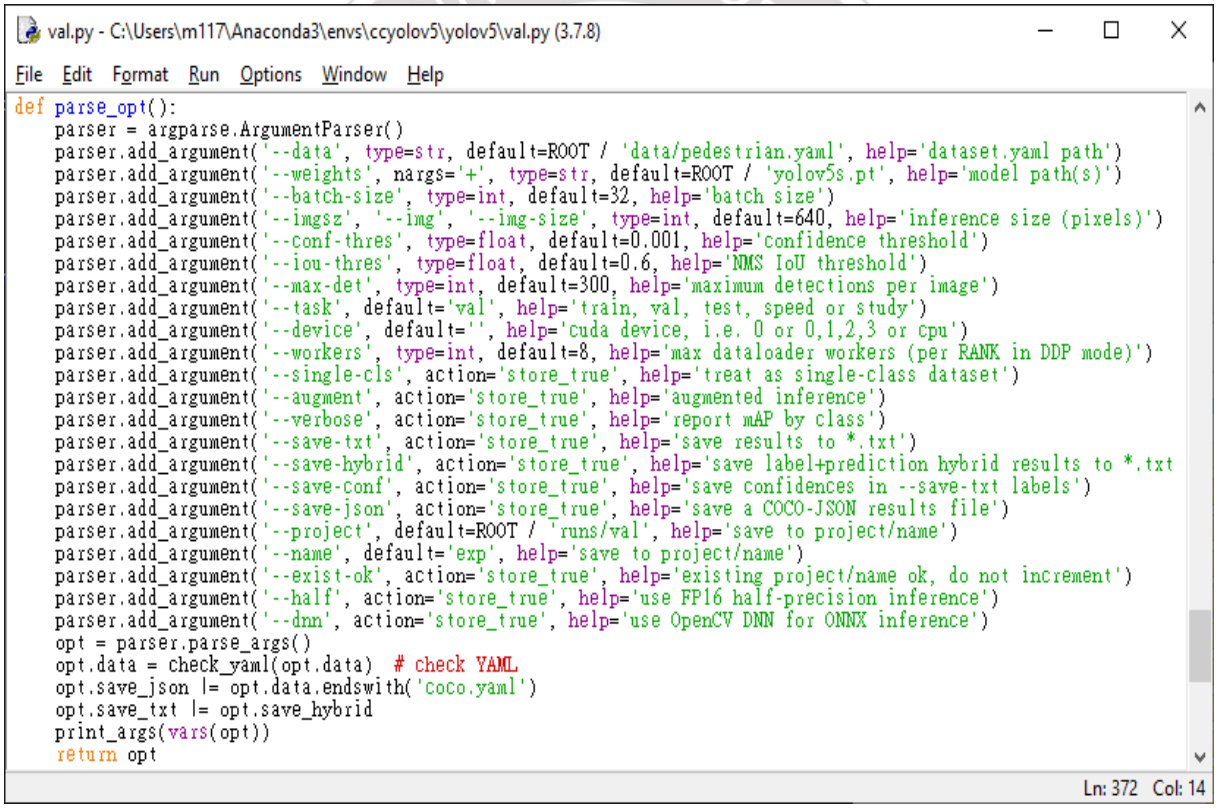

圖 30 節錄 val.py 模型測試的參數設定

表 6 YOLOv5 的訓練、預測、測試參數

--weights：	存放權重的路徑
--cfg：	存放模型配置的 yaml 文件
--source：	指定網路輸入的路徑
--data：	存放資料集配置的 yaml 文件
--epoch：	訓練週期
--batch：	batch_size 大小
--img：	網路輸入圖像大小
--single-cls：	訓練單個類別的資料集
--iou-thres：	設定 IoU 門檻值
--conf-thres：	設定信心門檻值
--name：	預測結果保存的文件夾名稱
--max-det：	最大偵測數量，默認是最多偵測 1000 個目標
--save-txt：	是否將偵測結果的坐標以 txt 文件形式保存

在 YOLOv5 模型的實驗階段中總共蒐集了 786 張行人圖像，一開始會先執行程式碼將訓練以及測試圖像劃分為 8 比 2。圖 31 為訓練資料集，而圖 32 為測試資料集。

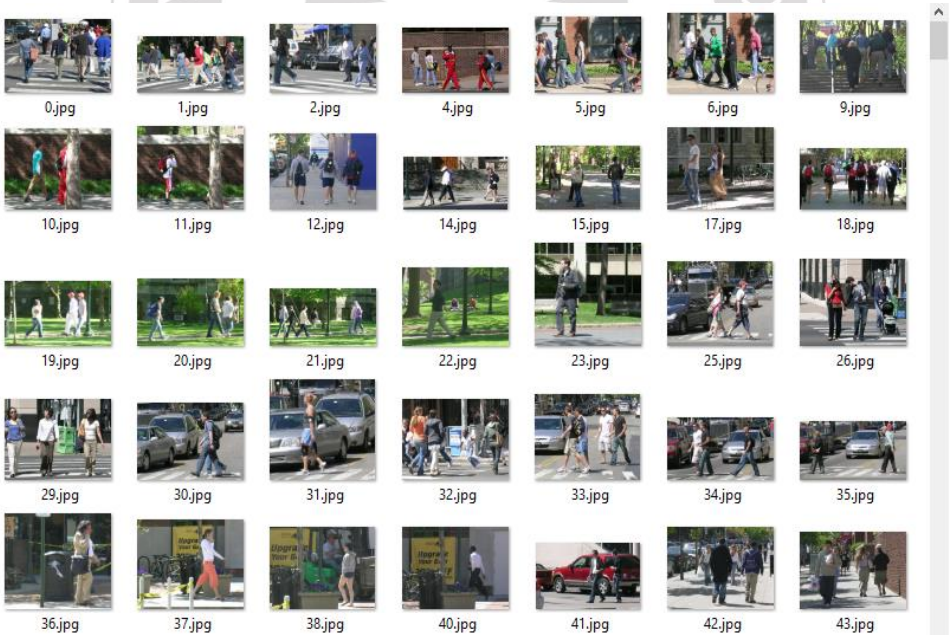

圖 31 YOLOv5 訓練資料集

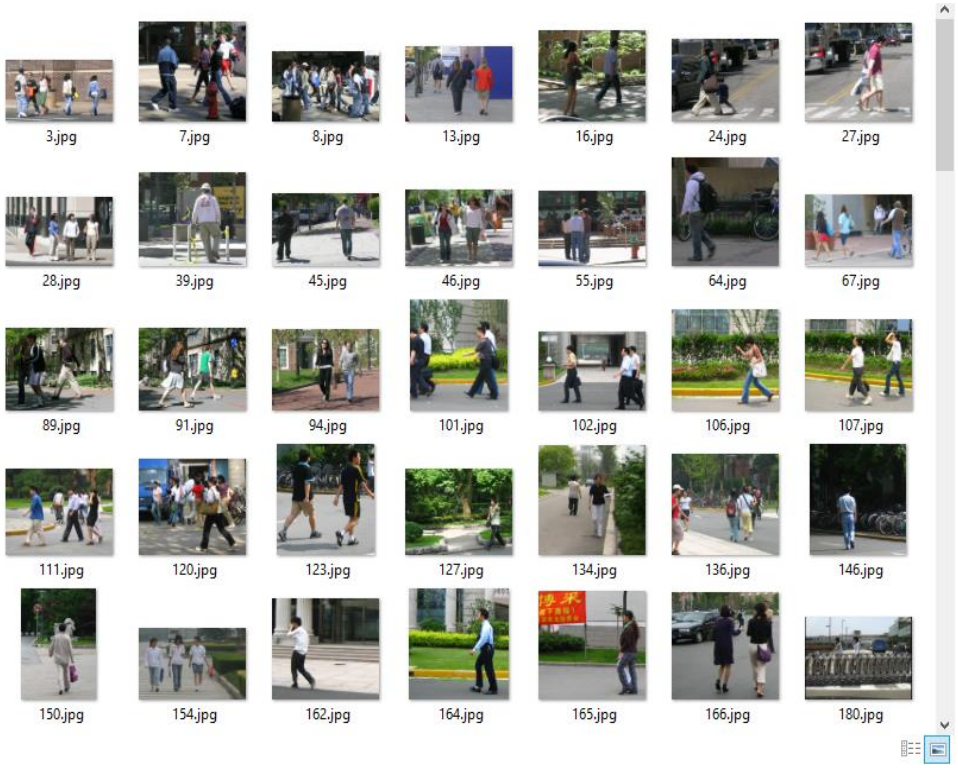

圖 32 YOLOv5 測試資料集

YOLOv5 模型在訓練過程中將 Epochs 訓練總輪數設定為 200 輪，每一輪都會使用數據集裡的測試樣本訓練一次，數量越多所花費的訓練時間越久，但精確度也越高。在 YOLOv5 模型訓練終將 Batch-size 批次的大小設定為 8，數字大小取決於顯示卡的效能，低階顯示卡建議調整 8~16 左右。Image-size 輸入圖像解析度的值，則使用預設為 640 x 640 的設定。圖 33 為使用 YOLOv5s 模型進行測試後的結果。表 7 是 YOLOv5 模型的訓練結果。表 8 是 YOLOv5 模型的測試結果。

圖 33 YOLOv5s 模型測試結果

表 7 YOLOv5 的訓練結果

	Hours	Precision	Recall	F1 Score	mAP
YOLOv5s	0.492	0.899	0.949	0.923	0.96
YOLOv5m	0.662	0.918	0.935	0.926	0.958
YOLOv5l	0.893	0.906	0.952	0.928	0.963
YOLOv5x	1.466	0.922	0.94	0.931	0.963
YOLOv5n	0.511	0.917	0.918	0.917	0.963

表 8 YOLOv5 的測試結果

	Precision	Recall	F1 Score	mAP
YOLOv5s	0.909	0.941	0.925	0.963
YOLOv5m	0.929	0.934	0.931	0.959
YOLOv5l	0.925	0.934	0.929	0.968
YOLOv5x	0.929	0.932	0.930	0.964
YOLOv5n	0.909	0.921	0.915	0.963

4.3 U-Net 的實驗結果

U-Net 模型輸入的圖像為 8 位元深度的灰階圖像，行人標記的圖像則是 8 位元深度的灰階圖像，在訓練過程中會將圖像統一 Resize 為 256 x 256。

(1) YOLOv5s 串接 U-Net 的訓練與測試結果

在 YOLOv5s 串接 U-Net 的實驗中總共切割了 3240 張的行人圖像，在此一階段中會先使用程式碼將訓練資料集以及測試資料集劃分為 8 比 2，如圖 34 、圖 35。圖 36 為 YOLOV5s 串接 U-Net 模型對切割後的行人圖像進行訓練後的測試結果。圖 37 YOLOV5s 串接 U-Net 模型 accuracy、loss、Dice Coefficient、Precision、Recall 和 F1 Score 的折線圖。表 9 為 YOLOV5s 串接 U-Net 模型的訓練與測試結果。

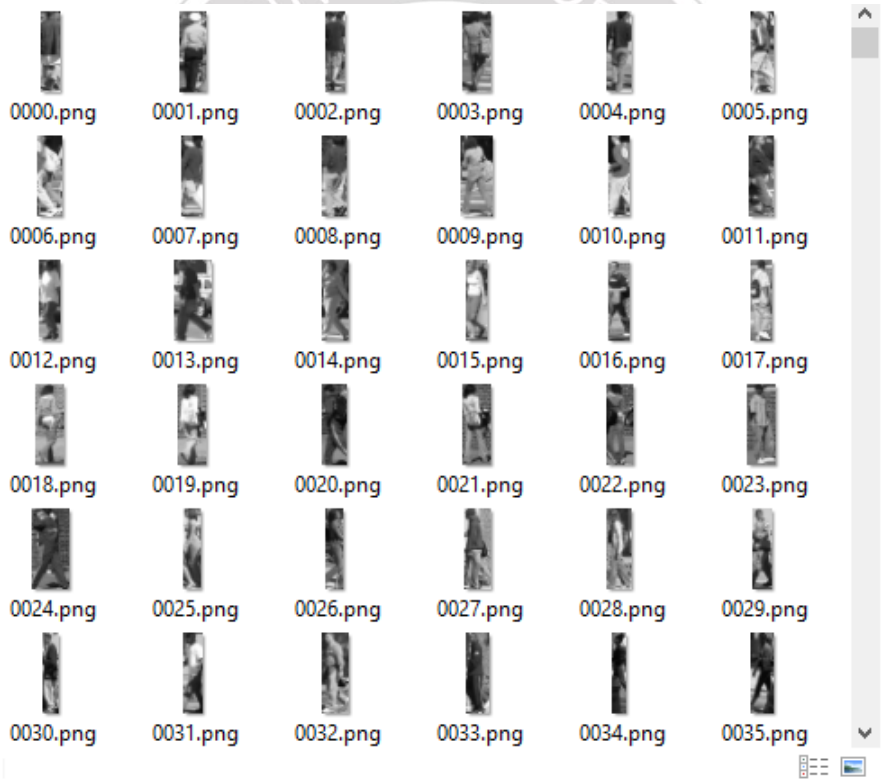

圖 34 YOLOv5s 串接 U-Net 的訓練資料集

圖 35 YOLOv5s 串接 U-Net 的測試資料集

圖 36 YOLOv5s 串接 U-Net 的測試結果

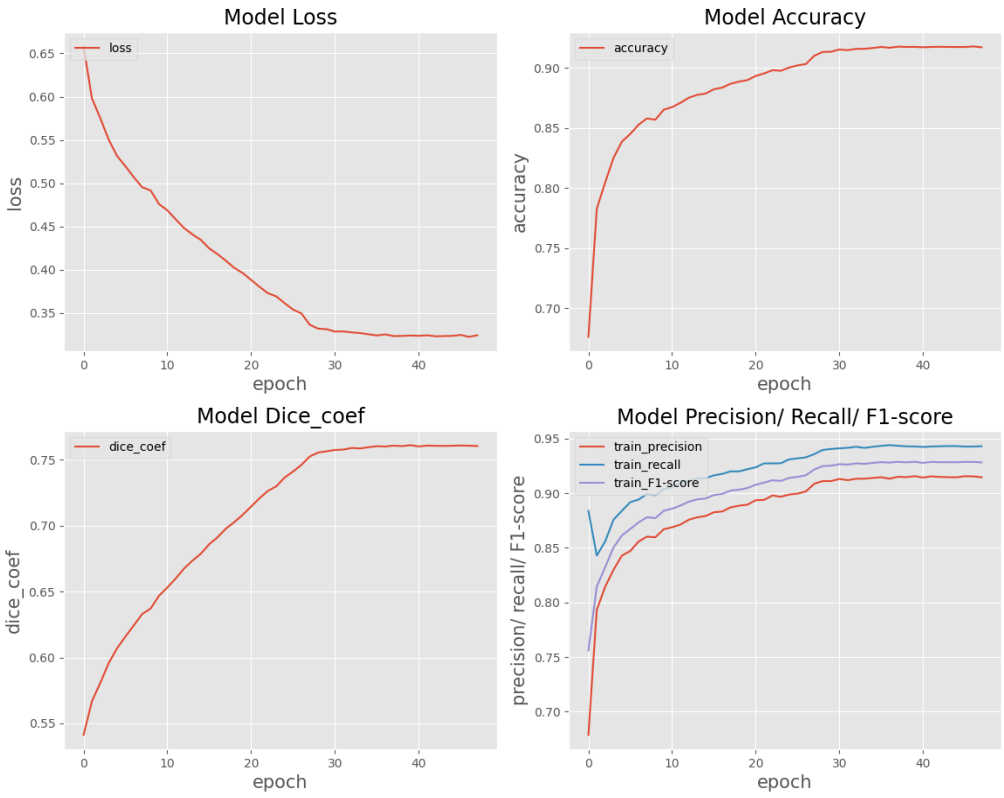

圖 37 YOLOv5s 串接 U-Net 模型折線圖

表 9 YOLOv5s 串接 U-Net 的訓練與測試結果

	Accuracy	Dice Coefficient	Precision	Recall	F1 Score
Training	0.9173	0.7605	0.9146	0.9430	0.9283
Testing	0.8868	0.7459	0.8936	0.9213	0.9069

(2) U-Net 的訓練與測試結果

在 U-Net 的實驗中的訓練與測試資料集一共為 811 張的行人圖像，在此一階段中會使用程式碼將訓練資料集以及測試資料集劃分為 8 比 2，如圖 38 、圖 39。圖 40 為 U-Net 模型對行人圖像進行訓練後的測試結果。圖 41 U-Net 模型 accuracy、loss、Dice Coefficient、Precision、Recall 和 F1 Score 的折線圖。表 10 為單獨使用 U-Net 模型進行實驗所得到的訓練與測試結果。

圖 38 U-Net 訓練資料集

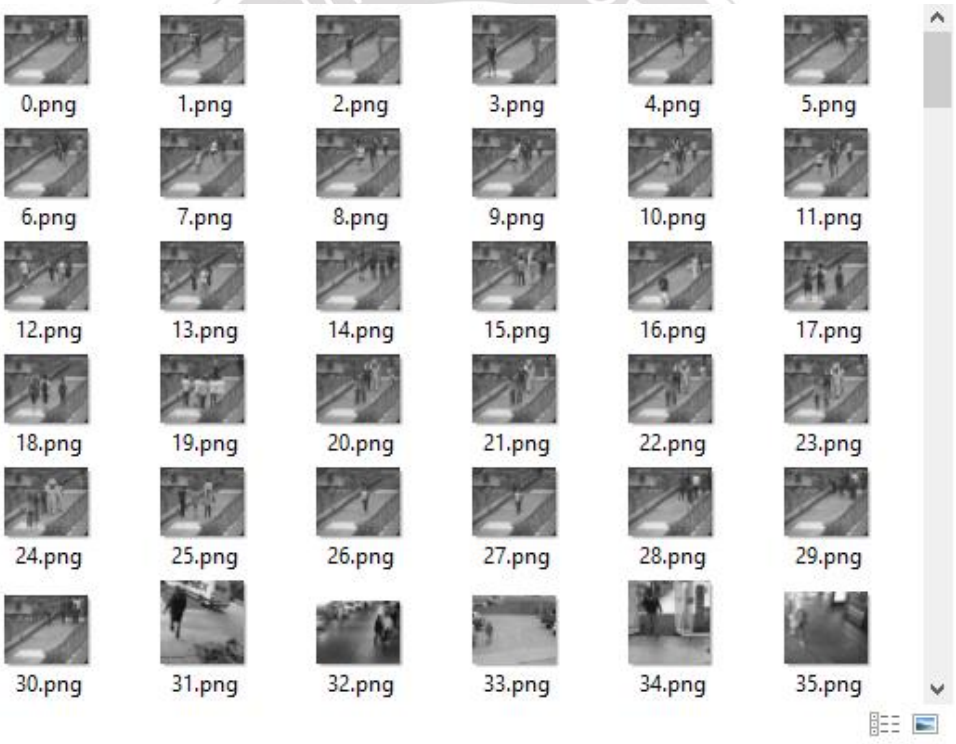

圖 39 U-Net 測試資料集

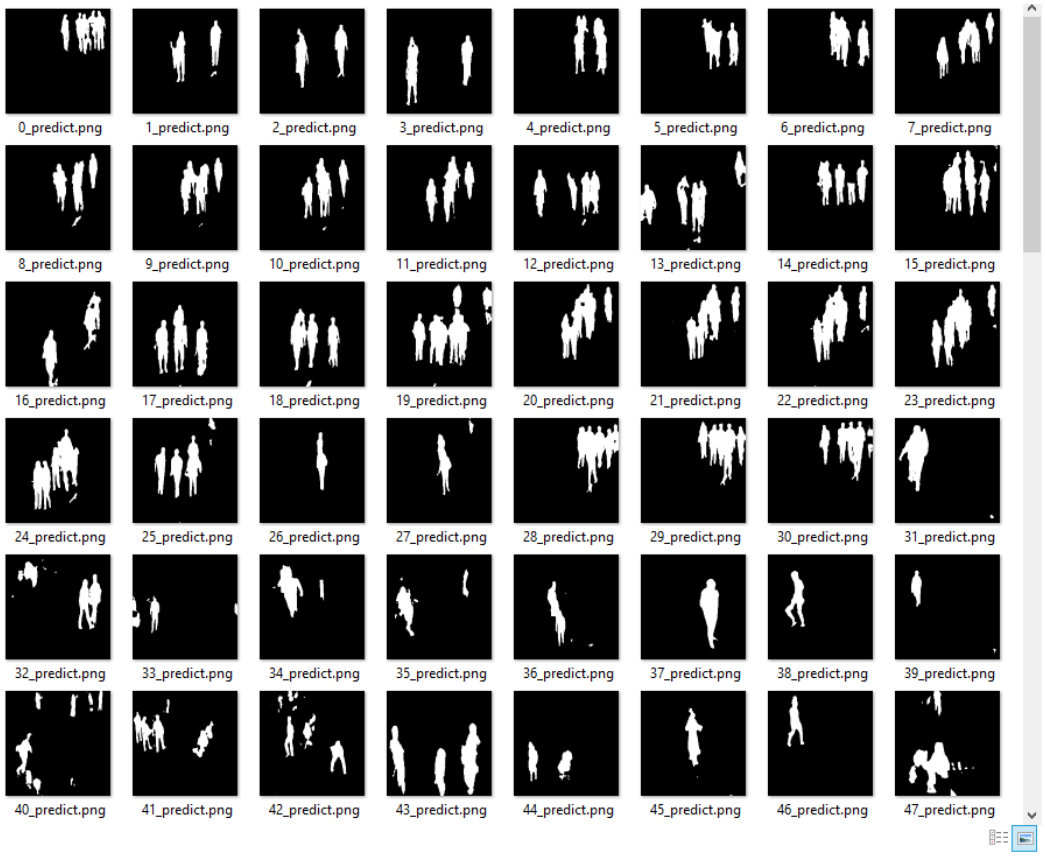

圖 40 U-Net 測試結果

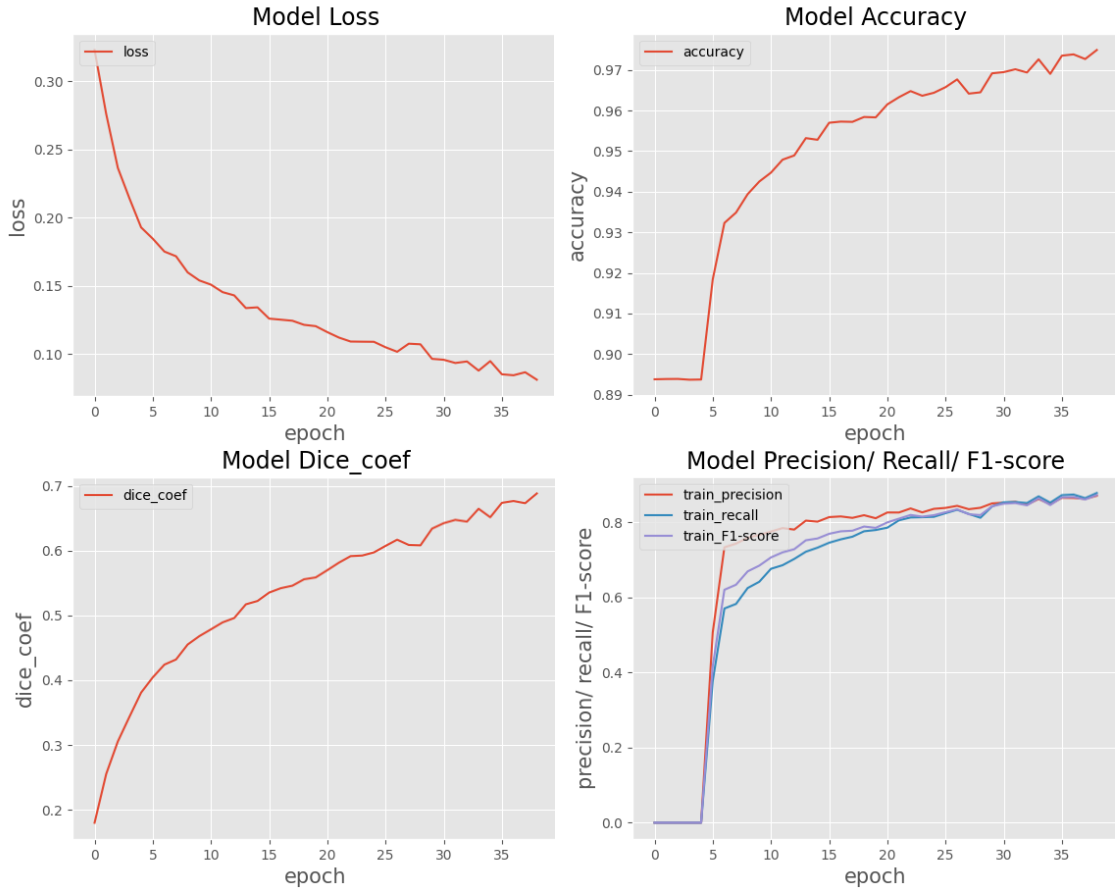

圖 41 U-Net 模型折線圖

表 10 U-Net 的訓練與測試結果

	Accuracy	Dice Coefficient	Precision	Recall	F1 Score
Training	0.9749	0.6881	0.8708	0.8781	0.8727
Testing	0.9656	0.5458	0.7037	0.7422	0.7075

4.4 實驗結果討論

在 4.2 的實驗中，首先將 24 位元的行人圖像輸入 YOLOv5 模型進行訓練和測試，然後會獲得經過 YOLOv5 模型處理後的行人區域 Bounding Box，這樣一來就可以確認行人在圖像中的精確位置。在表 7 中，我們針對 YOLOv5s、YOLOv5m、YOLOv5l、YOLOv5x 和 YOLOv5n 等模型進行了訓練並比較了訓練結果，結果顯示 YOLOv5s 所花費的時間最短，且訓練

結果的評估指標與其他四個模型的結果相近，因此我們選擇了 YOLOv5s 作為串接 U-Net 的實驗模型。在 4.3 的實驗中，分別進行了單獨使用 U-Net 的實驗和將 YOLOv5s 和 U-Net 串接的實驗，從實驗結果顯示可以發現，單獨使用 U-Net 的實驗的多項評估指標明顯低於將 YOLOv5s 和 U-Net 串接的實驗。

第五章 結論和未來研究

在本章節中，我們將針對本論文提出的行人偵測系統進行優缺點總結，並提出未來研究方向。台灣目前存在交通法規不合適、道路設計不良等問題，上述問題都不是能短時間得到解決的，而且這些因素容易導致各地交通事故頻繁發生。因此，本研究提出了一種結合 YOLOv5 模型與 U-Net 模型的行人偵測方法，結合了深度學習的圖像識別技術和人工智慧先進的駕駛輔助系統或智慧型視訊監控系統，有助於駕駛者注意到可能發生的馬路突發狀況，提高用路人的警覺並降低交通事故發生的可能性。在未來的研究中，我們希望能收集更多不同類型的行人圖像，包括小孩、老人、行動不便的身障者等，來豐富行人實驗資料集，以確保實驗的可用性和實用性，從而提升行人偵測的準確度，並改善本研究中可能存在的不足之處。

參考文獻

[1] *提高罰則、科技執法還不夠直搗核心被稱「行人地獄」的台灣——為何我們的馬路設計不安全*，Available: https://www.twreporter.org/a/sidewalk-pedestrian-traffic-safety

[2] *道安資訊查詢網*，Available: https://roadsafety.tw/Dashboard/Custom?type=%E7%B5%B1%E8%A8%88%E5%BF%AB%E8%A6%BD%E5%9C%96%E8%A1%A8#dash_item_1876

[3] W. Chang, *Taiwan's 'living hell' traffic is a tourism problem, say critics*, Retrieved from https://edition.cnn.com/travel/article/taiwan-traffic-war-tourism-intl-hnk/index.html/, last accessed 2023/3/10

[4] J. Redmon, S. Divvala, R. Girshick, A. Farhadi (2016), "You only look once: Unified, real-time object detection," *the 29th IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, pp.779-788.

[5] G. Jocher, *YOLOv5 by Ultralytics (Version 7.0) [Computer software]*, Retrieved from https://doi.org/10.5281/zenodo.3908559/, last accessed 2023/3/10

[6] W. Guo, N. Shen, and T. Zhang (2022), "Overlapped Pedestrian Detection Based on YOLOv5 in Crowded Scenes," *In 2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 412-416.

[7] M. Rieder, and R. Verbeet (2019), "Robot-human-learning for robotic picking processes," *Proceedings of the Hamburg International Conference of Logistics (HICL)*, vol.27. Berlin: epubli GmbH, pp.87-114.

[8] O. Ronneberger, P. Fischer, and T. Brox (2015), "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*. Springer,

Cham, pp.234–241.

[9] G. Li, Y. Yang, and X. Qu (2019), "Deep learning approaches on pedestrian detection in hazy weathe," IEEE Transactions on Industrial Electronics, vol.67, no.10, pp.8889-8899.

[10] L. Chen, S. Lin, X. Lu, D. Cao, Wu. Hangbin, C. Guo, C. Liu, and Wang. F. Y. (2021), "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," IEEE Transactions on Intelligent Transportation Systems, vol.22, no.6, pp.3234-3246.

[11] B. Han, Y. Wang, Z. Yang, and Gao, X. (2019), "Small-scale pedestrian detection based on deep neural network," IEEE transactions on intelligent transportation systems, vol.21, no.7, pp.3046-3055.

[12] W. Wang (2020), "Detection of panoramic vision pedestrian based on deep learning," Image and Vision Computing, vol.103, pp.103986.

[13] Z. Shao, G. Cheng, J. Ma, Z. Wang, J. Wang, and D. Li (2021), "Real-time and accurate UAV pedestrian detection for social distancing monitoring in COVID-19 pandemic," IEEE transactions on multimedia, vol.24, pp.2069-2083.

[14] TensorFlow Install. Retrieved from https://www.tensorflow.org/install/source#common_installation_problems /, last accessed 2023/6/1

[15] *Deep Learning in Medical Imaging V*. Retrieved from https://medium.datadriveninvestor.com/deep-learning-in-medical-imaging-3c1008431aaf/, last accessed 2022/9/20

[16] *Penn-Fudan Database for Pedestrian Detection and Segmentation*. Retrieved from https://www.cis.upenn.edu/~jshi/ped_html/, last accessed 2022/9/20

[17] *Robust Multi-Person Tracking from Mobile Platforms*. Retrieved from https://data.vision.ee.ethz.ch/cvl/aess/dataset/, last accessed 2023/3/10

[18] *CUHK Occlusion Dataset*. Retrieved from https://www.ee.cuhk.edu.hk/~xg

wang/CUHK_pedestrian.html/, last accessed 2023/3/10

[19] *Human Detection Dataset*. Retrieved from https://www.kaggle.com/datasets /constantinwerner/human-detection-dataset/, last accessed 2023/3/10

[20] P. Dollar, C. Wojek, B. Schiele, and P. Perona (2011), "Pedestrian detection: An evaluation of the state of the art," IEEE transactions on pattern analysis and machine intelligence, vol.34, no.4, pp.743-761.

[21] A. Ess, B. Leibe, and L. V. Gool (2007), "Depth and appearance for mobile scene analysis," In 2007 IEEE 11th international conference on computer vision, pp. 1-8.

[22] C. Wojek, S Walk, and B. Schiele (2009), "Multi-cue onboard pedestrian detection," In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp.794-801.

[23] N. Dalal, and B. Triggs (2005), "Histograms of oriented gradients for human detection," *In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol.1, pp.886-893.

[24] R. Fisher, *CAVIAR Test Case Scenarios*. Retrieved from https://groups.inf.ed.ac.uk/vision/DATASETS/CAVIAR/CAVIARDATA1/, last accessed 2023/3/10