基於深度學習模型之多車種辨識系統應用於嵌入式系統

Deep Learning-Based Multi-Class Vehicle Detection System Applied to Embedded Systems

王得貴
明志科技大學電機工程系助理教授

芮紹棠
明志科技大學電機工程系

摘要

隨著交通路網擴張與用車需求持續成長，行車安全與交通管理效率成為重要議題。近年來智慧運輸系統、先進駕駛輔助系統等技術快速發展，期望透過資訊化與自動化技術降低事故發生率並提升交通效率，此類應用高度仰賴對車流量、車流密度與車速等交通資訊之即時獲取與準確分析。現行智慧交通辨識系統多採用雲端運算架構，運作高度仰賴穩定之網路頻寬，且建置與維運成本較高；相較之下，邊緣運算架構可大幅降低延遲與網路頻寬負擔，並具備較低之長期維運成本。本研究提出一套基於邊緣運算之多車種即時辨識技術驗證架構，採用 YOLOv8 與 YOLOv11 之輕量化模型，並以台灣道路為場景自建車流資料集，將模型經過訓練與量化後部署於兩種嵌入式開發板；實驗設計本地驗證與實際場域即時辨識兩階段，先測試並歸納兼顧精度與效能之最佳配置，再將系統架設於天橋進行即時實地辨識，評估其於真實環境之效能、穩定度。研究成果驗證邊緣裝置應用於真實交通場景之可行性，並為以車流辨識作為決策依據之智慧交通管理系統提供實證基礎與配置參考。

關鍵詞：深度學習、物件偵測、模型量化、嵌入式系統

Abstract

As transportation networks expand and vehicle usage keeps growing, driving safety and traffic management efficiency have become important issues. Technologies such as Intelligent Transportation Systems and Advanced Driver-Assistance Systems are advancing rapidly to reduce accidents and improve traffic efficiency through informatization and automation. These applications rely on the real-time acquisition and accurate analysis of traffic information, such as traffic volume, density, and vehicle speed. Most existing traffic detection systems adopt cloud computing, which depends on stable network bandwidth and incurs high deployment and maintenance costs. In contrast, edge computing greatly reduces latency and bandwidth demands while lowering long-term maintenance costs. This research proposes an edge-computing-based real-time detection verification framework for multiple vehicle classes. Lightweight YOLOv8 and YOLOv11 models were trained on a self-built traffic flow dataset from Taiwanese roads, quantized, and deployed on two embedded development boards. The experiments were conducted in two stages: local validation and real-world field detection. The test set was first used to determine the configuration that best balances accuracy and performance, and the system was then mounted on a pedestrian overpass to evaluate its performance and stability in a real environment. The results confirm the feasibility of deploying edge devices in real traffic scenarios and provide an empirical basis and configuration reference for intelligent traffic management systems that use traffic flow detection for decision-making.

Keywords: Deep learning, Object detection, Model quantization, Embedded system

1.緒論

1.1邊緣 AI 與智慧交通辨識之發展

隨著交通路網持續擴張與用車需求逐年成長，行車安全與交通管理效率成為備受關注之議題[25]。近年來智慧運輸系統（Intelligent Transportation System, ITS）[4]與先進駕駛輔助系統（Advanced Driver Assistance Systems, ADAS）[12]等技術快速發展，期望以資訊化與自動化方法提升交通安全與效率，其成效取決於能否即時且準確地獲取車流量、車流密度與車速等交通資訊[23]。

於人工智慧領域中，電腦視覺為影像分析任務之核心技術[22]，其中物件偵測（Object Detection）負責定位影像中目標物件之邊界框並判別其類別[26]，為車輛辨識與車流統計等智慧交通應用之關鍵技術。物件偵測方法中，YOLO（You Only Look Once）系列[17]於辨識精度與推論速度間取得良好平衡，且架構具高度可擴充性，適用於即時性要求高且運算資源受限之場景，已成為即時影像辨識領域之主流方案[8]，亦為本研究採用之核心模型架構。

現行智慧交通辨識系統多採用雲端運算架構，將攝影機影像傳輸至雲端伺服器進行推論，此架構具備高運算能力，但高度仰賴穩定之網路頻寬，當部署規模擴大時，資料傳輸延遲與建置維運成本均顯著上升[19]。相較之下，邊緣運算架構由嵌入式裝置於本地端直接執行推論，可大幅降低延遲與網路頻寬負擔[2]，並具備體積輕巧與長期維運成本低廉等優勢，成為大規模部署之可行方案[1]。其中，NVIDIA Jetson 與 Raspberry Pi 系列等嵌入式平台因兼具低功耗與適中運算能力，已廣泛應用於交通辨識模型之部署[13]。邊緣裝置受限於體積、功耗與成本，運算能力與雲端仍有顯著差距，如何於資源受限之硬體上穩定執行高精度多車種辨識，為本研究待克服之核心課題。

基於上述動機，本研究旨在探討輕量化 YOLO 模型結合量化技術部署於邊緣裝置時，於台灣道路俯視場景進行即時多車種辨識之精度與效能表現。透過自建交通影像資料集訓練 YOLOv8 與 YOLOv11 之輕量化模型，並以不同量化精度部署於兩種邊緣平台，形成涵蓋硬體平台、模型版本與量化精度之系統性比較，以歸納兼顧精度與效能之最佳配置。為達成上述目的，本研究設定以下具體目標。

- 建構交通影像資料集：於天橋拍攝車流影像並完成多車種人工標註，建立可供訓練與驗證之自建資料集。
- 訓練基礎模型：以自建資料集訓練 YOLOv8n 與 YOLOv11n 輕量化模型，作為後續量化與部署之起始模型。
- 執行量化與跨平台轉換：透過轉換工具輸出 FP32、FP16 與 INT8 三種精度模型，以縮減體積並提升邊緣端推論效率。
- 本地端效能驗證：測試各硬體平台、模型版本與量化精度配置之辨識精度與推論效能，初步驗證系統可行性。
- 實際場域即時辨識：將系統架設於天橋進行多輪即時實驗，比較基準組與實驗組之運算效能、系統穩定度與辨識成果。

1.2深度學習物件偵測技術

深度學習於電腦視覺領域之興起，可追溯至 2012 年提出之卷積神經網路（Convolutional Neural Network, CNN），其透過卷積層、池化層與激勵函數所構成之多層結構，自原始像素中逐層提取由淺至深之階層式特徵，有效捕捉影像中之邊緣、紋理與幾何結構[11]。基於此一特徵提取能力，物件偵測方法進一步衍生為二階段與單階段兩類架構[26]：前者先生成候選區域再進行分類與邊界框回歸，精度較佳但流程繁複；後者則以單一神經網路一次完成位置與類別預測。其中，Redmon 等人於 2016 年提出之 YOLO（You Only Look Once）將物件偵測視為端到端之回歸問題，將輸入影像劃分為網格並於各網格單元同時預測邊界框與類別機率，使單張影像之推論得於毫秒級內完成[17]。整體而言，單階段方法於推論

效能明顯領先[8]；考量本研究聚焦之即時邊緣交通辨識任務對推論延遲具嚴格要求，單階段架構為部署於邊緣端之合理選擇。

YOLO 系列自初代提出後歷經多次架構與訓練策略之迭代演進，逐步成為單階段偵測領域之主流模型架構[21]。其中，由 Ultralytics 公司於 2023 年發布之 YOLOv8 為廣泛應用之代表性版本之一[20]；該公司並於 2024 年推出其後繼版本 YOLOv11[10]，架構改良使其於維持相近甚至更低參數量之條件下達成更佳之辨識精度[5][18]，尤適用於需處理小型車輛與機車等多尺度目標之即時交通辨識任務。於模型規模方面，兩版本均提供不同尺寸之預訓練版本，其中 n（Nano）版本參數量與運算複雜度最低，辨識精度雖略遜於大型版本，其輕量化特性使其於資源受限之邊緣裝置上仍能維持實用之推論效能。基於推論延遲與裝置功耗之限制，並權衡精度與效能之取捨，本研究選用 YOLOv8n 與 YOLOv11n 作為核心比較對象，系統性評估兩版本之辨識精度與運算效能差異。

1.3模型最佳化技術

於邊緣裝置部署深度學習模型時，模型量化為當前廣泛採用之最佳化技術之一，其透過降低模型權重與激勵值之表示精度，以縮減模型體積、降低記憶體頻寬需求並加快矩陣運算速度[7]。深度學習框架於訓練階段所產生之權重多以 FP32 格式儲存，於部署階段轉換為較低位元之表示形式即可減輕邊緣端之運算負擔，常見目標格式包含 FP16 與 INT8。依 IEEE 754 標準[6]，FP32 由 1 位元符號位、8 位元指數位與 23 位元尾數位組成，FP16 則對應為 1 位元符號位、5 位元指數位與 10 位元尾數位，位元寬度恰為 FP32 之半且格式結構具對應關係，故 FP32 至 FP16 之轉換相對單純、精度損失影響較低，其過程如圖 1 所示。

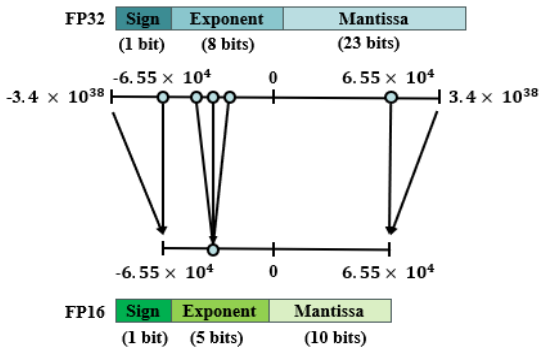


圖 1 FP32量化至 FP16示意圖

INT8 量化則須將連續分布之浮點數值映射至 −128 至 +127 之整數區間，其動態範圍相較 FP32 大幅縮減[7]，其過程如圖 2 所示。此映射雖能進一步縮減模型體積，並於支援整數運算之加速器上顯著提升推論速度，但對數值範圍之離散化截斷將不可避免地造成量化誤差[24]。

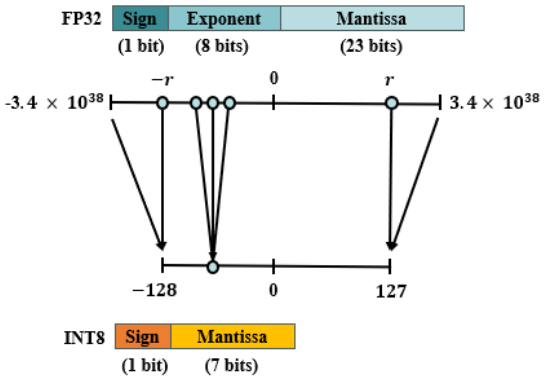


圖 2 FP32量化至 INT8示意圖

為平衡精度與效能，量化策略依是否涉及模型再訓練而有所區分，本研究採用訓練後量化（Post-Training Quantization, PTQ）。PTQ 於量化過程中不執行任何再訓練，僅透過數值轉換將權重與激勵值映射至較低位元之表示形式；於 INT8 等較低位元情境中，通常須輔以代表性校正資料集（Calibration Dataset），藉由推論統計各層激勵值之數值分布，據以計算各層最佳之量化參數[3]。

雖然量化技術能降低運算複雜度，欲使輕量化模型於邊緣硬體上達成最佳推論效能，仍須透過推論框架將模型最佳化並轉換為對應硬體之專用格式。其中，運算子融合（Operator Fusion）為

代表性技術，其將模型中連續之卷積、批次正規化與激勵函數層加以合併，縮減硬體核心函數（Kernel）之呼叫次數，藉此降低推論延遲[15]。針對本研究採用之兩種邊緣平台，TensorRT 對應 NVIDIA GPU 與 Jetson 系列裝置，Hailo DFC 則對應 Hailo 系列 NPU，兩者於編譯過程中均整合運算子融合與 PTQ 等最佳化機制。據此，本研究之 YOLOv8n 與 YOLOv11n 基礎模型於完成 FP32 訓練後，分別透過上述兩框架轉換為對應硬體之部署格式，其 INT8 量化均採用標準 PTQ 流程，全程不執行任何再訓練。

2.研究方法

2.1研究架構

為系統性地評估輕量化 YOLO 模型結合量化技術於邊緣裝置上之即時多車種辨識表現，本研究架構規劃為四個依序執行之階段。研究架構如圖 3所示。

- 資料集建構階段：於天橋上進行車流影片之錄製，並擷取為影像；隨後依車輛類別定義對影像進行人工標註，並依拍攝時序將標註資料依比例劃分為訓練集、驗證集與測試集，作為後續模型訓練與驗證之資料集。
- 模型訓練與最佳化階段：以前階段所建立之資料集為訓練資料，分別訓練 YOLOv8n 與 YOLOv11n 兩個輕量化模型，產出基礎模型權重；其後依目標邊緣硬體，將基礎模型轉換為對應之專用格式，並於轉換過程中執行量化以縮減模型體積並提升推論效能。
- 邊緣部署與本地端驗證階段：將完成轉換之模型分別部署至兩種邊緣運算平台；隨後以測試集為驗證資料，分別於 PC 端與各邊緣裝置上測試模型之精度與效能，並進行比較分析。
- 實際場域即時辨識階段：本研究將前階段完成模型部署之邊緣裝置與攝影機架設於天橋，依預先規劃之實驗設計，同時以兩組不同配置進行同步即時辨識拍攝，記錄即時辨識之影像結果、推論效能等資訊，藉此評估所提出之多車種辨識系統於真實場域運作之穩定性與實用性。

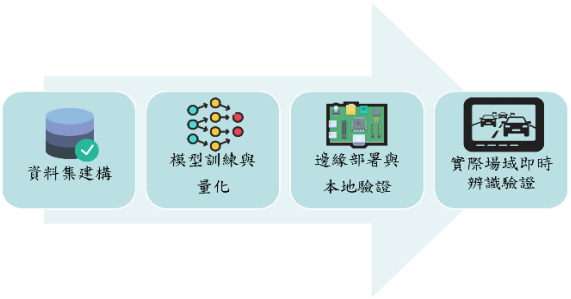


圖 3　研究架構圖

2.2實驗環境

2.2.1硬體環境

(1) 個人電腦（PC）

PC 平台配備 AMD Ryzen 9 5950X 16 核心 CPU、NVIDIA GeForce RTX 3090 GPU。本研究中，PC 平台用於執行 YOLOv8n 與 YOLOv11n 兩個基礎模型於自建交通影像資料集上之訓練流程。

(2) Jetson Xavier NX

NVIDIA Jetson Xavier NX 為嵌入式 AI 運算開發板，外觀如圖 4 所示，內建採用 Volta 架構之 GPU，搭載 6 核心 ARM Carmel CPU、8 GB 記憶體，於 INT8 精度下可提供達 21 TOPS 之推論效能。本研究中用於部署經 TensorRT 轉換之模型。



圖 4 Jetson Xavier NX

(3) Raspberry Pi5 8GB 與 Raspberry Pi AI HAT+ 2

Raspberry Pi 5 搭載 4 核心 Cortex-A76 CPU 與 8 GB 記憶體，外觀如圖 5 所示，其原生運算能力主要由 CPU 提供。本研究透過外接 AI 加速擴充板以提升其神經網路推論效能。Raspberry Pi AI HAT+ 2 為 Hailo 公司推出之第二

代 AI 加速擴充板，外觀如圖 6 所示，搭載 Hailo-10H NPU 與 8 GB 記憶體，於 INT4 精度下可提供達 40 TOPS 之推論效能。於本研究中，此配置構成另一組邊緣運算平台，用於部署經 Hailo DFC 轉換之模型，與 Jetson Xavier NX 形成硬體平台之比較基礎。



圖 5 Raspberry Pi 5



圖 6 Raspberry Pi AI HAT+ 2

(4) IPEVO P2V Ultra

IPEVO P2V Ultra 為一支援高解析度影像擷取之 USB 攝影機，外觀如圖 7 所示，採用 1300 萬畫素 CMOS 感光元件，支援 4K 畫質之動態錄製。於本研究中，該攝影機同時作為資料集建構之影像拍攝設備，以及實際場域即時辨識實驗中之即時影像拍攝裝置。



圖 7 IPEVO P2V Ultra 攝影機

2.2.2軟體與開發環境

於模型訓練階段，本研究採用 Ultralytics 所提供之 YOLO 框架，於 PC 端完成 YOLOv8n 與 YOLOv11n 之模型訓練。於模型轉換階段，依目標邊緣平台之不同分別採用對應工具鏈， Jetson Xavier NX 採用 NVIDIA 所提供之 JetPack 軟體開發套件，Raspberry Pi 5 平台採用 Hailo DFC 與 Hailo Model Zoo 工具完成模型專用格式之轉換。各項軟體之版本資訊如表 1 所示。

表 1 開發環境規格表

	PC	Jetson Xavier NX	Raspberry Pi 5 + Hailo-10H AI HAT
CUDA	12.1	11.4	-
Pytorch	2.10.0	1.11.0	-
TensorRT	10.15.1.29	8.5.2.2	-
HailoRT	-	-	5.1.1
Ultralytics	8.4.10	8.3.234	-
OS	Windows11 Education, 24H2	Ubuntu 20.04	Raspbian 13.4
Jetpack	-	5.1.5	-
Power mode	-	20W 6CORE	-

2.3資料集建構

2.3.1影像拍攝與擷取

本研究之影像資料取自於天橋上以俯視角度拍攝之實際道路車流影片，拍攝地點之選擇主要考量以下三項條件。

- 視野須清楚涵蓋多線道路段且無明顯遮擋，以完整擷取車輛特徵。
- 路段車流量須具一定密度，以蒐集足量之車輛樣本。
- 路段須為公車或大客車經常行駛之路線，以確保資料集涵蓋多種車輛類別。

影像以攝影機於 3840 × 2160（4K UHD）解析度、30 FPS 之幀率錄製，再透過 FFmpeg 以每秒 2 張之頻率擷取為靜態影像，藉此避免相鄰影像因車輛位移過小而特徵高度相似，並控制人工標註之負擔。經上述流程共獲得 6,103 張 PNG 格式影像，作為後續標註與劃分之原始資料，實際車流影片之截圖如圖 8 所示。



圖 8 天橋俯視角度實際拍攝結果

2.3.2車輛類別定義與影像標註

依本研究多車種辨識之目標並考量拍攝場景之車流組成，將辨識類別依尺寸特徵與用途屬性劃分為以下四項。

- 一般車輛（Car）：涵蓋轎車、休旅車與廂型車等出現頻率最高之常見一般用途車型。
- 機車（Motorbike）：涵蓋普通輕型、普通重型與大型重型機車，僅含以引擎或電動驅動之兩輪機動車輛，不含自行車與電動自行車。
- 大客車（Bus）：涵蓋以乘客運輸為主之公車與遊覽車。
- 大型車輛（Other_car）：涵蓋卡車、貨車、水泥車、油罐車與工程車等以載貨或工程用途為主之大型車輛，及非屬前三類之車輛。

影像標註以 LabelImg 工具進行並採 YOLO 格式儲存，使用者透過繪製邊界框並指定對應類別完成標註作業，每張影像之標註結果以純文字檔記錄各目標之類別編號、邊界框中心座標與寬高。經人工標註共產生 75,743 筆車輛邊界框標註資料，各類別之數量與所佔比例如表 2 所示。其中機車與一般車輛合計逾 95%，反映拍攝路段之主要車流組成；大客車與大型車輛之標註數量則相對稀少，此一類別不平衡分布將對模型於各類別之辨識精度造成差異化影響，後續於第四章實驗結果進一步分析。

表 2 標註資料量與比例

	標註量	標註比例
Car	35318	46.63%
Motorbike	37211	49.13%
Bus	2578	3.40%
Other_car	636	0.84%
All	75743	

2.3.3資料集劃分

資料集劃分方式為依拍攝時序將 6,103 張影像以 7：2：1 之比例劃分為訓練集 4,272 張、驗證集 1,220 張與測試集 611 張，即拍攝時序最前段之 70% 作為訓練集、中段 20% 作為驗證集、最末段 10% 作為測試集。採時序劃分可使各子集於影像內容上明確區隔，確保所量測之辨識精度更貼近模型於實際部署之泛化表現[9]。

2.4模型選用與訓練策略

2.4.1模型選用

承緒論 1.2 所述，本研究選用 YOLOv8n 與 YOLOv11n 輕量化模型作為核心比較對象。儘管 YOLOv11n 為新版本，於架構設計上引入更完善之特徵提取機制，理論上具備較佳之辨識精度，然其網路架構相對複雜，於運算資源受限之邊緣裝置上，是否仍能維持相應之推論效能則有待驗證，故新版本於精度上之理論優勢，未必能直接轉化為實際部署之最佳選擇，此亦為本研究選用兩版本進行比較之核心考量。此外，兩版本於輕量化部署與多尺度目標辨識之適用性、版本間之可比較性，以及對 TensorRT 與 Hailo 等推論框架之部署相容性，亦為選用依據。

本研究將兩版本模型分別以不同量化精度部署至兩種邊緣平台，系統性評估其於多車種辨識任務之精度與效能差異，以及量化技術對兩模型於不同硬體上之精度衰減與推論加速幅度。透過上述比較，得以確切掌握兩版本模型於本研究硬體平台與辨識情境下之適用性，並歸納兼顧精度與效能之最佳模型版本與部署配置。

2.4.2模型參數設定與訓練

YOLOv8n 與 YOLOv11n 採用相同之訓練參數配置，以確保後續實驗之精度與效能差異得以歸因於模型架構本身，而非訓練設定之不一致。

訓練週期、批次大小與輸入影像尺寸等項目之設定詳如表 3。為避免過擬合（Overfitting），本研究將早停等待週期（Patience）設為 50，當驗證集評估指標連續 50 個訓練週期未見改善時即自動終止訓練，以兼顧模型收斂之充分性與訓練效率。

表 3 基本參數設定

訓練週期（Epochs）	400
批次大小（Batch Size）	16
輸入影像尺寸（Image Size）	640
運算裝置（Device）	0
早停等待週期（Patience）	50

本研究採用 AdamW 最佳化器，其核心在於將權重衰減(Weight Decay)自損失函數中分離，使其不受自適應學習率之縮放影響，於收斂速度與泛化能力上較傳統 Adam 更具優勢[14]。學習率方面，初始學習率設定為 0.01，屬訓練起始階段之較高設置，使模型得以於較廣之參數空間中快速探索；隨訓練進行，學習率逐步衰減至初始值與最終學習率之乘積，即 0.0001，使模型於收斂階段之最後以極低之學習率細緻趨近最佳解，詳細設定如表 4 所示。

表 4 最佳化器與學習率參數設定

最佳化器（Optimizer）	AdamW
初始學習率（lr0）	0.01
最終學習率比例（lrf）	0.01

資料增強方面，使用 Ultralytics 框架提供之多項增強策略，涵蓋色調、飽和度與亮度等色彩空間調整，以及水平翻轉、平移與縮放等幾何變換。前者使模型得於不同光照與色彩條件下學習車輛外觀，後者則模擬俯視場景中車輛因行駛方向與遠近所形成之外觀與尺度變化，藉此提升模型對多時段與多尺度交通場景之泛化能力，參數設定如表 5 所示。

表 5 資料增強參數設定

色調增強（hsv_h）	0.015
飽和度增強（hsv_s）	0.7
亮度增強（hsv_v）	0.4
隨機水平翻轉（fliplr）	0.5
平移（translate）	0.1
縮放（scale)	0.5
Mosaic 增強（mosaic）	1.0

2.5模型轉換與量化

本研究將訓練得之基礎模型 YOLOv8n.pt 與 YOLOv11n.pt 依目標邊緣平台分別轉換為對應之專用格式並執行量化處理。量化精度配置方面，對應 Jetson 硬體之 TensorRT 推論框架同時支援 FP32、FP16 與 INT8 三種運算精度，本研究於 Jetson 平台上分別部署此三種精度之 .engine 模型；而 Hailo-10H AI HAT 之官方軟體開發工具針對物件偵測模型之轉換預設將量化目標設定為 INT8，故本研究於該平台僅部署 INT8 精度之 .hef 模型。前述量化精度之配置設計使本研究得以於 Jetson 平台上完整觀察 FP32、FP16 與 INT8 三種精度之精度衰減與效能提升幅度，並於 Hailo 平台上以 INT8 模型之表現，作為跨平台精度與效能比較之依據，具體轉換與量化資訊如表 6 所示。

表 6 模型轉換與量化比較表

硬體平台	模型推論框架	模型轉換格式	目標量化精度
Jetson Xavier NX	TensorRT	.engine	FP32, FP16, INT8
Raspberry Pi 5 + Hailo-10H AI HAT	HailoRT	.hef	INT8

2.6邊緣部署與驗證

2.6.1實驗模型配置與部署

本節彙整三個運算平台之模型部署配置，完整內容如表 7 所示。PC 平台部署根據 2.4 節以自建交通資料集訓練所得 YOLOv8n 與 YOLOv11n 之基礎模型，作為各邊緣平台之精度與效能對照基準。Jetson Xavier NX 除部署相同之基礎模型外，另依轉換流程將兩版本模型轉為 FP32、FP16 與 INT8 三種量化精度之 TensorRT 專用格式模型，以觀察格式轉換與量化對辨識精度及推論效能之影響。Raspberry Pi 5 搭載 Hailo-10H AI HAT，將兩版本轉換為對應之 INT8 專用格式。

表 7 硬體平台模型配置表

硬體平台	配置模型
PC	YOLOv8n.pt
	YOLOv11n.pt
Jetson Xavier NX	YOLOv8n.pt
	YOLOv8n.engine (FP32)
	YOLOv8n.engine (FP16)
	YOLOv8n.engine (INT8)
	YOLOv11n.pt
	YOLOv11n.engine (FP32)
	YOLOv11n.engine (FP16)
	YOLOv11n.engine (INT8)
Raspberry Pi 5 + Hailo-10H AI HAT	YOLOv8n.hef (INT8)
	YOLOv11n.hef (INT8)

2.6.2模型訓練關鍵指標說明

混淆矩陣為偵測任務評估之基礎，其依模型預測結果與實際情形之對應關係，將辨識樣本區分為四類如表 8 所示。真陽性（True Positive, TP）為模型預測為正且實際亦為正之樣本，假陽性（False Positive, FP）為模型預測為正但實際為負之樣本，真陰性（True Negative, TN）為模型預測為負且實際亦為負之樣本，假陰性（False Negative, FN）則為模型預測為負但實際為正之樣本。

表 8 評估指標之混淆矩陣

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

交並比則用以量化模型所輸出之預測框與真實標註框之間的重疊程度，其定義為兩框交集面積與聯集面積之比值，公式如式(1)所示[16]，B_p 為模型預測框，B_{gt} 為真實標註框。於物件偵測任務中，當預測框與真實標註框之 IoU 高於設定門檻時，該預測判定為 TP，反之則判定為 FP。

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{1}$$

精確率（Precision）為模型所有正向預測中實際為正之樣本比例，反映模型預測之準確程度，如式(2)所示。召回率（Recall）為所有實際為正之樣本中被模型正確預測之比例，反映模型對目標物之偵測完整程度，如式(3)所示[16]。

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

平均精確率（Average Precision, AP）為單一類別於精確率與召回率曲線下之面積，AP50 為 IoU 門檻設定為 0.5 時所計算之平均精確率，如式(4)所示，R_n 為精確率-召回率曲線上，第 n 個取樣點的召回率數值。R_{n+1} 為緊鄰於 R_n 的下一個召回率數值。此兩點間的差距 $(R_{n+1} - R_n)$，即構成計算 AP 面積時的一個微小區間寬度。$P_{interp}(R_{n+1})$ 為召回率 R_{n+1} 處之插值（Interpolation）精確率，定義為所有大於等於 R_{n+1} 之召回率對應精確率中的最大值。

平均召回率（Average Recall, AR）之計算原理與 AP 相同，本研究以 AP50、AR50 作為精度評估指標[16]。

$$AP = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1}) \tag{4}$$

mAP50 為所有類別之 AP50 取平均所得之平均精度均值，用於評估模型於多類別偵測任務之綜合精度表現，如式(5)所示[16]。

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{5}$$

效能指標方面，本研究以每秒影像處理張數（Frames Per Second, FPS）與平均推論時間評估系統之即時性。FPS 為系統每秒所能處理之影像數，數值越高代表即時辨識越流暢，為評估交通辨識系統即時性之重要依據[8]，如式(6)所示，N_{frames} 為辨識過程中所處理之影像總影像數，T_{total} 為完成所有影像辨識所耗費之總時間，單位為秒。平均推論時間則為各影像於模型推論階段所耗費時間之平均值，其能直接比較模型本身之推論速度，於跨平台效能比較上尤具參考價值，如式(7)所示，$t_{inf,\ i}$ 為第 i 張影像於模型推論階段所耗費之時間，單位為毫秒。

$$FPS = \frac{N_{frames}}{T_{total}} \quad (6)$$

$$T_{inf} = \frac{1}{N_{frames}} \sum_{i=1}^{N_{frames}} t_{inf,\ i} \quad (7)$$

2.7驗證實驗規劃與流程設計

為驗證本研究提出之邊緣交通辨識系統之可行性，規劃精度與效能本地驗證實驗，以及實際場域即時辨識驗證實驗。前者以測試集測試各運算平台與模型配置之辨識精度與推論效能，後者則將系統架設於真實天橋場景進行即時辨識，以評估其於實際環境下之穩定性與運作表現。

2.7.1精度與效能本地驗證實驗

精度與效能本地驗證實驗依 2.6.1 節之三個運算平台與其對應之模型配置，於本研究測試集上量測辨識精度與推論效能。由於三平台模型之運行涵蓋不同框架，為確保精度評估之一致性，本研究統一採用物件偵測領域廣泛使用之 pycocotools 工具計算精度指標，並記錄 FPS 與平均推論時間等效能數據。

為兼顧視覺呈現與精度計算之完整性，各配置對測試集進行兩階段推論，於交並比參數統一設定為 0.7 的情況下，先以信心度閾值 0.25 產生視覺化辨識影像，再以 0.001 之極低閾值保留各信心度層級之完整預測框，以正確繪製精確率與召回率曲線並計算精度指標，避免因捨棄低信心度預測導致召回率不完整而使數值產生偏差[16]。上述結果可反映模型經最佳化部署於邊緣裝置後之效能變化，藉以驗證模型最佳化之有效性與系統實際部署之可行性，完整流程如圖 9 所示。

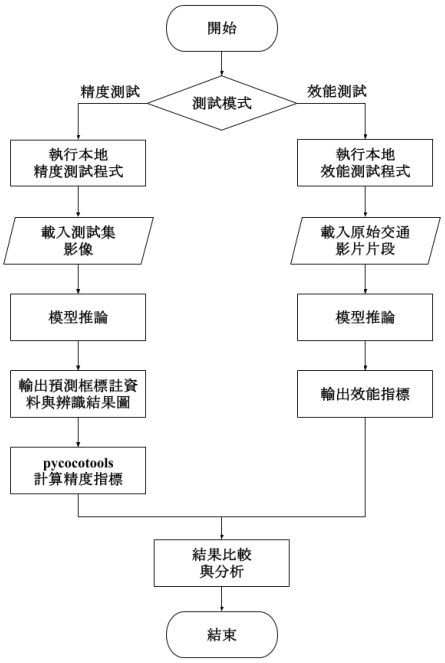


圖 9 本地驗證實驗流程圖

2.7.2實際場域即時辨識驗證實驗

實際場域即時辨識驗證實驗旨在驗證所提出之交通辨識系統於真實環境下之運作表現，實驗流程如圖 10 所示。實際場域實驗可檢視系統於不同光照、裝置長時間運作升溫，以及同時進行影像串流擷取、即時辨識結果儲存等高負載下之可靠性與穩定性，並觀察其是否因長時間運行而效能衰退。

本實驗共設計六輪，以兩支相同規格之攝影機分為基準組與實驗組同步拍攝辨識。基準組於各輪固定採用 Jetson Xavier NX 搭配 YOLOv8n 之 INT8 模型作為共同比較基準。實驗組則依比較目的分為兩階段：前兩輪採用 Raspberry Pi 5 搭配 Hailo-10H AI HAT，分別執行 YOLOv8n 與 YOLOv11n 之 INT8 模型，於相同量化精度下比較兩種邊緣平台之效能差異；後四輪固定採用 Jetson Xavier NX，分別執行兩版本之 INT8 與 FP16 模型，藉以驗證 YOLOv11n 相較 YOLOv8n 之精度提升，並比較 INT8 與 FP16 於同一平台之效能與辨識差異。各輪配置如表 9 所示，實驗過程並記錄 FPS 與記憶體用量等數據，用以評估系統於實際場域長時間運作之穩定性與辨識能力是否符合部署需求。

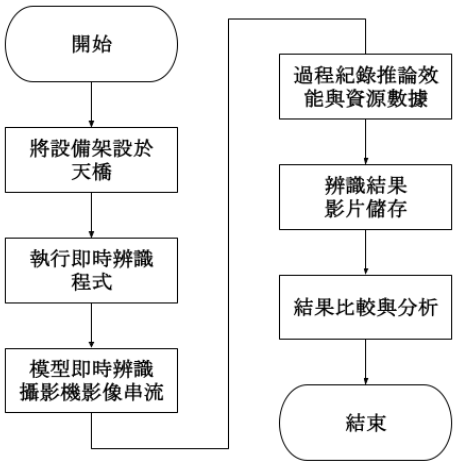


圖 10 實際場域即時辨識實驗流程圖

表 9 實際場域即時辨識裝置與模型配置表

實驗輪數	基準組		實驗組	
	設備	模型	設備	模型
第一輪	Jetson	YOLOv8n .engine (INT8)	Pi 5 + AI HAT	YOLOv8n .hef (INT8)
第二輪	Jetson	YOLOv8n .engine (INT8)	Pi 5 + AI HAT	YOLOv11n .hef (INT8)
第三輪	Jetson	YOLOv8n .engine (INT8)	Jetson	YOLOv8n .engine (INT8)
第四輪	Jetson	YOLOv8n .engine (INT8)	Jetson	YOLOv11n .engine (INT8)
第五輪	Jetson	YOLOv8n .engine (INT8)	Jetson	YOLOv8n .engine (FP16)
第六輪	Jetson	YOLOv8n .engine (INT8)	Jetson	YOLOv11n .engine (FP16)

3.實驗成果

3.1模型訓練與收斂成果

本節呈現 YOLOv8n 與 YOLOv11n 於自建交通資料集上之訓練與收斂成果，藉由訓練過程之收斂曲線評估兩模型於訓練階段所展現之辨識能力基礎。

兩模型均依 2.4 節之參數與方法進行訓練。兩者最終皆以驗證集表現最佳之權重作為基礎模型。就最佳權重之精確率、召回率與 mAP50 而言，兩模型表現相當接近，詳細收斂曲線分別如表 10 與表 11 所示。就曲線特性而言，YOLOv8n 之精確率震盪幅度較大，召回率於訓練後段約 50 個週期呈下降且 mAP 未見明顯增長；YOLOv11n 之震盪幅度較小，mAP 約於第 100 個週期達最佳後維持穩定。上述後期指標趨於平緩或微降之現象，反映早停機制已適時於模型達最佳泛化表現後終止訓練，兩模型於辨識能力上之具體差異將於後續章節進一步比較與分析。

表 10 YOLOv8n 模型訓練結果

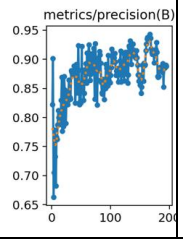
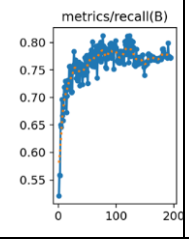
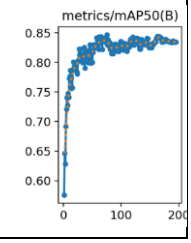
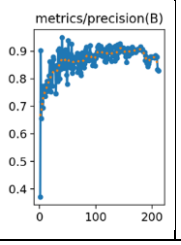
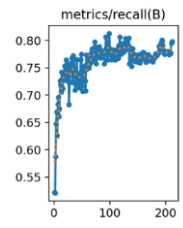
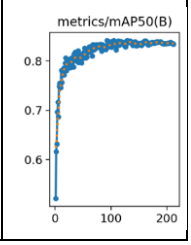
訓練平台：PC 預訓練模型：Ultralytics YOLOv8n.pt		
Precision 收斂曲線	Recall 收斂曲線	mAP50收斂曲線
收斂結果：0.903	收斂結果：0.777	收斂結果：0.841
		

表 11 YOLOv11n 模型訓練結果

訓練平台：PC 預訓練模型：Ultralytics YOLOv11n.pt		
Precision 收斂曲線	Recall 收斂曲線	mAP50收斂曲線
收斂結果：0.905	收斂結果：0.77	收斂結果：0.841
		

3.2模型對測試集之測試結果

本節依 2.7.1 節之本地驗證方法，呈現 PC、Jetson Xavier NX 與 Raspberry Pi 5 三個運算平台所部署之各模型配置對測試集之辨識精度與推論效能測試結果，並依總體辨識精度、各車輛類別辨識精度與推論效能等面向依序分析。

3.2.1辨識精度與推論效能分析

各模型依本地驗證方法對測試集所測得之精度指標比較如圖 11 所示。基礎模型於 PC 與 Jetson Xavier NX 兩平台之各項指標幾乎一致，未出現跨平台之精度衰減，可確認後續之精度變化源自格式轉換與量化，而非運算平台之差異。就兩版本之比較而言，以 Jetson 平台之基礎模型為例，YOLOv11n 之 mAP50 較 YOLOv8n 提升約 4.8%、AR50 提升約 4.4%，顯示其對受遮擋與小尺度目標之偵測完整度更佳，整體辨識能力優於 YOLOv8n。

格式轉換與量化之影響方面，模型轉換為 TensorRT 之 FP32 專用格式後精度指標幾乎未見變動，顯示格式轉換本身對辨識精度甚微；進一步量化至 FP16 時，兩模型之 mAP50 與 AR50 變動幅度均在 0.5% 以內，維持近乎無損之精度。然而量化至 INT8 時精度出現較明顯衰減，YOLOv8n 與 YOLOv11n 之 mAP50 相較 FP16 分別下降約 1.96% 與 5.52%，顯示主要精度損失發生於 INT8 階段。

最後比較兩種邊緣平台之 INT8 模型，以 Jetson 平台之 INT8 模型為共同基準與 Raspberry Pi 5 平台對照。就 mAP50 而言，YOLOv8n 於 Pi 5 較 Jetson 下降約 6.41%，YOLOv11n 則提升約 2.15%；於 AR50 方面，Pi 5 之兩模型均明顯較高，YOLOv8n 提升約 8.6%、YOLOv11n 提升約 13.9%。整體而言，兩平台於精確率指標表現相當，而經 Hailo 框架轉換之 Pi 5 模型於召回率較高，代表其辨識結果涵蓋較多車輛目標，此一差異將於後續以實際辨識結果圖進一步比較與分析。

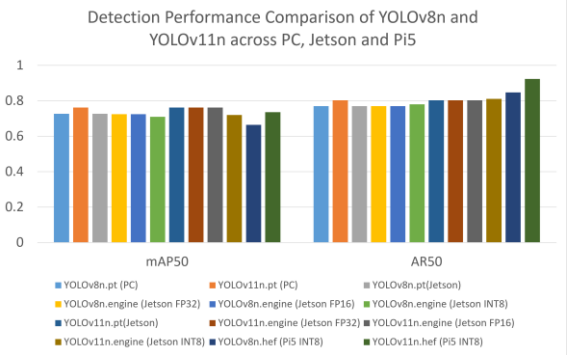


圖 11 各模型辨識精度測試結果比較圖

於推論效能方面，各模型之 FPS 與平均推論時間比較如圖 12 所示。PC 平台憑高效能 GPU 展現最高推論效能，兩基礎模型之 FPS 均逾 42；Jetson Xavier NX 平台則明顯較低。以 Jetson 平台觀察量化之效能效益，模型由原始格式經格式轉換並量化至 FP16 為效能提升之主要階段，YOLOv8n 與 YOLOv11n 之 FPS 相較基礎模型分別提升約 7.61% 與 16.18%、平均推論時間分別縮短約 25.61% 與 37.87%；然而由 FP16 進一步與 INT8 相比，兩模型之 FPS 與推論時間變動均不及 1%，未帶來相應之效能增益。此結果顯示 INT8 量化雖伴隨明顯精度衰減，於 Jetson 平台卻未換取顯著之效能提升。

進一步比較兩種邊緣平台之 INT8 模型，搭配 Hailo-10H NPU 之 Raspberry Pi 5 於 FPS 與平均推論時間上均優於 Jetson 平台，YOLOv8n 與 YOLOv11n 之 FPS 較 Jetson 對應 INT8 模型分別提升約 22.45% 與 26.12%，顯示 Hailo NPU 之硬體加速於推論運算上具明顯優勢。

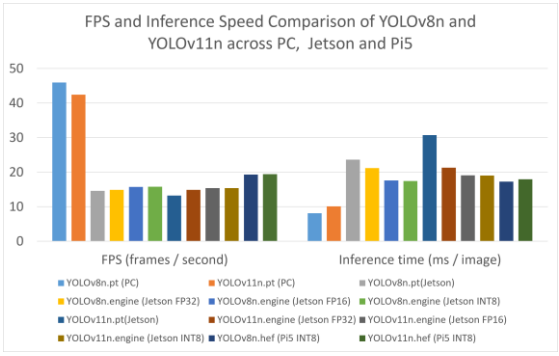


圖 12 各模型推論效能測試結果比較圖

3.2.2各類別車輛之辨識精度分析

為進一步觀察模型對個別車輛類別之辨識能力，本節以 Jetson 平台之 YOLOv8n 與 YOLOv11n 兩種 FP16 模型為代表，比較其對四類車輛之 AP50，結果如圖 13 所示。整體而言，模型於 Car、Motorbike 與 Bus 三類常見車輛均展現高水準之辨識精度，其 AP50 明顯高於 Other_car 類別。其中，Car 與 Bus 兩類因於畫面中尺度較大、外觀特徵清晰，表現最佳，AP50 達 0.96 上下之高水準；Motorbike 雖因尺度較小而精度略低於前二類，但仍維持於 0.92 以上。

相對地，Other_car 類別之 AP50 於各配置下均明顯偏低，其主因在於該類別之標註量僅佔總標註量之 0.84%，樣本數量顯著不足，且其涵蓋卡車、貨車、水泥車、油罐車與工程車等多種大型車輛，車種繁多而彼此之尺度與外觀特徵差異甚大，使模型難以建立穩定之辨識能力。此一限制源自樣本稀少與類別內特徵差異，而非模型架構本身之缺陷。

綜合各類別之辨識結果，本研究之模型於 Car、Motorbike 與 Bus 三類常見車輛上，均維持高水準之辨識精度，足以因應實際交通辨識應用之需求；Other_car 類別之表現受限，可藉由後續增加該類別之標註樣本並細分車種類別加以改善。

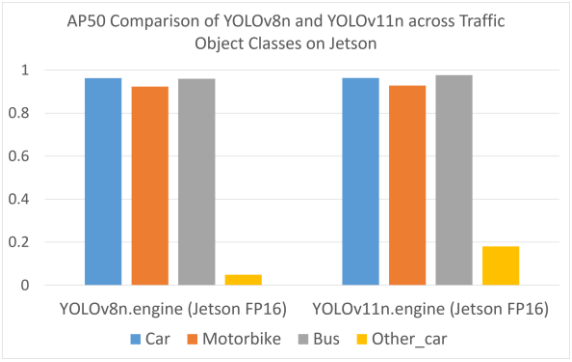


圖 13 各類別物件之 AP50測試結果比較圖

3.2.3模型辨識測試集圖片結果比較

為觀察各模型於實際影像上之辨識表現，本節呈現代表性配置對測試集影像之推論結果，信心度門檻均設定為 0.25。

於 Jetson 平台之 TensorRT 專用格式模型方面，YOLOv8n 與 YOLOv11n 之 FP16 辨識結果分別如圖 14 與圖 15 所示。其中 YOLOv11n 於人行道行人處產生一低信心度之機車預測框，而 YOLOv8n 未出現此誤判，顯示 YOLOv11n 較強之特徵提取能力雖提升辨識靈敏度，亦可能增加對非車輛物件之誤判風險。整體而言，兩模型之預測框定位皆準確，近處車輛之辨識信心度多達 0.9 以上，未出現明顯之辨識錯誤或漏檢。

Raspberry Pi 5 搭載 AI HAT 之 Hailo 框架專用格式模型，其 YOLOv11n 之 INT8 辨識結果如圖 16 所示。畫面中車輛物件之標註準確、未見明顯誤判，小尺度或遠處目標出現重複預測框之現象，此現象可能與 Hailo 框架之模型轉換流程，以及 INT8 低精度量化對辨識結果之影響有關，至於一般車輛等尺度較大、特徵清晰之目標則辨識正確，未出現重複預測框。

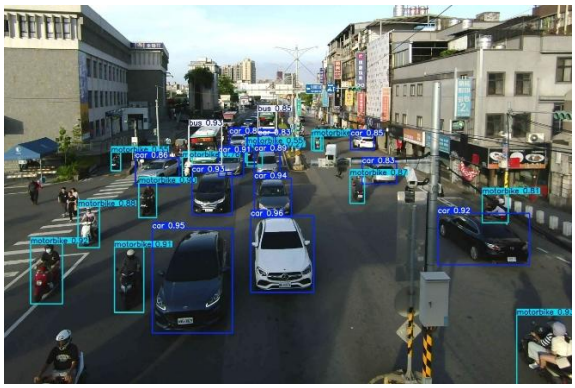


圖 14 Jetson YOLOv8n.engine (FP16) 測試集圖片辨識結果

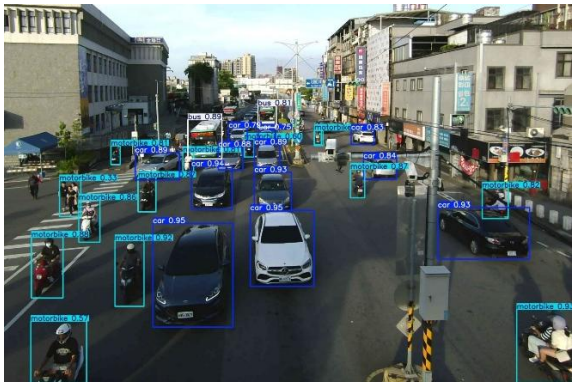


圖 15 Jetson YOLOv11n.engine (FP16) 測試集圖片辨識結果

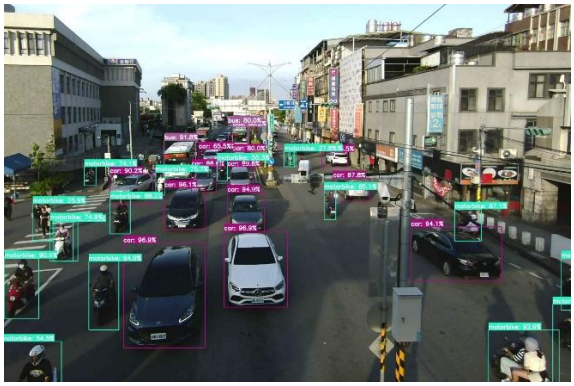


圖 16 Pi5 YOLOv11n.hef (INT8) 測試集圖片辨識結果

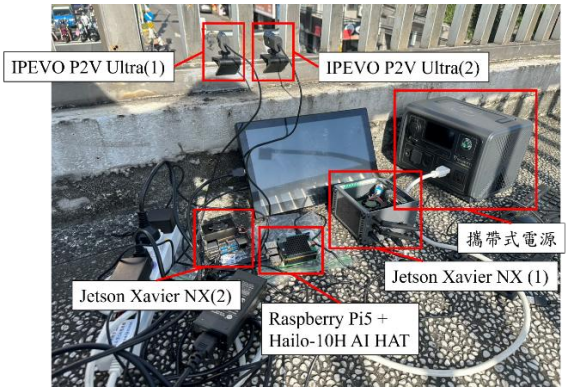


圖 18 實際場域即時辨識實驗硬體實際配置

3.3實際場域即時辨識測試結果

本節依 2.7.2 節之實驗方法，將邊緣裝置與攝影機部署於資料集拍攝所在之天橋，以相同俯視角度對實際車流進行即時辨識，並就系統架設、即時辨識效能數據與系統穩定度等面向依序分析。

3.3.1實驗環境與系統架設

實際場域即時辨識實驗之硬體架構如圖 17 所示，以攜帶式電源為邊緣裝置供電，並透過筆記型電腦以 SSH 遠端控制各邊緣裝置。兩支攝影機之連接依實驗階段調整：前兩輪分別接至基準組之 Jetson 與實驗組之 Raspberry Pi 5，後四輪則改由兩台 Jetson 同步辨識。各輪即時辨識進行 300 秒，完成後隨即更換配置，直至完成六輪，所有實驗均以 1920 × 1080 解析度拍攝，實際配置如圖 18 所示。

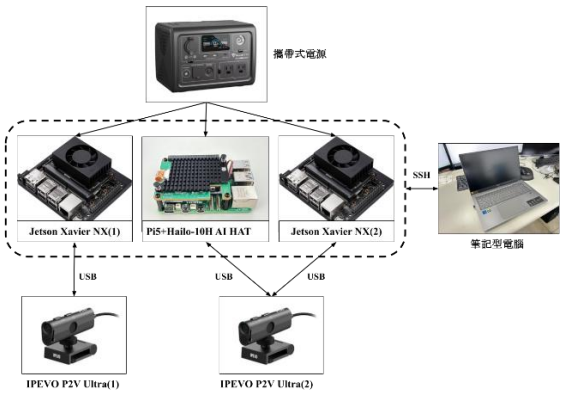


圖 17 實際場域即時辨識實驗硬體架構圖

3.3.2即時辨識測試數據與分析

即時辨識實驗各組配置之 FPS 與平均推論時間比較分別如圖 19 與圖 20 所示。實驗過程中，各邊緣裝置須同時接收並處理攝影機之車流影像串流、執行模型推論，並儲存辨識結果影片，此一高負載流程對裝置運算資源形成相當負擔。

整體而言，六輪實驗之 FPS 均偏低，介於 4.64 至 4.99 之間，組別間差異甚小；其中以搭配 Hailo-10H NPU 之 Raspberry Pi 5 執行 YOLOv8n 與 YOLOv11n 之模型表現最佳，FPS 均達 4.99。各組 FPS 表現於 5 以下，主因在於裝置須同時處理影像串流接收、前處理、模型推論、後處理與影片儲存等多項工作，運算負擔過重，致使模型推論之效能難以反映於整體 FPS。

比較兩台 Jetson Xavier NX，實驗組之 FPS 維持於 4.95 至 4.97、與 Raspberry Pi 5 相當，基準組則明顯較差且自第二輪之 4.87 逐輪下降至第六輪之 4.64。由於兩台裝置之硬體規格與軟體版本完全相同，推測此差異源自基準組先前曾用於本地測試與模型轉換，其殘留之套件、容器環境與資料對運作效能造成影響。

值得注意的是，量化精度對推論時間之影響並非絕對；理論上 INT8 應快於 FP16，第四輪之 YOLOv11n INT8 平均推論時間反較第六輪之 FP16 略慢，推測與各輪之裝置溫度、實驗間隔及當下車流量所致之運算負擔差異有關。整體而言，即便 YOLOv8n 與 INT8 於推論時間略具優勢，各組 FPS 差異仍甚微，顯示裝置運算能力主要消

耗於影像串流處理與影片儲存，使推論環節之優勢未能反映於整體 FPS；綜合本節數據，實驗組之 Raspberry Pi 5 搭配 .hef 模型展現最佳效能，並與實驗組之 Jetson 相當。

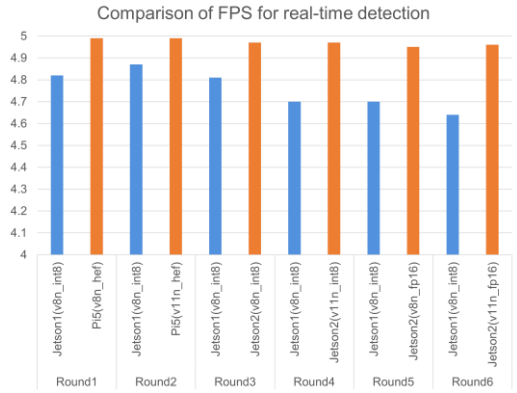


圖 19 即時辨識實驗 FPS 比較圖

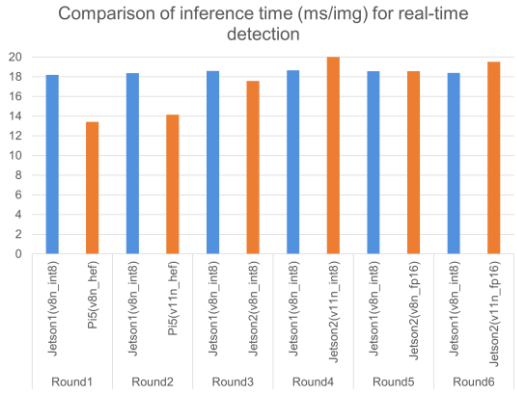


圖 20 即時辨識實驗推論時間比較圖

3.3.3系統穩定度分析

為評估即時辨識過程之效能穩定度，本節選取跨平台比較最具代表性之第一輪實驗，就基準組 Jetson Xavier NX 與實驗組 Raspberry Pi 5 之每秒 FPS 與記憶體用量進行分析。就每秒 FPS 而言，基準組 Jetson 之幀數如圖 21 所示，於實驗開始約 50 秒起出現較大幅度之波動與掉幀，其後呈現穩定與波動反覆之趨勢；實驗組 Raspberry Pi 5 則如圖 22 所示，幀數穩定維持於 5 FPS，僅出現極微小且短暫之掉幀並隨即恢復。此顯示 Jetson 之辨識穩定度較低，而 Raspberry Pi 5 搭配 Hailo-10H NPU 之配置更能滿足即時交通辨識對穩定性之需求。

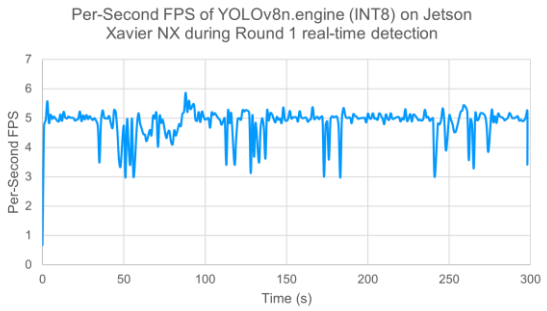


圖 21 第一輪基準組 Jetson YOLOv8n INT8 每秒 FPS 折線圖

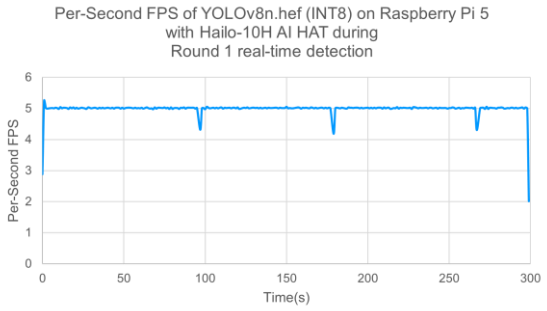


圖 22 第一輪實驗組 Pi5 YOLOv8n INT8 每秒 FPS 折線圖

於記憶體用量方面，基準組 Jetson 之數據如圖 23 所示，呈現隨實驗進行而持續上升之趨勢，此一趨勢與其 FPS 逐輪下降之現象於時序上相符，但現有數據尚不足以確認兩者間之因果關係，其確切成因仍待後續透過重複實驗、設備重新開機或系統資源監測等方式進一步驗證；實驗組 Raspberry Pi 5 之本體記憶體用量如圖 24 所示，整體偏低且未見持續上升。推測此差異與兩平台之記憶體架構有關：Jetson Xavier NX 採統一記憶體架構，CPU 與 GPU 共享同一組實體記憶體，故前處理、推論、後處理等所有流程之記憶體使用皆反映於同一數值中；Raspberry Pi 5 搭載之 Hailo-10H AI HAT 則為獨立模組，其內建 8 GB 記憶體專供 NPU 內部推論運算使用，與 Pi 5 本體 8 GB 記憶體相互獨立，並非可共享調度之單一資源池。因此，本研究所量測之 Raspberry Pi 5 本體記憶體用量，僅反映運算流程之負載，NPU 端之推論運算負載則獨立於此數值之外，故其用量偏低且穩定之現象。

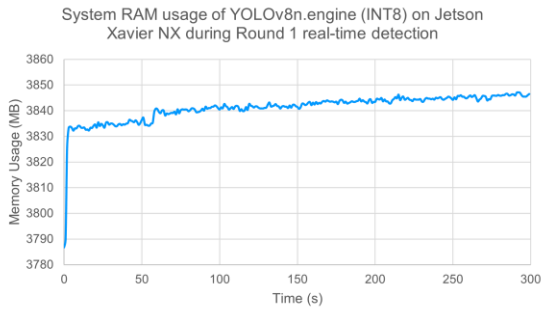


圖 23 第一輪基準組 Jetson YOLOv8n INT8 RAM 用量折線圖

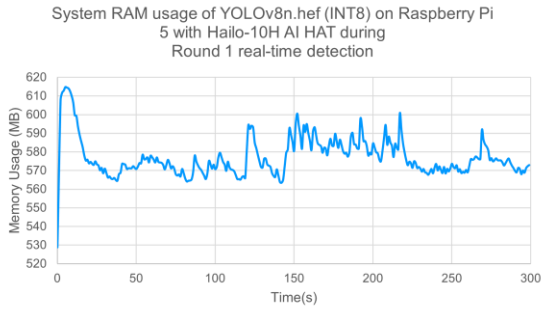


圖 24 第一輪實驗組 Pi5 YOLOv8n INT8 RAM 用量折線圖

3.3.4即時辨識結果比較

本節比較各輪實驗中基準組與實驗組於相同時間點所擷取之實際辨識結果。第一輪之基準組與實驗組辨識結果分別如圖 25 與圖 26 所示；即時辨識雖以 1920 × 1080 解析度拍攝、與資料集之 3840 × 2160 不同，辨識結果仍具一定準確度，顯示模型於不同解析度下仍具可靠辨識能力。基準組之 Jetson 搭配 YOLOv8n 於近處車輛之辨識信心度多達 0.8 以上，並能正確辨識畫面中之卡車此一於本地驗證中精度偏低之大型車輛，且維持良好之預測框定位；實驗組之 Raspberry Pi 5 搭配 YOLOv8n 整體信心度則較低，部分機車出現重複預測框，卡車亦於其大型車輛預測框外另現一低信心度之一般車輛預測框。

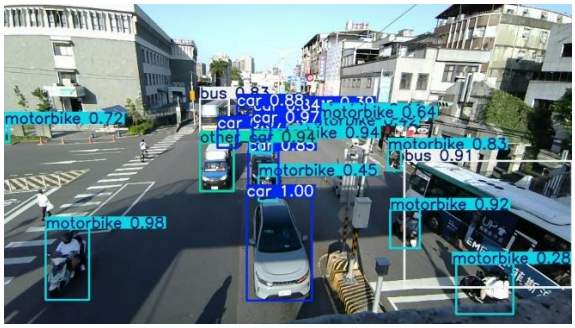


圖 25 第一輪基準組 Jetson YOLOv8n INT8 辨識結果

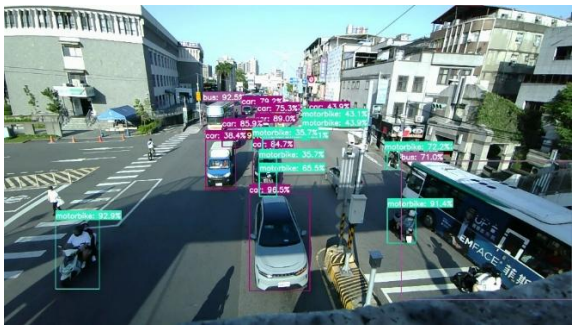


圖 26 第一輪實驗組 Pi5 YOLOv8n INT8 辨識結果

第六輪之基準組為 Jetson YOLOv8n INT8、實驗組為 Jetson YOLOv11n FP16，辨識結果分別如圖 27 與圖 28 所示。兩模型於此輪之辨識信心度互有高低、整體差異不大，基準組對畫面左側之卡車產生重複預測框，實驗組則對最近處之卡車出現預測框過大之情形，二者各有失誤。

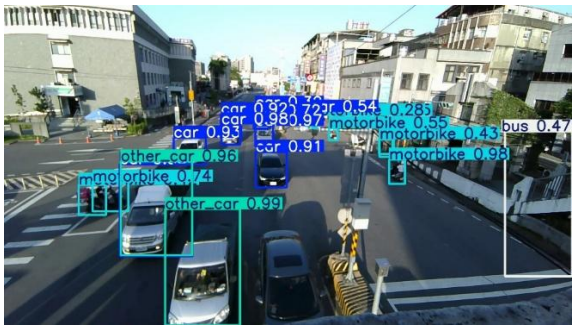


圖 27 第六輪基準組 Jetson YOLOv8n INT8 辨識結果

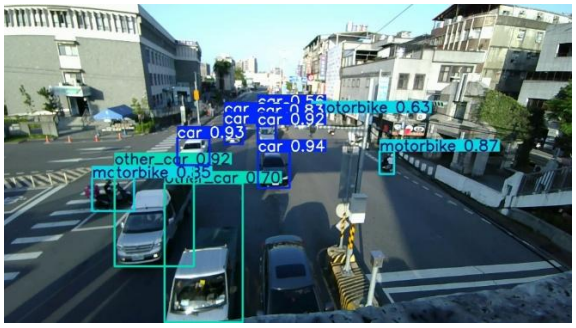


圖 28 第六輪實驗組 Jetson YOLOv11n FP16 辨識結果

4.結論與未來展望

4.1結論

本研究以天橋俯視角度為場景，建置一套部署於邊緣運算平台之多車種即時辨識技術驗證架構。研究以自建之多車種車流資料集訓練兩種 YOLO 輕量化模型，透過格式轉換與量化將其部署於 Jetson Xavier NX 與搭載 Hailo-10H AI HAT 之 Raspberry Pi 5 兩種邊緣裝置，並經本地驗證與實際場域即時辨識兩階段實驗，從辨識精度、推論效能與系統穩定度等面向進行系統性評估。

辨識精度方面，YOLOv11n 整體優於 YOLOv8n，以 Jetson 平台之基礎模型為例，其 mAP50 較 YOLOv8n 提升約 4.8%。就量化之影響而言，模型量化至 FP16 時兩模型之精度近乎無損，卻能明顯提升效能，YOLOv8n 與 YOLOv11n 之 FPS 相較基礎模型分別提升約 7.61% 與 16.18%；然而由 FP16 進一步與 INT8 相比，mAP50 分別下降約 1.96% 與 5.52%，FPS 之提升卻不及 1%，故於 Jetson 平台上 YOLOv11n 之 FP16 模型為兼顧精度與效能之最佳配置。進一步分析，Raspberry Pi 5 搭配 Hailo-10H NPU 於效能上相較 Jetson 明顯占優，YOLOv8n 與 YOLOv11n 之 FPS 較 Jetson 對應 INT8 模型分別提升約 22.45% 與 26.12%，展現最佳之推論效能。

針對各車輛類別，模型對 Car、Motorbike 與 Bus 三類常見車輛之精度均達實際交通辨識應用所需之水準；大型車輛類別則因標註樣本稀少且類別內車種之尺度與外觀差異甚大而精度偏低，後續可藉由增加樣本並細分車種加以改善。

實際場域即時辨識中，因裝置須同時處理多項任務造成之高負載，各配置之 FPS 均在 5 以下；其中以 Raspberry Pi 5 搭配 Hailo-10H NPU 之配置效能最佳，且每秒 FPS 穩定、記憶體用量低而無持續累積，Jetson Xavier NX 則出現 FPS 波動掉幀與記憶體持續累積、不利長時間運作之情形。整體而言，Raspberry Pi 5 搭配 Hailo-10H NPU 於效能與運作穩定性上符合即時交通辨識系統實際部署之需求，但其辨識信心度與預測框精準度仍略低於 Jetson，且較易出現重複預測框、模型轉換流程亦較繁瑣，顯示 Hailo 之轉換框架相較成熟之 TensorRT 仍有發展空間。綜上所述，本研究之具體貢獻如下。

- 建置一套可部署於邊緣運算平台之多車種交通車輛即時辨識技術驗證架構，整合資料集建置、模型訓練、量化與跨平台部署之完整流程。
- 系統性比較與分析 Jetson Xavier NX 與 Raspberry Pi 5 搭載 Hailo-10H AI HAT 兩種邊緣裝置於不同量化精度下之辨識精度與推論效能，歸納出最佳硬體平台與模型之配置。
- 透過本地驗證與實際場域即時辨識兩階段實驗設計，驗證所建置架構於真實交通場景下之辨識能力與運作穩定度，為智慧交通管理系統之邊緣裝置部署提供實證參考。

本研究成果驗證邊緣裝置結合輕量化 YOLO 模型應用於真實交通場景之可行性，可作為後續智慧交通管理系統於邊緣裝置選型與部署之參考依據。

4.2未來展望

本研究之資料蒐集與辨識測試，均於同一天橋、單一俯視角度，且於晴天午後之光照條件下進行，故本研究所呈現之辨識精度與效能表現，僅能反映模型於此一特定場域與拍攝條件下之辨識結果，尚不足以證明模型於不同道路類型、不同攝影機架設角度、不同光照條件或高遮蔽等交通環境

下，亦能維持相同水準之辨識能力，此為本研究現階段之限制。後續研究可於上述多樣化道路、時段、天候與拍攝角度之條件下蒐集資料並建立測試集，以進一步驗證模型於各類交通環境下之泛化能力，亦可將場景拓展至快速道路、巷弄與匝道等情境。

模型最佳化方面，本研究以格式轉換與量化為主，後續可進一步結合注意力機制、模型剪枝或混合式架構以精簡並強化模型結構；同時，YOLO 系列已陸續推出 YOLOv12、YOLOv13 與 YOLOv26 等新版本，未來可採用不同版本與尺寸之模型並於運算能力更高之邊緣平台上測試，以追求更佳之精度與效能。

實務應用方面，本研究所建立之辨識基礎可延伸至白牌機車誤入偵測、逆向行駛等違規偵測、交通事故辨識、車流量與密度估計，以及車距與行人禮讓行為判別等情境；於硬體方面，後續可改用 CSI 介面攝影機以突破 USB 傳輸限制，或採用深度攝影機進行距離偵測，亦可將系統部署於無人機等載具，以實現目標追隨與巡檢等更廣泛之交通辨識應用。

5.參考文獻

1. Alfikri, M. D., & Kaliski, R. (2024). "Real-time pedestrian detection on IoT edge devices: a lightweight deep learning approach", pp. 3-9.
2. Barthélemy, J., Verstaevel, N., Forehead, H., & Perez, P. (2019). "Edge-computing video analytics for real-time traffic monitoring in a smart city". Sensors, 19(9), 2048, pp. 5-21.
3. Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). "A survey of quantization methods for efficient neural network inference". In Low-power computer vision, pp. 291-326.
4. Guerrero-Ibáñez, J., Zeadally, S., & Contreras-Castillo, J. (2018). "Sensor technologies for intelligent transportation systems". Sensors, 18(4), 1212, pp. 5-8.
5. Z. He, K. Wang, T. Fang, L. Su, R. Chen and X. Fei. (2025). "Comprehensive performance evaluation of YOLOv11, YOLOv10, YOLOv9, YOLOv8 and YOLOv5 on object detection of power equipment". 37th Chinese Control and Decision Conference (CCDC), Xiamen, China, pp. 1281-1286.
6. IEEE STD 754-2019. (2019). "IEEE standard for floating-point arithmetic". IEEE, Piscataway, NJ, USA, pp. 1-84.
7. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). "Quantization and training of neural networks for efficient integer-arithmetic-only inference". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2704-2713.
8. L. Jiao et al. (2019). "A survey of deep learning-based object detection". IEEE Access, vol. 7, pp. 128837-128868.
9. Kapoor, S., & Narayanan, A. (2023). "Leakage and the reproducibility crisis in machine-learning-based science". Patterns (New York, N.Y.), 4(9), 100804, pp. 1-11.
10. Khanam, R., & Hussain, M. (2024). "YOLOv11: an overview of the key architectural enhancements", pp. 1-6.
11. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. (2017). "ImageNet classification with deep convolutional neural networks". Commun. ACM, 60(6), pp. 84-90.
12. V. K. Kukkala, J. Tunnell, S. Pasricha and T. Bradley. (2018). "Advanced driver-assistance systems: a path toward autonomous vehicles". IEEE Consumer Electronics Magazine, vol. 7, no. 5, pp. 18-25.

13. S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang and W. Shi. (2019). "Edge computing for autonomous driving: opportunities and challenges". Proceedings of the IEEE, vol. 107, no. 8, pp. 1697-1716.

14. Loshchilov, I., & Hutter, F. (2017). "Decoupled weight decay regularization", pp. 2-9.

15. Mills, K. G., Fetrat Qharabagh, M., Qiu, W., Han, F. X., Salameh, M., Lu, W., Jui, S., & Niu, D. (2025). "Applying graph explanation to operator fusion", pp. 1-6.

16. R. Padilla, S. L. Netto and E. A. B. da Silva. (2020). "A survey on performance metrics for object-detection algorithms". International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, pp. 237-242.

17. J. Redmon, S. Divvala, R. Girshick and A. Farhadi. (2016). "You only look once: unified, real-time object detection". IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 779-788.

18. Sapkota, R., & Karkee, M. (2024). "Comparing YOLOv11 and YOLOv8 for instance segmentation of occluded and non-occluded immature green fruits in complex orchard environment", pp. 1-4.

19. W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu. (2016). "Edge computing: vision and challenges". IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646.

20. Sohan, M., Sai Ram, T., Rami Reddy, C.V. (2024). "A review on YOLOv8 and its advancements". In: Jacob, I.J., Piramuthu, S., Falkowski-Gilski, P. (eds) Data Intelligence and Cognitive Informatics. ICDICI 2023. Algorithms for Intelligent Systems. Springer, Singapore, pp. 4-9.

21. Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). "A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS". Machine Learning and Knowledge Extraction, 5(4), pp. 1680-1716.

22. Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). "Deep learning for computer vision: a brief review". Computational intelligence and neuroscience, 7068349, pp. 1-3.

23. M. Won. (2020). "Intelligent traffic monitoring systems for vehicle classification: a survey". IEEE Access, vol. 8, pp. 73340-73358.

24. Wu, H., Judd, P., Zhang, X., Isaev, M., & Micikevicius, P. (2020). "Integer quantization for deep learning inference: principles and empirical evaluation", pp. 1-6.

25. Zhu, Li & Yu, Fei & Wang, Yige & Ning, Bin & Tang, Tao. (2018). "Big data analytics in intelligent transportation systems: a survey". IEEE Transactions on Intelligent Transportation Systems, pp. 1-16.

26. Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye. (2023). "Object detection in 20 years: a survey". Proceedings of the IEEE, vol. 111, no. 3, pp. 257-276.