逢甲大學學生報告 ePaper

運用強化學習技術規劃幸福巴士路線之研究－以南投市為例

A Comparative Study of Traditional and AI-Based for Nantou City DRTS Route Planning

作者：盧廷昀、陳昱萱、白婕瑜、陳唯賓

系級：運輸與物流三乙

學號：D1160084、D1160437、D1160467、D1160173

開課老師：蘇昭銘

課程名稱：整合性專題實作(二)

開課系所：運輸與物流學系

開課學年：113 學年度 第 2 學期

中文摘要

本研究針對偏鄉地區交通服務不足的問題，探討以深度強化學習（Deep Q-Network, DQN）為基礎的智慧化公車路線規劃模式，並以南投市為實證場域。由於傳統固定式公車路線在地形破碎、人口分布零散的地區難以有效覆蓋居民需求，導致交通可及性低落、公共運輸資源使用效率不彰。為此，本研究整合票證資料、手機信令資料與問卷調查，建立符合在地需求的路網與節點評分機制，進而應用 DQN 模型進行幸福巴士路線規劃。

為驗證人工智慧方法於偏鄉公車路線規劃的成效，本研究以涵蓋率、服務高需求點、總路徑長度及重疊比例等指標，比較 DQN 模型與專家路線的表現。結果顯示，DQN 所生成的幸福公車路線成功覆蓋專家規劃重點區域東山里與內興里，涵蓋率分別由 44.51% 提升至 73.17%，及由 36.82% 提升至 91.33%；同時在路線總長度與彎繞度上略優於人工規劃，並將整體路線與個別路段與既有公車路線的重疊比例分別降至 5.11% 與 5%。此外，敏感度分析結果亦顯示，獎勵函數中的「重要點位權重」與「重疊既有路線懲罰」為關鍵參數。若移除重要點位權重會犧牲公共運輸可及性，而取消重疊懲罰則會降低資源使用效率。綜合評估可確認，DQN 模型在整體路線規劃品質上優於傳統方法設計之專家路線，並具高度調適性與可行性，未來可作為偏鄉地區導入智慧交通系統的重要參考依據。

關鍵詞：幸福公車、偏鄉交通、深度強化學習、路線規劃

Abstract

This study explores an intelligent bus route planning model based on Deep Q-Network (DQN) to address insufficient transportation services in rural areas, with Nantou City as the case study. Traditional fixed-route buses are often ineffective in fragmented terrains with dispersed populations, resulting in poor accessibility and inefficient utilization of transport resources. To overcome these challenges, the research integrates smart card data, mobile signaling data, and questionnaire surveys to develop a demand-oriented network and node scoring mechanism, which is then applied to the DQN model for Demand Responsive Transit System (DRTS) route planning.

The study evaluates DQN-generated routes against expert-designed ones using indicators such as coverage rate, service to high-demand points, total route length, and overlap ratio. Results indicate that the DQN-generated DRTS routes successfully covered the key areas emphasized in expert planning—Dongshan and Neixing Villages—raising coverage rates from 44.51% to 73.17% and from 36.82% to 91.33%, respectively. At the same time, the routes slightly outperformed manual planning in terms of total length and curvature, while reducing the overlap with existing bus services to 5.11% for the overall network and 5% for individual segments. Sensitivity analysis further revealed that “importance weighting of key nodes” and “overlap penalties” are critical parameters, as removing them sacrifices accessibility and resource efficiency. Overall, the findings confirm that the DQN model outperforms expert-designed routes in planning quality, demonstrating strong adaptability and feasibility, and serving as a valuable reference for implementing intelligent transportation systems in rural regions.

Keywords: Deep Reinforcement Learning (DRL), DRTS, Route Planning, Rural Transportation

目錄

圖目錄

表目錄

第一章　緒論

隨著社會與城市快速發展，城鄉之間在資源分配與交通可及性上的差距日益擴大，偏鄉地區交通不便問題已逐漸成為限制居民生活品質與經濟發展的重要因素。在公共資源有限的情況下，如何有效規劃與分配交通服務，已是各地政府與相關單位需面對的課題。本研究不僅以優化南投市幸福公車路線為目標，更希望建立一套可複製、可調整之智慧化路線規劃流程與模型，使其未來得以應用於其他交通資源不足地區，作為提升偏鄉交通可及性重要參考工具。

本章將說明本研究之發展背景、研究動機與問題意識，並進一步闡述研究目的、研究範圍與研究方法，最後透過研究流程圖，介紹本研究整體架構與執行步驟，以作為後續章節分析與討論的基礎。

1.1 研究動機

近年來，因應高齡化社會與偏鄉資源不均問題，台灣各縣市政府積極推動「幸福公車」政策，以改善偏遠地區民眾在就醫、就學、採購等方面的交通不便。然而在實務操作上，多數公車路線仍依賴傳統以問卷調查、人工判斷與固定式規劃的方式進行設計，導致在面對地理條件破碎、人口分布零散或需求變動快速的區域時，常出現服務涵蓋不足、資源重複浪費或無法即時調整等問題。

研究關注於傳統人工規劃方法與人工智慧方法在公車路線設計上的效率差異。隨著人工智慧技術發展日趨成熟，強化學習（Reinforcement Learning）具備從經驗中學習、動態調整並優化決策策略的能力，特別適合應用於具連續狀態與複雜約束的交通路線規劃問題。深度強化學習中的 Deep Q-Network（DQN）方法可透過模擬環境與獎懲機制，持續提升策略效益，突破傳統方法在規模與彈性上的限制。

本研究即以 DQN 為核心演算法，建立一套可自動化學習之公車路線規劃模型，並以南投市為驗證案例，比較 AI 與人工規劃於涵蓋率、服務高需求點命中率、總路徑長度與重複區域避開能力等多項指標的表現差異。研究目的在於評估深度強化學習於實務路線設計中的可行性與優勢，並建立具通用性與可複製性之規劃流程，為未來交通政策制定與偏鄉運輸服務優化提供參考依據。

1.2 研究目的

本研究旨在建構一套結合深度強化學習與實際需求資料之幸福公車智慧規劃系統，並以南投市為應用場域，驗證其效能與可行性。具體研究目的如下：

以下三點為本研究之研究目的：

一、分析了解南投市公共運輸的服務現況：透過票證資料與路線資訊掌握現有路線的運行情況與資源分布。

二、找出南投市幸福公車的需求：運用問卷調查與信令資料找出具潛力且現階段未被涵蓋的高需求區域。

三、運用深度強化學習進行幸福公車最佳化規劃：設計環境狀態與獎懲機制，利用 DQN 訓練模型進行最佳化路徑搜尋。

1.3 研究範圍與限制

一、 研究範圍：研究對象為南投縣南投市各里村里居民的交通需求概況，和常去的地點頻率及往返的運輸方式。

二、 信令資料時間限制：108 年 4 月 20 日至 4 月 27 日及 108 年 9 月 21 日至 9 月 28 日。

三、 票證資料時間限制：112 年 1 月至 112 年 12 月。

四、 訪談對象限制：南投市各村里長在收集民意後，代表該里填答之。

五、 服務限制：幸福公車僅限於南投市內運行，不延伸至他縣市。

1.4 研究流程

本研究流程如圖 1 所示，步驟說明如下：

一、 確定專題研究方向：確定本研究範圍之現況、相關議題及現有實例並確定研究問題。

二、 背景及文獻探討：對於本研究相關之文獻內容進行回顧及彙整，內容包括偏鄉公車營運規劃、DQN 於路線之應用研究及其他應用研究。

三、 南投市基本資料調查：蒐集研究區域之基礎資料，建立環境背景，包括：進行交通盤點、公共運輸涵蓋率、年齡分布。

針對本研究進行研究方法設計，分成南投市交通盤點、信令資料研究、問卷調查三部分。

四、 研究方法探討：針對本研究進行研究方法設計，分成票證資料分析、信令資料研究、問卷調查三部分。

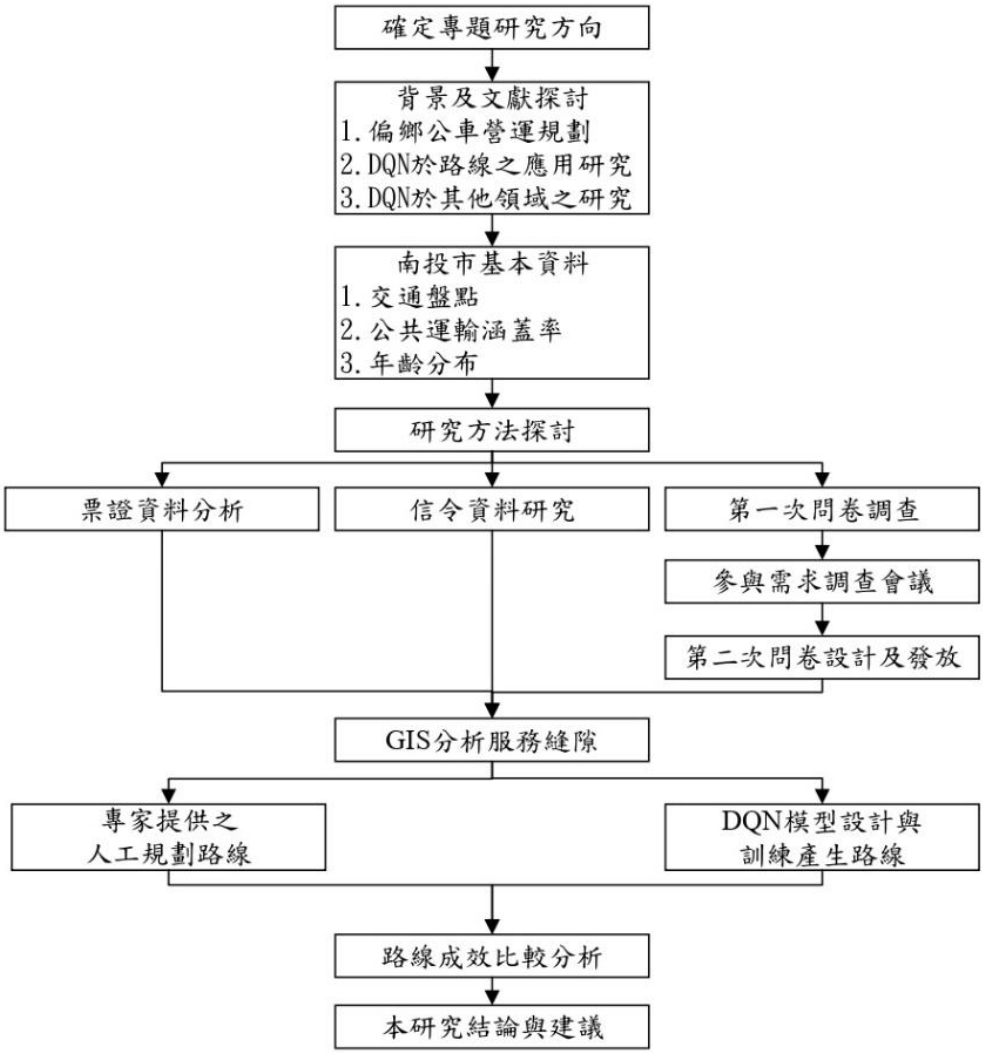
五、 GIS 分析服務縫隙：透過地理資訊系統（GIS）找出目前公共運輸服務較缺乏地區，並進行數據呈現，作為路線優化與模型訓練的重要參考依據。

六、 規劃南投市幸福公車路線及探討：此部分會分為專家提供人工路線規劃以及 DQN 模型設計與訓練兩種方式，比較二者結果，於公共運輸涵蓋率提升和最短路徑間評估，並得出最佳化之路線。

（一）專家提供之人工規劃路線：由逢甲大學區域研究中心提供人工規劃路線，供本團隊進行研究。

（二）DQN 模型設計與訓練產生路線：設計並訓練深度 Q-Network（DQN）模型，結合網格分數、交通資料與空間特徵，使模型能夠自動規劃高效合理的服務路線。

七、 本研究結論及建議：綜整研究成果，提出對南投市幸福巴士路線的具體建議，並就研究方法、模型應用、可複製性等面向提供後續研究建議。



資料來源：本研究整理

圖 1 研究流程圖

第二章　文獻回顧

臺灣偏鄉地區因地形複雜、道路條件不佳及人口稀少，導致傳統客運業者難以提供服務，公共運輸模式以固定班次和固定路線為主，缺乏彈性，無法有效滿足居民需求。這種情況讓當地民眾在日常生活、通勤、就醫及洽公等方面面臨諸多困難。為了改善偏鄉交通問題，政府近年積極推動相關計畫，期望提升偏鄉公共運輸的服務品質與居民的生活便利性，實現交通公平。

本章共分三節，透過整理與分析相關文獻，探索研究主題的核心概念與應用。第一節以偏鄉公車營運規劃，回顧早期偏鄉公車路線規劃的實務經驗與執行成果；第二節 DQN 於路線之應用研究，分析其在交通運輸領域中的實作情形；第三節 DQN 於其他領域之應用研究，作為本研究未來發展與跨域應用之參考依據，分別敘述如下：

2.1 偏鄉公車營運規劃

針對偏鄉地區的公車營運規劃問題，學者們提出多種結合 GIS 與數理模型之規劃方式。蘇昭銘等人（2015）利用 TOP 問題之觀念建構偏鄉公車路線設計模式，透過民意代表之間交流得出預設公車路線行經站點之具體位置，並結合 GIS 以及基因驗算法進行最佳化分析，加入如營運費用限額、每增加一個家戶所增加單位營運成本大小作為考量公車營運效益之依據，最終求解得出可供選擇的候選公車路線。

宮偉杰（2015）針對偏遠地區交通運輸進行規劃以新竹縣五峰鄉為案例，指出如花園村與竹林村等部落長期未有大眾運輸服務，導致約八成居民未被涵蓋。研究採用結合 GIS 之距離限制之 p 中位問題（DCPMP）模式，進行最適站點配置規劃。透過分析部落家戶密集度與道路條件，以 GIS 執行 500 公尺服務範圍可及性分析後，納入 DCPMP 模式進行評估，並以最小化居民通勤距離與時間為目標。此外，參考 Matisziw（2006）之敏感度分析架構，調整「車上時間權重」發現，當該權重增加時，站點數量會減少，有助於強化站點設置之效率與合理性。

蘇昭銘等人（2013）亦提出一套完整的公車路線評估指標體系，透過問卷調查進行重要度與滿意度分析，運用 GIS 空間分析與 IPA 方法，建構包含服務人口、路線彎繞度、行駛時間、重複路段、場站涵蓋率及可及性等八大指標。該研究強調，透過量化數據與地圖分析，可有效協助政府在偏鄉新增或停駛公車路線時，兼顧財政效益與民眾交通權益，進而增加路線的可行性與信服度。

2.2 DQN 於路線之應用研究

強化學習是指機器透過主體和環境互動過程中學習，加入獎懲機制，並考慮未來的狀態，達到預期的目標，以獲取最大化的預期利益。通常使用表格或是簡單的函數計算價值函數，當數據較為龐大時，以表格的形式計算價值函數會具有局限性。如 Q-learning 能夠較為簡易地評估出不同選擇下所帶來的結果，得出較為優秀的結果，但 Q-learning 需要透過 Q-table 以表格的形式儲存，由當前狀態下所選擇之動作所帶來的結果，計算結果會受到存儲形式的限制。

深度強化學習是在強化學習的基礎上，運用深度神經網路透過自動學習與參數訓練，使價值函數更接近最優解，進而處理更複雜的決策問題。深度強化學習具有從原始數據中找出特徵並自動學習的能力，更加善於處理較為複雜的問題。例如 DQN 使用深度神經網路 Q-network 代替原本 Q-learning 所使用的 Q-table，能在連續狀態空間中評估所有可能動作的回饋值，並透過主網路與目標網路分工學習與更新 Q 值，以提升模型穩定性。並透過經驗回放的能力打破數據之間的相關性，減少主觀意願對於研究結果的影響，更好地找出最優解。

林婉伃（2024）運用 Deep Q-Network（DQN）進行多目標強化學習，以優化生產、運輸批量與配送路徑。透過決策價值和自定義優先權重，提升 DQN 在單一目標獎勵機制的評估能力。本研究針對單一供應商對多個同質零售商的易腐商品生產存貨路徑規劃，考量成本、交期等多重目標，並根據不同範例調整參數。最終，利用 DQN 的損失值、平均獎勵與收斂速度等績效指標進行評估，並透過敏感度分析發現：零售商數量對不同目標具正向影響，時間週期對交貨時間影響最為顯著，而運輸車輛數則呈負相關，且部分因子間存在顯著交互作用。

柯威年（2021）整合利用 OpenAI 的 GYM 環境，設定獎勵機制並將圖資轉換成網格路徑，以 RobotMaster 移動機器人展示路徑，比較四種演算法（SARSA、Q-Learning、DQN、A2C）在避障礙路線的表現。SARSA 和 Q-Learning 是屬於表格方法中的時間差分學習演算法，透過逐步更新表格來學習最佳行動策略，適合小型網格地圖，學習速度快；DQN 和 A2C 則屬於近似方法中的深度強化學習，利用深度神經網路來學習近似策略與價值函數，適合處理大空間和複雜環境。本研究結果為 Q-Learning 在迭代次數、記憶體使用量以及達成目標所需步數方面表現最佳，顯示它在路線規劃和避開障礙物時特別有效率，並透過機器人進行實驗及驗證。

許詠翔（2021）利用 DQN 進行室內路徑規劃以及障礙避繞，透過鐳射掃描環境並建立 Simultaneous localization and mapping(SLAM)與 adaptive Monte Carlo localization（AMCL），建立虛擬環境並使用 DQN 進行訓練，透過數據交互的方式提升模型的穩定性，在逐步訓練的過程中使機器人獲得自我導航的能力。透過給機器人設定起始點與目標點，以週期少與時間短為高分的報酬作為機器人在進行路徑規劃和障礙避繞時的評量依據，進行 DQN 訓練後在 Gazebo 建立的情景中輸出動作，最後將訓練完成的模型應用於現實情景中，並得到出色的成績。

黃雅鈺（2021）在研究中運用深度強化學習技術 Deep Q-Network（DQN）進行植保機的農藥噴灑路徑規劃，並在特定範圍內結合 PyQt5 套件生成獎賞相關的 Excel 檔，作為強化學習的訓練資料來源。透過 TensorFlow 開發環境，建立 DQN 訓練模型，進一步探討兩層神經網路與三層神經網路的表現差異，以及不同飛行方向選擇對模型的影響。研究結果顯示，當獎賞與懲罰的設置越明確時，COST 曲線會更加平滑。在路徑規劃表現上，兩層神經網路能夠產生比三層神經網路更優化的路徑，但三層神經網路 COST 值變化較為平穩，其原因是三層神經網路考慮的參數過多導致 overfitting。在大範圍地圖環境下，行走方向選擇較少的，覆蓋率的提升更具優勢，能達到更佳的作業效率。

2.3DQN 於其他領域之應用研究

王智文（2020）研究如何在有限資源和時間內，針對平行機台調度問題（Parallel machine scheduling problem ,PMSP）進行組合最佳化。該問題的缺點是大規模的問題比較難快速求解，因本研究利用深度學習的 DQN 演算法，以深度神經網路逼近 Q 值函數取代 Q-Learing 提升效率，研究核心方法包括以下三部分：

一、經驗回放：把過去的數據隨機抽樣進行訓練，使模型更穩定。
二、目標網路：設立一個網格計算學習目標，定期更新、穩定學習目標。
三、貪婪策略：模型結合隨機選擇與最佳選擇，確保模型可以充分學習。

此外，本研究還有設置獎勵機制，使智能體能夠從過去的經驗不斷學習與更新狀態，以有效解決平行機台調度問題的核心目標-完工時間最小化，如果所有任務平均分配給所有機台，此決策結果對平行機台調度問題為最佳解。

沈靖筌（2023）比較注意力機制與深度強化式學習，用於自主無人機上的差異性，主要分為 Bahdanau Attention、GTrXL Attention、DQN 三種演算法，並利用 python 語法等，導入 AirSim 的虛擬環境中訓練，實驗結果可以得知 DQN 之優勢為不須先備知識或大量資料，便可在與環境互動的過程中求得 Q 值；其劣

勢為會專注於如何避障，而非探索路徑，可能導致期望與實際間有誤差，另有梯度消失、學習曲線不穩定等問題需要注意。

蔡秀榛（2019）針對無 GPS 信號的室內倉儲環境，透過 DQN 實現四旋翼飛行器的自主導航。利用如 RealSense D435i 等視覺捕捉設備以及 ORB-SLAM2.0 和 OctoMap 演算法的建構 3D 環境地圖。利用 DQN 演算法來計算獎勵和懲罰，如碰撞懲罰、障礙懲罰、時間懲罰、方向獎勵等，以這些獎勵影響四旋翼飛行器的動作。使用 Unreal Engine 和 AirSim 建構模擬環境，驗證 DQN 算法在不同場景下的表現。當到達目的地後，四旋翼飛行器將開始使用微型 YOLOv3 對庫存產品進行辨識，並且將結果傳送至 point of sale(POS)系統。

2.4 綜合探討

一、偏鄉地區民眾居住分散，常導致公車站點設立困難。為提升規劃效益，學者普遍結合 GIS 與數學最佳化模型進行分析。蘇昭銘等人（2015）利用 TOP 問題構建公車路線，結合民意代表交流與基因演算法，求解候選路線。宮偉杰（2015）則以五峰鄉為例，採用距離限制 p 中位問題（DCPMP）模式，搭配 GIS 進行 500 公尺服務範圍分析，並以最小化通勤時間為目標優化站點配置，進一步透過敏感度分析提升站點設置的彈性與效率。此外，蘇昭銘等人（2013）提出一套公車路線評估指標體系，結合問卷調查、IPA 方法與 GIS 分析，涵蓋服務人口、可及性、路線彎繞度等八項指標，提供偏鄉公車路線增設與裁撤之客觀依據。整體而言，前述研究均強調數據導向與空間分析在偏鄉運輸規劃的重要性，然而傳統模型需仰賴大量主觀問卷數據，易受偏誤影響。因此，本研究改採「涵蓋率」作為規劃依據，期望降低主觀因素，提升模型客觀性與實用性。

二、近年來，DQN 技術逐漸應用於路線規劃與運輸最佳化，如林婉仔（2024）運用 DQN 來調整物流配送的最佳路徑，根據成本、時間和需求動態變化來決策；柯威年（2021）則比較不同強化學習演算法在機器人導航中的表現，發現 DQN 及 A2C 在處理大規模路徑規劃上更具優勢。許詠翔（2021）研究 DQN 在室內路徑規劃中的應用，透過 SLAM 與 AMCL 技術，使機器人能夠自主學習與導航。黃雅鈺（2021）則將 DQN 應用於農藥噴灑機的路徑規劃，透過 PyQt5與 TensorFlow 進行模型訓練，驗證不同深度神經網路結構對路徑規劃的影響。這些研究表現出 DQN 在路線規劃問題上的學習與適應能力，可用於解決本研究中偏鄉公車路線規劃問題，因此本研究將 DQN 作為主要研究方法，希望透過研究驗證 DQN 在固定起訖點的環境中，對於偏鄉公車的路線規劃問題的應用是否具有正向的意義。

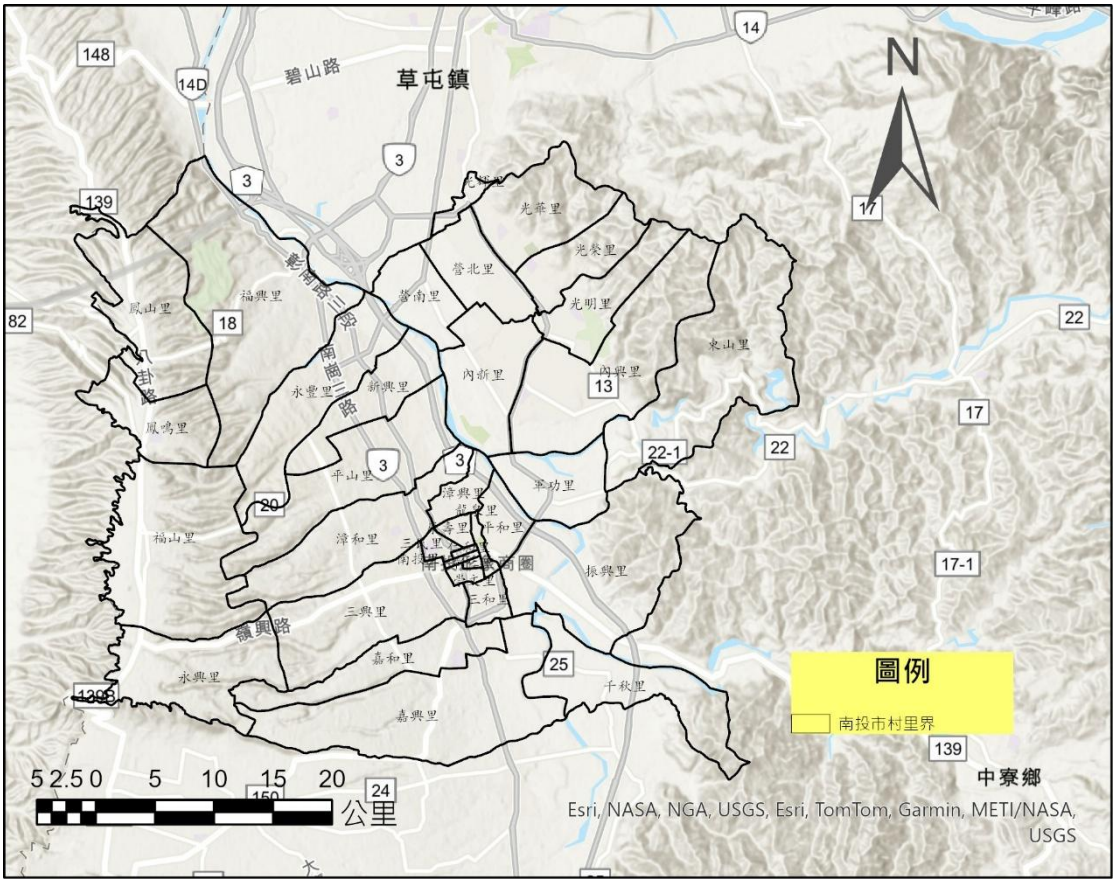
三、除了路線規劃應用，DQN 也廣泛的運用於不同領域，如王智文（2020）研究如何讓工廠的機台調度更有效率，透過 AI 來分配生產工作，減少延誤時間；沈靖筌（2023）探討 AI 在無人機導航上的應用，發現 DQN 雖然能幫助無人機避開障礙物，但可能忽略最短飛行路線；蔡秀榛（2019）研究如何在有 GPS 的倉儲環境中，讓飛行器學習自主導航與辨識貨物。以上研究顯示 DQN 不僅交通領域，也能應用於機器人導航、智慧倉儲等場景，未來可進一步探索其在偏鄉公車規劃中的潛力，如自適應調整路線與即時應變規劃等，讓本研究所規劃之路線及站點能達到最佳化。

四、隨著技術的進步，無須再透過空想或實測去測試路線可行性，可以進一步透過人工智慧去模擬，避免更多的不確定性和成本的投入，上述文獻中提及多種演算法，DQN 在障礙避繞問題、植保機針對蟲害噴灑農藥的路徑問題、最短路徑問題等多項研究問題上表現較為出色，在偏鄉路線設計上，需要滿足需求且有效的減少行駛的路徑，而 DQN 在這方面都表現得不錯，且 DQN 不需具備先備知識，適合用於新闢路線及站牌的選擇，所以本研究以 DQN 為主要的深度學習方法，運用 PyQt5建立環境並賦予獎勵機制，希望能透過此演算法，求得最符合南投市幸福公車需求之站點及路線。

第三章　南投市交通特性分析

3.1 社經資料

南投市位於南投縣西北側，全市共有 34 個里，村里分布圖如圖 2 所示，黑色區域為南投市之範圍，面積 71 平方公里，南北長與東西寬皆約 8.2 公里，屬於臺灣面積第三大的縣轄市，在南投縣所有行政區中排行第二小。南投市北接草屯鎮，東臨名間鄉，南接集集鎮和中寮鄉，西接名間鄉，不僅是平原地帶南北往來必經之所，也身處通往鄰近各鄉鎮的交通樞紐。為南投縣的政治、經濟與文化中心，有較多的旅次吸引量。



資料來源：本研究整理

圖 2 南投縣南投市區域圖

南投市總人口數約 93,884 人，全市各年齡人口分佈如表 1，表中顯示南投市高齡人口比例約達 20%，且受限於南投市內公共運輸服務之據點，主要集中於市中心龍泉里、康壽里、三民里、仁和里、南投里、彰仁里、崇文里、漳興里等八里。距離市中心較遠村落，不易搭乘公共運輸，而目前南投市居民大多數需使用私人運具，才能滿足購物洽公、通學、通勤等交通需求。

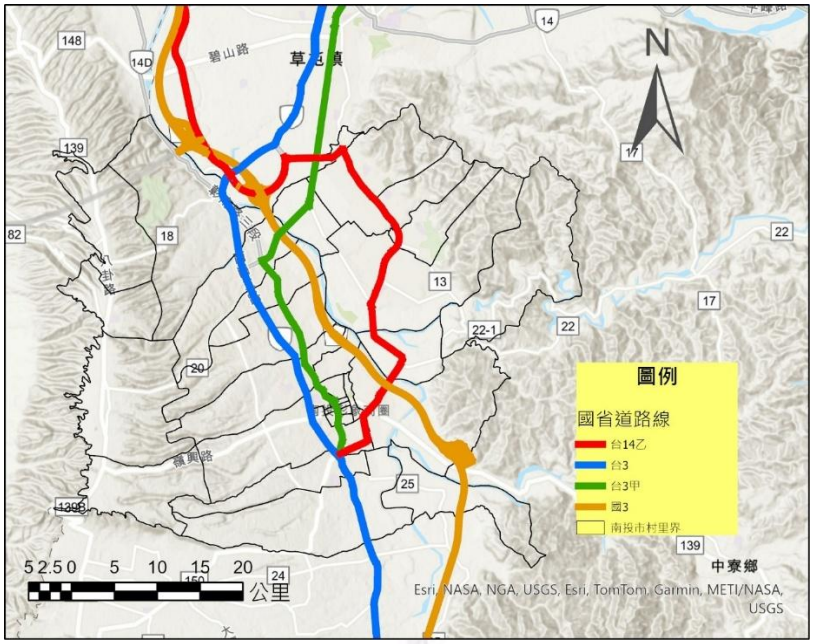
表 1 南投市 113 年 8 月人口數及年齡分布

年齡層	人口數(人)	百分比(%)
學齡前(0-5 歲)	3,381	3
國小學齡 06-12 歲	6,283	6
國中學齡 13-15 歲	2,668	3
高中學齡 16-18 歲	2,534	2
19-30 歲	14,336	15
31-40 歲	13,458	14
41-50 歲	14,559	15
51-64 歲	20,570	22
65 歲以上	19,476	20
總計	93,884	100

資料來源：南投縣政府人口統計資訊平台，本研究整理

3.2 南投市道路路網分布

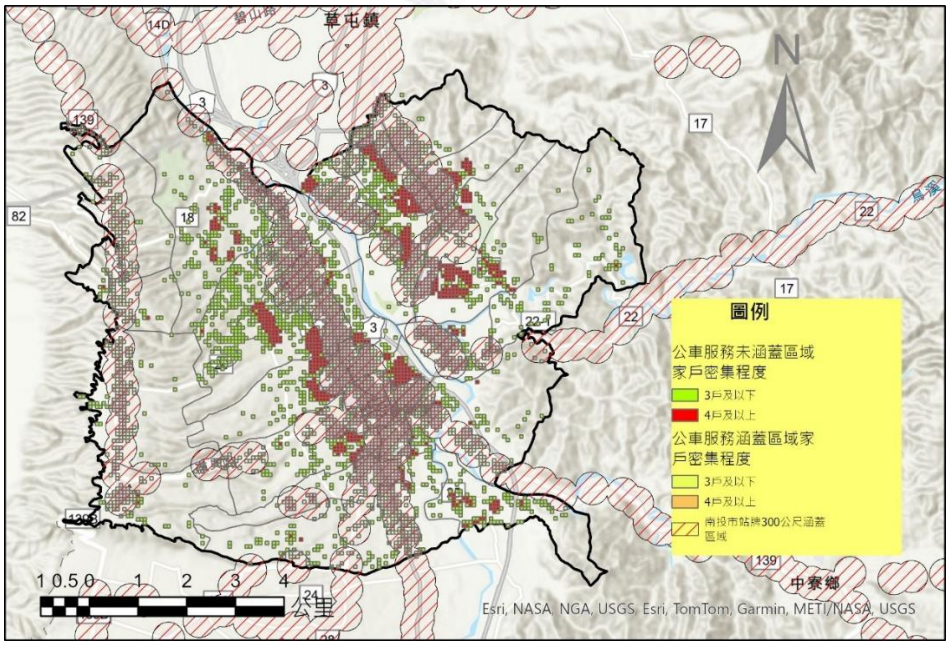
南投市主要交通要道為國道 3 號、台 3 線、台 3 甲線、台 14 乙線等，如圖 3 所示，國道 3 號為青藍色路線、台 3 線為深藍色路線、台 3 甲線為深綠色路線、台 14 乙線為紅色路線。市內公路汽車客運多行以行駛台 3 甲線及台 14 乙線為主。人口集中在市區中心，商業活動集中，人口數約為 9 萬 3 千多人，市民就醫、採買生活必需品或至外縣市就學、就職者，皆需至市區消費或轉乘。但因為臨近山區，既有的公車路線無法充分地滿足當地居民的需求。



資料來源：本研究整理

圖 3 主要行經南投市之國道、省道路線圖

南投市內公車路線多為南北向，較少東西向之公車路網，市內站牌涵蓋家戶如圖 4 所示，圖上綠色區塊為住戶較分散之地區、紅色區塊為住戶較集中之地區，而紅底斜線為公車站牌服務範圍，我們藉由家戶門牌數與站牌服務範圍得到南投市各里的公共運輸服務涵蓋率以及各里的家戶分佈情況，目前仍有多個地區之民眾尚未被服務。



資料來源：本研究整理

圖 4 南投縣南投市站牌涵蓋家戶圖

3.3 南投市內公共運輸現況

南投市既有公共運輸公路客運路線如表 2。根據表中可知現況公共運輸共 41 條（含延伸線），分別為市區公車 4 條、國道客運 5 條及一般公路客運 32 條，通往南投市區主要道路須行經台 3 線，因此主要通往南投市市區公車乃集中於台 3 線。既有公共運輸可分為 7 種路線，分別說明如下：

一、台中、草屯、南投—溪頭、杉林溪：3 路、3A 路、3B 路、5 路等 4 條路線。
二、台中、草屯、南投—竹山：1632 路、6742 路、6742A 路等 3 條路線。
三、基隆、台中—南投：1657 路、1806 路、1831 路、1831A 路等 4 條路線。
四、台中—水里、濁水：3622 路、6333 路、6333A 路、6333C 路、6333D 路、6333E、6333F 路等 7 條路線。
五、南投—埔里：6672 路、6672A 路、6875 路等 3 條路線。
六、草屯、南投—彰化：6916 路、6916A 路、6917 路、6917A 路、6917B 路、6918 路、6918A 路、6919 路、6919A 路、6924 路、6925 路、6925A 路、6926 路、6926A 路、6927 路、6928 路、6928A 路、6929 路等 18 條路線。
七、南投ㄧ內城：6920 路等 1 條路線

表 2 南投市既有公共運輸現況

類別	路線編號	路線名稱	班次	
			平日	假日
市區公車	3	草屯—南投—溪頭[經中興]	7	0
	3A	草屯—南投—溪頭[經彰南路]	3	0
	3B	南投ㄧ溪頭[經彰南路]	無資料	
	5	南投—水里	3	0
國道客運	1632	台北—草屯—竹山	2	2
	1657	高鐵臺中站ㄧ國道 3 號ㄧ南投	24	24
	1806	基隆ㄧ南投	2	2
	1831	臺北ㄧ南投	35	32
	1831A	臺北ㄧ南投[經霧峰]	2	2
一般公路客運	6322	台中—南崗—水里	2	2
	6333	台中—中興—水里	48	38
	6333A	台中—中興—水里[經國道 3 號]	30	28
	6333C	台中—中興—水里[經中興大學]	6	6

類別	路線編號	路線名稱	班次	
			平日	假日
	6333D	臺中—高鐵臺中站—水里[台灣好行集集線]	6	6
	6333E	台中—中興—南投	4	6
	6333F	台中—中興—濁水	4	0
	6672	南投—埔里(經國道 6 號)	2	4
一般公路客運	6672A	南投—埔里[經國道 6 號](繞駛中興高中)	2	0
	6742	台中—竹山	4	4
	6742A	台中—竹山[經南投高中]	2	0
	6871	臺中—杉林溪(經南投、鹿谷)	4	4
	6875	南投—中興—埔里	6	6
	6916	彰化—南投(經省訓團)	21	20
	6916A	彰化—南投(經省訓團)[經南投高中]	1	0
	6917	彰化—南投(經南崗)	15	12
	6917A	南投—彰化(經南崗)[經南投高中]	1	0
	6917B	草屯—彰化(經社口)	10	6
	6918	彰化—南投(經林子頭)	21	22
	6918A	彰化—南投(經林子頭)[經南投高中]	1	0
	6919	彰化—南投(經縣庄)	2	1
	6919A	彰化—南投(經縣庄)[經南投高中]	1	0
	6920	南投—內城	14	14
	6924	南投—員林(經碧山)	18	18
	6925	南投—員林(經樟普寮)	10	10
	6925A	南投—員林[經樟普寮、永興宮]	8	8
	6926	南投—赤水(經錦梓)	6	6
	6926A	南投—赤水[經神木社區]	6	6
	6927	南投—田中(延駛高鐵彰化站)[經田仔]	4	4
	6928	南投—赤水(經松柏坑)	12	12
	6928A	南投—赤水(經松柏坑)[延駛田中]	6	6
	6929	南投—鄉親寮	18	18

資料來源：113 年 9 月 18 日交通部公路局

南投市各村里的公共運輸服務涵蓋情形存在明顯落差，如表 3，其中，共有四個村里涵蓋率低於 50%，分別為內新里（50%）、東山里（32%）、內興里（29%）與千秋里（0%）。其中千秋里為全市唯一無任何公車路線行經之里別，交通資源嚴重不足；而內興里與東山里之涵蓋率亦明顯偏低，僅達 29% 與 32%，顯示該地區民眾通勤與日常出行高度受限。

此外，福興里（54%）、營北里（55%）與平山里（58%）等里別之涵蓋率亦偏低，顯示其公車路網尚未能全面覆蓋當地居民生活範圍。相對而言，包括康壽里、三民里、仁和里、南投里與彰仁里等五個村里涵蓋率達 100%，交通資源分布較為集中。整體而言，低涵蓋率地區多位於邊陲地帶，顯示傳統公車路線在服務範圍規劃上仍存有顯著縫隙，應作為未來幸福巴士優先規劃與資源投入的目標區域。

表 3 南投市各村里公共運輸服務涵蓋率彙整表

村里名	總門牌(戶)	總人口(人)	公共運輸服務涵蓋率(%)
康壽里	805	1,595	100
三民里	457	992	100
仁和里	225	535	100
南投里	374	626	100
彰仁里	207	389	100
崇文里	563	1,079	100
光華里	784	779	98
三和里	1,385	3,109	96
光輝里	328	592	96
漳興里	1,147	2,939	95
龍泉里	426	1,023	94
福山里	693	2,086	94
鳳山里	265	1,115	91
嘉和里	2,094	5,235	87
軍功里	1,951	5,184	85
永興里	255	836	85
嘉興里	901	2,407	75
永豐里	1,576	3,200	75
營南里	1,183	2,720	72
光榮里	929	1,017	70
振興里	868	2,330	67
光明里	952	904	67
平和里	3,063	7,051	63

村里名	總門牌(戶)	總人口(人)	公共運輸服務涵蓋率(%)
鳳鳴里	154	621	63
新興里	2,566	5,866	61
漳和里	3,153	8,252	60
三興里	2,370	6,313	60
平山里	2,614	6,215	58
營北里	2,865	7,114	55
福興里	1,482	3,417	54
內新里	1,472	3,941	50
東山里	165	411	32
內興里	2,337	6,334	29
千秋里	375	1,137	0

註：底色為藍色之里別公共運輸服務涵蓋率低於 50%

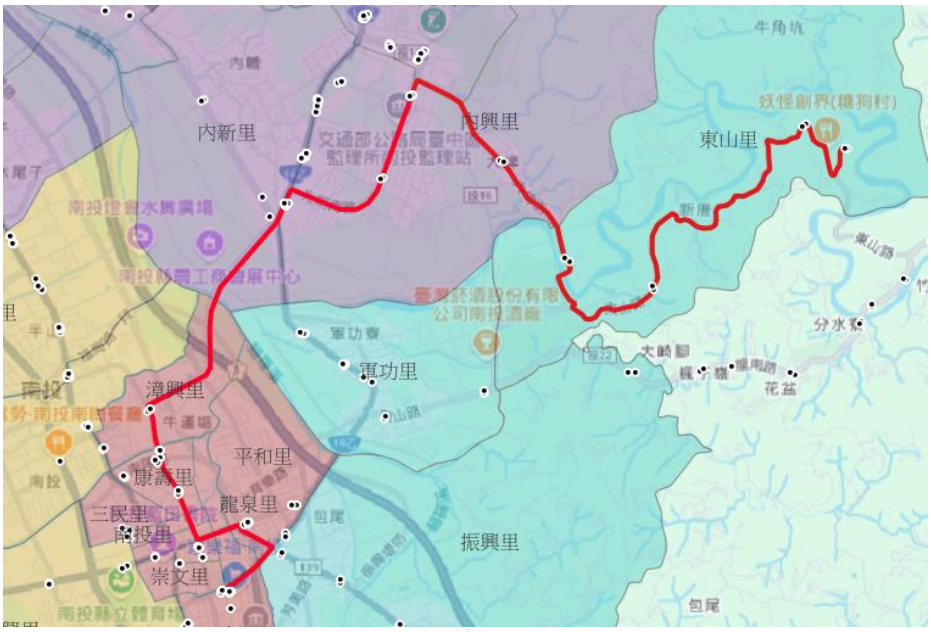
資料時間：2021 年 4 月

資料來源： 蘇昭銘等人（2019 年）

3.4 綜合探討

綜合分析結果顯示，南投市之主要道路呈南北向分布，公車路線多依附於台 3 線與台 14 乙線等聯外幹道，且大多會行經位於南投市中央之市區之村里如康壽里、三民里、仁和里等里，但東西向之公車路線卻相對不發達，造成市內雖有 41 條公車行經路線，卻仍有多數村里之公共運輸涵蓋率不足。市區與幹線沿線地區路網密集、服務重疊，與偏鄉地區形成明顯對比，造成涵蓋失衡的現象。

此外，南投市面臨高齡化與交通資源分布不均的雙重挑戰。高齡人口比例高達 20%，但市區外圍地區公共運輸涵蓋率明顯偏低，尤其以千秋里、東山里、內興里與內新里等邊緣村里最為嚴重。其中千秋里甚至無任何公車行經，可以推論當地居民在通勤、就醫、購物等日常活動上，已經習慣透過私有運具來解決日常問題。為改善此現況，南投市政府擬推動「幸福公車服務計畫」，由中區區域運輸發展研究中心協助，提出一條針對低涵蓋率地區之人工路線（如圖 5），後續將此路線簡稱為專家路線，服務範圍涵蓋千秋里、東山里與內興里，並盡量避開既有路線重疊區域，以補足現行公車服務的盲點。



資料來源：中區區域運輸中心

圖 5 專家規劃幸福巴士路線圖

本研究後續以此條專家路線作為比較基準，與深度強化學習（DQN）模型訓練所產出的路線結果進行比較。透過比較人工與 AI 規劃方法在涵蓋率、需求點覆蓋、路徑長度與既有路線重疊程度等多項指標上的表現，評估人工智慧於偏鄉交通規劃上的應用潛力與實務可行性，作為未來推動幸福巴士等新型運輸服務的重要參考依據。

第四章　強化學習技術之路線規劃方法

本章將說明本研究強化學習技術之路線規劃之整體設計與建構流程，包含模型環境設計、資料蒐集以及後續評估指標的設定與分析方式。首先於第 4.1 節說明 DQN 模型之規劃邏輯與架構，並於 4.2 節詳細介紹模型所依據的資料來源，包括門牌資料、信令資料與問卷調查等，作為環境狀態與節點獎勵機制設計之依據。最後於 4.3 節彙整本研究採用之路線評估指標，作為人工與 AI 規劃結果之比較基準，以系統性檢視兩者於實務應用上的可行性與效益差異。

4.1 幸福公車之路線規劃原則

在進行幸福公車路線規劃之強化學習模型設計前，需先瞭解本研究於路線設計上的基本規劃原則與目標，以作為後續模型架構、環境設定與獎勵設計之依據。本研究並非單純以最短路徑或最低成本為導向，而是希望結合「服務公平性」、「需求對應性」與「運行效率性」，透過深度強化學習技術建構具備可行性與智慧調適能力之偏鄉公車服務路線。其規劃原則說明如下：

一、服務公平性

本研究優先規劃南投市內涵蓋率明顯不足之區域，特別是內新里、東山里與、內興里等既有公車系統難以有效覆蓋之村里。在模型設計上把交通較缺乏地區之網格節點設定較高權重，以引導模型規劃能服務涵蓋率不足之區域路線，提升偏遠地區居民的基本交通可及性與資源分配公平性。

二、需求對應性

結合問卷調查、信令資料與家戶點位分布，本研究建構出各網格區塊之需求強度分數，模型將依據此分數優先涵蓋高移動需求、高人口密度或重要設施點位，如醫院、學校、政府機關、市場等民需之區域，確保幸福公車所提供之服務能服務在地居民的實際出行需求。

三、避免重疊與路線行駛效率

幸福公車定位為補足原有運輸不足處之交通工具，應避免與既有公車服務重疊或競爭。因此本研究於訓練中納入「與既有公車路線重疊」作為懲罰因子，避免模型產出覆蓋行為。同時亦設定每回合行動步數限制，並引入彎繞度與路線長度等效率指標，確保規劃結果兼顧覆蓋度與運行效益。

綜合上述原則，本研究嘗試將傳統人工路線規劃的經驗與邏輯，融合於人工智慧模型之中，透過合理設計之獎懲機制，引導強化學習模型於民眾需求與實際

數據之間達成最適化決策，進而產出兼顧公平性、需求性與效率性的智慧路線規劃成果。

4.2 規劃方法建構

為使深度強化學習模型能有效學習並產出合理的公車路線規劃結果，需先建立具備真實性的模擬環境。該環境須能具體呈現南投市區域內的道路結構、人口分布與公共運輸資源配置，並以可量化的方式，轉化為模型可理解之空間資訊結構與節點特徵。

本研究以「網格化」方式建構模擬環境，透過等距劃分空間區域，結合實際家戶、建築物、需求點與既有公車路線等多元資料。環境中每一個網格皆具有中心座標與屬性資訊，可作為學習過程中的決策節點。圖 6 為本研究所設計之環境設定流程圖，內容可分為三個主要階段：路網建構、資料統整、網格化轉換，說明如下：

一、路網建構：自開放街圖（OpenStreetMap）擷取道路資料，建立完整之路網結構，並以等距方式劃分為固定大小之網格。為提升環境精準度與模型運算效率，進一步篩選出具有實際通行路段之網格單元，排除空白或無效區域。

二、資料統整與屬性統計：分別蒐集與處理家戶點位、既有公車、重要建築、需求點位等以下四類資料，並依據網格分區進行統計：

（一）家戶點位資料：讀取望格中的家戶門牌位置，計算每格網格中之家戶數量，作為服務涵蓋率與潛在服務需求指標。

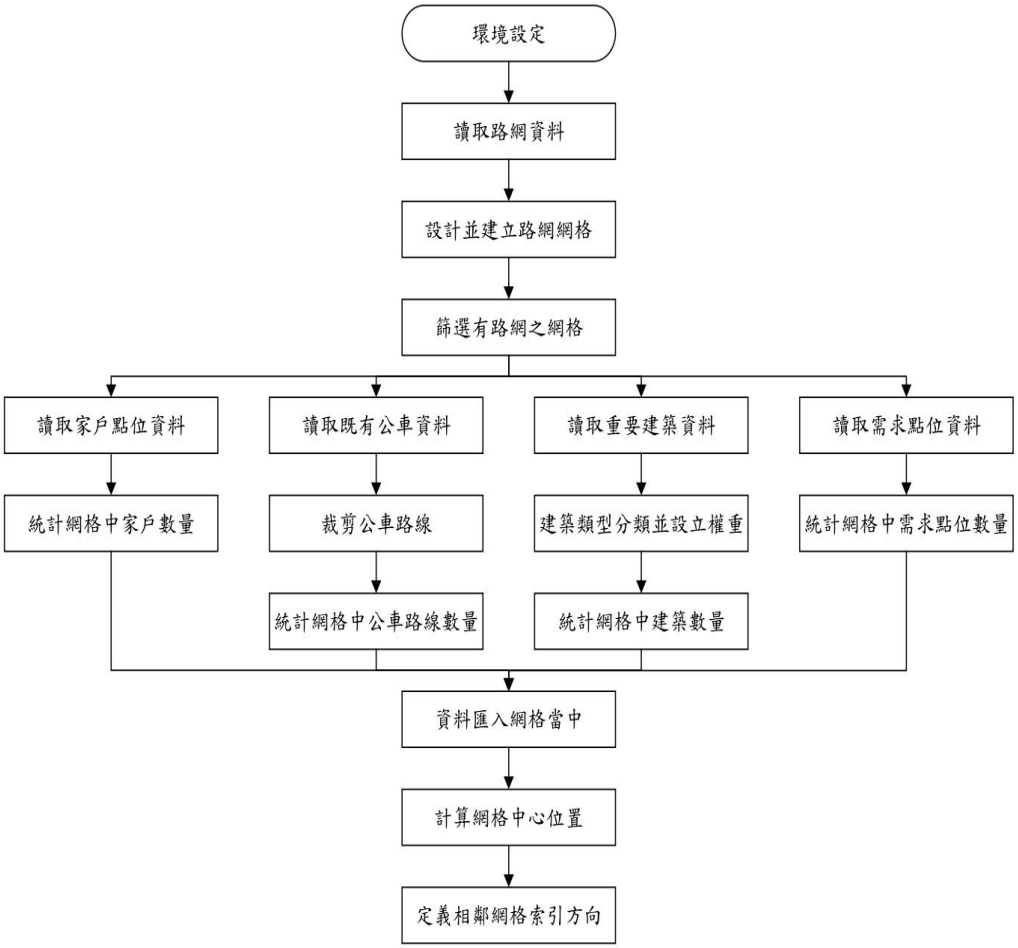
（二）既有公車資料：擷取公車路線與站點資料，裁剪出通過範圍之路線，統計每格網格中所經之公車路線數，作為既有服務之評估依據。

（三）重要建築資料：將醫療院所、學校、政府機關等重要民需據點分類後納入權重計算，並統計其在各網格之分布數量，作為後續獎勵機制的評分依據。

（四）需求點位資料：以問卷與信令資料所分析之高流動性據點為基礎，統計其在各網格內之出現頻率，補充數據中未涵蓋之主觀與實際需求資訊。

三、網格化整合：將所有屬性資料匯入對應之網格單元，並計算每格幾何中心座標，作為模型可選擇之潛在節點。同時，建立相鄰網格之索引方向與移動成本，形成強化學習中可操作的空間行動結構。

透過此流程，本研究將地理與社會資料有效轉化為模型訓練所需之網格化環境，使強化學習模型能根據節點評分與移動規則，進行路線探索與學習，進而產出兼顧效率與服務公平性的公車規劃結果。



資料來源：本研究整理

圖 6 環境建構與資料處理流程圖

為使 DQN 模型能有效學習並生成符合實務需求之路線規劃結果，本研究進一步設計完整的策略網路與獎勵函數運作流程（如圖 7）。流程可分為神經網路初始化、參數設置、學習與行動機制、獎勵與更新策略等等，說明如下：

一、神經網路與參數初始化

（一）設定神經網路結構：設計用於 Q 值預測的深度神經網路，包含輸入層（例如網格狀態向量）、隱藏層、輸出層（動作 Q 值）。

（二）定義主網絡與決策網絡：主網路負責實際決策與 Q 值計算，目標網路則用來穩定訓練，避免 Q 值估算震盪。

（三）定義決策網路同步率：每隔固定步數將主網路參數同步至目標網路，

以穩定學習過程。

二、學習與探索參數設定

（一）定義經驗獎勵池：建立經驗回放儲存模型經歷過的狀態、動作、獎勵、下一狀態資料，用於後續訓練。

（二）定義學習率：設定模型參數更新的速率，影響收斂速度與穩定性。

（三）定義探索率與衰退率：透過 ε 探索策略讓模型在初期有較高機率隨機探索不同動作，後期逐步收斂至穩定策略。

三、模型訓練與學習流程

（一）建立計步器：用以追蹤每一回合的步數與訓練次數，作為停止條件與同步判斷依據。

（二）定義模型行動：根據目前狀態與 ε 值從 Q 值最大或隨機中選擇動作，模擬實際決策過程。

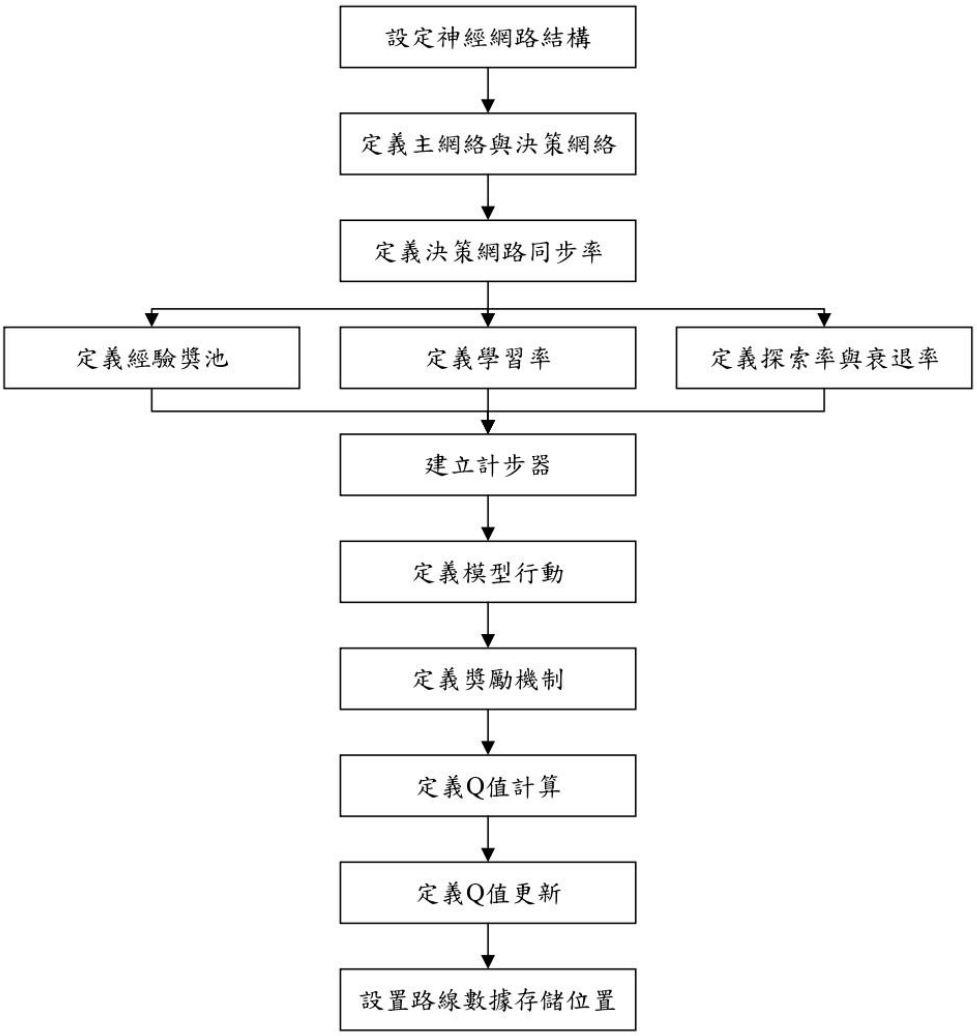
（三）定義獎勵機制：根據不同網格節點的屬性（如人口密度、重要設施、需求熱點等）設計回饋分數，鼓勵模型優先覆蓋服務弱區。

四、學習結果更新

（一）定義 Q 值計算：計算當前狀態與動作所對應的目標 Q 值。

（二）定義 Q 值更新：比較預測 Q 值與目標 Q 值之差異，反向傳播更新主網路參數。

（三）設置路線數據儲存位置：將每次訓練產生的路線結果與其對應評分指標紀錄於資料庫，供後續分析與繪圖。



資料來源：本研究整理

圖 7 策略網路與獎勵函數設計

為提升強化學習模型在幸福公車路線規劃上的實務應用能力，本研究設計一套完整且具可複製的訓練與評估流程（如圖 8）。整體流程整合專家提供之路線規畫、網格化路網資訊、獎勵回饋邏輯與強化學習技術，主要可區分為五個部分：專家資料處理與向量映射、訓練參數設定與初始判斷、訓練過程初始化與行動邏輯、訓練結束條件與路線評估，以及結果輸出與性能比較，說明如下：

一、專家資料處理與向量映射

（一）讀取專家路線資料：將區域中心提供之專家路線使用 GIS 工具繪製並匯出為 SHP 格式，供模型讀取與比對使用。

（二）映射至網格獲取向量資料：將專家路線轉換至與訓練用網格一致的向量格式，使其可與 DQN 模型輸出路線進行比較分析。

二、訓練參數設定與初始判斷

（一）設定最大訓練次數與步數：每一次訓練包含一條完整路線的模擬，限制最大步數以控制訓練時間與效率。

（二）訓練次數是否到達最大值：判斷是否達到設定的最大訓練次數，若達到則跳出訓練流程，進行結果評估；否則繼續訓練。

三、訓練過程初始化與行動邏輯

（一）初始化 Agent 值：將模型位置歸位至起點，並將該輪訓練的累積獎勵初始化為 0。

（二）由起點開始行動：從起點出發，根據當前策略與探索機率 ε 選擇行動方向。

（三）選擇前進方向：根據主網路輸出之 Q 值決策行動，並紀錄目前所處網格。

（四）紀錄重複抵達之網格數量：若再次進入已訪網格，進行統計並設定懲罰以降低該路線獎勵。

四、訓練結束條件與路線評估

（一）是否到達終點或最大步數：若成功抵達終點或超過最大步數即終止該回合訓練，並計算該回合的總獎勵值。

（二）計算本次訓練獎勵 Q 值：根據走訪路徑與回饋規則計算總體 Q 值。

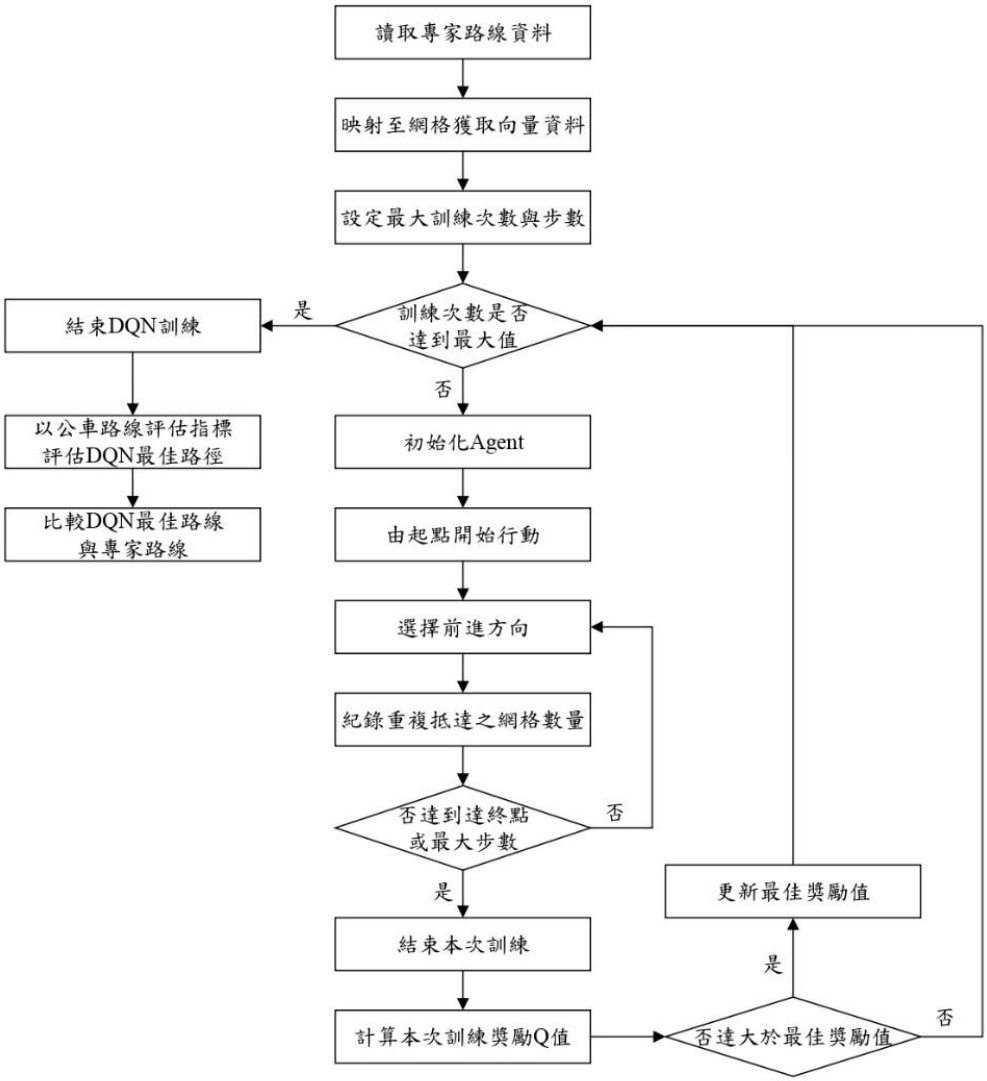
（三）是否大於最佳獎勵：若目前獎勵高於歷次最佳，則更新最佳路線與其獎勵數值，作為最終輸出的路線。

五、結果輸出與性能比較

（一）結束 DQN 訓練：當訓練次數達上限後，輸出最佳 Q 值對應路線並進行儲存。

（二）以公車路線評估指標評估 DQN 最佳路徑：使用如覆蓋率、服務盲點降低率、節點彎曲度等指標評估模型表現。

（三）比較 DQN 最佳路線與專家路線：將 DQN 輸出與專家路線進行重疊比對，探討相似性與差異性，作為模型優化依據。



資料來源：本研究整理

圖 8 強化學習訓練迴圈與性能評估

4.3 基本資料收集

為了有效建構適用於南投市路網環境之 DQN 強化學習模型，本研究蒐集並整合多項資料來源，作為環境狀態設計、網格特徵建構與獎懲機制制定之依據。資料來源包含門牌與站點位置資料、手機信令資料以及民眾問卷調查，三者相互補充，有助於提高模型對實際需求之掌握與規劃結果之準確性。

一、門牌資料與公車站點點位資料：本研究使用逢甲大學中區區域運輸發展研究中心所提供之家戶門牌位置資料，結合南投市政府開放資料平台公布的既有公車站點資料，建立完整之居住密度分布與現行站點服務範圍圖層。透過比對家戶點位與站點服務半徑，可有效辨識未被既有公車系統涵蓋之高密度區域，作為網格分數設定與服務盲點判斷之依據，協助模型判斷哪些地點值得優先拜訪或納入路線。

二、信令資料：為掌握居民實際的跨里移動行為與旅次分布，本研究採用某一週期範圍內之手機信令資料，分析南投市內各行政區之間的人口流動量與移動方向。該資料能補足票證資料無法覆蓋的非公車旅次需求，並作為 DQN 環境中節點需求強度（如獎勵權重）的依據，協助模型強化高移動需求區域之服務密度。

三、問卷調查：為補足難以掌握的主觀需求與居民偏好，本研究亦參考南投市交通問卷調查結果，內容涵蓋民眾對公車路線的滿意度、常用站點、希望新增服務區域等資訊。此資料可作為模型設計中「民眾期望經過之地點及時間段」等人本考量之參考。

4.4 路線評估方式

為系統性比較人工規劃與深度強化學習（DQN）所產生之路線規劃成效，本研究參考文獻回顧所整理之評估指標，加以修改為適用於本研究中之標準。整體評估指標分為站點評分機制和路線服務指標，並進一步分為服務涵蓋性、路線效率性與路網重疊性三大面向，詳細說明如下：

一、站點評分機制

本研究欲針對涵蓋率較低之村里進行幸福公車規劃，故將公共運輸服務涵蓋率納入節點評分標準。涵蓋率越低之地區，其服務需求之迫切性越高，評分亦越高。評分等級如表 4 所示：

表 4 公共運輸服務涵蓋率評分基準

公共運輸服務涵蓋率	分數
0–20%	4 分
21–40%	3 分
41–60%	2 分
61–80%	1 分
81–100%	0 分

資料來源：本研究整理

此外，再依各站點對應之家戶數與新增住戶數加權評分，以建立節點總分，作為 AI 模型選點與優先服務排序之依據。

二、路線服務指標

（一）服務家戶人口數(人)：此指標用於查看大眾運輸站牌在特定的步行距離範圍內所涵蓋的家戶人口數，式 1 中 A 的部分為審議路線，J 是路線所行經行政分區，j 為各區，HC_{Aj}是家戶涵蓋數，POP_j是該分區人口數，兩者之乘積總和為SP_A。

$$SP_A = \Sigma_{j=1}^{J} HC_{Aj} \times POP_j = \Sigma_{j=1}^{J} \frac{\sum_{h}^{H} H_j H_{jh} \times f_{Ajh}}{H_j} \times POP_j \quad (1)$$

（二）服務家戶人口重疊率(%)：此算式以服務家戶人口數作為依據並進行延伸，透過審議路線與既有的路線去比較查看是否有高度的服務一樣的居民，式 2 中，將審議路線 A 之服務人口數與行經行政區之其他既有路線服務人口數之交集，除以既有路線服務人口數之比例值，f_{nowjh}為各 j 分區家戶涵蓋率，為該行政區內 H 個家戶被現有客運路線涵蓋之函數，f_{nowjh}值為 1 表示有涵蓋，否則為 0，所以當F_{Ajh}與f_{nowjh}為 1 時即表示第 h 個家戶同時被 A 路線與現有路線涵蓋。

$$SPO_A = \frac{\Sigma_{j=1}^{J} \frac{\Sigma_{h=1}^{H} H_j H_{jh} \times F_{Ajh} \times f_{nowjh}}{H_j} \times POP_j}{\Sigma_{j=1}^{J} \frac{\Sigma_{h=1}^{H} H_j H_{jh} \times f_{nowjh}}{H_j} \times POP_j} \times 100\% \quad (2)$$

（三）路線彎繞度：此指標用於衡量審議路線直捷度指標，式 3 將審議路線(A)的起訖最短路徑(l_{sp})去除以審議綠線的長度(l_A)，假設算出來的值為 1 則為路線以經走最短的路徑。

$$R_A = \frac{l_A}{l_{sp}} \tag{3}$$

（四）整體路網重複比例(%)：此指標用於計算審議路線(A)與既有的公車路線之重複路段的長度，最後轉變為百分比呈現，式 4 先把審議路線分成 n 個路段，那每個路段為的長度為L_{Ai}，那fo_i是第 i 個路段之路網重複函數，如有重複，fo_i值為 1 無重複為 0，累積加起來得到一個值後得到式 5，再除以營運路線的長度，值越大表示重複的路段越長。

$$LO_{AR} = \Sigma_{i=1}^{n} L_{Ai} \times fo_i \tag{4}$$

$$RO_{AR} = \frac{LO_{AR}}{l_A} \times 100\% \tag{5}$$

（五）個別路網重複比例(%)：此指標主要用於彌補整體路網重複的部分，整體路網重複比例可能因審議路線過短，短導致重複度過高，因此設立此指標用於彌補，式 6 計算方法為將審議路線分為 n 段，L_{A_i}為每條路線長度，fo_{r_i}為第 i 個路段與客運路線 r 重複的函數，那路段中有客運經過值則為 1，否則為 0。式 7 可以算出RO_{Ar}這函數是整體路網重複路段長度與 r 客運路線長度 (l_r) 之比例值。

$$LO_{Ar} = \Sigma_{i=1}^{n} = L_{Ai} \times fo_{r_i} \tag{6}$$

$$RO_{Ar} = \frac{LO_A}{l_r} \times 100\% \tag{7}$$

4.5 綜合探討

本章旨在建立一套應用於幸福公車之深度強化學習路線規劃模型，透過環境模擬、資料彙整、網格建構與模型設計，逐步建立出一個具備實務涵義的強化學習訓練架構。在模擬環境建構方面，本研究採用網格化手法，將區域內的道路網路、家戶分布、公共服務據點及需求點位轉換為接近現實的學習狀態空間，此方法能夠較容易地統整不同維度的資料，藉此提升模型對於區域特性的辨識能力。同時，亦透過 OpenStreetMap、問卷資料與信令數據等多元資料進行整合，補足單一資料不足造成的盲點。

策略網路設計上，本研究利用雙網路架構（主網路與目標網路）以提升模型的穩定性，並透過貪婪策略達成探索；獎勵機制則依據涵蓋率、重要設施經過次數、與既有公車路線重疊程度等多項評估準則進行設計，使模型能兼顧服務公平性、路線效率與資源最適化三者之平衡。此外，訓練流程亦考量實務應用面需求，設定最大步數與終點條件，以確保所產生之路線具備完整性與可行性，避免無效繞行與重複拜訪等行為。此訓練架構將作為後續第五章實驗驗證與效能評估的基礎，進一步探討模型在實際應用場景中的表現與改進空間。

第五章　實例應用

5.1 模型設計與訓練概況

為驗證深度強化學習（DQN）於幸福公車路線規劃之適用性與效能，本研究依據第四章所建構之環境資料與評分架構，設計完整之 DQN 訓練模型，並建立模擬訓練流程。整合空間資訊、交通服務缺口與民眾實際需求，使模型自動學習符合幸福公車最佳服務路線。本節將詳述模型訓練所需之環境建構、資料結構、神經網路設計與訓練策略等核心技術內容。具體說明分為以下三部分：

一、訓練環境建構與網格資料設計

為使模型可以在真實交通特性之空間環境中進行學習行動決策，本研究根據 OpenStreetMap 之南投市路網，於環境中建立南投市路徑圖，並以等距 63.8 公尺劃分網格。僅保留具實際道路涵蓋之格點，以利路線模擬之真實性與效率。

每一網格節點均匯入以下七類屬性資料作為狀態向量輸入：

（一）家戶數
（二）公車路線數量
（三）重要點位加權分數
（四）問卷需求點有無
（五）到中途點與終點的距離
（六）到終點的距離
（七）是否已經拜訪過中途點

此外，解決南投市之偏鄉交通問題，本研究特別根據第 4.2 節資料，針對涵蓋率低於 50%之村里（如千秋里、東山里、內興里）設定較高初始評分與正向獎勵值，導引模型主動探索交通服務缺口區域。問卷調查與信令資料所辨識出之高頻移動區亦給予較高權重，同時在模型行動過程中設計方向性獎勵（朝終點方向移動給予加分），以提升路線任務達成效率。

二、策略網路結構與學習機制

本研究所建構之策略網路為三層全連接神經網路，輸入為上 7 維狀態向量，隱藏層分別設定 64 與 32 個神經元，激活函數採用 ReLU，輸出層對應 8 個可行動作（上下左右與四個對角方向）之 Q 值。為穩定學習過程與避免 Q 值估算震盪，模型採用主網路與目標網路之雙網路架構，並設定每間隔固定回合進行參數同步，確保學習穩定性。

在探索策略上，模型採用貪婪策略控制行動選擇的隨機性與學習性，初期設定 ε=1.0，以利模型廣泛探索環境；隨訓練進行逐步下降至最小值 0.05，使策略漸趨收斂。同時建構經驗回放獎池以儲存模型歷史互動經驗，並透過隨機抽樣進行批次學習，降低樣本間相關性，提升學習效率與收斂穩定性。

本研究完整設定 DQN 模型相關訓練與學習參數如表 5 所示：

表 5 南投市 DQN 路線規劃模型之參數設定

參數名稱	設定值
隱藏層數	2
神經元個數	64、32
激活函數	ReLU
折扣因子 (γ)	0.997
探索率初始值 (ε)	1.0
探索率最小值	0.05
探索率衰減步數	500 次
學習率 (learning rate)	0.001
批量大小 (batch size)	64
經驗獎池區容量	50,000
訓練回合總數	3,000
每回合最大步數	250
目標網路同步頻率	每 5000 步

資料來源：本研究整理

三、獎勵函數設計與參數設定

為規劃出兼具服務性與效率性的幸福巴士路線，本研究設計獎勵函數，綜合考量民眾交通需求、地理分布、重要點位等目標。該獎勵機制透過正向回饋引導模型主動涵蓋交通資源缺口區域，並抑制不必要的重複行經與無效探索，確保整條路線具備可行性。

其中，「重要點位加權分數」針對民眾需求設定差異性比重，使模型能更精準地反映居民日常活動之實際需求。加權設定如下：「醫院」加權 5 分為最高，「學校」為 4 分，「市場」為 3 分，「公家機關」為 2 分，「超市（如全聯、家樂福）」則為 1 分。每一網格若重疊多種設施，其建築分數會累計加總，並乘以設定之權重值納入整體評分。完整獎勵項目與分數設定如表 6 所示：

表 6 獎勵函數各分數設定

項目	分數	目標說明
家戶數	×0.45	提升服務高人口區域
重要點位加權分數（醫院、學校等）	×0.275	鼓勵行經生活機能較強區域
問卷需求點	×0.275	強化回應在地實際需求
中途點、終點命中	+500	必經之區域，加權鼓勵完成任務
每步前進處罰	−1.0	避免模型亂繞或無效探索
重複拜訪網格	−50	提高路徑效率，符合路線需求
重疊既有公車網格數	−5 分	降低與現有路線重疊的機率

資料來源：本研究整理

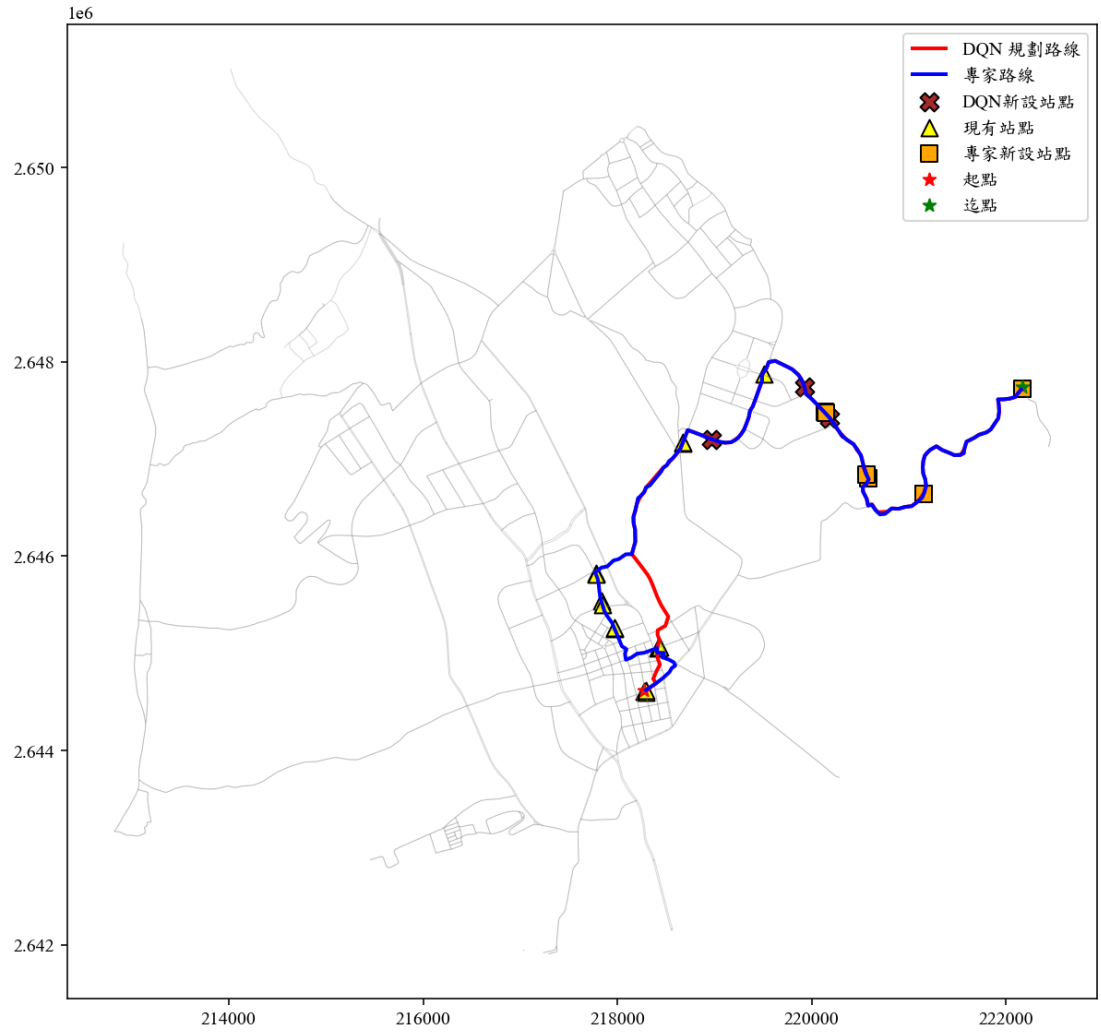
5.2 DQN 規劃結果展示

完成 5.1 節模型訓練後，本節將展示 DQN 於南投市幸福公車路線規劃中之應用成果。並針對路線長度、家戶涵蓋數、既有公車重疊數等等，和專家規劃路線進行比較，以探討 AI 模型在提升服務公平性、效率的差異，進一步驗證深度強化學習於偏鄉交通規劃領域之應用可行性。

圖 9 為本研究所建立之 DQN 模型路線規劃成果示意圖，路線起訖點固定於東山里與南投市家樂福，依據人工規劃結果進行設定。圖中紅色線條為 DQN 模型所生成之最佳路線，藍色線條則為專家以傳統人工方式規劃之路線。地圖中亦標示現有站點、專家新設站點、DQN 新設站點，以及起訖位置。

從圖中觀察可知，DQN 模型生成的路線傾向優先涵蓋公車涵蓋率較低的村里地區，如東山里與千秋里等地區，並有效串連問卷調查與移動數據中反映出高出行需求的區域。相較於專家規劃偏向主幹道路與既有站點之安排，DQN 模型展現更高的彈性能主動辨識服務空缺，並於高需求之區域新增站點（如圖中紅色叉號所示），顯示其具備由資料中學習與開發潛在服務節點的能力。

此外，在路線中段的設計上，DQN 模型能有效避開重複繞行區段，提升整體行駛效率與涵蓋範圍。整體而言，DQN 所產生的規劃結果展現出路徑創新性，並強化邊緣區域與高需求點之間的公共運輸連結，提升偏鄉地區的服務公平性與交通可及性。

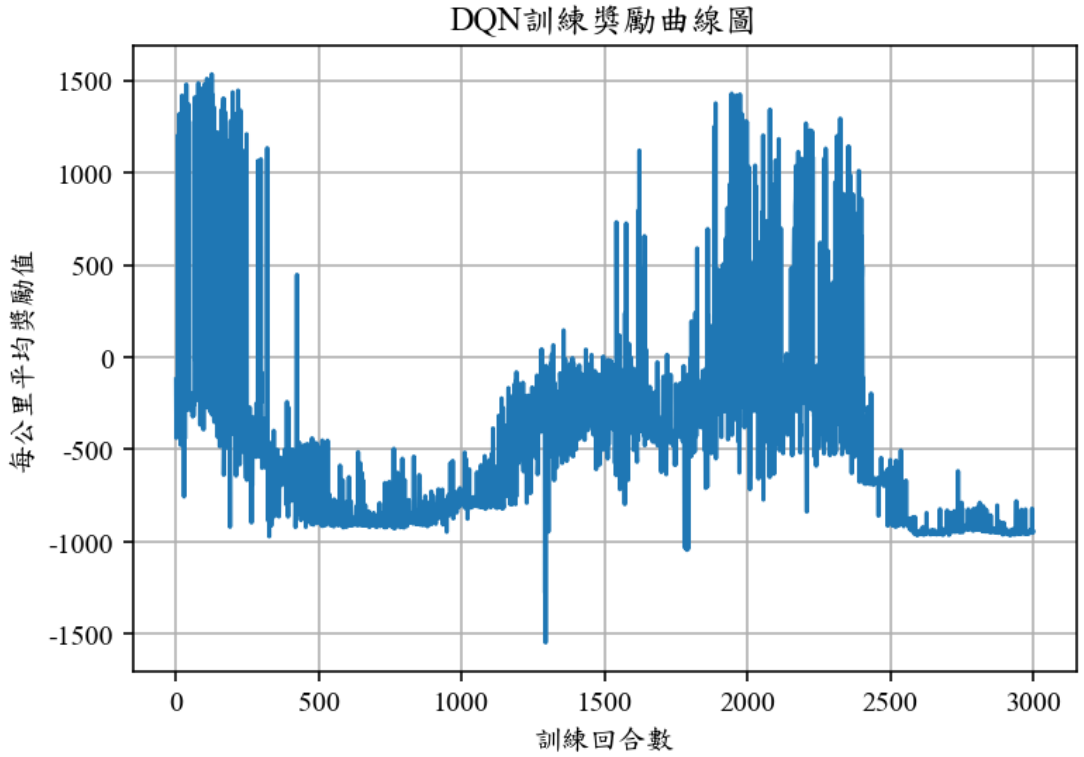


資料來源：本研究整理

圖 9 專家路線與 DQN 路線圖

為觀察 DQN 模型在訓練過程中的學習趨勢與收斂情形，本研究繪製圖 10 所示之訓練過程平均獎勵變化曲線圖。橫軸為訓練回合數，縱軸為每公里平均獲得之獎勵值。初期模型仍處於隨機探索階段，獎勵波動劇烈且多為負值，代表路線多為低效率或重複路徑，尚未學得有效策略。

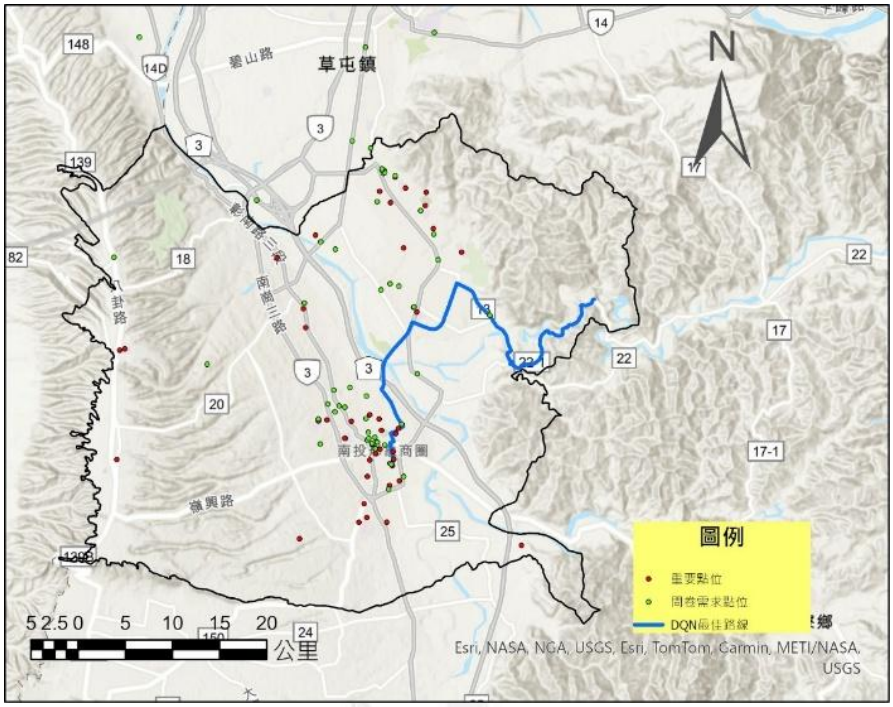
隨訓練次數增加，模型逐步累積經驗並強化有效行動行為，平均獎勵值於約於第 1500 回合後明顯提升，並在約第 2000 至 2300 回合期間達到相對高峰，顯示模型已能產生較具效率與涵蓋性的路線規劃。此後獎勵雖略有下降，但整體仍維持在合理範圍，反映模型已趨近穩定收斂。



資料來源：本研究整理

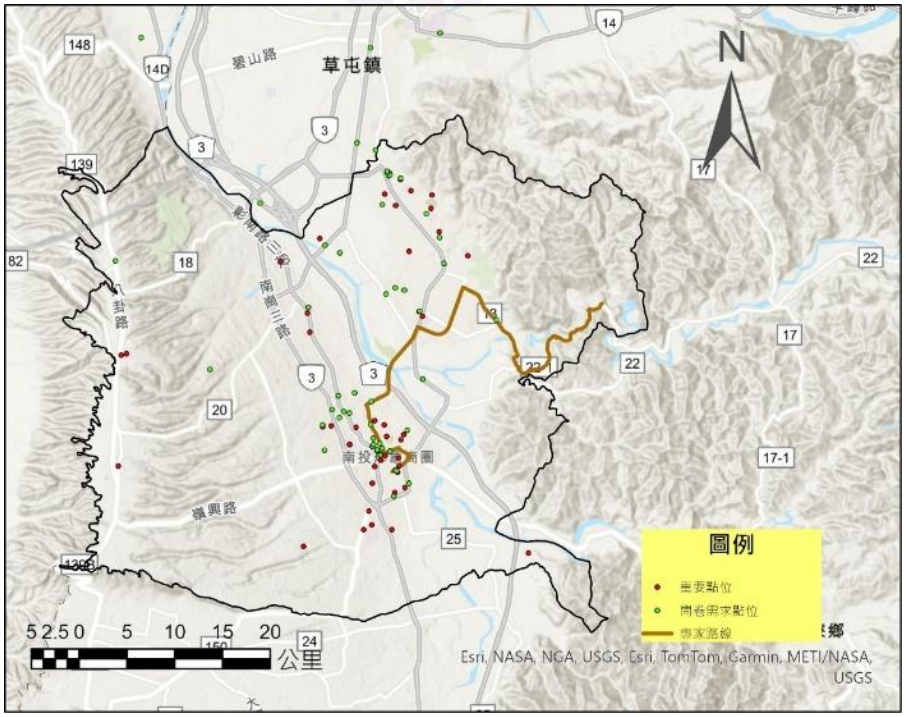
圖 10 每次訓練之平均獎勵值

圖 11 為 DQN 訓練之最佳路線，圖 12 為與之對比的專家路線，圖中紅色點位為南投市重要建築點位；綠色點位為區域中心問卷之需求點位。由圖可見 DQN 之最佳路線於南投家樂福出發，行經至內興里後之路線與專家路線一致，兩路線僅於市區及近市區有著些許的差異。專家路線著重於滿足問卷需求點位，而 DQN 最佳路線則是選擇在滿足較多問卷需求點位與重要建築點位的前提下，尋求更短的公車路線。



資料來源：本研究整理

圖 11 DQN 最佳路線



資料來源：本研究整理

圖 12 專家規劃路線

表 7 為比較使用 DQN 模型規劃出來的最佳公車路線與專家設計之路線兩者的詳細數據。首先為路線長度，DQN 的路線大約是 10.13 公里，而專家路線為 11.23 公里，由於 DQN 模型會在各需求基礎上追求以較短路徑規劃出最佳化路線，所以其路線長度略短於專家路線；涵蓋的家戶數值之比較，DQN 路線覆蓋了 673 戶，而專家路線涵蓋了 981 戶，相較於專家路線，DQN 更強調的是涵蓋重要據點和需求點，其中 DQN 路線涵蓋了重要點位和問卷需求點各 4 處，皆優於專家路線，表示 DQN 在服務公平性和重點區域的覆蓋上表現比較好，能有效掌握居民的核心需求。與現有公車路線的重疊程度，DQN 路線與 17 個網格之現有公車路線重疊，專家路線有 23 個網格與之重疊，這說明 DQN 能設計出更有補充性的路線，幫助填補現有公車網的不足；最後在平均獎勵分數方面，DQN 模型取得 1533.65 分，優於專家的 1371.69 分，代表其在綜合考量效率、服務涵蓋與路徑等多項指標後的表現更加出色。

表 7 路線涵蓋獎勵函數比較表

公車路線	路線長度	涵蓋家戶數	重要點位	問卷需求點位	重疊既有公車網格數	平均獎勵
DQN 最佳路線	10.13 公里	673	4	4	17 格	1533.65
專家路線	11.23 公里	981	1	6	43 格	1371.69

資料來源：本研究整理

5.3 規劃成效比較分析

在新增了 DQN 最佳路線以及專家路線以後，對於東山里與內興里都有較為明顯的提升，詳細資訊如表 8、表 9 所示：

一、 內興里：新增路段後，服務家戶數大幅提升至 2,128 戶，整體涵蓋率由 36.82%提升至 91.33%，增加了 54.51%；專家路線新增路段後，服務家戶數大幅提升至 2,155 戶，整體涵蓋率由 36.82%提升至 92.49%，增加了 55.67%。

二、 東山里：新增路段後，服務家戶提升至 120 戶，涵蓋率由 44.51%增至 73.17%，增加了 28.66%。

DQN 最佳路線對於內興里家戶服務有著重要的作用，基本上涵蓋了內興里大部分原本並無大眾運輸服務之區域。對於東山里方面，由於本身東山里家戶數較少，對於涵蓋率增加後實際地新增家戶有限。而相交於專家路線，內興里的家戶涵蓋數較少，但實際情況兩條路線行經內興里與東山里的路線是相同的。經過

推斷後，判斷可能得原因為專家路線在投影地圖當中是可能出現了些許的偏差，也間接地影響了後續家戶涵蓋的計算。

表 8 DQN 路線在東山里和內興里的涵蓋率變化

項目 / 里名	家戶數	服務家戶門牌數			服務涵蓋率(%)		
		現況	新增路段後	增加值	現況	新增路段後	增加值
東山里	165	73	120	47	44.51	73.17	28.66
內興里	2,337	8,58	2,128	1,270	36.82	91.33	54.51

資料來源：本研究整理

表 9 專家路線在東山里和內興里的涵蓋率變化

項目 / 里名	家戶數	服務加戶門牌數			服務涵蓋率(%)		
		現況	新增路段後	增加值	現況	新增路段後	增加值
東山里	165	73	120	47	44.51	73.17	28.66
內興里	2,337	8,58	2,155	1,297	36.82	92.49	55.67

資料來源：本研究整理

DQN 模型所訓練之路線結果如表 10 所顯示，服務家戶人口數為 6,732 人，服務家戶人口重疊率為 16.50%，彎繞度為 1.09，近似於理想值 1，顯示路線十分精簡。整體路網重複比例為 5.11%，而在個別路線的重複情形中，選取重複比例最高的兩條路線為 1657 路和 6916 路，其重複比例分別為 4.89%與 4.39%。

表 10 DQN 路線在東山里和內興里之分析結果

類型 / 指標名稱	現況 指標值	路線新增 指標值	指標 值變化	備註
服務家戶人口數 (人)		6,732		
服務家戶人口重疊率 (%)			16.50	
路線彎繞度		1.09		
整體路網重複比例 (%)	0	5.11		
個別路線重複比例 (%)	0	4.89		1657 路線
	0	4.39		6916 路線

資料來源：本研究整理

如表 11 中所示，專家路線的服務家戶人口數為 8,292 人，服務家戶人口重疊率為 21.18%，彎繞度為 1.21，相較於理想值 1 較為接近。整體路網重複比例為 13.13%，而在個別路線的重複情形中，選取重複比例最高的兩條路線為 6916A 路和 6871 路，其重複比例分別為 5.95%與 5.15%。

表 11 專家路線在東山里和內興里之分析結果

類型 / 指標名稱	現況指標值	路線新增指標值	指標值變化	備註
服務家戶人口數 (人)		8,292		
服務家戶人口重疊率 (%)			21.18	
路線彎繞度		1.21		
整體路網重複比例 (%)	0	13.13		
個別路線重複比例 (%)	0	5.95		6916A 路線
	0	5.15		6871 路線

資料來源：本研究整理

5.4 綜合比較與成效評估

一、指標量化分析

為進一步評估 DQN 模型規劃路線與專家規劃路線之表現差異，表 12 彙整了量化比較結果。

（一）服務家戶人口數：專家路線覆蓋 8,292 人，DQN 路線則為 6,732 人；專家路線的規劃考量更加著重與擴大服務家戶人口數之涵蓋範圍，在此項指標上佔有優勢。

（二）服務家戶人口重疊率：專家路線與既有路線之重疊率高達 21.18%，而 DQN 路線僅有 16.50%，在 DQN 模型下大幅降低了與現有路網的重疊情況，減少了與既有路線發生搶客問題的機率。

（三）路線彎繞度：專家路線彎繞度 1.21，DQN 路線僅 1.09，更接近理想值 1，在彎繞度方面的優勢顯示 DQN 路線較專家路線行駛效率更佳。

（四）整體路網重複比例：專家路線高達 13.13%，DQN 路線僅 5.11%，說明 DQN 路線能有效尋找補充性路段，減少重複覆蓋。

（五）個別路線重複比例：專家最高達 5.95%，DQN 路線僅 4.89%；同樣反映 DQN 路線在著重與各路線重疊程度的比較上也能比專家路線獲得更好的成效。

從表中可見，DQN 路線在「服務家戶人口數」與「服務家戶人口重疊率」方面皆低於專家路線，顯示在整體涵蓋面上略有不足。但在「路線彎繞度」、「整體路網重複比例」及「個別路線重複比例」三項指標中，DQN 均優於專家路線，為確保路線評估的準確與專業性，在個指標之間進行權重的分配，以此作為路線比較的評分依據。

表 12 專家路線與 DQN 路線之公車路線評估指標比較

類型 / 指標名稱	專家路線	DQN 路線	差異比較
服務家戶人口數 (人)	8,292	6,732	-1,560
服務家戶人口重疊率 (%)	21.18	16.50	-4.68
路線彎繞度	1.21	1.09	-0.12
整體路網重複比例 (%)	13.13	5.11	-8.02
個別路線重複比例 (%)	5.95	4.89	-1.06
	5.15	4.39	-0.76

資料來源：本研究整理

二、整體成效評估

綜合比較可知，雖然 DQN 路線在路線涵蓋服務家戶人口數方面上稍遜於專家路線，但其在避免與既有公車服務衝突、減少路線彎繞等方面佔有優勢，更貼近偏鄉交通資源最佳分配的需求。藉此也能證明 DQN 模型在路線規劃方面，針對降低重疊率與降低彎繞度提升行駛效率等關鍵評估指標上有較為突出的表現，相較於專家規劃，DQN 模型具備以下三項潛力：

（一）重複訓練：當需求資料（問卷、需求點、家戶數）更新時，可重新訓練模型，即時產出新路線。

（二）多目標優化能力：模型可同時考慮涵蓋率、路徑效率與公平性等多重目標，有助於提升公車資源配置的彈性與路線設計的效率。

（三）模型可複製性：DQN 模型能夠透過替換區域資料應用於其他地區，針對當地需求進行權重評分調整。

三、權重敏感度分析

在 DQN 模型的訓練過程中，獎勵函數中各項指標的權重設定對最終路線規劃結果具有關鍵影響。因此，針對兩種情境進行敏感度分析，了解各獎勵函數對於 DQN 模型規劃公車路線產生的影響。以本研究之權重為基礎假設以下情境：

情境一：將「重要點位權重分數」權重設為 0

在此設定下，情境模擬於訓練過程中將完全排除「重要點位」對網格評分的貢獻，即使網格中包含醫院、學校、市場等設施，其加總分數亦不再納入模型獎勵計算。此變動將原本設定的「重要點位總分 ×0.275」這項權重係數設為 0，並將 0.275 新增至家戶數的權重係數（如表 13），進而觀察在缺乏日常活動需求導向下，模型的學習與路線決策會產生何種移動。

表 13 情境一權重分數調整

	原始權重	調整過後
重要點位	×0.275	0
家戶數	×0.45	0.725

資料來源：本研究整理

將重要點位進行敏感度分析的路線結果，可參考圖 13，圖中可觀察到，情境一路線在初始路段與專家規劃路線存在明顯重疊，主要集中於市區範圍，顯示模型受市中心家戶人口密度較高的影響，傾向優先涵蓋此區以獲得較高的家戶數評分。然而，在進入中後段路線後，模型開始偏離專家路線，轉向接近中途點的區段。此一轉變可能與獎勵函數中所設置之「步數懲罰」有關；模型為了避免過長的行進距離而導致獎勵減少，故嘗試在路線長度控制下，最大化涵蓋評分，顯示其具備「評分導向」與「距離最小化」雙重目標傾向。

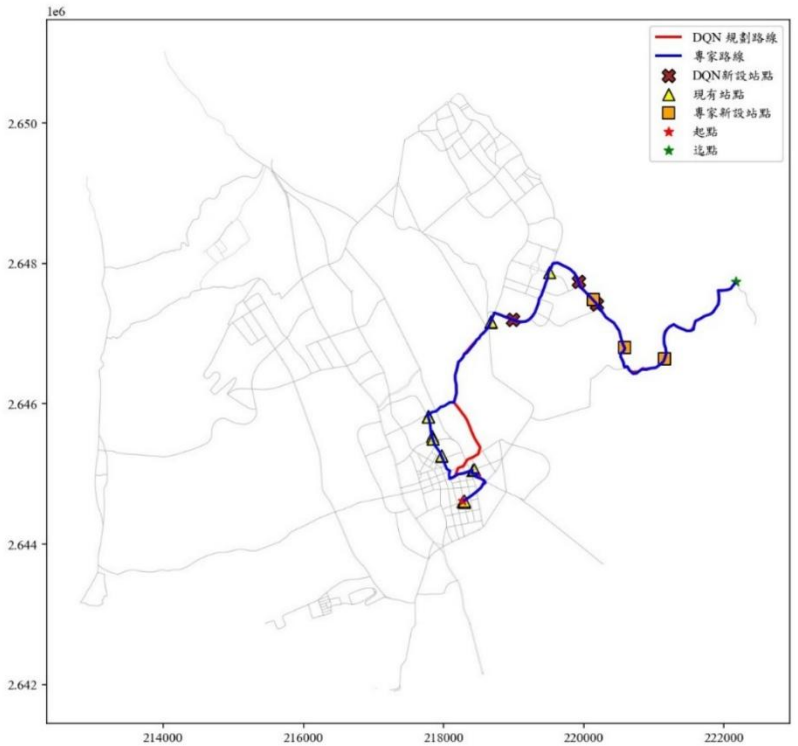


圖 13 專家路線與情境一路線圖

資料來源：本研究整理

比較三條路線（情境一路線、DQN 最佳路線與專家路線）於涵蓋家戶數、需求點、設施點與平均獎勵等多項指標之表現。由表 14 中數據可得以下分析：

（一）路線涵蓋與需求點分布

雖然情境一路線仍涵蓋 950 戶家戶，與專家路線相近，但問卷需求點位涵蓋僅有 4 處，低於專家路線之 6 處與 DQN 最佳路線之 4 處。當「重要點位」不納入評分後，模型學習過程缺乏導向性，偏重住宅密集區，而忽略了民眾實際生活需求分布的特徵。

（二）平均獎勵表現

情境一之平均獎勵為 1364.06，較專家路線（1371.69）與 DQN 最佳路線（1533.65）略低，說明在評估總體服務成效上，其配置效果並不理想。此亦反映出重要點位之設定，對模型總獎勵具有正向提升作用。

情境一分析結果推斷，排除重要點位並增加家戶數權重以後，模型雖可提升人口涵蓋率，但容易忽略居民實際的通勤、就醫與採買等日常需求空間，此外，模型因集中於人口密集區而導致與既有路線重疊率上升，因此，「重要點位」評分在路線規劃的模型中還是十分重要，可以根據民眾的日常需求進行規畫，而不是純粹涵蓋住宅區導致服務失衡與資源浪費。

表 14 情境一涵蓋獎勵函數比較表

公車路線	路線長度	涵蓋家戶數	重要點位	問卷需求點位	重疊既有公車網格數	平均獎勵
情境一路線	10.83 公里	950	2	2	27 格	1364.06
DQN 最佳路線	10.13 公里	673	4	4	17 格	1533.65
專家路線	11.23 公里	981	1	6	43 格	1371.69

資料來源：本研究整理

情境二：取消與既有公車路線重疊之獎懲值

由於南投市既有公車路線多行駛於主要幹道上，本情境設定中，取消原始訓練環境中針對「重疊既有公車網格數」之獎懲值「−5 分/格子」，觀察模型在缺乏此項懲罰引導下之學習行為與規劃結果。

對於「重疊既有公車網格數」之敏感度分析路線結果，因去除「重疊既有公車網格數」之獎懲值，模型不受既有公車路線所限，轉而選擇與中途點、終點交集之路線前進，以降低路線總長，從而提升獎勵分數的目的。可參考圖 14，除部分新增延伸至大埤與東山里之區段外，其餘大多數路線仍與既有公車系統高度重疊。此外，儘管「步數懲罰」的機制仍保留，在缺乏重疊懲罰的情況下，「步數懲罰」對模型的影響較低，模型更傾向規劃以最短路徑連接部分需求點位的策略，減少扣分並快速達成任務目標。

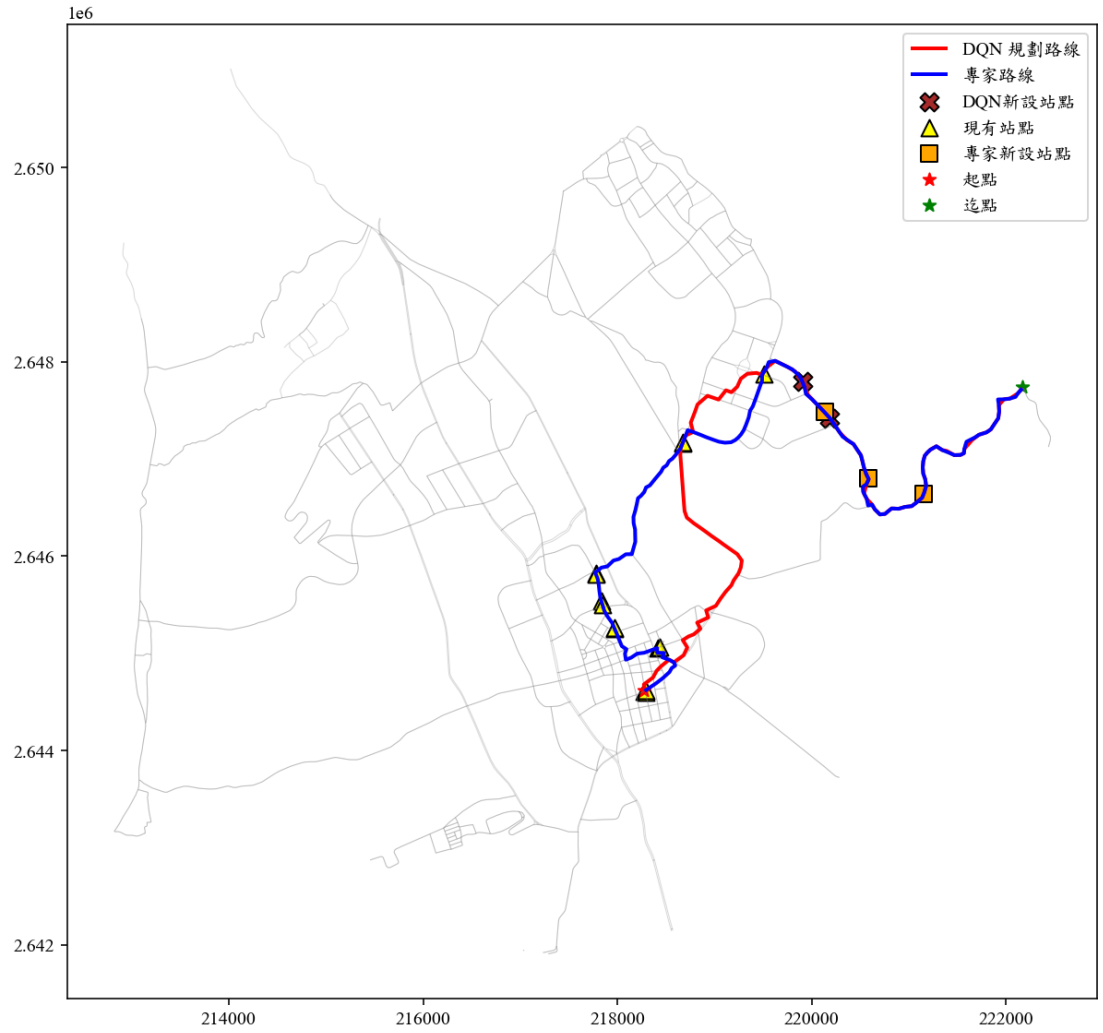


圖 14 專家路線與情境二路線圖

資料來源：本研究整理

表 15 為情境二、DQN 最佳路線與專家路線等三條路線獎勵函數比較表，分別為路線長度、涵蓋家戶數、重要點位、問卷需求點位、重疊既有公車網格數及平均獎勵值等多項指標之表現：

（一）重疊既有公車網格數

情境二在不考慮既有的公車路線下，路線的長度較短於 DQN 最佳路線，涵蓋家戶數略少於專家路線，但重疊既有公車網格數較多於其他兩路線，可以看出此模型對於避開和彎繞度上有較大的影響，重疊既有公車網格數略多於專家路線，但較 DQN 最佳路線多出 32 格，由於與既有公車路線並未設有獎懲值，導致訓練路線時不會避開原有路線，可能造成與原公車路線嚴重的競爭關係。

（二）平均獎勵值表現

情境二之平均獎勵為 1560.96，較專家路線（1371.69）與 DQN 最佳路線（1533.65）較高，說明在評估總體服務成效上有較好的表現，但是從重疊既有公車網格數遠高於 DQN 最佳路線，造成與既有公車高度重疊之現象。

雖然情境二之訓練模型在路線長度和平均獎勵較有優勢，但其規劃結果卻偏離幸福公車規劃原則。幸福公車是為補足既有公共運輸服務缺口而設立，若與既有公車路線重疊率過高、相互競爭，則與設立原則不相符；其次為達成服務公平性，DQN 最佳路線及專家路線皆以公共運輸涵蓋率未達 50%之里民需求作為主要規劃依據，在問卷需求中情境二僅滿足一點位；重要點位包含醫院、市場、學校等民需場所，若路線未能行經重要點位可能使民眾的搭乘意願下降，導致資源配置效益不彰，因此既有公車重疊仍需要加入獎懲值，避免路網重疊、新路線可行性低等情況。

表 15 情境二涵蓋獎勵函數比較表

公車路線	路線長度	涵蓋家戶數	重要點位	問卷需求點位	重疊既有公車網格數	平均獎勵
情境二路線	9.65 公里	941	1	1	49 格	1560.96
DQN 最佳路線	10.13 公里	673	4	4	17 格	1533.65
專家路線	11.23 公里	981	1	6	43 格	1371.69

資料來源：本研究整理

為探討模型訓練過程中既有公車重疊之獎懲值對於模型最終規劃之路線結果影響，特別運用蘇昭銘等人（2013）提出之公車路線評估指標中服務家戶人口重疊率指標，用於評估路線服務之家戶人口數與既有公車服務之家戶人口數的重疊比例，以此評估既有公車重疊之獎懲值，在路線規劃整體成效中產生的影響。其結果如表 16 所示，在去除既有公車重疊之獎懲值的情境二下，模型所規劃路線的服務家戶人口重疊率為 19.17%，相較於設有既有公車重疊之獎懲值的 DQN 最佳路線，情境二路線比 DQN 最佳路線的 16.50%高出 2.67%。

表 16 情境二服務家戶人口重疊率比較表

公車路線	服務家戶人口重疊率 (%)
情境二路線	19.17
DQN 最佳路線	16.50

資料來源：本研究整理

第六章　結論與建議

6.1 結論

本研究以南投市為實證場域，針對偏鄉地區公共運輸涵蓋不足的問題，結合問卷調查、手機信令、票證資料與 GIS 空間技術，導入 DQN 模型，發展一套智慧化、具多目標考量之幸福公車路線規劃模式。根據研究結果與分析，綜整出以下五項結論：

一、南投市現況分析結果

（一）人口結構與高齡化：南投市總人口約 93,884 人，其中 65 歲以上高齡人口佔比達 20%，而 0–18 歲學齡人口共約 14.6%，無法獨自使用私有運具的人口佔全市人口的 34.6%。因此南投市居民出行對於大眾運輸服務的需求極高。

（二）道路路網與公共運輸現況：主要交通要道包括國道 3 號、台 3 線、台 3 甲線與台 14 乙線，市區公路客運路線共 41 條（含延伸線），但大多依附於幹道行駛，因此南投市既有的公車服務大多為中部區域之南北向路線，未能形成完整之市內南北向網絡，也無東西向服務路線，導致偏鄉里別之連結性不足。

（三）服務涵蓋不均與私有運具依賴：南投市中心村里如康壽里、三民里、仁和里等八里之公共運輸涵蓋率可達 100%，但邊陲村里如千秋里、內興里、東山里與內新里涵蓋率分別只有 0%、29%、32%及 50%，其中千秋里更完全無公車行經，顯示傳統路線在偏遠地帶的資源盲點。由於公共運輸覆蓋不足，因此當地居民多依賴私有運具解決通勤、就醫與日常採買等需求。

二、找出南投市幸福公車的潛在需求區域

透過問卷與手機信令資料交叉比對後，找出未被服務但具高度出行需求的區域，尤其在千秋里、內興里、東山里等地。其中，千秋里雖為涵蓋率為 0%的區域，但根據問卷調查回覆「因轄內無公車行駛，民眾已習慣自行開車或騎車」，對幸福公車路線設置意願較低，故未將千秋里納入路線規劃範圍；而內新里、內興里及東山里則對公共運輸涵蓋感到不滿意，進而將路線迄點設置於較偏遠之東

山里，沿途行經內興里之需求點為大埤社區，滿足該地無公共運輸行經之處境，此次研究結果得知，對於偏鄉地區而言，問卷調查相較於票證資料等數值資料，更能準確了解當地民眾所需。

此外，利用 GIS 輔助，透過空間疊圖分析將問卷需求點、站牌點、家戶等資料疊加至南投市村里圖，協助辨識服務涵蓋盲點，更具體呈現需求分布之空間特徵，作為模型訓練中節點分數設定與獎懲機制設計的重要依據，有效提升模型的適用性與檢視當地需求。。

三、DQN 訓練並規劃路線

在 Python 環境中建立強化學習模擬環境，透過設計決策網路與獎勵函數，包含節點需求強度、既有站點、涵蓋率等變數的狀態空間，並將候選公車節點定義為行動空間，進一步設計獎勵函數以鼓勵模型選擇服務人口多、涵蓋率低、需求高的區域，進行多輪訓練與策略更新，最終成功使用 DQN 模型進行訓練，規劃出公車路線。儘管與中區區域運輸發展研究中心專家規劃的人工路線在市區部分略有差異，但透過引入蘇昭銘等人（2013）提出之公車路線評估指標進行比較分析後發現，DQN 模型在多項指標上表現更為優異，其規劃之路線在服務家戶人口重疊率方面相較與專家路線降低了約 5%，提升了整體服務；在彎繞度方面亦更接近理想值 1，代表路徑設計更具效率與可行性。

四、DQN 於偏鄉公車路線規劃之效能與有效性分析

DQN 在模擬環境中成功搜尋並優化南投市偏鄉地區之公車路線。研究結果顯示，DQN 模型在不同區域與服務條件下均展現穩定且良好的學習效果，特別是在公共運輸涵蓋率較低的地區，提升方面表現顯著，像是東山里的服務涵蓋率由原本的 44.51% 提升 73.17%，內興里則由 36.82%提升至 91.33%，涵蓋率提升代表更多居民可步行可及公車站點，對高齡與交通弱勢者特別重要。由研究的內容可以得知，DQN 模型在多目標需求的平衡上表現優異。透過對涵蓋率、彎繞度與路網重複率等多項指標的綜合考量，DQN 有效兼顧服務公平性與營運效率，整體與個別路段重複比例分別控制於 5.11%與 5%以內。

五、獎勵函數敏感度分析

為了解各獎勵參數對 DQN 模型行為與結果之影響，本研究設計兩組敏感度情境進行比對分析，並觀察其對路線策略與服務成效之影響。

（一）情境一：移除重要點位權重，模型轉向偏好家戶數高區域，服務涵蓋提升，但重要設施數由 4 個減少為 1 個，顯示路線涵蓋高家戶數的情況可能犧牲可及性。

（二）情境二：不懲罰重疊既有公車網格數，模型容易選擇現有路線，家戶涵蓋分數提升，但重疊率與資源效率下降。

綜合兩個情境結果可發現，獎勵函數中的「重要點位權重分數」與「重疊既有公車網格數」屬於高敏感變數，若未妥善設定獎勵權重，模型將失去對交通服務盲點與實際需求區的辨識能力，進而影響路線規劃品質。

六、資料整合對模型學習成效之影響

透過整合票證資料、問卷需求調查結果與 GIS 資料，用於建構具有代表性的訓練環境並提供模型足夠的狀態特徵。研究結果顯示，資料的完整性與多樣性對於模型學習成效具有關鍵影響，不僅提升探索效率，更有助於產出兼具可行性與實用性的路線設計成果。尤其在重要地標的涵蓋數量上，DQN 模型相較於專家路線，由 1 個提升至 4 個，進一步凸顯其對多指標環境的權衡問題上所具備的敏感度與回應能力。

儘管 DQN 模型整體表現優於傳統人工規劃，其方法仍存在部分限制。例如，其服務家戶人口重疊率仍達 16.5%，且路段重複比例亦約有 5%，顯示模型尚無法完全避免既有路線的重複覆蓋。此外，在面對突發性需求變動（如節慶活動或臨時接駁需求）時，模型的即時響應能力與動態調整機制尚需進一步強化，未來可朝即時學習或多階段決策系統方向發展，以提升智慧交通應用的彈性與韌性。

6.2 建議

本研究所運用的強化學習框架僅 DQN，其結構雖具備穩定性與良好表現，然而仍存在模型選擇單一之限制。後續研究可以採用不同的深度強化模型，對 DQN 模型進行比較，探索對於公車路線規劃這門議題，是否還有性能更好、成效更高的模型進行研究，為後續公車路線規劃自動化之研究做一份貢獻。

在獎懲機制的設定上，主要是依靠反覆地訓練及修改，根據不同獎懲機制下取得的 DQN 路線結果進行分析，得出相對合理且具科學依據的組合。然而，該過程仍需大量人工作業與經驗判斷，如果能在強化學習中引入自動化程式調整獎懲機制，會使獎懲機制的設定更為合理化，以達到更加接近路線最優解的可能。未來建議可採用 Pareto 最佳化策略，使模型在多目標指標（如涵蓋率、彎繞度、重複比例等）之間自動尋求平衡解，藉此找出最優解，以科學的角度提供多指標之間的衡量方案。

此外，對於營運成本與效益回報在公車路線規劃產生的影響相關問題上，其原因是受限於研究時間與資料可取得性，未將其納入考量因素，如公車營運過程中油料費用、駕駛人力成本及車輛維護等旅行成本進行整體考量。未來若能進一步建構旅行成本評估模型，並與 DQN 路線規劃模型整合，將有助於從成本效益角度對公車路線進行更全面的投入回報分析。此舉不僅可作為規劃決策之補充依據，亦能提升模型在實務應用層面的現實貼合度與實用價值。

參考文獻

1. 王智文（2020），Deep Q-network 演算法應用於並行機台調度問題，國立交通大學工業工程與管理學系碩士論文。
2. 沈靖筌（2023），應用注意力機制實作無人機自動導航飛行之研究比較，國立中興大學資訊工程學系碩士論文。
3. 林良泰等人（2023），區域運輸發展研究中心服務升級 2.0 計畫 (112-113 年)-中區區域計畫，交通部運輸研究所。
4. 林婉伃（2024），應用多目標強化學習於易腐性商品之生產存貨路徑規劃問題，國立成功大學工業與資訊管理學系碩士論文。
5. 柯威年（2021），基於強化學習的路徑和導航優化，元智大學資訊管理學系碩士論文。
6. 宮偉杰（2015），新竹縣五峰鄉愛心巴士站牌調整之研究，中華大學運輸科技與物流管理學系碩士論文。
7. 許詠翔（2021），基於 ROS 無人載具之 DQN 導引學習，國立雲林科技大學電機工程系碩士論文。
8. 黃雅鈺（2021），應用增強式學習 DQN 於植保機之三維路徑規劃，國立成功大學工程科學所碩士論文。
9. 蔡秀榛（2019），應用於室內倉儲環境之自主四旋翼的自動駕駛與貨品辨識，國立中興大學電機工程學系碩士論文。
10. 蘇昭銘、邱裕鈞、張志鴻、王穆衡、王晉元、蔡欽同、沈美慧（2013），「公路汽車客運路線審議作業空間決策輔助指標之建立與應用」，運輸計劃季刊，第四十二卷第二期，頁 171-194。
11. 蘇昭銘、王張煒、何文基（2015），「中小型鄉鎮接駁公車之路線設計方法」，運輸計劃季刊，第四十四卷第四期，頁 313-332。
12. 蘇昭銘等人（2019），南投縣整體基本民行公車營運體系建置研究計劃案，南投縣政府。
13. 交通部公路局，公路客運即時動態資訊網，擷取日期：2024 年 9 月 18 日，網站：https://www.taiwanbus.tw/eBUSPage/Default.aspx?lan=C。
14. 南投縣政府，南投縣政儀表板，擷取日期：2024 年 9 月 30 日，網站：https://reurl.cc/vp6aGN
15. Matisziw(2006), ” Strategic route extension in transit networks, ” European Journal of Operational Research, No. 171, pp.661-673.

附錄

一、程式碼：

```
from collections import deque
import os
import numpy as np
import pandas as pd
import geopandas as gpd
import networkx as nx
import torch
import torch.nn as nn
import torch.optim as optim
import random
from shapely.geometry import Point, box, LineString, MultiLineString,
GeometryCollection
from shapely.ops import nearest_points
import matplotlib.pyplot as plt
import chardet
# === 環境設定 ===
os.environ["CUDA_VISIBLE_DEVICES"] = "0"
os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"
os.environ["OMP_NUM_THREADS"] = "1"
os.environ["MKL_NUM_THREADS"] = "1"
torch.set_num_threads(1)
torch.set_num_interop_threads(1)
ORIGINAL_CRS = "EPSG:4326"
PROJECTED_CRS = "EPSG:3826"
ROAD_SHP_PATH = r"C:/Users/willi/Desktop/南投市公車站點 PY/南投路網/路網
圖.shp"
torch.backends.cudnn.benchmark = True
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"使用裝置：{device}")
# 1. 載入道路 SHP
def load_shp_network(shp_path):
    print("載入本地道路 SHP 並建立圖形結構...")
    edges = gpd.read_file(shp_path).to_crs(PROJECTED_CRS)
```

```
    G = nx.MultiDiGraph()
    for _, row in edges.iterrows():
        geom = row.geometry
        lines = [geom] if isinstance(geom, LineString) else list(geom.geoms) if
isinstance(geom, MultiLineString) else []
        for line in lines:
            u = (line.coords[0][0], line.coords[0][1])
            v = (line.coords[-1][0], line.coords[-1][1])
            G.add_node(u, x=u[0], y=u[1])
            G.add_node(v, x=v[0], y=v[1])
            length = line.length
            G.add_edge(u, v, geometry=line, length=length)
            G.add_edge(v, u, geometry=LineString(list(line.coords)[::-1]),
length=length)
    print("路網圖已載入並轉換投影。")
    return G, edges
G, gdf_edges = load_shp_network(ROAD_SHP_PATH)
print("節點與邊資料已提取。")
# 偵測並讀取 CSV
def read_csv_with_detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        rawdata = f.read(10000)
        result = chardet.detect(rawdata)
    encoding = result['encoding']
    print(f"偵測到的編碼: {encoding}")
    return pd.read_csv(file_path, encoding=encoding)
# 載入公車站牌 CSV 並建立 GeoDataFrame
BUS_STOP_CSV = r"C:/Users/willi/Desktop/南投市公車站點 PY/確認資料
/nantou_bus_stops.csv"
stops_df = read_csv_with_detect_encoding(BUS_STOP_CSV)
stops_geom_wgs = gpd.points_from_xy(stops_df.lon, stops_df.lat,
crs=ORIGINAL_CRS)
stops_gdf = gpd.GeoDataFrame(stops_df,
geometry=stops_geom_wgs).to_crs(PROJECTED_CRS)
# 載入新增站牌 CSV 並建立 GeoDataFrame
NEW_STOP_CSV = r"C:/Users/willi/Desktop/網格評分測試版 Nv31（修正評估指
標）/新增站點資訊(東).csv"
```

```
new_df = read_csv_with_detect_encoding(NEW_STOP_CSV)
new_geom_wgs = gpd.points_from_xy(new_df.lon, new_df.lat,
crs=ORIGINAL_CRS)
bus_csv_new_stops = gpd.GeoDataFrame(new_df,
geometry=new_geom_wgs).to_crs(PROJECTED_CRS)
# 2.建立網格並篩選含道路的格子
def create_and_filter_grid(edges_gdf, cell_size=63.8):
    print("建立並篩選網格：僅保留含道路者...")
    xmin, ymin, xmax, ymax = edges_gdf.total_bounds
    num_x = int((xmax - xmin) / cell_size) + 1
    num_y = int((ymax - ymin) / cell_size) + 1
    grids = []
    for i in range(num_x):
        for j in range(num_y):
            minx = xmin + i * cell_size
            miny = ymin + j * cell_size
            grids.append(box(minx, miny, minx + cell_size, miny + cell_size))
    grid = gpd.GeoDataFrame(geometry=grids, crs=PROJECTED_CRS)
    grid['has_road'] = grid.geometry.apply(lambda c: edges_gdf.intersects(c).any())
    road_grid = grid[grid['has_road']].reset_index(drop=True)
    print(f"網格總數：{len(grid)} -> 含道路：{len(road_grid)}")
    return road_grid
grid_gdf = create_and_filter_grid(gdf_edges)
# 3. 載入家戶資料並計數
def load_and_link_households(path):
    print("載入家戶資料...")
    df = pd.read_csv(path)
    geom = gpd.GeoSeries.from_wkt(df['geometry'],
crs=ORIGINAL_CRS).to_crs(PROJECTED_CRS)
    households = gpd.GeoDataFrame(df, geometry=geom, crs=PROJECTED_CRS)
    grid_gdf['households_count'] = grid_gdf.geometry.apply(lambda cell:
households.within(cell).sum())
    print("家戶數量標記完成。")
    return households
households_gdf = load_and_link_households(r"C:/Users/willi/Desktop/網格評分測
試版 Nv31（修正評估指標）/南投市家戶點位資料-20250522T145005Z-1-001/南
投市家戶點位資料/household_data_updated.csv")
```

```
# 4. 載入並裁剪公車路線
def load_and_link_bus_routes(shp, boundary_shp):
    print("載入公車路線並裁剪...")
    routes = gpd.read_file(shp).to_crs(PROJECTED_CRS)
    boundary = gpd.read_file(boundary_shp).to_crs(PROJECTED_CRS)
    clipped = routes.clip(boundary)
    grid_gdf['bus_route_count'] = grid_gdf.geometry.apply(lambda cell:
clipped.intersects(cell).sum())
    print("公車路線標記完成。")
    return clipped
bus_routes_gdf = load_and_link_bus_routes(
    r"C:/Users/willi/Desktop/網格評分測試版 Nv31（修正評估指標）/既有公車
路線-20250522T150136Z-1-001/既有公車路線/TDX_THBTNC_SHAPE.shp",
    r"C:/Users/willi/Desktop/網格評分測試版 Nv31（修正評估指標）/南投市邊
界-20250522T145413Z-1-001/南投市邊界/南投市界.shp"
)
# 5. 載入重要建築並計算加權分數 + 計數
building_weights = {"市場":3, "學校":4, "公家機關":2, "醫院":5, "全聯、家樂福
":1}
def load_and_score_buildings(path):
    print("載入重要建築點並計算分數、數量...")
    df = pd.read_csv(path)
    geom = gpd.points_from_xy(df.經度, df.緯度,
crs=ORIGINAL_CRS).to_crs(PROJECTED_CRS)
    builds = gpd.GeoDataFrame(df, geometry=geom, crs=PROJECTED_CRS)
    for typ, w in building_weights.items():
        mask = builds['類別'] == typ
        grid_gdf[f'{typ}_exists'] = grid_gdf.geometry.apply(lambda c:
builds[mask].intersects(c).any()).astype(int)
    grid_gdf['building_score'] = sum(grid_gdf[f'{typ}_exists'] * w for typ, w in
building_weights.items())
    grid_gdf['building_count'] = grid_gdf.geometry.apply(lambda c:
builds.within(c).sum())
    print("重要點位加權分數及建築數量計算完成。")
    return builds
buildings_gdf = load_and_score_buildings(r"C:/Users/willi/Desktop/網格評分測試
版 Nv31（修正評估指標）/重要點位.csv")
```

```
# 6. 載入需求點並標記
def load_and_link_demand(path):
    print("載入需求點資料...")
    df = pd.read_csv(path)
    geom = gpd.points_from_xy(df.經度, df.緯度,
crs=ORIGINAL_CRS).to_crs(PROJECTED_CRS)
    demand = gpd.GeoDataFrame(df, geometry=geom, crs=PROJECTED_CRS)
    grid_gdf['demand_score'] = grid_gdf.geometry.apply(lambda c:
demand.intersects(c).any()).astype(int)
    print("需求點標記完成。")
    return demand
demand_gdf = load_and_link_demand(r"C:/Users/willi/Desktop/網格評分測試版
Nv31（修正評估指標）/需求點經緯度.csv")
# 7. 建立 8 方位鄰接網格
def build_adjacency(grid):
    print("建立鄰接關係...")
    grid['centroid'] = grid.geometry.centroid
    cell = grid.geometry.iloc[0].bounds[2] - grid.geometry.iloc[0].bounds[0]
    sidx = grid.sindex
    dirs = [(0,cell),(0,-cell),(-cell,0),(cell,0),(cell,cell),(-cell,-cell),(cell,-cell),(-
cell,cell)]
    nbrs = []
    for idx, row in grid.iterrows():
        cx, cy = row.centroid.x, row.centroid.y
        pts = []
        for dx, dy in dirs:
            p = Point(cx+dx, cy+dy)
            cands = list(sidx.intersection((p.x-0.1,p.y-0.1,p.x+0.1,p.y+0.1)))
            nn, md = -1, float('inf')
            for c in cands:
                d = grid.centroid.iloc[c].distance(p)
                if d < md:
                    nn, md = c, d
            pts.append(nn)
        nbrs.append(pts)
    grid['neighbors'] = nbrs
    print("鄰接完成。")
```

```
        return grid
grid_gdf = build_adjacency(grid_gdf)
# 8. 設定起訖點、距離及中途點
start_pt = Point(218302.1050, 2644626.8971)
end_pt      = Point(222150.3313, 2647721.8971)
mid_pt      = Point(219530.7755, 2647929.3744)
grid_gdf['dist_start'] = grid_gdf.geometry.distance(start_pt)
grid_gdf['dist_mid']      = grid_gdf.geometry.distance(mid_pt)
grid_gdf['dist_end']      = grid_gdf.geometry.distance(end_pt)
start_idx = grid_gdf['dist_start'].idxmin()
mid_idx      = grid_gdf['dist_mid'].idxmin()
end_idx      = grid_gdf['dist_end'].idxmin()
end_center = grid_gdf.loc[end_idx, 'centroid']
mid_center = grid_gdf.loc[mid_idx, 'centroid']
print(f"起點 idx={start_idx}, 中途必經 idx={mid_idx}, 終點 idx={end_idx}")
# DQN 模型與 Agent 定義
class DQN(nn.Module):
        def __init__(self, in_f, out_f=8):
                super().__init__()
                self.net = nn.Sequential(
                        nn.Linear(in_f, 64), nn.ReLU(),
                        nn.Linear(64, 32),    nn.ReLU(),
                        nn.Linear(32, out_f)
                )
        def forward(self, x):
                return self.net(x)
class DQNAgent:
        def __init__(self, in_f):
                self.policy = DQN(in_f).to(device)
                self.target = DQN(in_f).to(device)
                self.target.load_state_dict(self.policy.state_dict())
                self.opt = optim.AdamW(self.policy.parameters(), lr=5e-5)
                self.memory = deque(maxlen=50000)
                self.eps, self.eps_min, self.eps_dec = 1.0, 0.05, 0.997
                self.gamma, self.batch, self.upd_freq = 0.99, 64, 5000
                self.step_ct = 0
                self.repeat_penalty = 50.0
```

```
def select_action(self, state, idx):
    nbrs = grid_gdf.loc[idx, 'neighbors']
    valid = [i for i,n in enumerate(nbrs) if n != -1 and n not in self.visited]
    repeated = False
    if valid:
        choices = valid
    else:
        choices = [i for i,n in enumerate(nbrs) if n != -1]
        repeated = True
    if not choices:
        return None, None, False
    if random.random() < self.eps:
        a = random.choice(choices)
    else:
        with torch.no_grad():
            qv = self.policy(state.unsqueeze(0))[0]
            a = choices[torch.argmax(qv[choices]).item()]
    nxt = nbrs[a]
    self.visited.add(nxt)
    return a, nxt, repeated
def store(self, s, a, r, s2, done):
    self.memory.append((
        s.cpu(),
        torch.tensor([a], device=device, dtype=torch.long),
        torch.tensor([r], device=device, dtype=torch.float32),
        s2.cpu() if s2 is not None else None,
        torch.tensor([float(done)], device=device, dtype=torch.float32)
    ))
def train(self):
    if len(self.memory) < self.batch: return
    batch = random.sample(self.memory, self.batch)
    S    = torch.stack([b[0] for b in batch]).to(device)
    A    = torch.cat([b[1] for b in batch])
    R    = torch.cat([b[2] for b in batch])
    nxtS = torch.stack([b[3] for b in batch if b[3] is not None]).to(device)
    D    = torch.cat([b[4] for b in batch])
    with torch.no_grad():
```

```
                tq    = self.target(nxtS).max(1)[0]
                tgt = R + (1-D)*self.gamma*tq
            cur    = self.policy(S).gather(1, A.unsqueeze(1)).squeeze()
            loss = nn.MSELoss()(cur, tgt)
            self.opt.zero_grad()
            loss.backward()
            torch.nn.utils.clip_grad_norm_(self.policy.parameters(), 1.0)
            self.opt.step()
            self.eps = max(self.eps_min, self.eps*self.eps_dec)
            self.step_ct += 1
            if self.step_ct % self.upd_freq == 0:
                self.target.load_state_dict(self.policy.state_dict())
def get_state(idx, visited_mid):
    d = grid_gdf.loc[idx]
    return torch.tensor([
            d.households_count ,
            d.bus_route_count    ,
            d.building_score       ,
            d.demand_score         ,
            d.dist_mid                ,
            d.dist_end                ,
            float(visited_mid)
    ], device=device, dtype=torch.float32)
# 獎勵函數
mid_weight = 0.8
end_weight = 5.0
mid_bonus    = 50.0
end_bonus    = 100.0
# 公車路線懲罰權重
bus_penalty_weight = 1.0
def calc_reward(cur, nxt, visited_mid):
    if not visited_mid:
            imp = (grid_gdf.loc[cur, 'dist_mid'] - grid_gdf.loc[nxt, 'dist_mid']) *
mid_weight
    else:
            imp = (grid_gdf.loc[cur, 'dist_end'] - grid_gdf.loc[nxt, 'dist_end']) *
end_weight
```

```
        # 各項權重
        hb      = grid_gdf.loc[nxt,'households_count'] * 0.45
        bb      = grid_gdf.loc[nxt,'building_score']    * 0.275
        db      = grid_gdf.loc[nxt,'demand_score']       * 0.275
        # 過中途、到終點的額外獎勵
        bonus = 0.0
        if nxt == mid_idx and not visited_mid:
            bonus += mid_bonus
        if nxt == end_idx and visited_mid:
            bonus += end_bonus
        # 每步固定扣分
        step_penalty = -1.0
        # 既有公車路線懲罰
        bp = grid_gdf.loc[nxt, 'bus_route_count'] * bus_penalty_weight
        # 最終報酬
        r = imp + hb + bb + db + bonus + step_penalty - bp
        return r
# 10. 公車路線評估 & 視覺化
def sample_points_along_line(line, dist=50):
        pts, L = [], line.length
        n = int(L / dist)
        for i in range(n+1):
            pts.append(line.interpolate(min(i*dist, L)))
        return pts
def get_grid_path_from_route(route_gdf, gg):
        geom = route_gdf.geometry.iloc[0]
        pts = []
        if geom.geom_type == 'MultiLineString':
            for ln in geom.geoms:
                pts += sample_points_along_line(ln)
        else:
            pts = sample_points_along_line(geom)
        clean = []
        for p in pts:
            sel = gg[gg.geometry.contains(p)]
            idx = sel.index[0] if not sel.empty else gg.geometry.distance(p).idxmin()
            if not clean or idx != clean[-1]:
```

```
                clean.append(idx)
        return clean
def evaluate_bus_route_reward(shp, gg, repeat_penalty=20.0):
        br = gpd.read_file(shp).to_crs(PROJECTED_CRS)
        route = get_grid_path_from_route(br, gg)
        tot_r, dist = 0.0, 0.0
        visited_mid = False
        visited_set = {route[0]}
        for i in range(1, len(route)):
                cur, nxt = route[i - 1], route[i]
                repeated = nxt in visited_set
                if not repeated:
                        visited_set.add(nxt)
                step_distance = gg.centroid.loc[cur].distance(gg.centroid.loc[nxt])
                dist += step_distance
                r = calc_reward(cur, nxt, visited_mid)
                if repeated:
                        r -= repeat_penalty
                if nxt == mid_idx and not visited_mid:
                        visited_mid = True
                tot_r += r
                if nxt == end_idx and visited_mid:
                        break
        # 計算平均每公里獎勵
        avg = tot_r / dist * 1000 if dist > 0 else 0
        return route, tot_r, dist, avg
grid_gdf['cell_reward'] = (
        grid_gdf['households_count'] * 0.45 +
        grid_gdf['building_score']       * 0.275 +
        grid_gdf['demand_score']          * 0.275
)
# 10. DQN 訓練迴圈
bus_shp = r"C:/Users/willi/Desktop/網格評分測試版 Nv31（修正評估指標）/東側
預設路線 shp-20250522T144830Z-1-001/東側預設路線 shp/預設路線.shp"
bus_route, bus_tot, bus_dist, bus_avg = evaluate_bus_route_reward(bus_shp,
grid_gdf)
print(f"Bus route length: {len(bus_route)}, avg reward: {bus_avg:.2f}")
```

```
imitation_prob = 0.9
imitation_prob_min = 0.2
imitation_prob_dec = 0.999
print("開始進行 DQN 訓練迴圈...")
agent       = DQNAgent(in_f=7)
EPISODES, MAX_STEPS = 3000, 250
best_avg, best_path    = -np.inf, []
results = []
for ep in range(EPISODES):
    agent.visited          = {start_idx}
    visited_mid             = False
    state                    = get_state(start_idx, visited_mid)
    cur                      = start_idx
    total_r                  = 0.0
    path                     = [start_idx]
    done                     = False
    bus_penalty_steps    = 0
    for st in range(1, MAX_STEPS+1):
        if random.random() < imitation_prob and cur in bus_route:
            idx_in_route = bus_route.index(cur)
            if idx_in_route + 1 < len(bus_route):
                next_pos = bus_route[idx_in_route + 1]
                nbrs         = grid_gdf.loc[cur, 'neighbors']
                if next_pos in nbrs:
                    a             = nbrs.index(next_pos)
                    nxt           = next_pos
                    repeated = (nxt in agent.visited)
                    agent.visited.add(nxt)
                else:
                    a, nxt, repeated = agent.select_action(state, cur)
            else:
                a, nxt, repeated = agent.select_action(state, cur)
        else:
            a, nxt, repeated = agent.select_action(state, cur)
        if grid_gdf.loc[nxt, 'bus_route_count'] > 0:
            bus_penalty_steps += 1
        r = calc_reward(cur, nxt, visited_mid)
```

```
            visited_mid |= (nxt == mid_idx)
            if repeated:
                r -= agent.repeat_penalty
            done = (nxt == end_idx and visited_mid)
            ns = get_state(nxt, visited_mid)
            agent.store(state, a, r, ns, done)
            agent.train()
            total_r += r
            path.append(nxt)
            state, cur = ns, nxt
            if done:
                break
        dist = sum(
            grid_gdf.loc[path[i-1], 'centroid'].distance(
            grid_gdf.loc[path[i],     'centroid'])
            for i in range(1, len(path))
        )
        avg_r = total_r / dist * 1000 if dist > 0 else 0
        stats = {
            'episode': ep+1,
            'reward': avg_r,
            'visited_nodes': len(path)-1,
            'householdcovered_count': sum(grid_gdf.loc[i,'households_count'] for i in
path),
            'facility_weight_sum':      sum(grid_gdf.loc[i,'building_count']       for i in
path),
            'buscovered_count':            sum(grid_gdf.loc[i,'bus_route_count']    for i in
path),
            'demand':                          sum(grid_gdf.loc[i,'demand_score']       for i
in path),
            'reached_mid': visited_mid,
            'reached_end': done
        }
        results.append(stats)
        if avg_r > best_avg:
            best_avg     = avg_r
            best_path    = path.copy()
```

```
            best_stats = stats.copy()
        print(
            f"回合 {ep+1} 結束：平均獎勵={avg_r:.2f}，"
            f"中途點={'✓' if visited_mid else '✗'}，"
            f"終點={'✓' if done else '✗'}，"
            f"家戶={stats['householdcovered_count']}，"
            f"建築={stats['facility_weight_sum']}，"
            f"需求={stats['demand']}，"
            f"既有公車數量={bus_penalty_steps}"
        )
        imitation_prob = max(imitation_prob_min, imitation_prob *
imitation_prob_dec)
print("訓練完成，最佳平均獎勵:", best_avg)
#初始化 DQN 自動補站清單
forced_indices = set()
forced = []
candidates = []
# 2.1 找出路徑中已有的站牌
path_cells = grid_gdf.loc[best_path]
stops_on_path = stops_gdf[stops_gdf.geometry.apply(
    lambda p: any(p.within(cell) for cell in path_cells.geometry)
)]
threshold = 600.0
MAX_FORCED = 5
cum_dist = 0.0
last_idx = best_path[0]
existing = stops_gdf.copy()
for idx in best_path[1:]:
    step = grid_gdf.centroid.loc[last_idx].distance(grid_gdf.centroid.loc[idx])
    cum_dist += step
    # 若此格已有站牌，重置累積距離
    if any(pt.within(grid_gdf.geometry.loc[idx]) for pt in existing.geometry):
        last_idx = idx
        cum_dist = 0.0
        continue
    # 累積距離超過閾值且未達上限，補站
    if cum_dist >= threshold and len(forced) < MAX_FORCED:
```

```
            start = min(best_path.index(last_idx), best_path.index(idx))
            end   = max(best_path.index(last_idx), best_path.index(idx))
            segment = best_path[start:end+1]
            if segment:
                pick = max(segment, key=lambda j: grid_gdf.loc[j, 'cell_reward'])
                if pick not in forced_indices:
                    cent = grid_gdf.at[pick, 'centroid']
                    if all(cent.distance(pt) > 300 for pt in existing.geometry):
                        forced_indices.add(pick)
                        forced.append(pick)
                    # 加入 existing，用於後續距離檢查
                        existing = pd.concat([
                            existing,
                            gpd.GeoDataFrame(
                                [{'geometry': grid_gdf.at[pick, 'centroid']}],
                                geometry='geometry', crs=PROJECTED_CRS
                            )
                        ], ignore_index=True)
                last_idx = pick
                cum_dist = 0.0
# 2.2 強制補站
forced_gdf = gpd.GeoDataFrame(
    {'grid_idx': forced, 'type': ['forced'] * len(forced)},
    geometry=[grid_gdf.centroid.loc[i] for i in forced],
    crs=PROJECTED_CRS
) if forced else gpd.GeoDataFrame({'grid_idx':[], 'type':[]}, geometry=[],
crs=PROJECTED_CRS)
# 2.3 候選補站：當路徑上沒有既有站牌時才啟動
if stops_on_path.empty:
    rewards = grid_gdf.loc[best_path, ['cell_reward']].sort_values('cell_reward',
ascending=False)
    for idx in rewards.head(10).index:
        center = grid_gdf.at[idx, 'centroid']
        if all(center.distance(p) > 1000 for p in existing.geometry):
            candidates.append(idx)
        if len(candidates) >= 3:
            break
```

```
# 2.4 候選補站
candidates_gdf = gpd.GeoDataFrame(
    {'grid_idx': candidates, 'type': ['candidate'] * len(candidates)},
    geometry=[grid_gdf.centroid.loc[i] for i in candidates],
    crs=PROJECTED_CRS
) if candidates else gpd.GeoDataFrame({'grid_idx':[], 'type':[]}, geometry=[],
crs=PROJECTED_CRS)
# 2.5 合併 forced 與 candidates
dqn_new_stops = pd.concat([forced_gdf, candidates_gdf], ignore_index=True)
# 2.6 轉回 WGS84 並輸出經緯度
new_stops_wgs = dqn_new_stops.to_crs(ORIGINAL_CRS)
new_stops_wgs['lon'] = new_stops_wgs.geometry.x
new_stops_wgs['lat'] = new_stops_wgs.geometry.y
print(new_stops_wgs[['grid_idx','type','lon','lat']])
# 結果比較
import geopandas as gpd
plt.rcParams['font.family'] = ['Times New Roman', 'DFKai-SB']
plt.rcParams['axes.unicode_minus'] = False
def plot_combined_routes(
    dqn_route, bus_route, eg, gg,
    stops_old,
    stops_dqn_new,
    stops_bus_old,
    stops_bus_new
):
    fig, ax = plt.subplots(figsize=(10,10))
    eg.plot(ax=ax, linewidth=0.5, color='gray', alpha=0.3)
    for route, c, label in [(dqn_route,'red','DQN 規劃路線'), (bus_route,'blue','專家
路線')]:
        pts = [nearest_points(gg.centroid.loc[i], eg.unary_union)[1] for i in route]
        ln  = LineString([(p.x,p.y) for p in pts])
        gpd.GeoSeries([ln], crs=gg.crs).plot(ax=ax, linewidth=2, color=c,
label=label)
    if not stops_dqn_new.empty:
        pts = [nearest_points(pt, eg.unary_union)[1] for pt in
stops_dqn_new.to_crs(gg.crs).geometry]
        xs, ys = [p.x for p in pts], [p.y for p in pts]
```

```
            ax.scatter(xs, ys, marker='X', s=100, edgecolor='black', color='brown',
label='DQN 新設站點')
        if not stops_bus_old.empty:
            pts = [nearest_points(pt, eg.unary_union)[1] for pt in
stops_bus_old.to_crs(gg.crs).geometry]
            xs, ys = [p.x for p in pts], [p.y for p in pts]
            ax.scatter(xs, ys, marker='^', s=80, edgecolor='black', color='yellow', label='
現有站點')
        if not stops_bus_new.empty:
            pts = [nearest_points(pt, eg.unary_union)[1] for pt in
stops_bus_new.to_crs(gg.crs).geometry]
            xs, ys = [p.x for p in pts], [p.y for p in pts]
            ax.scatter(xs, ys, marker='s', s=80, edgecolor='black', color='orange', label='
專家新設站點')
        dqn_start = gg.centroid.loc[dqn_route[0]].coords[0]
        ax.scatter(*dqn_start, marker='*', s=50, color='red', label='起點')
        end_coords = gg.centroid.loc[end_idx].coords[0]
        ax.scatter(*end_coords, marker='*', s=50, color='green', label='迄點')
        ax.legend()
        plt.show()
path_cells = grid_gdf.loc[best_path, 'geometry']
bus_cells   = grid_gdf.loc[bus_route,'geometry']
# 1. 既有站牌
stops_on_path_old = stops_gdf[
        stops_gdf.geometry.apply(lambda p: any(p.within(cell) for cell in path_cells))
]
# 2. DQN 補的站牌（
stops_dqn_new = dqn_new_stops[
        (dqn_new_stops['type']=='forced')
        & dqn_new_stops.geometry.apply(lambda p: any(p.within(cell) for cell in
path_cells))
]
# 3. 專家路線既有站牌
stops_on_bus_old = stops_gdf[
        stops_gdf.geometry.apply(lambda p: any(p.within(cell) for cell in bus_cells))
]
# 4. 專家路線候選站牌
```

```
stops_bus_new = bus_csv_new_stops[
    bus_csv_new_stops.geometry.apply(lambda p: any(p.within(cell) for cell in
path_cells))
]
# 印出數量
print(f"1. DQN 路線上的既有站牌：{len(stops_on_path_old)}")
print(f"2. DQN 補的新站牌：  {len(stops_dqn_new)}")
print(f"3. Bus 路線既有站牌：  {len(stops_on_bus_old)}")
print(f"4. Bus 候選站牌：  {len(stops_bus_new)}")
# 畫圖
plot_combined_routes(
    best_path, bus_route,
    gdf_edges, grid_gdf,
    stops_on_path_old,
    stops_dqn_new,
    stops_on_bus_old,
    stops_bus_new
)
# --- 在最終結果比較前，先計算 DQN 路徑距離 ---
if best_path:
    dqn_dist_m = sum(
        grid_gdf.loc[best_path[i-1], 'centroid'].distance(
            grid_gdf.loc[best_path[i], 'centroid']
        )
        for i in range(1, len(best_path))
    )
    dqn_dist_km = dqn_dist_m / 1000
    dqn_sum_households = sum(grid_gdf.loc[i, 'households_count'] for i in
best_path)
    dqn_sum_buildings  = sum(grid_gdf.loc[i, 'building_count']    for i in
best_path)
    dqn_sum_demand     = sum(grid_gdf.loc[i, 'demand_score']      for i in
best_path)
    dqn_reached        = (best_path[-1] == end_idx)
    dqn_bus_penalty_steps = sum(
        1 for i in best_path
        if grid_gdf.loc[i, 'bus_route_count'] > 0
```

```
    )
    print("=== DQN 最終結果比較 ===")
    print(f"路徑長度：{dqn_dist_km:.2f} 公里")
    print(f"平均獎勵：{best_avg:.2f}")
    print(
        f"  家戶總數={dqn_sum_households}、"
        f"建築數量={dqn_sum_buildings}、"
        f"問卷需求點數量={dqn_sum_demand}，"
        f"{'已' if dqn_reached else '未'}到達訖點、"
        f"既有公車數量={dqn_bus_penalty_steps}"
    )
else:
    print("Warning: best_path 為空，無法比較 DQN 結果。")
if bus_route:
    bus_dist_km = bus_dist / 1000
    bus_sum_households = sum(grid_gdf.loc[i, 'households_count'] for i in
bus_route)
    bus_sum_buildings  = sum(grid_gdf.loc[i, 'building_count']      for i in
bus_route)
    bus_sum_demand     = sum(grid_gdf.loc[i, 'demand_score']        for i in
bus_route)
    bus_reached        = (bus_route[-1] == end_idx)
    bus_bus_penalty_steps = sum(
        1 for i in bus_route
        if grid_gdf.loc[i, 'bus_route_count'] > 0
    )
    print("=== 公車路線最終結果比較 ===")
    print(f"路徑長度：{bus_dist_km:.2f} 公里")
    print(f"平均獎勵：{bus_avg:.2f}")
    print(
        f"  家戶總數={bus_sum_households}、"
        f"建築數量={bus_sum_buildings}、"
        f"問卷需求點數量={bus_sum_demand}，"
        f"{'已' if bus_reached else '未'}到達訖點、"
        f"既有公車數量={bus_bus_penalty_steps}"
    )
else:
```

```
        print("Warning: bus_route 為空，無法比較公車路線結果。")
import pandas as pd
df = pd.DataFrame(results)
plt.figure()
plt.plot(df['episode'], df['reward'])
plt.xlabel('訓練回合數')
plt.ylabel('每公里平均獎勵值')
plt.title('DQN 訓練獎勵曲線圖')
plt.grid(True)
plt.show()
import geopandas as gpd
zones = gpd.read_file(r"C:/Users/willi/Desktop/南投市公車站點 PY/專題大三下/村
(里)界(TWD97_121 分
帶)1130807/VILLAGE_NLSC_121_1130807.shp").to_crs(PROJECTED_CRS)
hh_zone = gpd.sjoin(
        households_gdf[['geometry']],
        zones[['VILLNAME','geometry']],
        how='left',
        predicate='within'
)
pop = hh_zone.groupby('VILLNAME').size().rename('population')
zones = zones.merge(pop, on='VILLNAME')
def compute_SP_A(route, grid_gdf, households_gdf, buffer_dist=300):
        pts = [grid_gdf.centroid.loc[i] for i in route]
        buf = LineString([(p.x, p.y) for p in pts]).buffer(buffer_dist)
        return int(households_gdf.geometry.apply(lambda p: buf.intersects(p)).sum())
def compute_SPO_A_zone(
        route,
        grid_gdf,
        households_gdf,
        bus_routes_gdf,
        zones_gdf,
        walk_dist=300
):
        # 1. 審議 A 路線的 buffer
        pts_A = [grid_gdf.centroid.loc[i] for i in route]
        buf_A = LineString([(p.x, p.y) for p in pts_A]).buffer(walk_dist)
```

```
    # 2. 找出 A 線經過的里
    zones_passed = zones_gdf[zones_gdf.geometry.intersects(buf_A)]
    zone_union = zones_passed.geometry.union_all()
    # 3. 篩選出行經相同里之其他既有路線
    candidate_exist =
bus_routes_gdf[bus_routes_gdf.geometry.intersects(zone_union)].copy()
    merged_exist = candidate_exist.geometry.union_all()
    buf_exist = merged_exist.buffer(walk_dist)
    # 4. 標記家戶
    hh = households_gdf.copy()
    hh['in_A']     = hh.geometry.within(buf_A)
    hh['in_exist'] = hh.geometry.within(buf_exist)
    num_A_exist = int((hh['in_A'] & hh['in_exist']).sum())
    num_exist   = int(hh['in_exist'].sum())
    return (num_A_exist / num_exist * 100.0) if num_exist > 0 else 0.0
def compute_R(route, grid_gdf, G, start_pt, mid_pt, end_pt):
    """
    計算路線效率比 R_A，並考慮必經中途點：
    分母為 起點->中途 + 中途->終點 的網絡最短路徑長度總和。
    """
    # 1. 計算 A 線實際折線距離 l_A
    l_A = sum(
        grid_gdf.centroid.loc[route[i-1]].distance(
            grid_gdf.centroid.loc[route[i]]
        ) for i in range(1, len(route))
    )
    # 2. 最近節點函式
    def nearest_node(p):
        return min(G.nodes, key=lambda n: Point(n).distance(p))
    # 3. 映射起點、中途、終點到圖上的節點
    u = nearest_node(start_pt)
    m = nearest_node(mid_pt)
    v = nearest_node(end_pt)
    # 4. 分段計算最短路徑長度
    l1 = nx.shortest_path_length(G, source=u, target=m, weight='length')
    l2 = nx.shortest_path_length(G, source=m, target=v, weight='length')
    l_sp_via = l1 + l2
```

```
    # 5. 回傳效率比 R_A
    return l_A / l_sp_via if l_sp_via > 0 else float('inf')
def compute_RO_A(route, grid_gdf, bus_routes_gdf, seg_len=50):
    pts = [grid_gdf.centroid.loc[i] for i in route]
    lnA = LineString([(p.x, p.y) for p in pts])
    lA  = lnA.length
    n   = max(int(lA // seg_len), 1)
    segs = [LineString([lnA.interpolate(i/n, normalized=True),
lnA.interpolate((i+1)/n, normalized=True)])
            for i in range(n)]
    results = {}
    for sub_id, grp in bus_routes_gdf.groupby("SubRouteNa"):
        geom_r = grp.unary_union if hasattr(grp, 'unary_union') else
grp.geometry.union_all()
        lr      = geom_r.length or 0.0
        overlap = sum(seg.length for seg in segs if geom_r.intersects(seg))
        results[sub_id] = (overlap / lr * 100) if lr > 0 else 0.0
    return results
def compute_RO_Ar(route, grid_gdf, bus_routes_gdf, seg_len=50):
    pts = [grid_gdf.centroid.loc[i] for i in route]
    lnA = LineString([(p.x, p.y) for p in pts])
    lA  = lnA.length or 0.0
    n   = max(int(lA // seg_len), 1)
    segs = [LineString([lnA.interpolate(i/n, normalized=True),
lnA.interpolate((i+1)/n, normalized=True)])
            for i in range(n)]
    results = {}
    for sub_id, grp in bus_routes_gdf.groupby("SubRouteNa"):
        geom_r = grp.unary_union if hasattr(grp, 'unary_union') else
grp.geometry.union_all()
        overlap = sum(seg.length for seg in segs if geom_r.intersects(seg))
        results[sub_id] = (overlap / lA * 100) if lA > 0 else 0.0
    return results
# === 執行評估 ===
sp_dqn = compute_SP_A(best_path, grid_gdf, households_gdf)
sp_bus = compute_SP_A(bus_route, grid_gdf, households_gdf)
```

```
spo_dqn = compute_SPO_A_zone(best_path, grid_gdf, households_gdf,
bus_routes_gdf, zones)
spo_bus = compute_SPO_A_zone(bus_route,    grid_gdf, households_gdf,
bus_routes_gdf, zones)
r_dqn = compute_R(best_path, grid_gdf, G, start_pt, mid_pt, end_pt)
r_bus = compute_R(bus_route,      grid_gdf, G, start_pt, mid_pt, end_pt)
ro_ar_dqn = compute_RO_A(best_path, grid_gdf, bus_routes_gdf)
ro_ar_bus = compute_RO_A(bus_route,      grid_gdf, bus_routes_gdf)
ro_ar_indiv_dqn = compute_RO_Ar(best_path, grid_gdf, bus_routes_gdf)
ro_ar_indiv_bus = compute_RO_Ar(bus_route,      grid_gdf, bus_routes_gdf)
overall_ro_ar_dqn = sum(ro_ar_dqn.values())/len(ro_ar_dqn) if ro_ar_dqn else 0.0
overall_ro_ar_bus = sum(ro_ar_bus.values())/len(ro_ar_bus) if ro_ar_bus else 0.0
# 合併所有子路線計算整體既有網絡 SPO_A
all_geom = bus_routes_gdf.geometry.union_all()
if isinstance(all_geom, LineString):
    merged = MultiLineString([all_geom])
elif isinstance(all_geom, MultiLineString):
    merged = all_geom
elif isinstance(all_geom, GeometryCollection):
    merged = MultiLineString([g for g in all_geom.geoms if isinstance(g,
LineString)])
else:
    raise ValueError(f"無法處理的幾何類型：{all_geom.geom_type}")
overall_route_gdf = gpd.GeoDataFrame({'geometry': [merged]},
crs=PROJECTED_CRS)
overall_bus_route = get_grid_path_from_route(overall_route_gdf, grid_gdf)
spo_overall = compute_SPO_A_zone(best_path, grid_gdf, households_gdf,
bus_routes_gdf, zones)
spo_overall_bus = compute_SPO_A_zone(bus_route, grid_gdf, households_gdf,
bus_routes_gdf, zones)
print("=== DQN 最佳路線 評估指標 ===")
print(f"SP_A                    : {sp_dqn}")
print(f"SPO_A (%)               : {spo_overall:.2f}%")
print(f"R_A                     : {r_dqn:.2f}")
print(f"RO_AR (平均%)      : {overall_ro_ar_dqn:.2f}%")
for sub_id, pct in ro_ar_dqn.items(): print(f"    {sub_id}: {pct:.2f}%")
print("RO_Ar (%) vs 各 SubRouteNa：")
```

```
for sub_id, pct in ro_ar_indiv_dqn.items(): print(f"    {sub_id}: {pct:.2f}%")
print("\n===  專家路線  評估指標  ===")
print(f"SP_A                    : {sp_bus}")
print(f"SPO_A (%)               : {spo_overall_bus:.2f}%")
print(f"R_A                     : {r_bus:.2f}")
print(f"RO_AR (平均%)       : {overall_ro_ar_bus:.2f}%")
for sub_id, pct in ro_ar_bus.items(): print(f"    {sub_id}: {pct:.2f}%")
print("RO_Ar (%) vs  各  SubRouteNa：")
for sub_id, pct in ro_ar_indiv_bus.items(): print(f"    {sub_id}: {pct:.2f}%")
import geopandas as gpd
from shapely.geometry import LineString, MultiLineString
import pandas as pd
#  讀里界並計算家戶數（放在最前面，只要執行一次）
zones = zones = gpd.read_file(r"C:/Users/willi/Desktop/南投市公車站點 PY/專題大
三下/村(里)界(TWD97_121 分
帶)1130807/VILLAGE_NLSC_121_1130807.shp").to_crs(PROJECTED_CRS)
hh_zone = gpd.sjoin(
    households_gdf[['geometry']],
    zones[['VILLNAME','geometry']],
    how='left',
    predicate='within'
)
pop = hh_zone.groupby('VILLNAME').size().rename('population')
zones = zones.merge(pop, on='VILLNAME')
# 1.  定義  compute_zone_cov（如前）
def compute_zone_cov(route_geoms, zones_gdf, households_gdf, buffer_dist=300):
    #  合併成  MultiLineString
    if isinstance(route_geoms, (LineString, MultiLineString)):
        merged = route_geoms
    else:
        merged = MultiLineString(route_geoms)
    buf = merged.buffer(buffer_dist)
    #  家戶是否被覆蓋
    hh = households_gdf[['geometry']].copy()
    hh['in_cov'] = hh.geometry.intersects(buf)
    #  空間連結到里
    hh_zone = gpd.sjoin(
```

```
            hh.set_geometry('geometry'),
            zones_gdf[['VILLNAME','geometry']],
            how='left',
            predicate='within'
        )
        covered =
hh_zone[hh_zone['in_cov']].groupby('VILLNAME').size().rename('covered')
        pop = zones_gdf.set_index('VILLNAME')['population']
        df = pd.concat([pop, covered], axis=1).fillna(0)
        df['covered'] = df['covered'].astype(int)
        df['coverage_rate'] = df['covered'] / df['population'] * 100.0
        return df.reset_index()

# 2. 準備既有網路 geometry（改用 union_all）
all_geom = bus_routes_gdf.geometry.union_all()
exist_geom = (
        all_geom
        if isinstance(all_geom, (LineString, MultiLineString))
        else MultiLineString([g for g in all_geom.geoms if g.geom_type=='LineString'])
)
# 3. DQN 路線 LineString
pts_dqn = [grid_gdf.centroid.loc[i] for i in best_path]
ln_dqn = LineString([(p.x, p.y) for p in pts_dqn])
dqn_plus_exist = (
        MultiLineString([*exist_geom.geoms, ln_dqn])
        if isinstance(exist_geom, MultiLineString)
        else MultiLineString([exist_geom, ln_dqn])
)
# 4. 專家路線 LineString
pts_bus = [grid_gdf.centroid.loc[i] for i in bus_route]
ln_bus = LineString([(p.x, p.y) for p in pts_bus])
bus_plus_exist = (
        MultiLineString([*exist_geom.geoms, ln_bus])
        if isinstance(exist_geom, MultiLineString)
        else MultiLineString([exist_geom, ln_bus])
)
# 5. 呼叫函式
```

```
cov_exist = compute_zone_cov(exist_geom, zones, households_gdf)
cov_dqn   = compute_zone_cov(dqn_plus_exist, zones, households_gdf)
cov_bus   = compute_zone_cov(bus_plus_exist, zones, households_gdf)
# 6. 合併比較
comparison = (
    cov_exist[['VILLNAME','coverage_rate']]
    .merge(cov_dqn[['VILLNAME','coverage_rate']], on='VILLNAME',
suffixes=('_exist','_dqn'))
    .merge(cov_bus[['VILLNAME','coverage_rate']], on='VILLNAME')
    .rename(columns={'coverage_rate':'coverage_rate_bus'})
)
comparison['delta_dqn'] = comparison['coverage_rate_dqn'] -
comparison['coverage_rate_exist']
comparison['delta_bus'] = comparison['coverage_rate_bus'] -
comparison['coverage_rate_exist']
comparison.rename(columns={
    'VILLNAME':                里名',
    'coverage_rate_exist':     既有公車原始涵蓋率',
    'delta_dqn':               DQN 路線提升(%)',
    'delta_bus':               專家路線提升(%)',
    'coverage_rate_dqn':       加設 DQN 路線後涵蓋率',
    'coverage_rate_bus':       加設專家路線後涵蓋率'
}, inplace=True)
print(comparison)
import pandas as pd
# 調整 pandas 顯示設定，不折疊欄位
pd.set_option("display.max_columns", None)
pd.set_option("display.width", None)
# 把 cov_* 表的 covered 和 coverage_rate 改成你要的欄位名稱
cov_exist = cov_exist.rename(columns={
    "covered": "count_exist",
    "coverage_rate": "coverage_rate_exist"
})
cov_dqn = cov_dqn.rename(columns={
    "covered": "count_dqn",
    "coverage_rate": "coverage_rate_dqn"
})
```

```
cov_bus = cov_bus.rename(columns={
    "covered": "count_bus",
    "coverage_rate": "coverage_rate_bus"
})
# 合併所有欄位：里名、覆蓋家戶數、覆蓋率
comparison = (
    cov_exist[["VILLNAME", "count_exist", "coverage_rate_exist"]]
    .merge(cov_dqn[["VILLNAME", "count_dqn", "coverage_rate_dqn"]],
on="VILLNAME")
    .merge(cov_bus[["VILLNAME", "count_bus", "coverage_rate_bus"]],
on="VILLNAME")
)
# 計算增量（百分點差異）
comparison["delta_dqn"] = comparison["coverage_rate_dqn"] -
comparison["coverage_rate_exist"]
comparison["delta_bus"] = comparison["coverage_rate_bus"] -
comparison["coverage_rate_exist"]
# 把欄位改成中文
comparison = comparison.rename(columns={
    "VILLNAME": "里名",
    "count_exist": "既有覆蓋家戶數",
    "coverage_rate_exist": "既有公車原始涵蓋率",
    "count_dqn": "DQN 後覆蓋家戶數",
    "coverage_rate_dqn": "加設 DQN 路線後涵蓋率",
    "count_bus": "專家後覆蓋家戶數",
    "coverage_rate_bus": "加設專家路線後涵蓋率",
    "delta_dqn": "DQN 路線提升(%)",
    "delta_bus": "專家路線提升(%)"
})
# buffer 距離
tol = 10
# 對既有路網做 buffer
exist_buf = exist_geom.buffer(tol)
for name, route in [("DQN", ln_dqn), ("專家", ln_bus)]:
    overlap_line = route.intersection(exist_buf)
    overlap_len = overlap_line.length
    route_len   = route.length
```

```
ro_ar = overlap_len / route_len * 100
print(f"RO_AR ({name} 路線, buffer={tol}m) : {ro_ar:.2f}%")
print(f"   {name} 路線全長          : {route_len:.1f} m")
print(f"   跟既有路網 {tol}m 內重疊長度 : {overlap_len:.1f} m\n")
```

全文完