中原大學

資訊工程學系

113 學年度專題實驗期末報告

結合三角錐偵測與施工區域分割模型之道路施工區域之判斷

指導教授：教授

組員

資訊四甲 組員 A

資訊四甲 11027164 趙怡儒

專題報告目錄

主要目錄.....2
摘要.....4
導論.....4
研究目的.....4
研究中使用之模型相關文獻與環境介紹.....6
研究方法.....13
結果與討論.....22
參考文獻.....24

圖目錄.....2
圖 1、專題使用情境與簡易步驟說明.....5
圖 2、YOLOPv2 的網路結構圖.....7
圖 3、YOLOPv2 的流程圖.....8
圖 4、YOLOv8 與前幾代的比較圖，.....9
圖 5、YOLOv8 的架構圖.....10
圖 6、Rein 方法概述.....11
圖 7、以 labelme 標注 nuScenes 施工區域資料集.....13
圖 8、工程車資料集頁面.....14
圖 9、Roboflow 標記畫面.....15
圖 10、Roboflow 標記畫面.....15
圖 11、紐澤西圍欄資料集頁面.....15
圖 12、YOLOPv2 的實作結果.....16
圖 13、右圖為 Rein 之結果，左圖為 deeplab3 之結果.....16

圖 14、將 bounding box 底邊兩點放入 list 中，用所有的點畫出凸包表示警戒範圍.....17
圖 15、用網路上影片實作模型.....18
圖 16、用原圖執行 Rein.....19
圖 17、用 YOLOPv2+YOLOv8 的執行結果再執行 Rein.....19
圖 18、將三個模型同步執行之結果-1.....19
圖 19、將三個模型同步執行之結果--2.....20
圖 20、詳細系統執行流程圖.....21
表目錄.....3
表 1、YOLOPv2 與同類型之模型比較.....6
表 2、實作環境 1.....12
表 3、實作環境 2.....12
表 4、三類施工物品訓練結果表.....18

摘要

本專題的目標為基於自駕車之避障功能系統，包含透過深度學習模型辨識出可行駛區域，接著在畫面中出現施工區域以語意分割模型標出其位置，最後是三角錐、工程車、紐澤西護欄等施工區域常見物品的物件偵測與範圍標示之三項功能，標示出道路施工區域並且警示用路人。

導論

● 研究目的

近年來，科技進步的速度飛快，食衣住行育樂等等日常瑣碎都隨之發展。當中的「行」，隨著自駕車的問世與其功能日新月異，其中避障功能最為重要，畢竟車子在路上最重要的就是安全第一，因此如何加強避障功能是本專題的目標。然而我們發現對於常常不定時出現於道路上的施工區域研究較少。由於各種因素，例如天氣變化、人為破壞、定期維護等等，施工區域常常出現在道路上，但由於它不定時、地點也不固定，因此不一定在自駕車的系統中能即時更新，所以我們認為標示出施工區域範圍是極為重要的一件事。研究[1]，結合了基於影像的工作區偵測與基於雷射雷達的工作區偵測，使其偵測的結果以鳥瞰圖呈現。基於想使結果呈現的方式更加的具體與直觀，我們專題維持影像偵測的部分，設計了包含區域本身與其他地方的語義分割，再加上三角錐、工程車、紐澤西圍欄等施工物品的物件偵測與其圍出之區域來完整標示出警示範圍，最後加上自駕車最基本的該有的可行駛區域道路分割功能。

在本專題中，我們實現以下項目：

(1) 對道路進行分割，使自駕車走在正確的區域

為了享有更安全的駕駛環境，道路的正確分割一直以來都是自駕車需要面臨的課題，如何正確標記行駛範圍十分重要。除了遵照基本的馬路線劃分區域外，如何將道路上行走的人和車即時劃分在可行駛道路的區域之外，也是十分重要的。

(2) 施工區域的語意分割

在日常生活中，我們認為在道路上常見但不可預期的危險區域就是突然發生的道路施工，因此我們想要重視這一塊。第一步就是先將危險的施工區域分割出來，使車子可以同時走在可行駛道路與避免施工區域。

(3) 施工物品的物件偵測與區域標示

語意分割模型不一定能將危險範圍完整的標示出來，例如：主要施工範圍在後方 5 公尺處，但三角錐可能放在其前方 3 公尺處提醒用路人，而連三角錐圍出來的範圍也是用路人該注意的。因此，補足語意分割模型缺少的這一塊，我們偵測三角錐、工程車、紐澤西護欄等施工物品，偵測出來後同時將其範圍圍出來，形成更大塊該注意的區域。

圖 1、專題使用情境與簡易步驟說明

綜合上述，本專題結合道路分割、施工物品物件辨識及施工區域語意分割，用來面對不定時且不定地點發生的道路施工，提高用路人的安全，用更直觀的方式表示出施工區域。從 nuScenes[2]大量的街景資料集中挑選出含有施工區域的資料集，用了 1.參考 Roboflow、paperswithcode 等資料集網站 2.自行上路拍攝兩種方法蒐集了施工物品的影片與圖片，製作了包含 8 種工程車、2 種紐澤西圍欄與三角錐的偵測模型。

- 研究中使用之模型相關文獻與環境介紹

此專題用到的模型有負責可行駛區域分割的 YOLOPv2 [9]、物件偵測的 YOLOv8 [12]、工程區語意分割的 Rein[10]，而這三個模型都是基於 PyTorch[13]構建的，以下分別介紹。

1. YOLOPv2 [9]

YOLOPv2 是一種高效的多任務學習網路，專為全景駕駛感知設計，能同時進行交通物體檢測、可駕駛區域分割及車道線檢測。YOLOPv2 在前一代 YOLOP 的基礎上進行了改進，並且比其前一代更快、更準確，獲得了更好的性能評價，在與其他同性質的模型 HybridNets 比較中也十分突出。以下為其 github 中 YOLOP、HybridNets、YOLOPv2 在幾個分項中的比較數據。

表 1、YOLOPv2 與同類型之模型比較

Model	Size	Params	inference Speed (fps)	Object Detection Result mAP@0.5 (%)	Drivable Area Segmentation mIoU (%)	Lane Line Detection Accuracy (%)
YOLOP	640	7.9M	49	76.5	91.5	70.5
HybridNets	640	12.8M	28	77.3	90.5	85.4
YOLOPv2	640	38.9M	91	83.4	93.2	87.3

這篇論文基於現有模型如 YOLOP 和 HybridNets 設計了一個更高效的網路架構。YOLOPv2 受 YOLOP 和 HybridNets 啟發，保留了其核心設計概念，但使用了更強大的特徵提取主幹。此外，與過去的工作不同，這篇論文將每個任務分開處理，而非像以前那樣在同一個分支中同時進行可駕駛區域分割和車道偵測。這種做法的原因在於，交通區域分割和車道偵測的難度不一樣，因此它們在特徵層上有不同需求，這使得分開的網路結構更有效。在 YOLOPv2 的編碼器部分，它採用了 E-ELAN 設計來利用 group 卷積技術，這不僅減少了計算成本，還可以捕捉到多樣的特徵。與 YOLOP 使用 CSPDarknet 不同，YOLOPv2 通過 group 卷積學習更多不同層次的特徵，並在特徵金字塔網路（FPN）模組中融合不同語意層次的特徵。

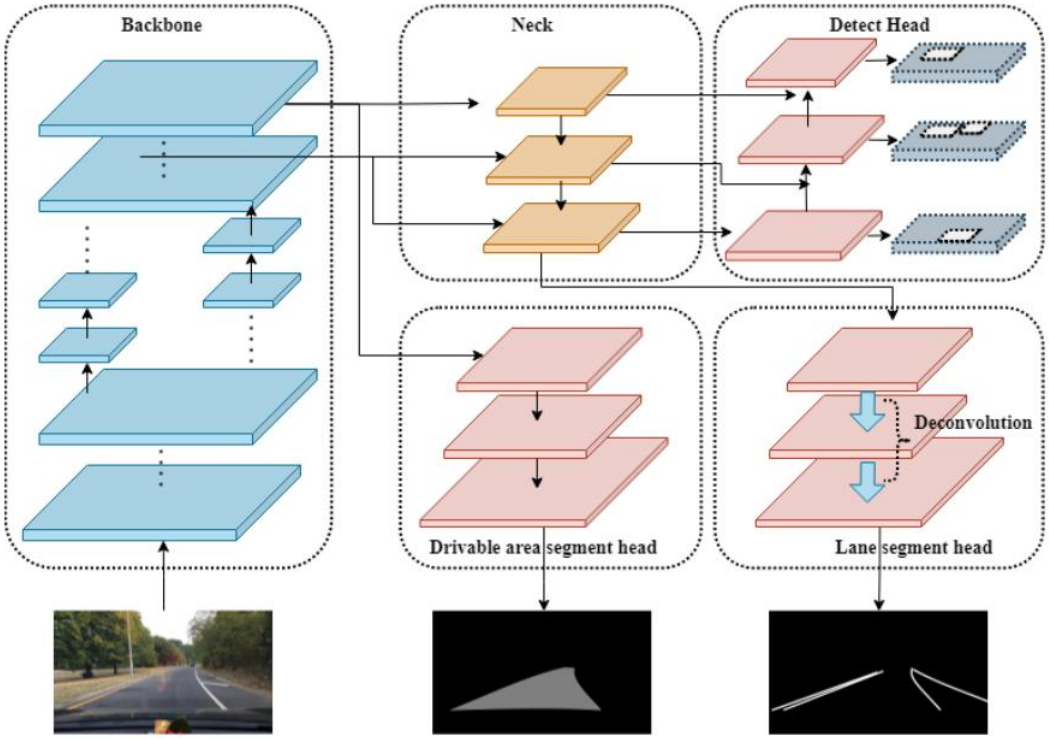

圖 2、YOLOPv2 的網路結構圖，來源 https://arxiv.org/pdf/2208.11434

在檢測頭方面，YOLOPv2 為每個任務設計了專門的解碼器頭，採用了類似於 YOLOv7 的 anchor-based 多尺度檢測方法。該設計結合了 PAN 和 FPN 的特徵，能更好地融合語意資訊與定位特徵。每個網格會分配三種不同縱橫比的錨點，並預測物體的位置信息、類別機率和置信度。

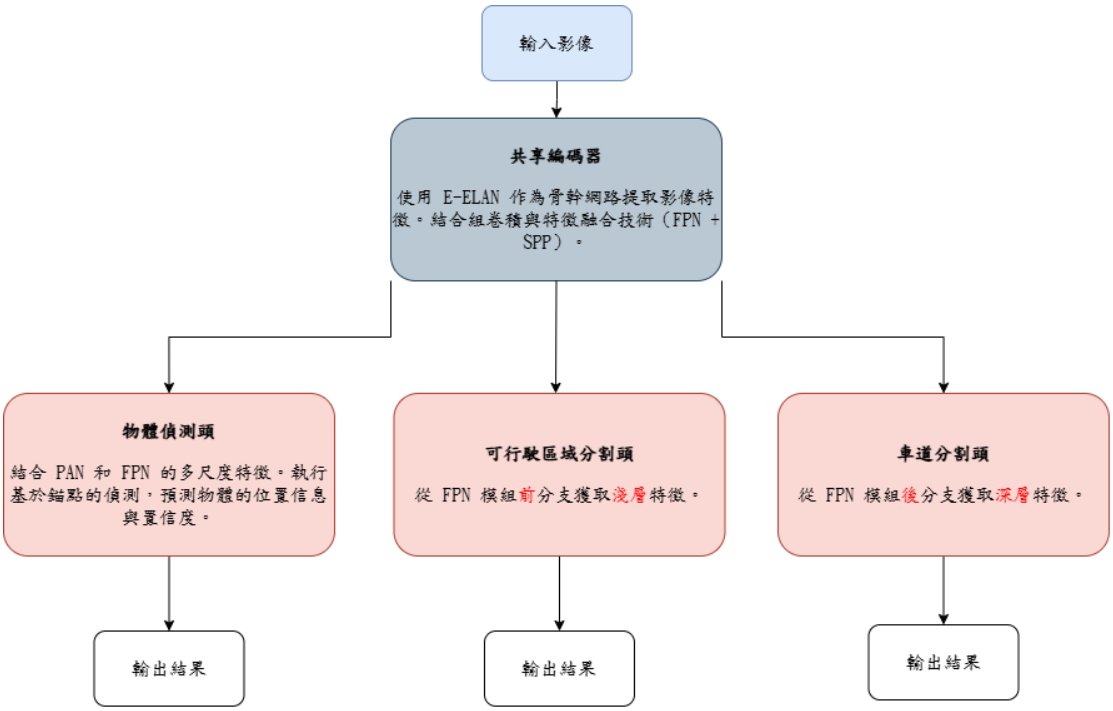

圖 3、YOLOPv2 的流程圖

可行駛區域分割和車道線分割分別在不同的任務頭中進行，並採用了不同的網路結構。與 YOLOP 相比，YOLOPv2 採用從 FPN 前的一層提取的特徵來進行可駕駛區域的分割，而不是使用更深層的特徵，這樣能避免增加模型訓練的難度。為了抵消這種變化帶來的可能損失，還應用了額外的上採樣層。對於車道線分割，該任務分支連接到 FPN 層的末端，因為車道線通常不明顯，需要更深層的特徵來檢測，並且在解碼器中使用反捲積進一步提升了性能。

2. YOLOv8 [12]

YOLOv8 是 YOLO 系列即時物體檢測器在 2023 年初發行的版本，在準確性和速度方面提供尖端性能。基於之前 YOLO 版本的進步，YOLOv8 引入了新功能和優化，使其成為各種物體檢測任務的理想選擇，適用於廣泛的應用領域。

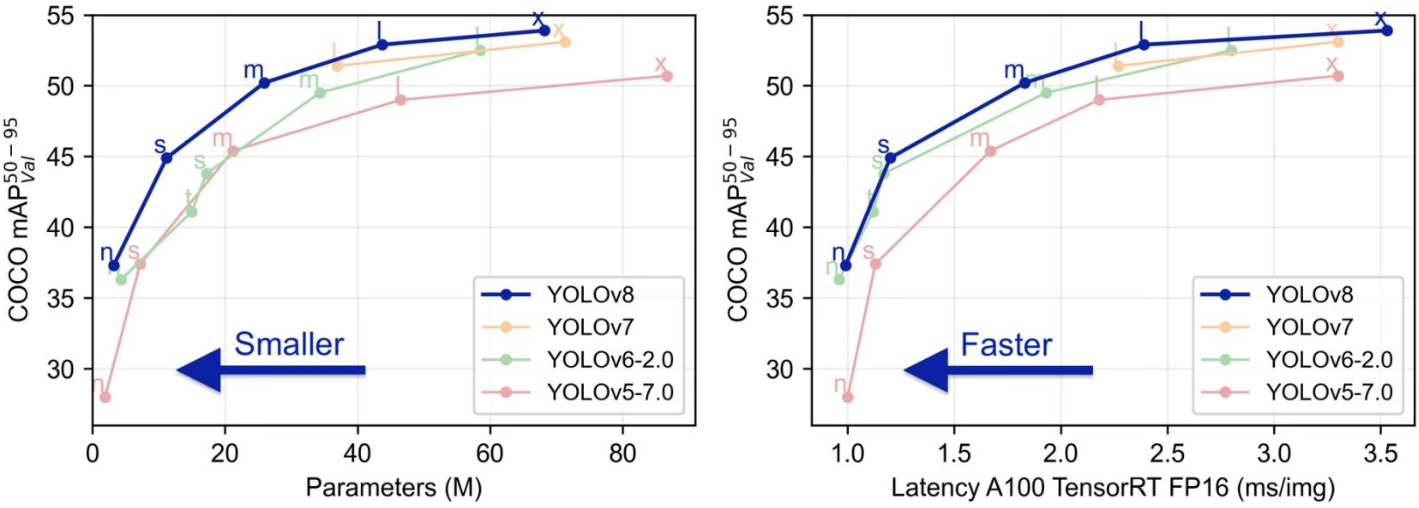

圖 4、YOLOv8 與前幾代的比較圖，來源 https://docs.ultralytics.com/models/yolov8/

YOLOv8 主要特點首先是先進的骨幹和頸部架構：YOLOv8 採用最先進的骨幹和頸部架構，從而改善特徵提取和物體檢測性能。再來是無錨點分割的 Ultralytics head：YOLOv8 採用無錨點的分割 Ultralytics head，這使得與基於錨點的方法相比，準確性更高且檢測過程更高效。接下來是優化的準確性與速度權衡：YOLOv8 專注於保持準確性與速度之間的最佳平衡，適合在多樣化應用領域中進行實時物體檢測任務。最後是多樣的預訓練模型：YOLOv8 提供一系列預訓練模型，以滿足各種任務和性能需求，使使用者更容易找到適合特定用例的模型。每個模型專門針對計算機視覺中的特定任務。這些模型旨在滿足各種需求，從物體檢測到更複雜的任務，如實例分割、姿勢/關鍵點檢測、定向物體檢測和分類。每個 YOLOv8 系列的變體都針對其各自的任務進行了優化，確保高性能和高準確性。此外，這些模型與各種操作模式兼容，包括推理、驗證、訓練和導出，便於在不同的部署和開發階段使用。

YOLOv8 的網路結構主要由以下部分組成：

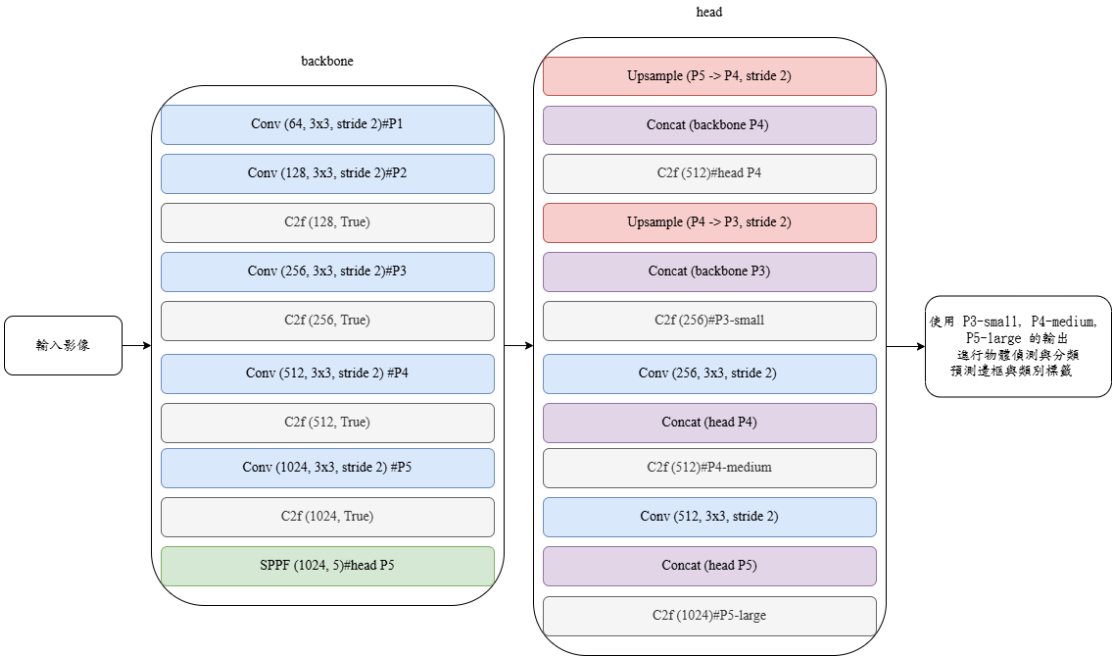

圖 5、YOLOv8 的架構圖，參考 YOLOv8 的 yaml 檔

Backbone：使用卷積和反卷積層來提取特徵並使用了殘差連接和瓶頸結構來減少網路的大小。使用 C2f 作為 basic unit，與前幾代 YOLOv5 的 C3 相比，C2f 具有更少的參數和更好的特徵提取。

Head：使用多尺度特徵融合，將 backbone 不同階段的特徵圖融合增強特徵表現能力。

最後的目標檢測和分類任務，包含了一個檢測頭和分類頭。檢測頭包含了卷積層和反卷積層，用於生成檢測結果。分類頭使用全局平均池化層對每個特徵圖分類

3. Rein[10]

此論文首先分析了多種視覺基礎模型（Vision Foundation Models, VFMs）在領域泛化語意分割（Domain Generalized Semantic Segmentation,DGSS）任務中的表現，包括 CLIP、MAE、SAM、EVA02 和 DINOv2。研究發現，即使在未進行微調的情況下，這些 VFMs 也能顯著超越過去最佳模型的表現水平。

為了克服 VFMs 在小型 DGSS 數據集上可能出現的過擬合問題，作者提出了一種名為 Rein 的新型微調方法。Rein 方法通過使用較少的可訓練參數來提高 VFMs 在 DGSS 任務中的泛化能力。

Rein 方法的核心機制是一組可學習的標記（tokens），每個標記直接對應到影像中的不同實例。這些標記通過與 VFMs 特徵的點積操作產生類似注意力的相似度圖，從而實現對每個實例的精確細化。此外，為了減少參數數量，Rein 方法在不同層之間共享 MLP 權重，並使用低秩矩陣生成標記序列。

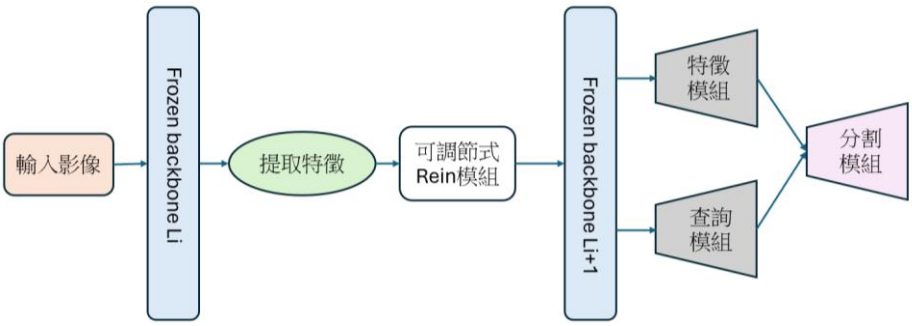

圖 6、Rein [10]方法概述，來源 https://arxiv.org/pdf/2312.04265

在多個資料集包括 Cityscapes、BDD100K 和 Mapillary 上的實驗來驗證 Rein 方法的有效性。實驗結果表明，Rein 在各種設定下都顯著超越了現有的 DGSS 方法，即使在只增加 1%的可訓練參數的情況下，也能達到與全參數微調相當的性能。

4. Pytorch[13]

PyTorch 是一個開源的 Python 機器學習庫，基於 Torch 庫，底層由 C++實現，應用於人工智慧領域，如電腦視覺和自然語言處理。它最初由 Meta Platforms 的人工智慧研究團隊開發，現在屬於 Linux 基金會的一部分。它是在修改後的 BSD 授權條款下發布的自由及開放原始碼軟體。 儘管 Python 介面更加完善並且是開發的主要重點，但 PyTorch

也有 C++介面，許多深度學習軟體都是基於 PyTorch 構建的。而 PyTorch 主要有兩大特徵： 第一是類似於 NumPy 的張量計算 Tensor，能在 GPU 或 MPS 等硬體加速器上加速，第二是基於帶自動微分系統的深度神經網路。

PyTorch 的設計直觀、線性且易於使用。每行程式碼都會即時執行，不涉及異步的運作模式，因此當進入偵錯器或遇到錯誤訊息時，堆疊追蹤清晰明瞭，能準確指向問題所在，減少調試時間。

PyTorch 框架的開銷極小，並整合了 Intel MKL 以及 NVIDIA 的 cuDNN 和 NCCL 等加速庫，極大提升運算速度。其核心的 CPU 和 GPU Tensor 以及神經網路後端經過多年測試，性能穩定，無論是小型還是大型神經網路，皆具備優秀的運行效率。

與 Torch 或其他替代方案相比，PyTorch 在記憶體使用方面更加高效。PyTorch 的 GPU 記憶體管理是自訂的，這讓深度學習模型能夠最大化地利用記憶體，使得訓練更大型的模型成為可能。

5. 使用環境

本專題到目前為止的訓練、實作皆在以下環境：

設備電腦一

主要實作 YOLOv8 的訓練、實作

表 2、實作環境 1

Items	規格
Paltform	ASUS TUF Gaming F15
CPU	12th Gen Intel(R) Core(TM) i7-12700H
System Ram	16GB
GPU	NVIDIA Geforce RTX 4050
OS	Window11
Python	3.11
Package	Torch, opencv, mmengine, mmseg, mmcv, mmsegmentation, mmdet, xformers

設備電腦二

主要實作 Rein 的訓練、實作

表 3、實作環境 2

Items	規格

Paltform	ASUS ROG ZEPHYRUS G14
CPU	AMD Ryzen 9 7940HS w/ Radeon 780M Graphics
System Ram	16GB
GPU	NVIDIA Geforce RTX 4080
OS	Window11
Python	3.11
Package	Torch, opencv, mmengine, mmseg, mmcv, mmsegmentation, mmdet, xformers

- 研究方法

1. 資料集來源與製作

(1) 施工區域

因主要參考論文 Work Zone Detection For Autonomous Vehicles [1]中提到他們是用 nuScenes[2]額外補充的工作區註釋，但在網路上沒有將其資料集與標示方法公開，nuScenes[2]中包含前、左前、右前、後、右後和左後之六個方向的鏡頭，所以我們決定從 nuScenes[2]的前方鏡頭中找出大約 1000 張的含有施工區域的照片，並以 labelme[3]為工具做出我們施工區域的資料集，標示好後再以水平翻轉的方式擴充資料集，最後共 2000 張左右。

圖 7、以 labelme 標注 nuScenes 施工區域資料集

(2) 施工物品

a. 三角錐

資料集來源為 paperswithcode 上面的 TraCon[4]，共 540 張左右。paperswithcode 成立於 2018 年 7 月，初衷是希望能夠幫助機器學習的愛好者追蹤最新發布的論文及源代碼，快速了解最新的技術進展。網站上廣泛涉及 ML 各個領域，包含 CV、NLP、醫療、語音、遊戲、時序、音訊、機器人、音樂、推理、計算機代碼等方面的內容。網站所有內容都是可編輯和版本化的。

b. 工程車

資料集來源為 Roboflow，它是一個專門管理影像數據的平台，目標是幫助使用者更有效地管理和處理圖像數據，我們原先採用了第一個 Construction Vehicle Detection Computer Vision Project[5]，在後面的實際操作發現它對於工程車背面的精準度不佳，因此多加了 Construction Vehicle Detection Dataset[6]還有自己拍的資料集[7]來補足，三個資料集加起來共 15000 張左右，種類分別有 Bulldozer 推土機, Dump Truck 砂石車, Excavator 挖土機, Grader 平土機, Loader 鏟土機, Mixer Truck 混泥土車, Mobile Crane 吊車, Roller 壓路機八種。

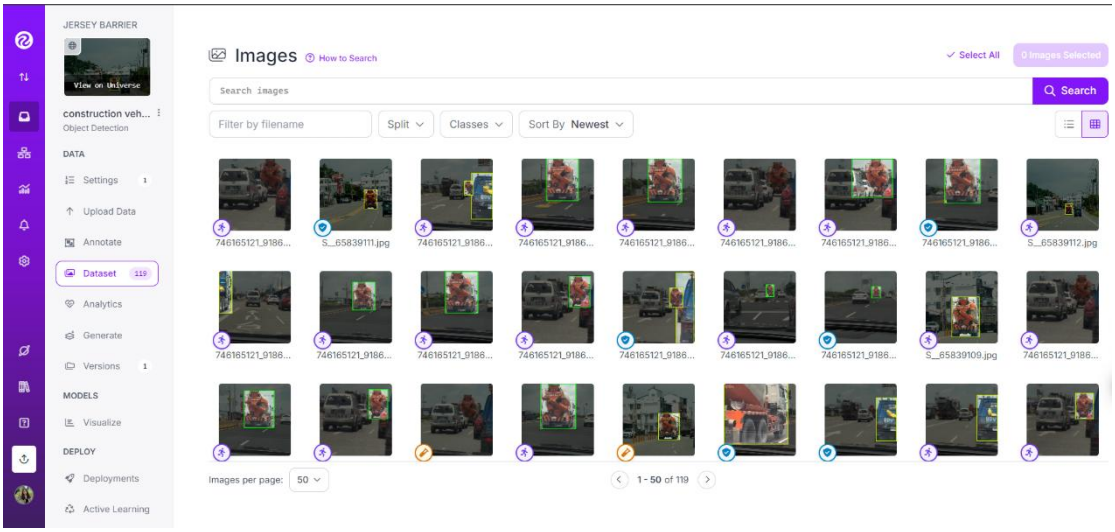

圖 8、工程車資料集頁面

c. 紐澤西圍欄

這個資料集在網路上找不太到相關資料，但我們一致認為在台灣的道路施工場域它是最常出現的物品之一。因此自己拍攝影片蒐集資料集，再藉由 Roboflow 的影片切畫面功能，將一秒切成 3-5 張照片不等，蒐集分別為水泥製的紐澤西圍欄與塑膠製的紐澤西圍欄兩種種類，並且藉由 Roboflow 標記，為 Jersey barrier Dataset[8]，共 1000 張左右。種類分別有水泥、塑膠的紐澤西圍欄兩種。

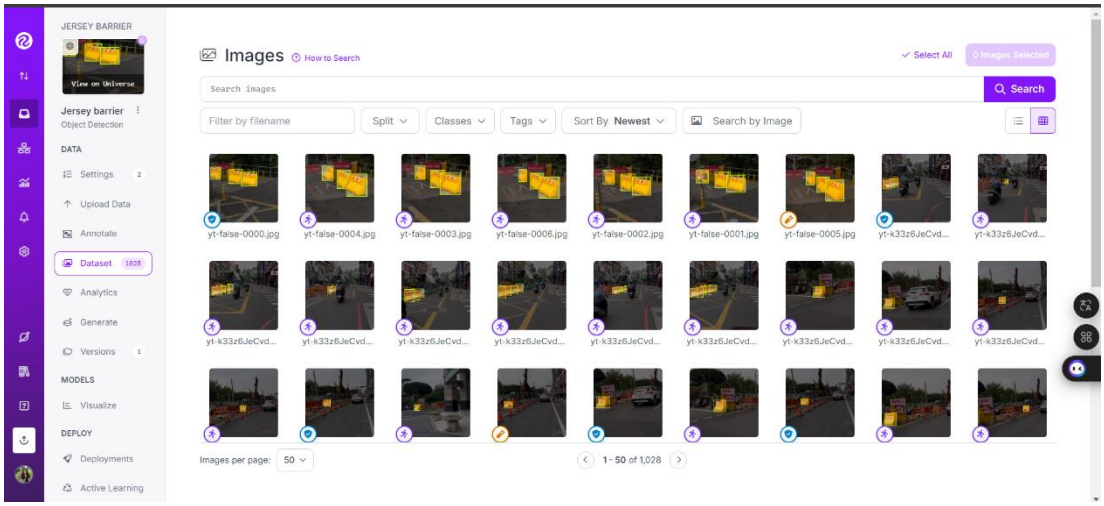

圖 11、紐澤西圍欄資料集頁面

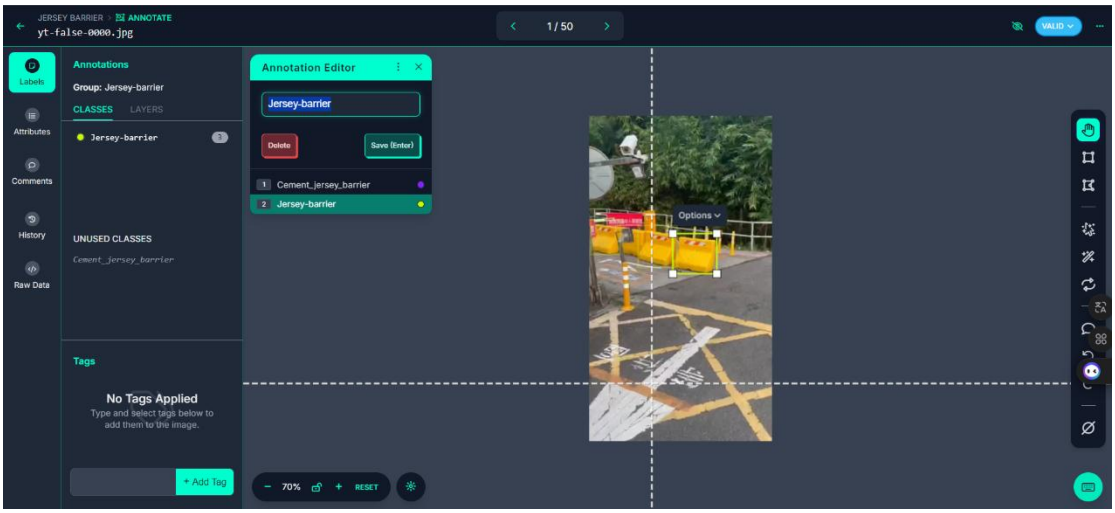

圖 10、Roboflow 標記畫面

2. 模型的選擇與運行結果

(1) 道路分割模型-- YOLOPv2 [9]

我們使用非常著名的預訓練模型 YOLOPv2 作為我們的道路分割模型，這是因為在自駕車技術中，分辨哪裡是可行駛的區域是其核心功能之一。自駕車需要準確地識別路面、車道線以及各種障礙物，以確保安全。因此，選擇一個已經在此方面非常成熟的預訓練模型來完成這一任務。

圖 12、YOLOPv2 的實作結果

(2) 施工區域的語意分割模型—Rein[10]

原本使用 deeplab3[11]做為語義分割模型，但它在有紅色的物體例如：紅色汽車、紅色招牌，出現時都會將其當作施工區域，結果與我們希望的不同，因此改使用 Rein[10]，下面圖例為左圖 deeplab3 右圖 Rein 的比較圖，可以發現 Rein 誤判率較低且範圍更完整。

圖 13、右圖為 Rein 之結果，左圖為 deeplab3 之結果

(3) 施工物品的物件偵測與區域標示-- YOLOv8 [12]

施工物品的物件偵測皆以 YOLOv8 [12]訓練，用模型偵測後，以演算法畫出施工物品所圍出的範圍，在前期實作中是將每個施工物品的 bounding box 底邊兩點放入 list 中，用所有的點畫出凸包表示警戒範圍，但遇到左右車道同時都有三角錐時，會從主要行駛的右側車道一路連到左側車道，與我們預想之結果不符，演算法為將每個施工物品的 bounding box 底邊兩點放入 list 中，先以畫面的一半(1280/2=640)的 pixel 值為基準，只要點 a 連去點 b 與點 c 的距離同時大於 640，就將點 a 移除，排除從右側車道連去左側車道的可能性，再將剩餘的點以凸包的方式圍出區域。

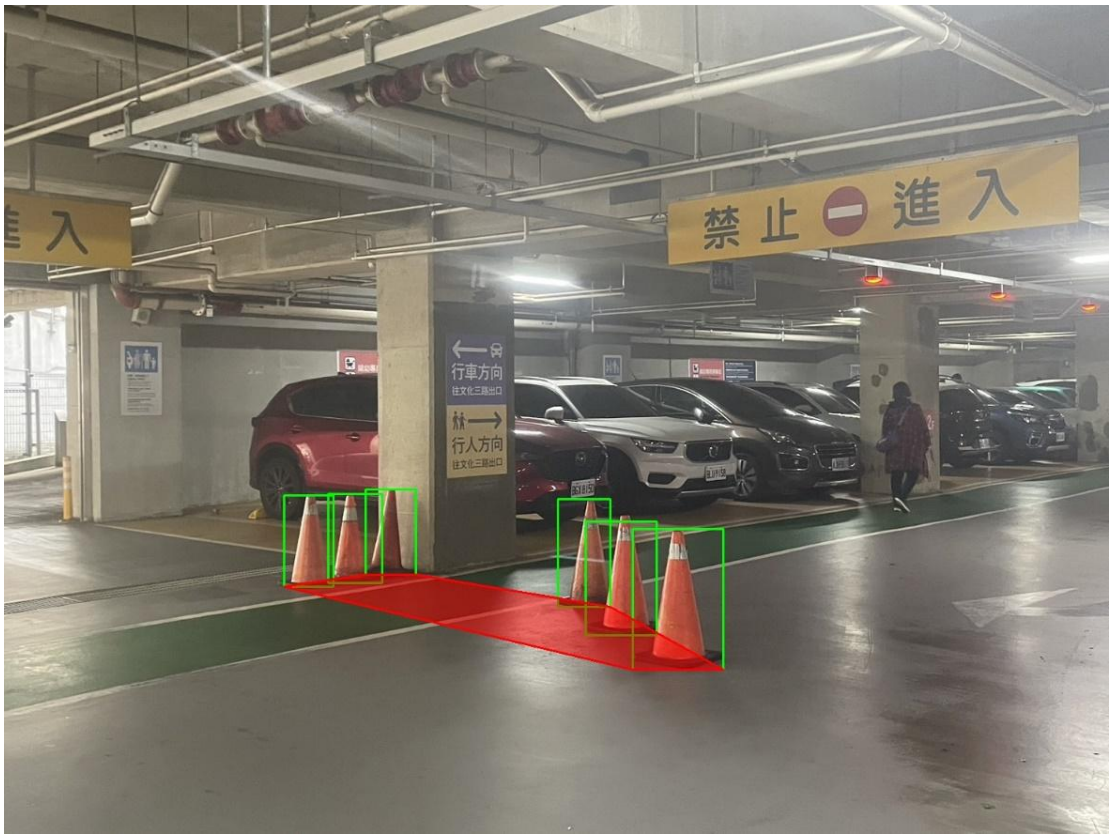

圖 14、將 bounding box 底邊兩點放入 list 中，用所有的點畫出凸包表示警戒範圍

訓練結果：

表 4、三類施工物品訓練結果表

種類	Epochs	Precision	Recall	mAP50
三角錐	100	0.98109	0.89946	0.95958
工程車	100	0.93602	0.90058	0.95021
紐澤西圍欄	100	0.95647	0.96492	0.98827

以 YOLOv8 [12]最輕量的模型 YOLOv8n 訓練資料集，訓練的主要參數為 epochs=100、batch=16、workers=8 等等，除了 epochs 其他的參數皆與其的預設參數相同，可以看到 mAP50 皆大於 95%。

(4) 結合以上模型

結合 YOLOPv2 [9] 與 YOLOv8 [12]後可以看到有些三角錐圍出的範圍(紅色半透明凸包)，YOLOPv2 依舊判定是可行駛範圍(綠色區域)，工作區的標示與警戒確實是一件該注意的事情。

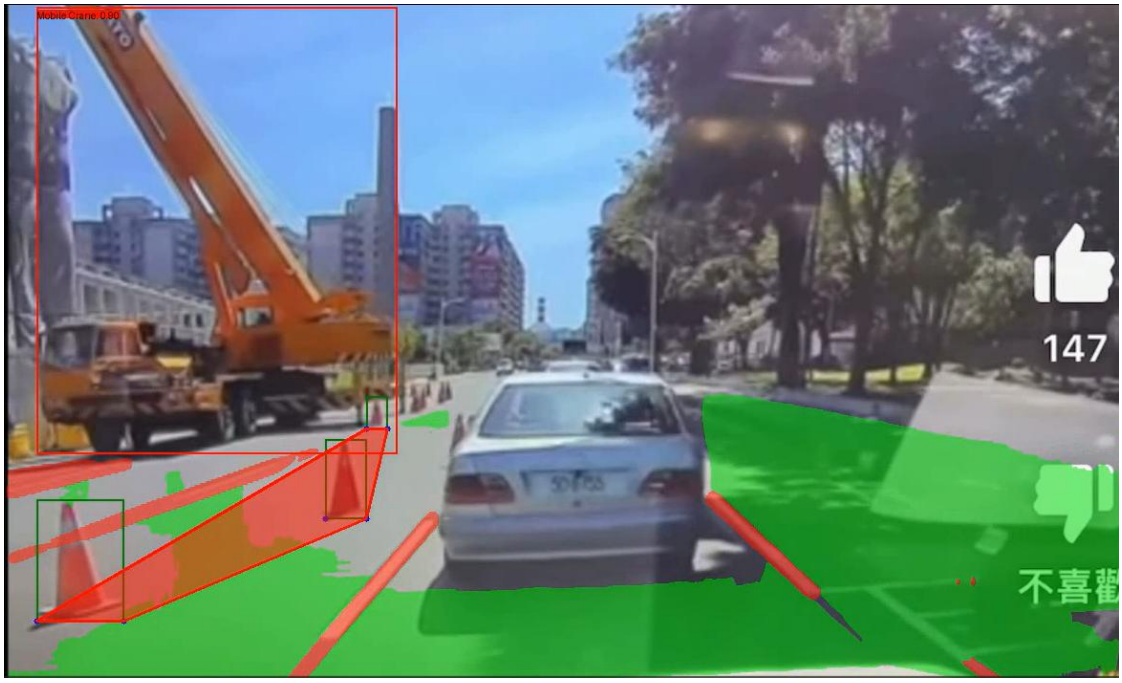

圖 15、用網路上影片實作模型，來源 https://www.youtube.com/shorts/ZwDq_SbLYcg

最終的目標是結合以及同步所有以上的模型使結果更加完善，下圖為結合 YOLOPv2 [9] 與 YOLOv8 [12]後，同時實驗原圖跑 Rein[10]和結合過後的結果圖再拿去跑 Rein[10]的比較，可以看到原圖跑的成效較好。

圖 16、用原圖執行 Rein

圖 17、用 YOLOPv2+YOLOv8 的執行結果再執行 Rein

以下的最終結果就是先執行 YOLOPv2 [9]，後執行 YOLOv8 [12]，再將 YOLOv8 [12]算出的結果存進 list 用我們寫的演算法畫出施工區域範圍，最後再執行 Rein[10]以得到完整的施工區域警戒圖。

圖 18、將三個模型同步執行之結果-1

圖 19、將三個模型同步執行之結果--2

3. 詳細的演算法與系統結構

在經過實作後，我們認為自駕車最重要的第一步依然是先分割出可行駛區域範圍，要先知道哪裡可以行駛才能為後續的動作打好基礎。接下來就是與發想過程不太一致的地方，因為在實驗的過程中，發現語意分割模型會佔走多數我們的模型執行時間，所以原本的想法是先語意分割出施工區域再用物件辨識模型補足結果，後來改成先用物件辨識模型確定畫面中有潛在的施工區域後再進行語意分割，其他時間語意分割就 6 秒執行一次即可，畢竟道路上並不是時時刻刻都會存在施工區域，這樣的運行結果可以更加即時且更加符合現實狀況。

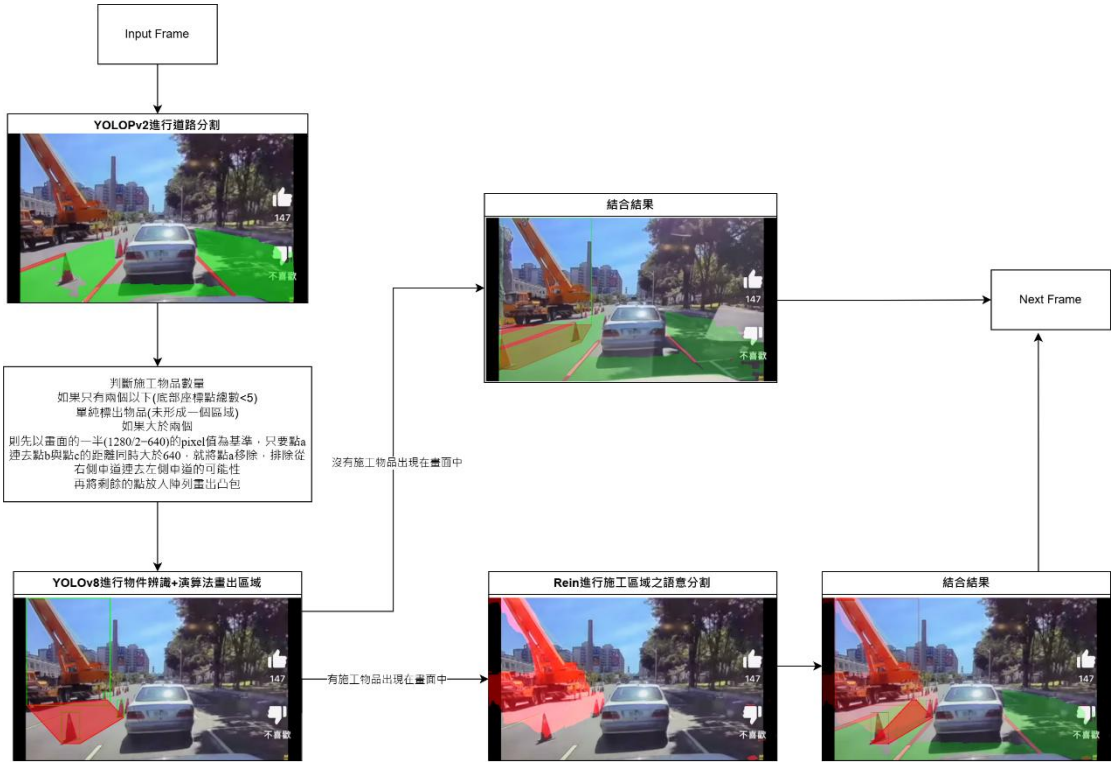

圖 20、詳細系統執行流程圖

而在 YOLOv8 中，我們為了避免連到對向車道的三角錐，我們嘗試了以下幾種方法，第一種是用數學方法算出離群值，這會導致若只有兩個三角錐分別在前方與對向的話，還是會連過去。第二中是以凸包之質心為圓心，設平均距離為半徑，會遇到若是對向車道的三角錐與同向的數量不分軒輊，依然會連到對面。最後決定以畫面的一半(1280/2=640)的 pixel 值為基準，只要點 a 連去點 b 與點 c 的距離同時大於 640，就將點 a 移除，排除從右側車道連去左側車道的可能性，再將剩餘的點以凸包的方式圍出區域。輸入可以有兩種格式，分別是 720P(1280*720)的圖片、720P 30fps 的影片。

最後，我們將所有功能以 GUI 的方法整合，只要有影片的路徑，就可以選取功能進行偵測。

圖 21、GUI 畫面呈現

總結以上來說，一開始是用 YOLOPv2 做自駕車最為重要的可行駛範圍分割，再來使用 YOLOv8 進行施工物品的偵測，偵測完後判斷施工物品數量，若只有兩個物品，則形成不了一個區域，所以僅標出施工物品的 bounding box，否則就將每個 bounding box 底邊兩點放入 list 中，再用我們的演算法排除可能連到對向車道的可能，最後 Rein 有兩個判斷的時機，第一時機是如果 YOLOv8 有結果傳到主程式的情況，Rein 就執行語意分割，第二個時機則是若連續六秒都沒有施工物品的情況，Rein 還是執行語意分割以避免沒有施工物品但有施工區域的情況，最後加上 GUI 以整合模型，以上就是我們的實作流程。

- 結果與討論

目前專題還在進行中，因此還未即時的同步全部功能。本專題預計包含施工區域辨識警告、行駛區域標記等功能並進行整合。攝影機捕捉畫面後交給深度學習模型規劃自駕車的可行駛區域，接著對畫面中的三角錐、工程車、紐澤西護欄等施工區域常見物品進行偵測，同時對施工區域加以標註與警示。在研究的過程中，一開始我們非常意外對於路上施工區域的研究少之又少，甚至沒有資料集能夠讓我們直接使用，所以在專題前期收集資料集花費了我們許多的精力與時間。在嘗試 YOLOv8 偵測後畫出所有 bounding box 的底邊兩點所形成的凸包，發現有可能連到對向車道的情

況，處理此情況也是我們專題中最棘手的任務。接下來三個模型的同步即時性也是本專題面臨的挑戰之一，目前還在嘗試解決，但相較於最一開始的一幀 6 秒已經快了 6 倍，希望在專題實驗的最後時間可以解決此問題。最後，期待此計畫能讓未來自走車的行駛和路徑規劃更加安全且有保障。

參考文獻

[1] Work Zone Detection For Autonomous Vehicles。取自 https://ieeexplore.ieee.org/document/9565073。

[2] nuScenes。取自 https://www.nuscenes.org/。

[3] Labelme。取自 https://github.com/wkentaro/labelme

[4] TraCon。取自 https://paperswithcode.com/paper/tracon-a-novel-dataset-for-real-time-traffic

[5] Construction Vehicle Detection Computer Vision Project。取自 https://universe.roboflow.com/capstone-lkzgq/construction-vehicle-detection-pxc7c

[6] Construction Vehicle Detection Dataset。取自 https://universe.roboflow.com/capstone-lkzgq/construction-vehicle-detection-pxc7c

[7][8]Roboflow。取自 https://roboflow.com/，其中自己的資料集有 https://universe.roboflow.com/jersey-barrier/construction-vehicle 跟 https://universe.roboflow.com/jersey-barrier/jersey-barrier

[9] YOLOPv2。程式碼取自 https://github.com/CAIC-AD/YOLOPv2

論文取自 https://arxiv.org/pdf/2208.11434

[10] Rein。程式碼取自 https://github.com/w1oves/Rein

論文取自 https://arxiv.org/pdf/2312.04265

[11] deeplab3。程式碼取自 https://github.com/VainF/DeepLabV3Plus-Pytorch

論文取自 https://arxiv.org/pdf/1802.02611v3

[12] Yolov8。取自 https://github.com/ultralytics/ultralytics

[13] pytorch，取自 https://github.com/pytorch/pytorch