

中華民國第 65 屆中小學科學展覽會

作品說明書

國中組 生活與應用科學科(一)

032803

運用生成式 AI 及 UAV 影像自動檢測道路裂縫
之可行性研究

學校名稱：臺中市私立明道高級中學(附設國中)

| | |
|---------------------------------|------------------|
| 作者： 國三 李羽弘 國一 鄭安捷 | 指導老師： 陳郁弦 |
|---------------------------------|------------------|

關鍵詞：生成式 AI、道路裂縫(Road crack)、鋪面狀況指數值(PCI)

摘要

本研究結合生成式 AI 與 UAV 技術，利用 Google Colab 雲端平台，自動檢測柔性鋪面裂縫並評估鋪面狀況指數 (PCI) 的可行性。團隊採用 ResNet152 預訓練模型進行遷移學習，並透過滑動窗口技術自動定位與分類裂縫。驗證結果顯示，模型在 15 張現場影像上的 F1 Score 達 0.815，具備穩定且優異的辨識能力。

團隊基於內政部營建署《柔性鋪面損壞調查手冊》標準，開發程式自動計算 PCI 值並分級評估路況。研究證實，本系統可低成本、快速且準確地替代傳統目視檢測，提供道路維護的自動化評估工具。此外，研究亦展現生成式 AI 在程式開發中的輔助效益，為未來 AI 道路檢測技術奠定基礎。

壹、前言

一、研究動機

台灣的道路品質長久以來讓人詬病不已，每次搭車被坑洞震得七葷八素時，心中總忍不住抱怨：「難道政府都不管嗎？」然而，查閱資料後發現，政府單位其實有在進行道路巡查，道路檢查員會依據道路破損狀況回報主管單位進行後續處理。但問題在於，現行的巡查方式；也就是傳統的目視檢測；存在諸多缺陷，不僅效率低下，還潛藏危機，這讓我們不禁思考：是否該尋找更聰明、更安全的替代方案？

根據內政部營建署的「柔性鋪面損壞調查手冊」，柔性鋪面(即瀝青混凝土鋪築的道路)的損壞調查分為兩大類：傳統調查法與改良式調查法。傳統調查法仰賴調查員步行或乘車，透過肉眼觀察道路破壞的類型、數量及嚴重程度並記錄下來。然而，這種方式的弊端顯而易見：

- (一) **漏檢危機**：調查員在車輛行駛中目視檢測，稍有不慎便可能因速度過快或視線受阻而錯過破損處。這些被漏掉的坑洞，輕則讓機車騎士摔得鼻青臉腫，重則引發國賠糾紛，讓政府單位背負沉重的心理與財政壓力。
- (二) **安全風險**：調查員在繁忙路段作業時，必須一邊注意來車、一邊觀察路面，稍有分神便可能危及自身安全，這不僅增加工作風險，也影響數據的可靠性。
- (三) **人力短缺**：許多基層單位因人力不足，無法定期巡檢，導致道路維護工作斷斷續續，甚至有名無實。

這些問題讓我們意識到，傳統目視檢測已無法滿足現代需求，於是團隊將目光轉向改良式調查法中的影像儀器技術。鑒於這二年無人飛行載具(Unmanned Aerial Vehicle, 以下簡稱 UAV)被大量使用在俄烏戰爭的戰場上，讓我們覺得 UAV 的運用是未來一個極具潛力的工具。另一方面 AI 技術的興起，改變了大家的生活形式，讓我們有將二者做一結合運用的想法。

所以團隊打算使用 UAV 快速拍攝大範圍道路影像，並上傳雲端，搭配我們訓練的深度學習模型，自動檢測並計算鋪面狀況指數值(Pavement Condition Index, 以下簡稱 PCI)。

這項技術究竟有何優勢，讓我們認為它能取代傳統方法？以下是我們的分析：

- (一) 成本低廉：UAV 設備價格親民且技術穩定，政府單位可輕鬆配發給基層人員，無需動輒數百萬的設備投資。
- (二) 高效覆蓋：透過預設飛行路徑，UAV 能在短時間內檢查廣大路段，大幅減少人力與時間成本，讓巡檢不再是苦差事。
- (三) 精準測量：傳統上，若將攝影設備架在汽車或機車上，需繁瑣校正才能確保拍攝面積準確，而 UAV 只需定高飛行，就能輕鬆提供精確的面積數據，為 PCI 值計算奠定基礎。
- (四) 自動化檢測：UAV 拍攝的影像上傳雲端後，深度學習模型能自動辨識裂縫等劣化現象，徹底杜絕漏檢風險。調查員不再需要耗時費力地人工判讀，既省時又省錢。

當然，深度學習模型的強大，離不開豐富的數據支持。柔性鋪面的劣化型態五花八門，從裂縫、車轍到坑洞、剝落，每一種都可能影響道路安全。為了驗證此方法的可行性，本研究先聚焦於最常見的三種面層裂縫，作為技術應用的起點。我們希望透過這項研究，證明 UAV 搭配 AI 能徹底革新道路巡查，為台灣的道路品質帶來實質改善。

二、研究目的

本研究旨在結合先進的 AI 技術與 UAV 應用，打造一套自動化、智能化的柔性鋪面裂縫檢測系統，取代傳統低效且主觀的目視檢測方法。具體而言，我們將聚焦於以下三個核心目標：

(一) 打造高效 AI 裂縫辨識模型

訓練並優化深度學習模型，使其能夠精準辨識柔性鋪面上的裂縫，並以測試集準確性(Test Accuracy)、與 UAV 拍攝柔性鋪面的裂縫辨識 F1 Score 等效能指標進行嚴格驗證，確保模型在多樣化真實場景中的穩定性與高準確性，為後續自動化檢測奠定技術基礎。

（二）實現雲端自動裂縫偵測

利用訓練完成的模型，在雲端環境下快速且準確地從 UAV 拍攝的大量柔性鋪面照片中自動標記裂縫位置。同時，進一步整合滑動窗口技術，提升裂縫偵測的完整性，確保檢測過程無死角、無遺漏，實現高效且全面的自動化分析。

（三）自動化 PCI 值評估與決策支援

自動計算照片中裂縫的面積比例，並依據內政部「柔性鋪面損壞調查手冊」標準，精確評估鋪面狀況指數(PCI)值。最終，將即時分析結果提供給調查人員，作為道路維護決策的科學依據，提升維護管理的效率與可靠性。

三、文獻回顧

為了選擇適合 UAV 拍攝柔性鋪面影像的裂縫辨識方法，我們借助 Google NotebookLM 整理並分析三篇關鍵研究，探討不同深度學習模型在裂縫檢測上的應用、優劣與對本研究的啟發。以下將分段呈現各篇研究的內容與評估，並明確說明本研究最終採用的方法及其理由。

Özgenel(2018)比較了七種預訓練的 CNN 模型(AlexNet、VGG16、VGG19、GoogleNet、ResNet50、ResNet101 和 ResNet152)在建築物裂縫檢測中的效能，並探討了訓練數據集大小、訓練週期數、卷積層數和可學習參數數量對效能的影響。研究結果表明，當訓練數據和測試數據相似時，即使使用有限的訓練樣本，預先訓練的網路也能夠通過微調實現高精度裂縫分類。然而當測試數據與訓練數據存在差異時，增加數據集大小可能會導致過度擬合。此外，具有 16 到 22 個卷積層的網路(VGG16、VGG19 和 GoogleNet)表現最佳，而層數過少(AlexNet)或過多(ResNet 網路)都可能導致效能下降。具有分層連接和多個全連接層的網路在不同條件下表現更佳，並有望實現通用的裂縫檢測框架，不受材料類型的限制。



圖 1：使用 Google NotebookLM 協助整理文獻重點 (由所有作者共同實作並截圖)

章皓鈞(2021)研究以 Mask R-CNN 深度學習，搭配 Python 與 OpenCV，開發道路破損檢測系統，能辨識龜裂、裂縫、補綻、變形及坑洞，部分類別召回率達 86% 以上，精確率超過 82%。此外，研究計算破損面積，提供道路維修量化指標，輔助人力巡查。

本研究為國內少數應用深度學習於道路檢測的碩士論文，採「物件偵測」技術評估破損範圍。然而，該方法在裂縫偵測易重複框選，計算量大且耗時。此外，透過車輛拍攝鋪面影像需繁瑣校正，應用於 PCI 值評估仍有技術挑戰待克服。

蔡佳勳(2023)在 Colab 訓練 28 組深度學習圖像分類模型以辨識混凝土裂縫，其中 ResNet152 搭配 Adagrad 優化器表現最佳，平均 F1 Score 達 0.957，準確率高達 99.5%。該研究提供一種經濟且快速的混凝土裂縫檢測方法，透過模型與優化函數選擇，實現裂縫分類與定位。團隊討論後認為該文獻雖然是使用較為簡易的「圖像分類」的深度學習模型來辨識混凝土裂縫，但只要能準確、快速識別裂縫的存在與位置，其實該文獻的方法也是能運用在柔性鋪面的裂縫辨識上。且該文獻使用圖像分類模型進行裂縫的分類和以滑動視窗進行裂縫定位，與我們研究目的「裂縫辨識並進行 PCI 值計算」相符，且運用雲端運算的特性也符合團隊需求，因此我們將採用該文獻的理念來開發系統。

為了簡化研究流程，我們在假設道路裂縫的特徵也跟混凝土相似的前提下，所以採用該論文研究的最佳模型(ResNet152)與其最佳優化函數(Adagrad)設定來訓練我們的模型。

貳、研究設備與器材

本研究使用以下設備與工具來完成 UAV 影像蒐集、深度學習模型訓練及 PCI 值計算：

- 一、UAV：採用大疆所開發之 Mavic Pro 2。
- 二、ChatGPT：是 OpenAI 開發的人工智慧聊天機器人程式，於 2022 年 12 月推出。是一種大型語言模型並以強化學習訓練。所以生成式 AI 技術的成熟，讓我們有能力以**提示工程 (Prompt Engineering)**的方式，在 ChatGPT 的協助下開發深度學習模型與 PCI 值相關計算程式。
- 三、Google NotebookLM：Google 實驗室推出的一款在線筆記本，NotebookLM 內置 Gemini，它可以根據用戶上傳的內容生成摘要、註解和用戶想要的答案。
- 四、Google Colaboratory：作為主要的模型訓練與程式開發環境，Colab 支援 GPU 運算，適用於深度學習模型訓練與推論。

參、研究過程與方法

一、ResNet152 特徵萃取模型訓練

(一) 訂航高

以Mavic Pro 2鏡頭以垂直路面的方式拍攝照片，將UAV的拍攝航高訂為6m，並以此高度採集了柔性鋪面照片並作為後續進行PCI值評估的基準高度。而訂為6m的理由如下：

- 1、以UAV飛行時所顯示之離地高度，可知馬路的紅綠燈與橫跨馬路的電線高度一般為5m左右，因此將UAV之拍攝航高訂為6m，則能躲避道路上大部分的障礙物(如圖2左為UAV相機鏡頭平行拍攝)。
- 2、以6m高度所拍攝之道路照片，剛好可涵蓋二個車道(如圖2右為UAV相機鏡頭垂直拍攝)，這種剛好涵蓋二個車道的拍攝高度，十分適合飛行路徑的規劃，可迅速的、大範圍的進行道路檢測工作。

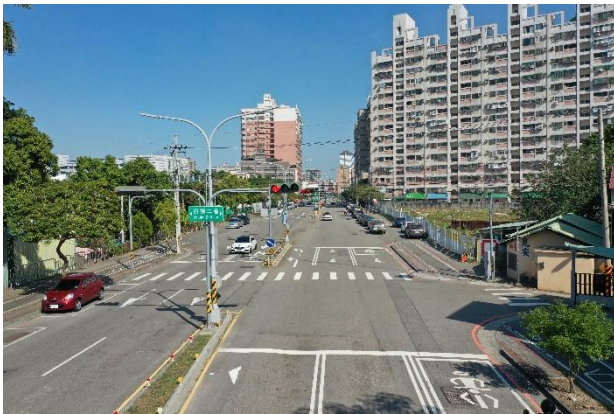


圖 2：6m 航高之平行拍攝與垂直拍攝 (由第二作者拍攝)

(二) 數據集製作

我們將UAV所拍攝柔性鋪面的照片後裁減成 224×224 像素大小的小切片，並去除了班馬線、車道線與人孔蓋等項目，僅針對柔性鋪面裂縫照片與純瀝青照片來製作訓練所需要的數據集。

製作數據集需要對柔性鋪面照片裁減成 224×224 像素大小的小切片，因此要撰寫一個小程序來執行。由於本研究的程式皆是採 ChatGPT 生成，因此簡述一下 ChatGPT 使用提示詞(Prompt)與提示工程(Prompt Engineering)之差異。

- 1、提示詞(Prompt)：僅以單純的提問或指令，缺少背景上下文、角色設定、呈現格式說明，也沒明確列出效能或相依套件等約束，通常輸出品質較不穩定，圖3為切片程式以提示詞

生成之範例。

請寫一個程式，對於路徑'/content/drive/MyDrive/Road'資料夾裡面
瀝遍所有的.JPG檔案，將每張照片逐一滑窗，裁成224*224的小切片，
將小切片存到與照片同名的子資料夾

圖 3：切片程式以提示詞生成(由第二作者截圖)

2、提示工程(Prompt Engineering)：為了能讓 ChatGPT 更精準、穩定地輸出符合預期的程式碼，可以使用提示工程完整地框定了程式碼預期的行為範圍，能有效提升生成程式碼的穩定性、可重現性和符合度。團隊採用「上下文(Context) /扮演角色(Role) /具體任務(Task) /呈現方式(Presentation) /限制條件(Constraints)」五個要素來滿足需求，圖4為切片程式以提示工程生成之範例。

Context (上下文)

你是一位深度學習工程師助理，專精於 TensorFlow / Keras 以及雲端 Notebook (例如 Google Colab) 。目前需要對模型輸出的預測圖做切片、整理，以方便後續分析。

Role (角色)

你將扮演我的技術顧問與程式碼生成器。

Task (任務 / Goal)

請幫我生成一段可在 Google Colab 上直接執行的 Python 程式碼，用於：

走訪路徑 '/content/drive/MyDrive/Road ' 中所有 .JPG 檔案

對每張照片以 224×224 的框做滑動切片 (步長同樣設為 224)

為每張原圖建立一個同名子資料夾，並將所有切片依序儲存至該子資料夾

Presentation (呈現 / Way)

以完整的 Colab code cell 格式呈現 (包含必要的 import)

每段 code cell 上方均加註解說明

Constraints (限制條件，Optional)

僅依賴 Colab 預先安裝之 Python 標準函式庫及 opencv-python
支援處理大量影像 (可容錯已存在的子資料夾)

圖 4：切片程式以 ChatGPT 提示工程生成(由第二作者截圖)

為了程式碼能符合我們的需求，之後所有對於 ChatGPT 提問皆是採用提示工程並以如下的五要素來生成：

1、上下文(Context)：簡要說明使用場景或前提。

- 2、扮演角色(Role)：明確 ChatGPT 扮演的身份。
- 3、具體任務(Task)：具體要 ChatGPT 做到什麼。
- 4、呈現方式(Presentation)：希望輸出的格式、風格或語氣。
- 5、限制條件(Constraints)：可選，用於加入額外限制，如版本、時間、資源等。

之後我們將切成 224×224 像素大小的小切片進行分類。我們將之分類為正類(Positive)；即為柔性鋪面裂縫照片，反類(Negative)；即為純瀝青照片。並以以下的數量製作成數據集：

- 1、訓練集(train data)：正類、反類各3200張，共6400張。
- 2、驗證集(validation data)：正類、反類各640張，共1280張。
- 3、測試集(test data)：正類、反類各640張，共1280張。

(三) ResNet152特徵萃取模型訓練

1、ResNet152模型概述

ResNet(殘差神經網)由微軟研究院開發，通過引入「殘差塊(Residual Block)」解決深層神經網路中的過擬合和梯度消失問題。殘差塊允許網路跳躍式傳遞訊息，將輸出值 $H(X)$ 表示為 $F(X) + X$ ，其中 $F(X)$ 是需學習的殘差部分(如圖 5)。

ResNet152 模型深度達 152 層，是 2015 年 ILSVRC 大賽的冠軍模型，Top-5 錯誤率僅 3.57%，低於人類的 5.1%。其優異表現使其成為圖像辨識任務的理想選擇。

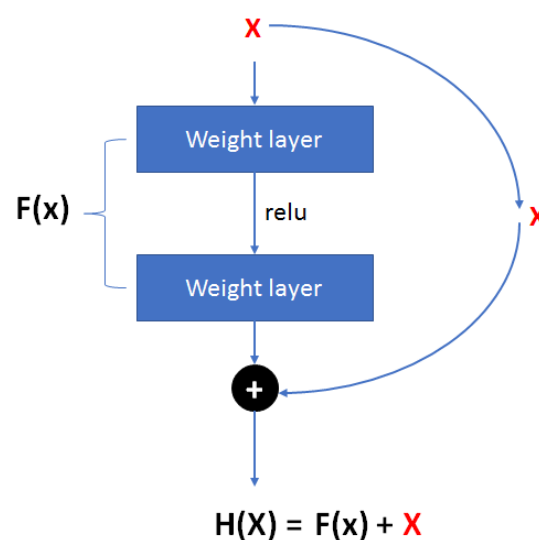


圖 5：殘差原理(由所有作者共同繪製並截圖)

2、模型訓練策略

為快速適應少量數據集，本研究使用預訓練網路(Pretrained Network)進行遷移學習；也就是特徵萃取技術。將 ResNet152 的卷積基底(Convolution Base)凍結，保留預訓練權重(基於 ImageNet 數據集)，並以隨機初始化方式建立新的分類器，適應柔性鋪面裂縫檢測任務(如圖 6)。

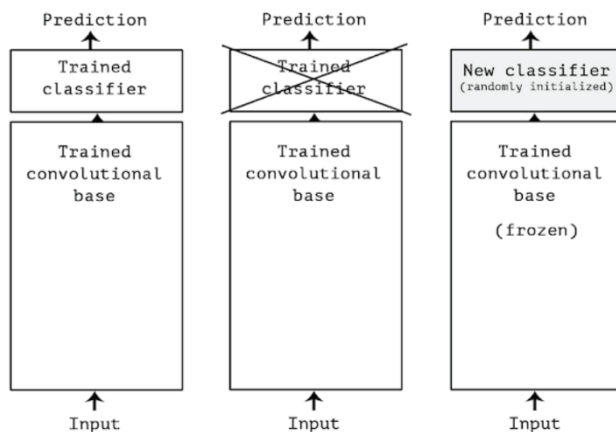


圖 6：模型遷移學習特徵萃取示意圖(由所有作者共同繪製截圖)

3、程式開發過程

程式開發參考 GitHub 上的所搭建的 Resnet50 程式，程式來自「**Detection of Surface Cracks in Concrete Structures using Deep Learning**」一文連結中，程式後半段並附有滑窗功能檢測混凝土裂縫(如圖 7 所示)。

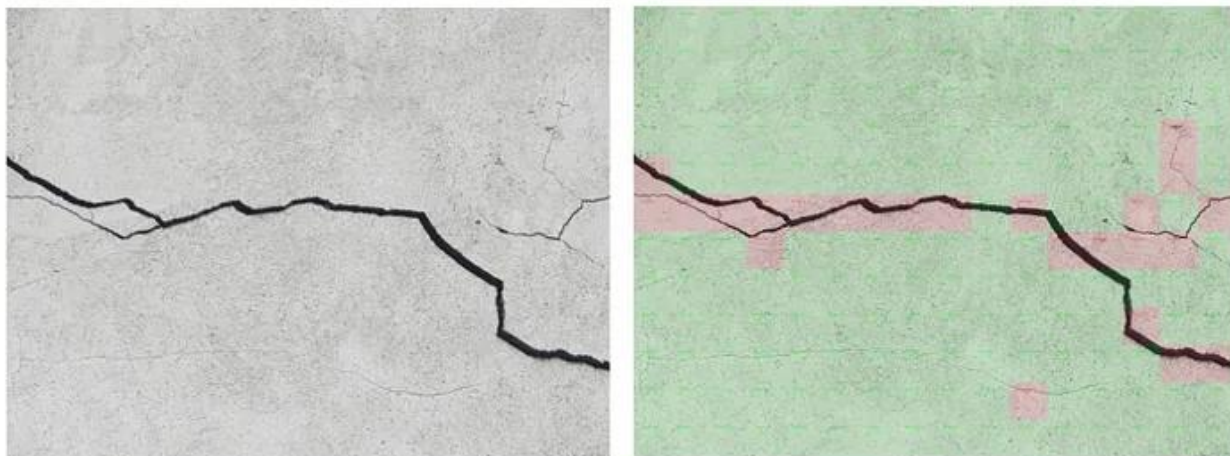


圖 7：網路 Resnet50 程式之滑窗結果(由所有作者共同截圖)

模型的程式撰寫是藉由生成式 AI 的輔助，我們以交談式的 ChatGPT 依以下的順序生成模型程式：

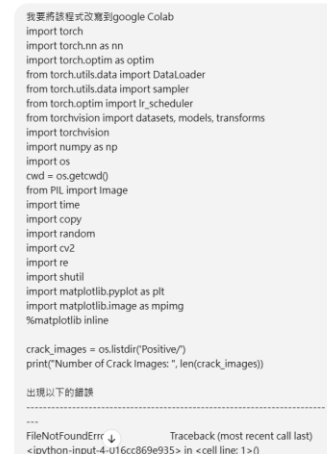
- (1) 傳ResNet50程式至ChatGPT，ChatGPT提示工程格式理解其架構與功能(如圖8 A)。

- (2) 將程式移植至Google Colab，調整適配雲端環境(如圖8 B)。
- (3) 執行程式時遇到錯誤，將錯誤訊息提交ChatGPT進行偵錯，反覆修正後成功運行(如圖8 C、D所示)。

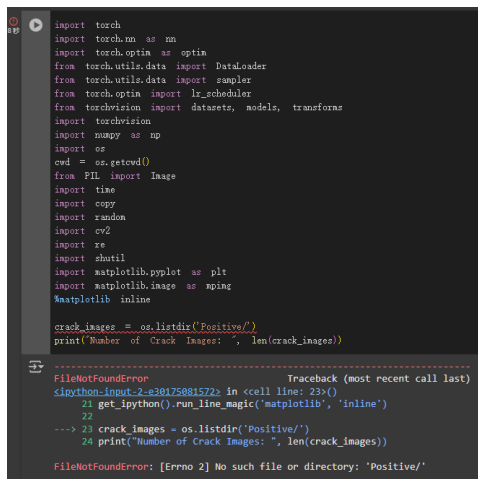
A、ChatGPT 解釋程式架構與功能



B、ChatGPT 改寫程式至 Google Colab



C、執行程式段落遇到錯誤



D、請 ChatGPT 就錯誤的部分進行改寫



圖 8：利用 ChatGPT 協助程式偵錯(由第一作者整理並截圖)

在老師的指導下，我們利用 ChatGPT 生成程式後，我們將程式的 Resnet50 預訓練模型以 Resnet152 來取代，即可完成我們模型訓練的程式碼。

4、訓練參數設置

參數設置參考蔡佳勳(2023)研究，詳情如下：

- (1) 優化器：採用Adagrad優化器，適合ResNet152模型，能根據參數更新頻率調整學習率。

(2) 批次大小(Batch Size)：為每批處理的樣本個數批次大小，設為32，平衡訓練效率與記憶體需求。

(3) 訓練週期(Epoch)：初始設定200 Epoch，考量道路裂縫形態複雜，另訓練500 Epoch 模型進行對比。

二、模型訓練成效評估

對於模型的訓練成效我們採用以下的二種方法來交叉評估模型：

(一) 對於數據集中的測試集資料夾進行測試：做法是在開發完前述 Resnet 程式移植後，加一個提示工程，要 ChatGPT 對特定路徑的資料夾進行測試。並將最後的 test acc 與 test loss 輸出到一個 csv 檔。

(二) 採用實際柔性鋪面裂縫照片來分析模型的實際訓練成果。將滑窗後的照片以前面所撰寫的程式對照片進行切片，再以人工對切片進行分類。

滑動窗口的檢測結果採用二元混淆矩陣 (Confusion Matrix)來當成效指標，二元混淆矩陣如圖 9 所示，其相對應驗證指標說明如下：

- 1、 True Positive (TP)「真陽性」：真實情況是切片區域「有裂縫」，模型滑窗結果「有裂縫」的數量。
- 2、 True Negative(TN)「真陰性」：真實情況是切片區域「沒有裂縫」，模型滑窗結果「沒有裂縫」的數量。
- 3、 False Positive (FP)「偽陽性」：真實情況是切片區域「沒有裂縫」，模型滑窗結果「有裂縫」的數量。
- 4、 False Negative(FN)「偽陰性」：真實情況是切片區域「有裂縫」，模型滑窗結果「沒有裂縫」的數量。
- 5、召回率(Recall)：樣本中有裂縫的數量有多少被預測準確，衡量的是召回率，預測對的正例數佔真正的正例數的比率：

$$\text{Recall} = \frac{\text{TP}}{(\text{TP}+\text{FN})}$$

- 6、精準率(Precision)：針對預測結果而言，預測為有裂縫的數量有多少是真正的有裂縫樣本，衡量的是查準率，預測正確的正例數佔預測為正例總量的比率：

$$\text{Precision} = \frac{\text{TP}}{(\text{TP}+\text{FP})}$$

- 7、F1 Score：是精確度和召回率的調和平均值。

$$F1\ Score = \frac{2(Precision \times Recall)}{Precision + Recall}$$

最後統計模型對實際照片滑窗的結果，得到 TP、TN、FP 與 FN 四個數值後，計算得到召回率(Recall)、精準率(Precision)與 F1 Score。


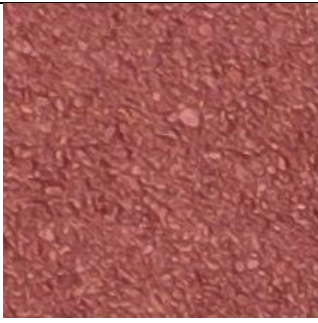


| | | Actual Values(真實值) | |
|------------------------|----------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| | | Positive | Negative |
| Predicted Values (預測值) | Positive |  |  |
| | | True Positive(TP) | False Positive(FP) |
| Negative | |  |  |
| | | False Negative(FN) | True Negative(TN) |

圖 9：二元混淆矩陣 Confusion Matrix(由所有作者共同繪製)

三、模型滑動窗口開發

(一) 照片調整

團隊想利用撰寫的程式載入前一步驟所訓練出來的模型，程式中會產生一個像素大小 224×224 滑窗逐一掃過照片，藉此判定照片中是否有裂縫存在。但 UAV 所拍攝照片的大小不一定是 224×224 的滑窗可以整除的，為能讓滑窗無遺漏的檢視整張相片有無裂縫，必須要對 UAV 所拍攝的照片作處理。本研究採用調整大小(Resize)法，將照片像素調整到可被滑窗可以整除的大小，因為我們之後做的 PCI 值評估是採用密度(%)，其定義就是裂縫在整張照

片的百分比，因此這種調整大小(Resize)的方法是不會對評估結果造成影響的。其程式開發邏輯如下：

- 1、在設定的資料夾裡讀取照片大小。
- 2、依據前述的照片大小找到最近224整除的大小函數。
- 3、計算新的大小，讓寬度和高度可以被224整除。
- 4、放大圖片並儲存到新的資料夾中。

此部分程式一樣採用 ChatGPT 提示工程協助生成，提示工程產生 Resize 程式過程詳如圖 10 所示。

A、Chatgpt 提示工程

Context (上下文) 你是一位深度學習工程師，專精於 TensorFlow / Keras 以及雲端 Notebook (例如 Google Colab)。

Role：你將扮演我的技術顧問。

Task：

Resize 程式

- 1、讀取google雲端硬碟路徑 "/content/drive/MyDrive/Road" 裡面所有的照片*.jpg
- 2、創建一個Road Resize資料夾。
- 3、將該照片放大為224大小可以整除的，並存到google雲端硬碟路徑 "/content/drive/MyDrive/Road Resize" 資料夾resize後的照片名稱跟原照片一樣。

Presentation (呈現 / Way)

舉例 如原始照片大小5472*3648，則resize後的照片為5600*3800，該照片是可以被224整除的

Constraints：請以你的專業腳色，幫我將上段程式碼改寫到colab執行。

B、Chatgpt 產生 Resize 程式

```
1 # Colab: Resize all JPG images in Google Drive so dimensions are multiples of 224
2
3 # 1. 掛載 Google 雲端硬碟
4 from google.colab import drive
5 import os
6 import cv2
7
8 drive.mount('/content/drive')
9
10 # 2. 定義原始和目標資料夾路徑
11 input_dir = '/content/drive/MyDrive/Road'
12 output_dir = '/content/drive/MyDrive/Road Resize'
13
14 # 3. 如果目標資料夾不存在，創建之
```

在聊天中回答 >

我已在 Colab 上提供一段完整的 Python 程式碼，會自動：

1. 掛載 Google 雲端硬碟
2. 在 /content/drive/MyDrive/Road 中掃描所有 *.jpg
3. 計算每張影像的最小放大尺寸 (向上取至能被 224 整除)
4. 使用 OpenCV 放大並保持長寬比
5. 儲存到 /content/drive/MyDrive/Road Resize，檔名不變

你只需將此程式碼貼到 Colab 執行，即可完成批次 resize。若有任何疑問或進一步需求，請隨時告訴我！

圖 10：ChatGPT 輸入提示工程產生 Resize 程式(由第一作者生成並截圖)

(二) 模型滑動窗口開發

前述網路上的程式包含模型訓練與滑動視窗二大部份，為了後續的研究便捷性，團隊將 Resize 程式與 Github 上的滑窗部分結合。此部份程式開發邏輯如下：

- 1、依224×224 窗口大小與步伐逐一滑過整張照片，讓模型判別該窗口是正類亦或是負類。
- 2、判定為負類之區域不進行任何處理，判別為正類的窗口，以紅色的透明方形在照片上標示出來，並將正類切片儲存到與原照片同名的資料夾。

ChatGPT 以提示工程協助生成程式過程詳如圖11所示，滑動窗口程式流程圖詳如圖12所示。

Context (上下文) 你是一位深度學習工程師，專精於 TensorFlow / Keras 以及雲端 Notebook (例如 Google Colab)。

Role: 你將扮演我的技術顧問。

Task: 前面的深度學習 resnet 模型，主要訓練模型來辨識裂縫。以下的程式碼，則是用滑窗來將裂縫找出來

10. 預測函式 (滑動窗口)

```
def predict(model, test_image, print_class = False):

    transform = chosen_transforms['val']

    test_image_tensor = transform(test_image)

    if torch.cuda.is_available():
        test_image_tensor = test_image_tensor.view(1, 3, 227, 227).cuda()
```

請以你的專業腳色，幫我將上段程式碼改寫到 colab 執行。

Presentation (呈現 / Way)

- 1、路徑/content/drive/MyDrive/65/Road PCI Resize，裡面所有的照片*.JPG進行滑窗
- 2、滑窗窗口大小為224，步長也是224
- 3、判定為Positive則照原本紅色透明反白，判定為Negative則不用任何處理。
- 4、判定為Positive的區塊則切取出來，儲存到跟原本照片同名的子資料夾
- 5、滑窗處理後的照片與子資料夾全部存到模型儲存的路徑，資料夾的名稱與路徑為/content/drive/MyDrive/65/resnet25Epoch/predictions

Constraints: 用原有的程式碼下去修正。

圖 11：ChatGPT 輸入提示工程協助生成滑動窗口程式(由第一作者生成並截圖)

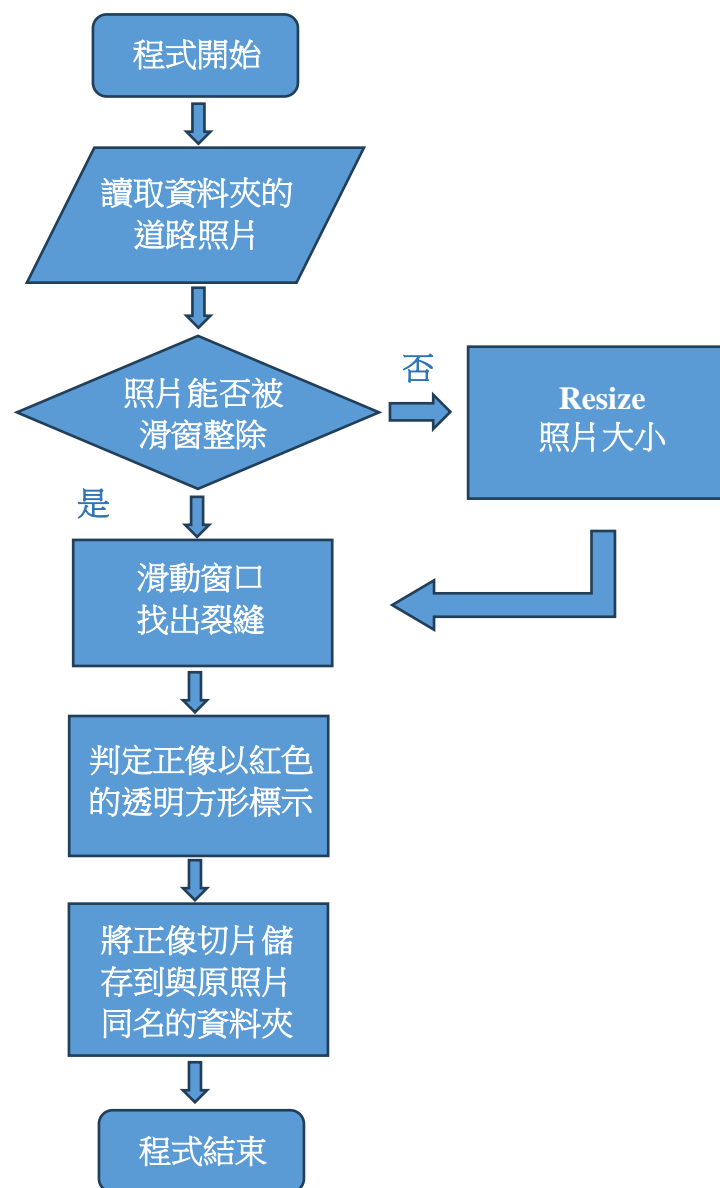


圖 12：滑動窗口程式流程圖(由第一作者製作)

四、裂縫自動 PCI 值計算

以上述所訓練出來的最佳模型來進行柔性鋪面的裂縫辨識並計算 PCI 值。在開發裂縫自動 PCI 值計算程式之前，先說明內政部「柔性鋪面損壞調查手冊」(2002)中所載之市區道路柔性鋪面損壞調查表(如下表 1)。

表 1：市區道路柔性鋪面損壞調查表(由第二作者製作並截圖)

| 市區道路柔性鋪面損壞調查表 | | | | | | | | | 草圖： | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----|---|--|--|--|--|--|-----|--------|------|
| 區編碼：文心路 段編碼：四段 | | | | | | | | | | | |
| 樣本單位編碼：01 ，面積：300m ² | | | | | | | | | | | |
| 調查人員：林世泰 | | | | | | | | | | | |
| 調查日期：民國 90 年 10 月 15 日 | | | | | | | | | | | |
| <p style="text-align: center;">損壞種類及型式</p> <div style="display: flex; flex-wrap: wrap;"> <div style="width: 33%;">1.龜裂</div> <div style="width: 33%;">6.補綻及管線回填</div> <div style="width: 33%;">10.波浪狀鋪面</div> <div style="width: 33%;">2.縱向及橫向裂縫</div> <div style="width: 33%;">7.推擠</div> <div style="width: 33%;">11.車道與路肩分離</div> <div style="width: 33%;">3.塊狀裂縫</div> <div style="width: 33%;">8.隆起與凹陷</div> <div style="width: 33%;">12.滑溜裂縫</div> <div style="width: 33%;">4.坑洞及人孔高差與薄層剝離</div> <div style="width: 33%;">9.冒油</div> <div style="width: 33%;">13.骨材剝落</div> <div style="width: 33%;">5.車轍</div> </div> <p style="text-align: right;">嚴重程度 輕級：L、中級：M、重級：H</p> | | | | | | | | | | | |
| 損壞類型及嚴重程度 | 數 量 | | | | | | | | 總數 | 密度 (%) | 折減值 |
| 1M | 6 | 9 | | | | | | | 15 | 5.0 | 39.0 |
| 1L | 10 | 12 | | | | | | | 22 | 7.3 | 30.0 |
| 2M | 2 | 6 | | | | | | | 8 | 2.7 | 7.5 |
| 2L | 10 | 2 | 2 | | | | | | 14 | 4.7 | 3.5 |
| 4M | 0.5 | 0.5 | | | | | | | 1 | 0.3 | 15.0 |
| 4L | 1 | 1 | | | | | | | 2 | 0.6 | 14.0 |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

在實施柔性鋪面損壞調查過程中可由該表結果換算出目視檢測區域之 PCI 值。要對一段道路進行 PCI 值評估並需要進行以下的步驟：

(一) 基本資料(表 1 橘色方框處)，包含調查區域、日期、人員與評估面積等。

因「柔性鋪面損害手冊」的市區道路柔性鋪面損壞調查表有用到面積的評估值，雖然我

們開發的評估方法裡是用不到這欄面積的，但我們還是想辦法求出評估的面積值。我們採用的方法是使用一個已知尺寸的物體，在 **6m** 航高下拍攝照片中計算其照片中的所佔像素的大小反推該張照片的面積。

團隊在討論過後，決定採用一個圓形物體來做研究，我們思考到籃球場的中圈是個合宜的選擇，因此我們使用 **Mavic Pro 2** 在 **6m** 航高下拍攝照片(如圖 13 左)，接著再以小畫家進行裁切，裁切後可得知之中圈像素平均為 **2725**(如圖 13 右)。

實際測量中圈直徑為 **353cm**，因為 **Mavic Pro 2** 所拍攝的照片尺寸為 **5472×3648**，因此推估在 **6m** 高度下 **1Pixel** 的長度為 **1.25mm**，所以反推其拍攝的面積為：

$$\left[\frac{(5472 \times 1.25)}{1000} \right] \times \left[\frac{(3648 \times 1.25)}{1000} \right] = 6.78 \times 4.56 = 30.92 \text{m}^2。$$

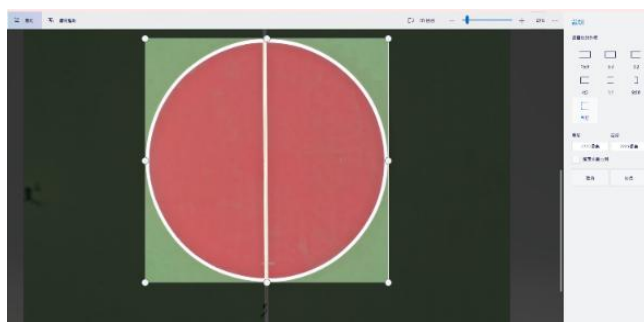
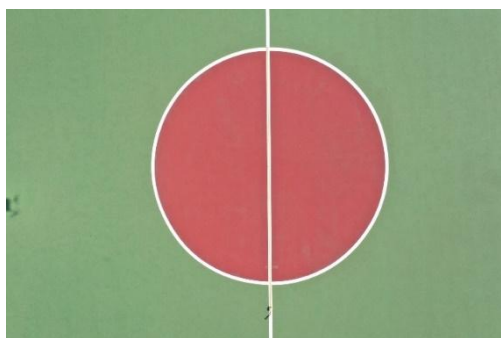


圖 13：6m 航高之 **Mavic Pro 2** 垂直拍攝籃球場中圈並用小畫家 3D 裁切(由第二作者繪製)

(二) 嚴重程度(表 1 紫色方框處)。

輕級以 **L** 表示，中級以 **M** 表示，重級以 **H** 表示，以縱向裂縫(Longitudinal Cracks)來說；輕級需小於 **0.3** 公分以下，中級無填補之裂縫平均寬度須介於 **0.3** 公分至 **0.5** 公分，重級無填補之裂縫平均寬度會大於 **0.5** 公分。

(三) 密度%(表 1 紅色方框處)，密度%為總數分別除以樣本單位面積，得各自之面積百分比。如表 1 中 **1M** 是表示中級龜裂，該行有 **6、9** 二個數字，表示在本次調查的 **300m²** 面積中，中級龜裂其面積分別為 **6 m²** 與 **9 m²**，因此其損壞密度分別為 $15/300 = 5\%$ 。

(四) 折減值(表 1 綠色方框處)：依前述求得之密度%，再依嚴重程度。可在手冊中查表求得鋪面各類損壞模式的折減值。

(五) 計算最大折減值(HDV)後計算最大容許損壞折減值數量：由表 1 最右側可知折減值的最大數為 **39**，此值即為最大折減值(HDV)，將該值代入 **m** 值計算公式。

$$m=1+(9/98)(100-HDV)$$

當 HDV39，其 $m=1+(9/98)(100-39)=6.6$

m 值採整數且小數部份一律進位，故選取 7 個折減值作為運算根據，但實際上表中只有 6 個折減值，故全部採用。

(六) 依據表 1 的損壞型式折減值，由大至小之順序填入表 2 中第一列內。

表 2：柔性鋪面之修正折減值計算範例(由第二作者截圖)

| # | 折減值 (Deduct Value) | | | | | | | | | | Total | q | CDV |
|---|--------------------|----|----|----|-----|-----|--|--|--|--|-------|---|------|
| 1 | 39 | 30 | 15 | 14 | 7.5 | 3.5 | | | | | 109.0 | 6 | 54.0 |
| 2 | 39 | 30 | 15 | 14 | 7.5 | 2 | | | | | 107.5 | 5 | 56.0 |
| 3 | 39 | 30 | 15 | 14 | 2 | 2 | | | | | 102.0 | 4 | 58.0 |
| 4 | 39 | 30 | 15 | 2 | 2 | 2 | | | | | 90.0 | 3 | 57.0 |
| 5 | 39 | 30 | 2 | 2 | 2 | 2 | | | | | 77.0 | 2 | 56.0 |
| 6 | 39 | 2 | 2 | 2 | 2 | 2 | | | | | 49.0 | 1 | 49.0 |
| 7 | | | | | | | | | | | | | |

(七) 將上步驟之折減值橫向加總後，填入表 2 的第一列的 Total 欄內。本例之 Total 為 109。

(八) 計算出取樣單位大於 2.0 折減值之個數，填入表 2 的第一列的 q 值欄內。本例開始之 q 值為 6。

(九) 依據 q 值及總折減值，可在手冊中查表或代入公式得到修正折減值(CDV)，填入表 2 的第一列的 CDV 欄內。本例之 CDV 值為 54。

(十) 進行第二次計算柔性鋪面之 CDV，仍將第一列之損壞型式折減值抄錄至第二列，僅將最末一項大於 2.0 之損壞型式折減值以 2.0 代替(本例之最後一項原為 3.5，於第二列中以 2.0 代替之)，然後將所有第二列折減值加總後，填入表 2 的第二列的 Total 欄內，再以查表或代入公式得到對應之 CDV。

(十一) 再依據 7 至 10 等步驟，依序算出 q 值及修正折減值，填入表 2 的第二列的第三及四列，直至 q=1 時為止。

(十二) 當取樣單位之 q=1 時為完成修正折減值計算作業。由表 2 中 CDV 欄內取其最大值(本例為 58)，本例之 PCI 值即為(100-58=42)。

(十三) 鋪面經過 PCI 指標評估後，鋪面狀況可依據 PCI 指標之值，由表 3 查得該路段之鋪面現況。本例之 PCI 指標為 42，其鋪面狀況等級為「尚可」。

表 3：鋪面狀況等級與 PCI 值對照表(由第二作者截圖)

| PCI 值 | 等級 (Rating) | 圖示 (Chart) |
|--------|----------------|------------|
| 85~100 | 最佳 (Excellent) | |
| 70~85 | 很好 (Very good) | |
| 55~70 | 好 (Good) | |
| 40~55 | 尚可 (Fair) | |
| 25~40 | 差 (Poor) | |
| 10~25 | 很差 (Very poor) | |
| 0~10 | 不合格 (Failed) | |

了解以上之評估方法後，我們依以下的流程進行 PCI 指標評估程式的開發：

(一) 密度%計算：

密度%的算法是滑窗模型所切取出來每張照片中資料夾中裂縫總數，除以每張照片的滑窗總數。

(二) 嚴重程度假設：

因為柔性鋪面損壞調查手冊中的損壞型式包含了面層裂縫三種、面層變形五種、面層損壞三種與其他二種。因訓練的模型僅能針對道路的裂縫做辨識與判定，因此擬針對三種面層裂縫來做評估，面層裂縫包含龜裂、縱向及橫向裂縫與塊狀裂縫等三者，而損壞型式分輕(L)、中(M)及重級(H)等三種。

要計算出照片中的裂縫寬度，除了要有該照片精準的空間解析度外，在定位切取出裂縫的位置後，也要進行相當複雜的數位影像處理，方能求取出裂縫寬度，限於難度，本研究將不著手該部分的計算，我們將統一將辨識出的裂縫設為重級(H)，以最保守的方法進行 PCI 值計算。

(三) 折減值與最大折減值：

因我們訓練出的 Resnet152 模型為二分類問題(正類與反類)；但「柔性鋪面損壞調查手冊」中與面層裂縫有關的損害種類包含龜裂、縱向及橫向裂縫與塊狀裂縫三種裂縫形式。所以我們將計算出每張照片三種裂縫形式其所有裂縫皆為重級(H)的假設下計算出其折減值後取最大值，此值即為在最保守狀況下所算出之最大折減值(HDV)。

依「柔性鋪面損壞調查手冊」中折減值的求法如下：

1、折減值計算

依「柔性鋪面損壞調查手冊」中第 44、45 頁所列，圖四十四_龜裂折減曲線圖、圖四十五_縱向及橫向裂縫折減曲線圖與圖四十六_塊狀裂縫折減曲線圖查表其相關數值，先查表得其相關數值。圖 14 為龜裂折減曲線圖，查表可知龜裂百分比為 1 的話其折減值為 31；10 的話其折減值為 62；餘類推。

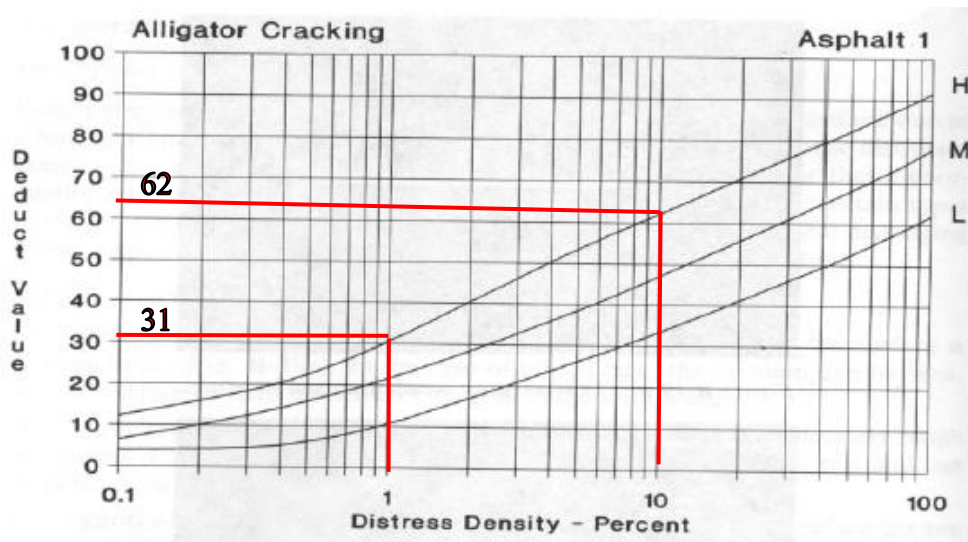


圖 14：龜裂折減曲線圖(由第二作者截圖並後製)

2、鋪面損壞折減值圖

依三種面層裂縫之鋪面損壞折減值圖，我們手工查表列出各關鍵值的對應值，以內插法求出折減值，如圖 15 所示寫入程式中。

```
# 定義折減值表
reduction_data1 = {
    0.1: 0, 0.2: 1.3, 0.3: 2.4, 0.4: 2.8, 0.5: 3.3, 0.6: 4.4, 0.7: 5.1, 0.8: 5.5,
    0.9: 6.4, 1: 6.8, 2: 12.7, 3: 16, 4: 18.7, 5: 20.7, 6: 23.1, 7: 24.7, 8: 26.7,
    9: 28.1, 10: 30, 20: 42, 30: 48.7, 40: 54.7, 50: 59.1, 60: 62.7, 70: 65.3, 80: 68,
    90: 70, 100: 71.5
}

reduction_data2 = [
    0.1: 0, 0.2: 0, 0.3: 0, 0.4: 4.2, 0.5: 5, 0.6: 5.8, 0.7: 6.7, 0.8: 7.5,
    0.9: 8, 1: 8.3, 2: 14.2, 3: 17.5, 4: 20.8, 5: 23.3, 6: 25, 7: 27.5, 8: 29.2,
    9: 32.5, 10: 34.2, 20: 49.2, 30: 59.2, 40: 67.5, 50: 63.3, 60: 77.5, 70: 80,
    80: 82.5, 90: 85, 100: 85.8
]

reduction_data3 = {
    0.1: 12, 0.2: 16, 0.3: 18, 0.4: 20, 0.5: 22, 0.6: 24, 0.7: 26, 0.8: 28,
    0.9: 29, 1: 31, 2: 40, 3: 46, 4: 50, 5: 54, 6: 56, 7: 58, 8: 59, 9: 61,
    10: 62, 20: 71, 30: 76, 40: 79, 50: 82, 60: 86, 70: 88, 80: 89, 90: 90,
    100: 92
}

# 將折減值表轉換為 numpy array 以進行內插
keys1, values1 = zip(*reduction_data1.items())
keys2, values2 = zip(*reduction_data2.items())
keys3, values3 = zip(*reduction_data3.items())
```

圖 15：鋪面損壞折減值圖求出折減值(由第二作者截圖)

(四) 計算 **m** 值與 **q** 值：

因為只有一個最大折減值，故可填**m**為1，**q**也為1，總折減值也不需修正。

(五) 計算 **CDV** 值：

代入**q**值為1時的總折減值，此時**CDV**值將等於**HDV**，將100減去**CDV**值即為**PCI**值。

(六) 計算 **PCI** 值：

將求出之**PCI**值依數值大小進行評估獲得鋪面狀況等級。

Context (上下文)：簡要說明使用場景或前提。
你將扮演我的技術顧問與程式碼生成器，協助我撰寫可直接在 Colab 執行的腳本，完成柔性鋪面 PCI 值的自動化計算。 Condition Index) 值。

Role (角色)：明確模型的身份。

Role (角色)：
你將扮演我的技術顧問與程式碼生成器，協助我撰寫可直接在 Colab 執行的腳本，完成柔性鋪面 PCI 值的自動化計算。

Task (任務 / Goal)：具體要模型做到什麼。

Task (任務)：
請生成一段 Python 程式碼，可在 Google Colab 上直接執行，實現以下功能：

1. 遍歷目錄 /content/drive/MyDrive/65/ Road 500Epoch predictions/ 下所有子資料夾，統計每個子資料夾中 .jpg 檔案數量，並產生欄位 Folder Name 與 jpg Count。
2. 計算 Crack Percentage = (jpg Count / 425) * 100。
3. 使用提供之 three sets of reduction data，對 Crack Percentage 做線性內插，分別計算 Reduction Value1、Reduction Value2、Reduction Value3：

- reduction_data1：塊狀裂縫折減值
- reduction_data2：縱橫向裂縫折減值
- reduction_data3：龜裂折減值

4. 計算 HDV 為前述三個折減值的最大值。
5. 設定 CDV = HDV。
6. 計算 PCI = 100 - CDV。
7. 根據 PCI 值給出 Rating：

- PCI >= 85：最佳 Excellent
- 70 <= PCI < 85：很好 Very good
- 55 <= PCI < 70：好 Good
- 40 <= PCI < 55：尚可 Fair
- 25 <= PCI < 40：差 Poor
- 10 <= PCI < 25：很差 Very poor
- PCI < 10：不合格

8. 將結果輸出為 CSV 文件至 /content/drive/MyDrive/65/ Road 500Epoch predictions/PCIv.csv，包含欄位：

- Folder Name, jpg Count, Crack Percentage, Reduction Value1, Reduction Value2, Reduction Value3, HDV, CDV, PCI, Rating。

Presentation (呈現 / Way)：希望輸出的格式、風格或語氣。

Presentation (呈現)：

- 以完整的 Colab code cell 格式呈現，包含必要的 import 與路徑掛載說明。
- 每段程式碼 cell 上方加入註解說明功能。
- 最後展示程式執行結果的範例（如 DataFrame head）。
- 語氣專業、簡潔。

Constraints (限制條件, Optional)：可選，用於加入額外限制。

圖 16：ChatGPT輸入提示工程協助生成PCI值計算程式(由第一作者生成並截圖)

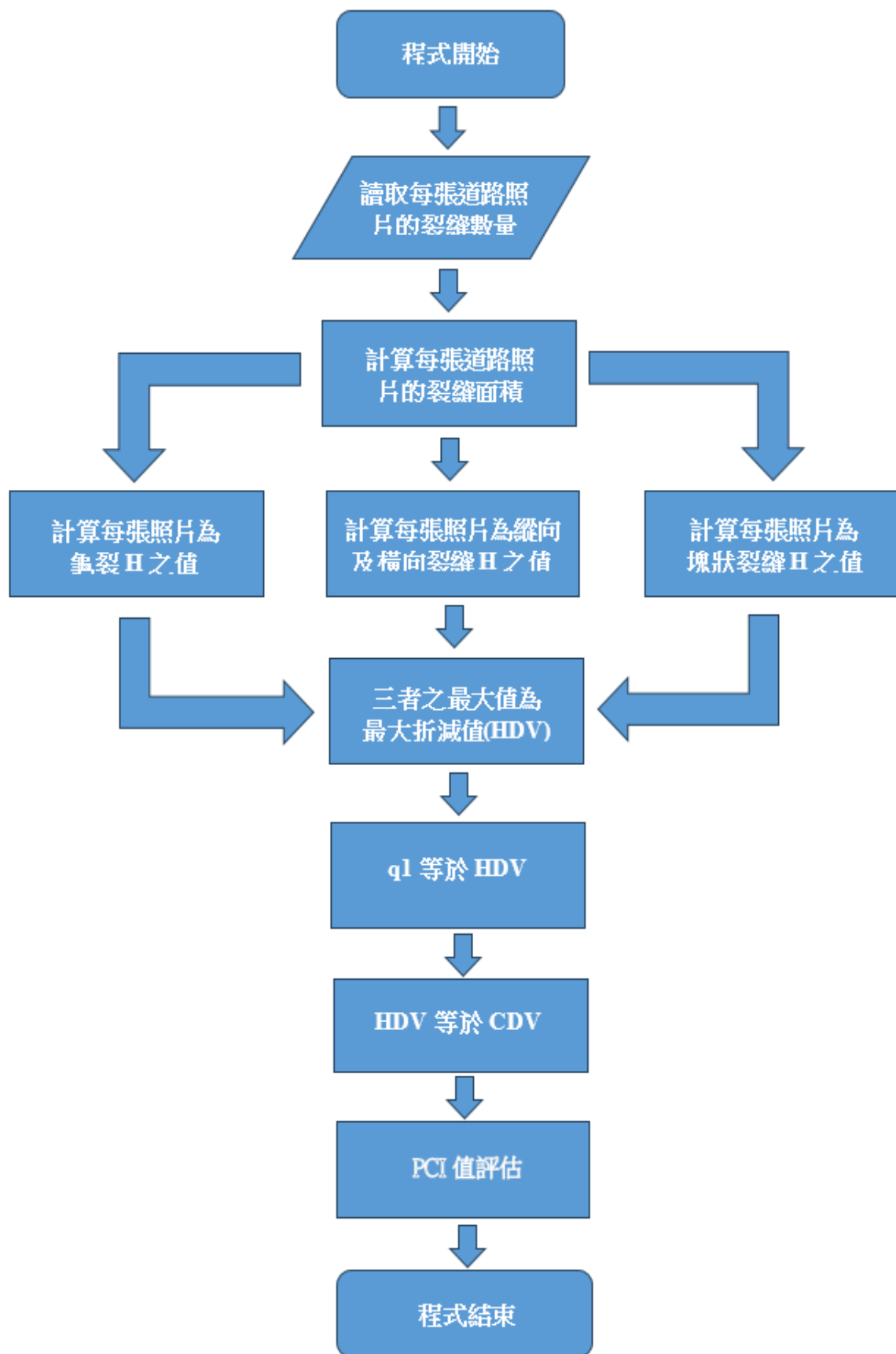


圖 17：PCI值計算程式流程圖(由第一作者製作)

肆、研究結果

一、Resnet152 模型訓練

將前述提示工程協助生成的Resnet152程式分別訓練200Epoch與500Epoch，圖18左為GitHub上原始程式對於數據集增強可視化(Visualize the Augmented Data Set)的執行結果，圖18右為我們將之改寫至Colab上執行的結果。

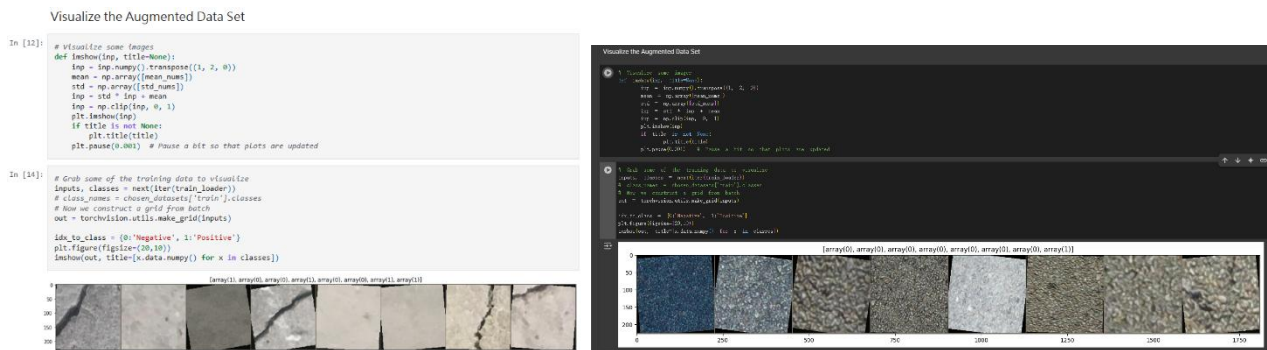


圖 18：GitHub 原始程式與作者改寫至 Colab 上執行的結果 (由所有作者共同截圖)

(一) 使用測試集評估模型

將訓練好的模型以測試集評估訓練結果，200 epoch 的 Test Accuracy 為 0.9234，Test Loss 為 0.2285 (如圖 19 左)，而 500 epoch 的 Test Accuracy 為 0.9294，Test Loss 為 0.2229 (如圖 19 右)，二者 Test Accuracy 差異小於 0.001，幾乎可以忽略。

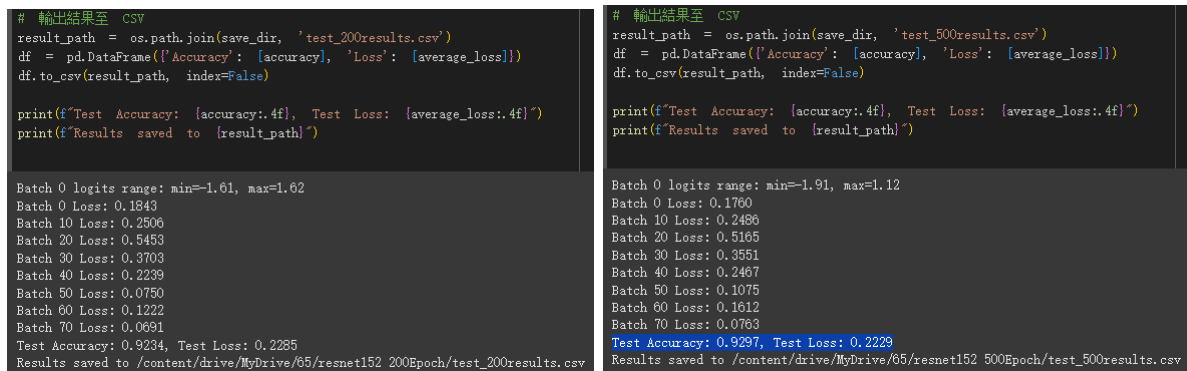


圖 19：模型對測試集的測試結果(由所有作者共同截圖)

(二) 使用二元混淆矩陣評估模型

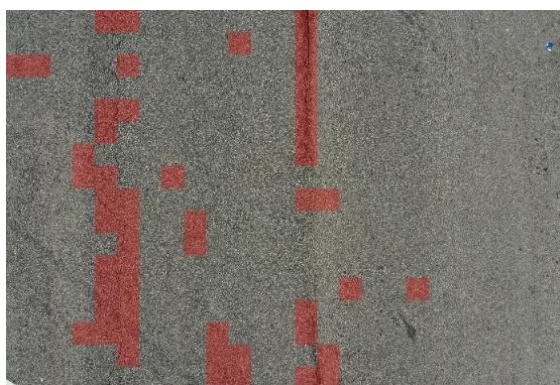
因以Test Accuracy無法明確評估二個模型的優劣，團隊討論後改採用 F1 Score 來評估滑窗結果的優劣。

採用 F1 Score 而不是 Test Accuracy 來評估的優劣的原因是，採用滑動視窗偵測 UAV 所拍攝的裂縫照片時，因為照片中無裂縫的區域大於有裂縫的區域，因此造成負類數量(無裂縫)遠大於正類數量(有裂縫)，因此 Test Accuracy 無法呈現出模型的優劣，此時應以 F1 Score 來做為評估指標，因為 F1 Score 是由召回率(Recall)、精準率(Precision)二個指標的調和平均值，可以較精確的評估出模型採用滑動視窗來標定出裂縫的優劣(蔡佳勳2023)。

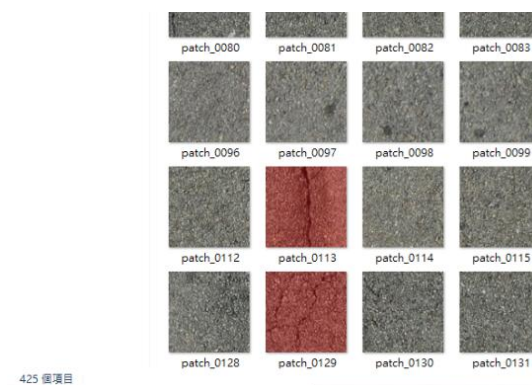
我們的做法是採用柔性鋪面照片經二模型滑窗後的照片，以前面所撰寫的切片程式進行切片，再以人工目視分類出每張照片中 TP、TN、FP、FN 之數量後計算該張照片之二元混淆矩陣。流程如下：

- (一) 採用 15 張 Mavic Pro 2 拍攝的柔性鋪面照片將其依次命名 01 至 015。
- (二) 以 224×224 的滑動窗口逐一掃描照片，標示裂縫區域(如圖 20 A)。
- (三) 將上項滑照片以切片程式裁成 425 張切片，用於後續二元混淆矩陣評估(如圖 20 B)。
- (四) 以人工目視分類出每張照片中 TP、TN、FP、FN 之數量(如圖 20 C)。
- (五) 以 EXCEL 試算表計算二元混淆矩陣(如圖 20 D)。

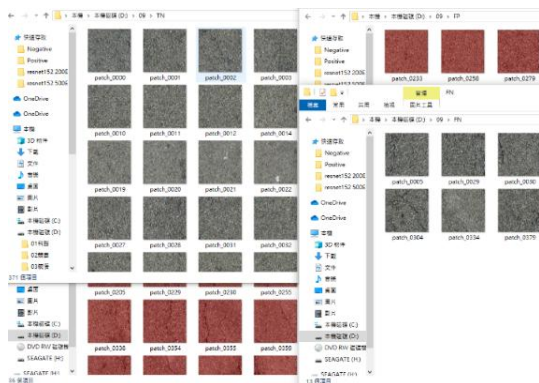
A、採用 224×224 滑動窗口標示裂縫區域



B、以切片程式裁成 425 張切片



C、人工目視分類



D、計算二元混淆矩陣

| | A | B | C | D | E | F | G | H | I |
|----|-------------|-----|-----|-----|----|-----|--------|-----------|----------|
| 1 | Folder Name | TP | TN | FP | FN | 總數 | Recall | Precision | F1 Score |
| 2 | 1 | 247 | 29 | 140 | 9 | 425 | 0.965 | 0.638 | 0.768 |
| 3 | 2 | 219 | 44 | 156 | 6 | 425 | 0.973 | 0.584 | 0.730 |
| 4 | 3 | 240 | 76 | 62 | 47 | 425 | 0.836 | 0.795 | 0.815 |
| 5 | 4 | 158 | 171 | 0 | 96 | 425 | 0.622 | 1.000 | 0.767 |
| 6 | 5 | 111 | 253 | 2 | 59 | 425 | 0.653 | 0.982 | 0.784 |
| 7 | 6 | 87 | 307 | 0 | 31 | 425 | 0.737 | 1.000 | 0.849 |
| 8 | 7 | 92 | 323 | 0 | 10 | 425 | 0.902 | 1.000 | 0.948 |
| 9 | 8 | 53 | 345 | 2 | 25 | 425 | 0.679 | 0.964 | 0.797 |
| 10 | 9 | 36 | 371 | 5 | 13 | 425 | 0.735 | 0.878 | 0.800 |
| 11 | 10 | 42 | 350 | 1 | 32 | 425 | 0.568 | 0.977 | 0.718 |
| 12 | 11 | 26 | 377 | 7 | 15 | 425 | 0.634 | 0.788 | 0.703 |
| 13 | 12 | 38 | 361 | 4 | 22 | 425 | 0.633 | 0.905 | 0.745 |
| 14 | 13 | 21 | 392 | 10 | 2 | 425 | 0.913 | 0.677 | 0.778 |
| 15 | 14 | 28 | 384 | 0 | 13 | 425 | 0.683 | 1.000 | 0.812 |
| 16 | 15 | 15 | 399 | 3 | 8 | 425 | 0.652 | 0.833 | 0.732 |

圖 20：將滑動窗口後的照片轉換成小切片(由所有作者共同製作並截圖)

1、訓練 200Epoch 之 Resnet152 模型

15張柔性鋪面照片以訓練200Epoch的Resnet152模型來滑窗後，得到的二元混淆矩陣結果如下表4。F1 Score最小的為標號11的0.703，最大的為標號07的0.948，所有照片的F1 Score皆大於**0.700**，表示模型的訓練結果不錯，15張柔性鋪面照片F1 Score平均值為0.783。

表 4：模型訓練 200Epoch 後之評估指標(由所有作者共同製作)

| Folder Name | TP | TN | FP | FN | Recall | Precision | F1 Score |
|--------------|-----|-----|-----|----|--------|-----------|--------------|
| 01 | 247 | 29 | 140 | 9 | 0.965 | 0.638 | 0.768 |
| 02 | 219 | 44 | 156 | 6 | 0.973 | 0.584 | 0.730 |
| 03 | 240 | 76 | 62 | 47 | 0.836 | 0.795 | 0.815 |
| 04 | 158 | 171 | 0 | 96 | 0.622 | 1.000 | 0.767 |
| 05 | 111 | 253 | 2 | 59 | 0.653 | 0.982 | 0.784 |
| 06 | 87 | 307 | 0 | 31 | 0.737 | 1.000 | 0.849 |
| 07 | 92 | 323 | 0 | 10 | 0.902 | 1.000 | 0.948 |
| 08 | 53 | 345 | 2 | 25 | 0.679 | 0.964 | 0.797 |
| 09 | 36 | 371 | 5 | 13 | 0.735 | 0.878 | 0.800 |
| 10 | 42 | 350 | 1 | 32 | 0.568 | 0.977 | 0.718 |
| 11 | 26 | 377 | 7 | 15 | 0.634 | 0.788 | 0.703 |
| 12 | 38 | 361 | 4 | 22 | 0.633 | 0.905 | 0.745 |
| 13 | 21 | 392 | 10 | 2 | 0.913 | 0.677 | 0.778 |
| 14 | 28 | 384 | 0 | 13 | 0.683 | 1.000 | 0.812 |
| 15 | 15 | 399 | 3 | 8 | 0.652 | 0.833 | 0.732 |
| F1 Score 平均值 | | | | | | | 0.783 |

2、訓練 200Epoch 之 Resnet152 模型

15張柔性鋪面照片以訓練500Epoch的Resnet152模型來滑窗後，得到的二元混淆矩陣結果如下表5。該模型F1 Score最小的為標號11的0.703，滑窗結果最大的為標號07的0.945。15張柔性鋪面照片F1 Score平均值為0.815。

表 5：模型訓練 500Epoch 後之評估指標(由所有作者共同製作)

| Folder Name | TP | TN | FP | FN | Recall | Precision | F1 Score |
|--------------|-----|-----|-----|----|--------|-----------|--------------|
| 01 | 265 | 13 | 143 | 4 | 0.985 | 0.650 | 0.783 |
| 02 | 235 | 19 | 165 | 6 | 0.975 | 0.588 | 0.733 |
| 03 | 258 | 68 | 79 | 20 | 0.928 | 0.766 | 0.839 |
| 04 | 174 | 175 | 0 | 76 | 0.696 | 1.000 | 0.821 |
| 05 | 131 | 251 | 6 | 37 | 0.780 | 0.956 | 0.859 |
| 06 | 96 | 313 | 3 | 13 | 0.881 | 0.970 | 0.923 |
| 07 | 94 | 320 | 3 | 8 | 0.922 | 0.969 | 0.945 |
| 08 | 61 | 344 | 5 | 15 | 0.803 | 0.924 | 0.859 |
| 09 | 37 | 363 | 14 | 11 | 0.771 | 0.725 | 0.747 |
| 10 | 49 | 360 | 1 | 15 | 0.766 | 0.980 | 0.860 |
| 11 | 32 | 366 | 18 | 9 | 0.780 | 0.640 | 0.703 |
| 12 | 44 | 358 | 5 | 18 | 0.710 | 0.898 | 0.793 |
| 13 | 44 | 358 | 5 | 18 | 0.710 | 0.898 | 0.793 |
| 14 | 31 | 383 | 3 | 8 | 0.795 | 0.912 | 0.849 |
| 15 | 16 | 396 | 3 | 10 | 0.615 | 0.842 | 0.711 |
| F1 Score 平均值 | | | | | | | 0.815 |

我們將表4與表5的F1 Score做比較，當模型訓練由200Epoch增加為500Epoch後，F1 Score 平均值由0.783增加為0.815。500Epoch的模型除了Test Accuracy略高於200Epoch的模型外，在實際滑窗應用中有更高的 F1 Score，代表它更準確地識別裂縫，也更適合部署在現實應用中。所以我們以**500Epoch**的模型來做為自動PCI值評估的模型。

二、自動 PCI 值評估

圖21為PCI值評估執行程式截圖，圖22為十五張面層裂縫PCI值滑窗結果，表6為PCI值計算結果，經由我們開發程式的評估計有DJI_0005與DJI_0006二張照片是不合格的，DJI_0001、DJI_0004、DJI_0013三張照片示很差(Very Poor)的狀態，八張評級為差(Poor)，只有二張為尚可(Fair)。另由表6可以觀察到Reduction Value3(代表龜裂)的值在三種裂縫中恆為最大，所以面層裂縫為H的狀態下，龜裂的形式是影響最大的。

| PCiv.csv | | | | | | | | | | |
|----------|-------------|-----------|------------------|------------------|------------------|------------------|-------|-------|-------|-----------|
| | A | B | C | D | E | F | G | H | I | J |
| 1 | Folder Name | jpg Count | Crack Percentage | Reduction Value1 | Reduction Value2 | Reduction Value3 | HDV | CDV | PCI | Rating |
| 2 | 1 | 408 | 96 | 70.9 | 85.48 | 91.2 | 91.2 | 91.2 | 8.8 | Failed |
| 3 | 2 | 400 | 94.12 | 70.62 | 85.33 | 90.82 | 90.82 | 90.82 | 9.18 | Failed |
| 4 | 3 | 337 | 79.29 | 67.81 | 82.32 | 88.93 | 88.93 | 88.93 | 11.07 | Very poor |
| 5 | 4 | 174 | 40.94 | 55.11 | 67.1 | 79.28 | 79.28 | 79.28 | 20.72 | Very poor |
| 6 | 5 | 137 | 32.24 | 50.04 | 61.06 | 76.67 | 76.67 | 76.67 | 23.33 | Very poor |
| 7 | 6 | 99 | 23.29 | 44.21 | 52.49 | 72.65 | 72.65 | 72.65 | 27.35 | Poor |
| 8 | 7 | 97 | 22.82 | 43.89 | 52.02 | 72.41 | 72.41 | 72.41 | 27.59 | Poor |
| 9 | 8 | 66 | 15.53 | 36.64 | 42.49 | 66.98 | 66.98 | 66.98 | 33.02 | Poor |
| 10 | 9 | 51 | 12 | 32.4 | 37.2 | 63.8 | 63.8 | 63.8 | 36.2 | Poor |
| 11 | 10 | 50 | 11.76 | 32.12 | 36.85 | 63.59 | 63.59 | 63.59 | 36.41 | Poor |
| 12 | 11 | 50 | 11.76 | 32.12 | 36.85 | 63.59 | 63.59 | 63.59 | 36.41 | Poor |
| 13 | 12 | 49 | 11.53 | 31.84 | 36.49 | 63.38 | 63.38 | 63.38 | 36.62 | Poor |
| 14 | 13 | 40 | 9.41 | 28.88 | 33.2 | 61.41 | 61.41 | 61.41 | 38.59 | Poor |
| 15 | 14 | 34 | 8 | 26.7 | 29.2 | 59 | 59 | 59 | 41 | Fair |
| 16 | 15 | 19 | 4.47 | 19.64 | 21.98 | 51.88 | 51.88 | 51.88 | 48.12 | Fair |

圖 21：PCI 值評估執行程式截圖(由第一作者截圖)

表 6：PCI 值計算結果(由第一作者製作)

| Folder Name | jpg Count | Crack Percentage | Reduction Value1 | Reduction Value2 | Reduction Value3 | HDV | CDV | PCI | Rating |
|-------------|-----------|------------------|------------------|------------------|------------------|--------|--------|--------|-----------------|
| 01 | 408 | 0.960 | 70.900 | 85.480 | 91.200 | 91.200 | 91.200 | 8.800 | 不合格 |
| 02 | 400 | 0.941 | 70.618 | 85.329 | 90.824 | 90.824 | 90.824 | 9.176 | 不合格 |
| 03 | 337 | 0.793 | 67.809 | 82.324 | 88.929 | 88.929 | 88.929 | 11.071 | 很差 Very poor |
| 04 | 174 | 0.409 | 55.114 | 67.105 | 79.282 | 79.282 | 79.282 | 20.718 | 很差 Very poor |
| 05 | 137 | 0.322 | 50.041 | 61.055 | 76.671 | 76.671 | 76.671 | 23.329 | 很差 Very poor |
| 06 | 99 | 0.233 | 44.207 | 52.494 | 72.647 | 72.647 | 72.647 | 27.353 | 差 Poor |
| 07 | 97 | 0.228 | 43.892 | 52.024 | 72.412 | 72.412 | 72.412 | 27.588 | 差 Poor |
| 08 | 66 | 0.155 | 36.635 | 42.494 | 66.976 | 66.976 | 66.976 | 33.024 | 差 Poor |
| 09 | 51 | 0.120 | 32.400 | 37.200 | 63.800 | 63.800 | 63.800 | 36.200 | 差 Poor |
| 10 | 50 | 0.118 | 32.118 | 36.847 | 63.588 | 63.588 | 63.588 | 36.412 | 差 Poor |
| 11 | 50 | 0.118 | 32.118 | 36.847 | 63.588 | 63.588 | 63.588 | 36.412 | 差 Poor |
| 12 | 49 | 0.115 | 31.835 | 36.494 | 63.376 | 63.376 | 63.376 | 36.624 | 差 Poor |
| 13 | 40 | 0.094 | 28.882 | 33.200 | 61.412 | 61.412 | 61.412 | 38.588 | 差 Poor |
| 14 | 34 | 0.080 | 26.700 | 29.200 | 59.000 | 59.000 | 59.000 | 41.000 | 尚可 Fair |
| 15 | 19 | 0.045 | 19.641 | 21.976 | 51.882 | 51.882 | 51.882 | 48.118 | 尚可 Fair |

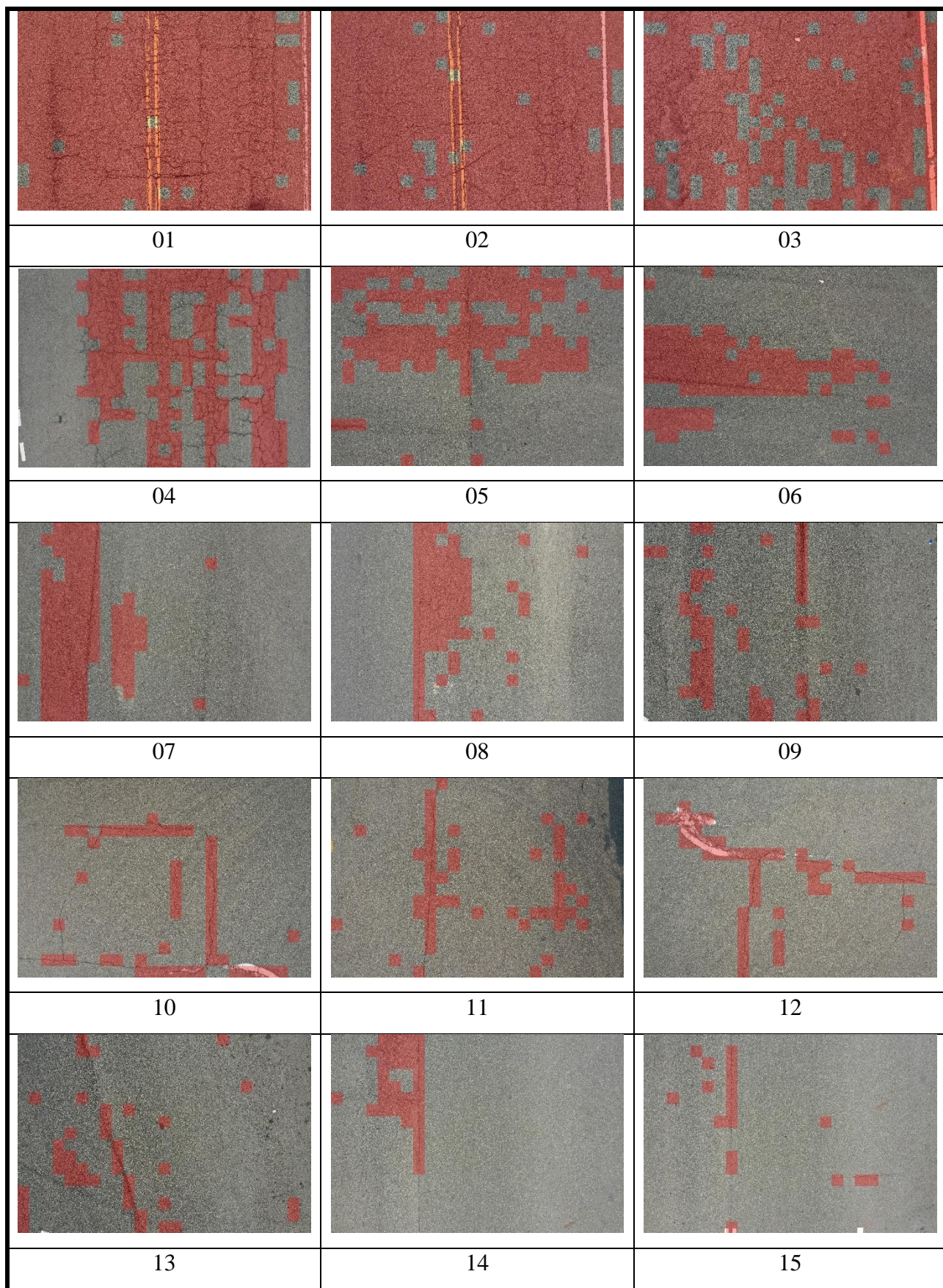


圖 22：十五張面層裂縫 PCI 值滑窗結果(由第一作者截圖)

伍、討論

以下針對訓練結果、滑窗分析及自動 PCI 值評估三方面進行討論。

一、ResNet152 模型訓練與滑窗分析討論

(一) 我們以 15 張面層裂縫的滑窗結果來取代傳統的測試集，目的是想知道模型對於道路裂縫的實際辨識能力。500Epoch 的 F1 Score 平均值為 0.815，訓練結果算是不錯。

但對於將其運用在自動 PCI 值的實際運用上，我們見表 7 所示，其第一欄為照片編號，第二欄為模型滑窗結果 TP+ FP 這代表著模型實際辨識裂縫的能力，第三欄為人工目視 TP+ FN 表示人工目視檢測出這些照片上有裂縫的切片正確數目，之後二欄分別為二個值的差值與誤差百分比，最後一欄則為 PCI 值的評級。由表 7 我們觀察到幾點現象：

- 1、編號 01 與編號 02 其 PCI 評級為不合格，誤差百分比分別為 33% 與 37% 多達三成。
- 2、編號 03 至 05，其 PCI 評級為很差，誤差在正負 7% 至 18%；明顯較前述有較之低誤差值。
- 3、其餘十張照片誤差皆在正負 3% 以內，這些照片的裂縫誤差皆很小。

以上現象，顯示路面狀況越差，造成我們的模型越容易誤判，若 PCI 評級在差(Poor)以上，則我們的模型有很高的準確率。

(二) 我們採用 Resize 方法來調整照片大小，該方法讓滑窗可以完整的檢查到整張照片有沒有裂縫，而且我們所開發的評估方法是計算裂縫區域佔整張照片的百分比，因此 Resize 法對於評估結果的影響甚微。

(三) 為了簡化研究歷程，我們假設柔性鋪面裂縫與混凝土裂縫相似的前提下，採用既有混凝土裂縫模型訓練時的相關參數設置，但研究過程發現其實二種是有相當程度差異的。道路裂縫紋理複雜，背景噪聲較多，這也是道路裂縫的深度學習模型相關研究遠較混凝土裂縫少很多的原因，其訓練難度相較混凝土是不同等級的。因此部分涉及深度學習模型訓練，專業程度不是國中程度的我們可以處理的，建議有興趣的團隊對於此方面進行更為深入的研究。

表 7：模型辨識裂縫與人工目視結果(由第一作者製作)

| Folder Name | A 滑窗結果 TP+ FP | B 人工目視 TP+ FN | 差值 A-B | 誤差% | Rating |
|-------------|------------------|------------------|-----------|-------|-----------------|
| 01 | 408 | 269 | 139 | 33 % | 不合格 |
| 02 | 400 | 241 | 159 | 37 % | 不合格 |
| 03 | 337 | 278 | 59 | 14 % | 很差 Very poor |
| 04 | 174 | 250 | -76 | -18 % | 很差 Very poor |
| 05 | 137 | 168 | -31 | -7 % | 很差 Very poor |
| 06 | 99 | 109 | -10 | -2 % | 差 Poor |
| 07 | 97 | 102 | -5 | -1 % | 差 Poor |
| 08 | 66 | 76 | -10 | -2 % | 差 Poor |
| 09 | 51 | 48 | 3 | 1 % | 差 Poor |
| 10 | 50 | 64 | -14 | -3 % | 差 Poor |
| 11 | 50 | 41 | 9 | 2 % | 差 Poor |
| 12 | 49 | 62 | -13 | -3 % | 差 Poor |
| 13 | 40 | 26 | 14 | 3 % | 差 Poor |
| 14 | 34 | 39 | -5 | -1 % | 尚可 Fair |
| 15 | 19 | 26 | -7 | -2 % | 尚可 Fair |

二、自動 PCI 值評估

- (一) 鋪面損壞折減值圖我們是採用內插法的方式來計算出折減值，但內插法是有其誤差存在的，建議日後可以再針對鋪面損壞折減值圖開發出更精確的折減值計算法。
- (二) 龜裂的值在三種裂縫中恆為最大，所以面層裂縫為重級(H)的狀態下，龜裂的形式是影響最大的。
- (三) PCI值評估除了三種面層裂縫外，還包含了面層變形五種、面層損壞三種與其他二種。
此次研究證實了我們的想法是實際可行的，日後只要訓練出更強大的模型，並依照先前的開發邏輯，即可讓我們的自動檢測柔性鋪面PCI值系統實際運用於業界。

- (四) 我們全程採用開放框架、免費資源開發出這套系統，日後這套系統成熟時，也能協助檢測員進行外業後，只要有網路與筆電，立即可雲端解算做大量的初步評估，檢測員再根據評估結果，針對異常之處做複評，節省大量的時間。
- (五) 近年來 UAV 大量使用在如俄烏戰爭戰場等戰場上，本系統除了前述可協助檢測員快速檢測外。未來在戰場上當道路發生嚴重破壞時，指揮單位無法使用車輛進行道路損壞評估時，使用 UAV 搭配本系統是一個可行的選項。

三、生成式 AI 的應用

本研究全程採用開放框架及免費資源，並利用生成式 AI 工具輔助程式撰寫。

團隊使用 ChatGPT 將 GitHub 上的現有程式移植到 Google Colab 上執行，再搭配提示工程協助寫了三個小程式即完成了整個系統，展現了生成式 AI 在技術實踐中的強大輔助效益。

綜上，本研究證實了利用 ResNet152 結合滑動窗口技術與自動 PCI 值計算方法進行柔性鋪面裂縫智能檢測的可行性，並為未來系統優化與應用拓展提供了改進方向。

陸、結論

本研究以 ResNet152 模型實現了道路裂縫的自動檢測，並根據內政部營建署「柔性鋪面損壞調查手冊」所訂評估方法，自動計算面層裂縫的 PCI 值。透過 UAV 大規模航拍、雲端資料處理及免費框架的運用，我們建立了一套全自動化系統，可在外業初步評估路面狀況，並供檢測員針對異常部分進行複評。我們收集國內相關資料與文獻後，發覺團隊是首先嘗試以深度學習模型結合政府規範來進行道路裂縫自動評估，並在文章中證明了可行性，這是我們團隊最大的成果。該系統成功將改良式空照法應用於柔性鋪面檢測，展現了自動化檢測技術在提高檢測效率與降低人力成本方面的潛力，為未來智慧道路維護提供了堅實技術基礎。

柒、參考文獻資料

一、中文部分

- 1、財團法人中國生產力中心國立台灣大學土木工程學研究所（2002）。市區道路管理維護與技術規範手冊研究【內政部柔性鋪面損壞調查手冊】。內政部營建署。
- 2、章皓鈞（2021）。基於深度學習之路面破損檢測。〔碩士論文。國立臺灣師範大學〕臺灣博碩士論文知識加值系統。取自 <https://hdl.handle.net/11296/3224nc>。
- 3、蔡佳勳（2023）。基於雲端計算之深度學習混凝土裂縫檢測定位研究。〔碩士論文。國立中興大學〕臺灣博碩士論文知識加值系統。取自 <https://hdl.handle.net/11296/9hd45a>。
- 4、鄭安捷、劉千琳、李羽弘（2025）。運用生成式 AI 結合 UAV 自動檢測柔性鋪面 PCI 值之可行性研究。
- 5、鄭安捷、劉千瑀、劉千琳（2025）。全方位 AI 裂縫警示系統。

二、英文部分

- 1、Özgenel, Ç. F., & Sorguç, A. G. (2018). Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings. In *35th International Symposium on Automation and Robotics in Construction (ISARC 2018)*. doi:10.22260/ISARC2018/0094

三、網路資源

- 1、Keras-Team（2024）。Keras 中文文檔。取自 <https://keras-zh.readthedocs.io/optimizers/>。
- 2、OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. Retrieved from <https://chat.openai.com/chat>.
- 3、Priya Dwivedi (2020). Detection of Surface Cracks in Concrete Structures using Deep Learning. Retrieved from <https://medium.com/towards-data-science/detection-of-surface-cracks-in-concrete-structures-using-deep-learning-f8f85cd8ac8b>
- 4、sunlei（2020）。深度學習混凝土結構裂紋檢測。取自 <https://www.atyun.com/49187.html>
- 5、priya-dwivedi (2024). Deep-Learning/crack_detection/Crack Detection Model.ipynb. Retrieved from [https://github.com/priya-dwivedi/Deep-Learning/blob/master/crack_detection/Crack Detection Model.ipynb](https://github.com/priya-dwivedi/Deep-Learning/blob/master/crack_detection/Crack%20Detection%20Model.ipynb)

【評語】 032803

1. 本作品完成度高，展現出優良的實用性，有助於減少道路檢查所需的人力成本，並有效提升檢測的效率與安全性。
2. 若能進一步提升資料影像判斷的精確度，並克服雜訊、光線與解析度等可能造成誤判的因素，將有助於讓系統更趨穩定與可靠。
3. 建議未來可嘗試將裂縫的寬度與深度納入模型分析中，這將進一步強化作品的應用深度與實務價值，值得持續發展。

作品海報



運用生成式AI及UAV影像 自動檢測道路裂縫之可行性研究

研究動機

台灣的道路品質長久以來讓人詬病，現行的巡查方式是傳統的目視檢測，有以下的缺點：

1 漏檢危機：

調查員在車輛行駛中目視檢測，稍有不慎便可能因速度過快或視線受阻而錯過。

2 安全風險：

調查員在繁忙路段作業時，必須一邊注意來車、一邊觀察路面。

3 人力短缺：

許多基層單位因人力不足，無法定期巡檢，導致道路維護工作斷斷續續。

以上缺陷，不僅效率低下，還潛藏危機，這讓我們不禁思考是否有更聰明、更安全的替代方案？

研究目的

結合先進的AI技術與UAV應用，打造一套自動化、智能化的柔性鋪面裂縫檢測系統

1 打造高效AI裂縫辨識模型：

訓練並優化Resnet152模型，使其能夠精準辨識柔性鋪面上的裂縫。

2 實現雲端自動裂縫偵測：

雲端環境下快速且準確地從UAV拍攝的柔性鋪面照片中自動標記裂縫位置。

3 自動化PCI值評估與決策支援：

依據內政部「柔性鋪面損壞調查手冊」標準，精確評估鋪面狀況指數(PCI)值。將即時分析結果提供給調查人員，作為道路維護決策的科學依據，提升維護管理的效率與可靠性。

研究方法

由於本研究的程式皆是採ChatGPT生成，為了能讓 ChatGPT 更精準、穩定地輸出符合預期的程式碼，使用提示工程(Prompt Engineering)完整地框定程式碼預期的行為範圍，能有效提升生成程式碼的穩定性、可重現性和符合度。團隊採用以下五大要素：

- 1 上下文(Context)：簡要說明使用場景或前提。
- 2 扮演角色(Role)：明確ChatGPT扮演的身份。
- 3 具體任務(Task)：具體要ChatGPT做到什麼。
- 4 呈現方式(Presentation)：希望輸出的格式、風格或語氣。
- 5 限制條件(Constraints)：可選，用於加入額外限制。

本研究所有程式碼之ChatGPT提示工程均詳細列於右圖之手冊中。



所有程式之提示工程^[注1]

ResNet152特徵萃取模型訓練

一、數據集製作

- 1 訓練集(train data)：正類、反類各3200張，共6400張。
- 2 驗證集(validation dtat)：正類、反類各640張，共1280張。
- 3 測試集(test dtat)：正類、反類各640張，共1280張。

二、ResNet152特徵萃取模型訓練

程式開發參考GitHub上的所搭建的Resnet模型，程式後半段並附有滑窗功能檢測裂縫。利用ChatGPT移植至Google Colab，調整適配雲端環境。

將前述提示工程協助生成的Resnet152程式分別訓練200Epoch與500Epoch二個模型。

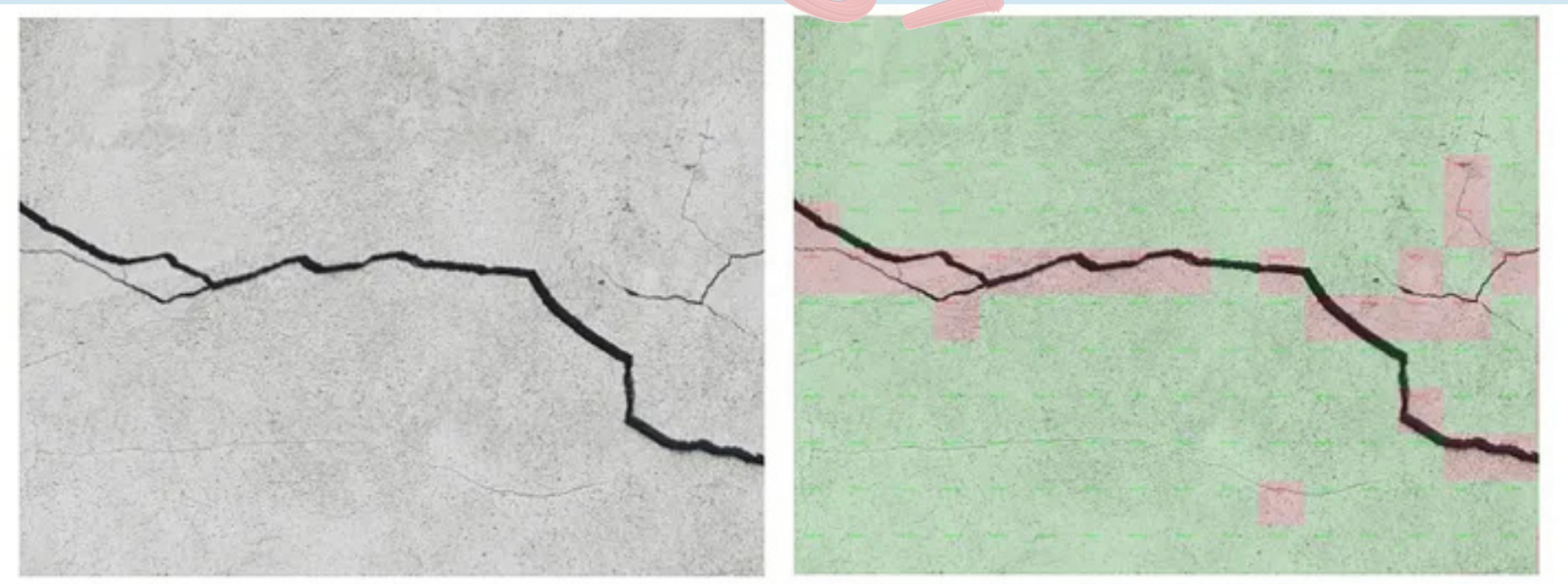
優化器：採用Adagrad優化器。

批次大小(Batch Size)：為每批處理的樣本個數
批次大小，設為32。

```
Context (上下文) 你是一位深度学习工程師，專精於 TensorFlow / Keras 以及雲端 Notebook (例如 Google Colab) 。
Role: 你將扮演我的技術顧問。
Task:
Resize程式
1、讀取google雲端硬碟路徑 "/content/drive/MyDrive/Road"裡面所有的照片*.jpg
2、創建一個Road Resize資料夾。
3、將該照片放大為224大小可以整除的，並存到google雲端硬碟路徑 "/content/drive/MyDrive/Road Resize"資料夾，resize後的照片名稱跟原照片一樣。

Presentation (呈現 / Way)
舉例 如原始照片大小5472*3648，則resize後的照片為5600*3800，該照片是可以被224整除的
Constraints: 請以你的專業腳色，幫我將上段程式碼改寫到colab執行。
。
```

切片程式提示工程^[注2]



GitHub上的Resnet程式滑窗結果^[注3]

模型成效評估

使用測試集評估與二元混淆矩陣二種方法來交叉評估模型。

- 1 對於數據集中的測試集資料夾進行測試，以 test acc與test loss來評估。
- 2 採用15張實際柔性鋪面裂縫照片來分析模型的實際訓練成果。將滑窗後的照片以切片程式進行切片，再以人工對切片進行分類。

召回率

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$





精準率

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

二者的調和平均值

$$\text{F1 Score} = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

二元混淆矩陣

| | | Actual Values(真實值) | |
|------------------------|----------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| | | Positive | Negative |
| Predicted Values (預測值) | Positive |  |  |
| | Negative |  |  |
| | | True Positive(TP) | False Positive(FP) |
| | | False Negative(FN) | True Negative(TN) |

二元混淆矩陣^[注4]

滑動窗口程式流程



PCI值計算程式流程



研究結果

一、ResNet152模型訓練

500Epoch的模型除了Test Accuracy略高於200Epoch的模型外，在實際滑窗應用中有更高的 F1 Score，代表它能更準確地識別裂縫，也更適合部署在現實應用中。所以我們以500Epoch的模型來做為自動PCI值評估的模型。

200Epoch模型

Test Accuracy為 0.9234，Test Loss為0.2285。

模型訓練200Epoch後之評估指標^[注14]

| Folder Name | TP | TN | FP | FN | Recall | Precision | F1 Score |
|-------------|-----|-----|-----|----|--------|-----------|----------|
| 1 | 247 | 29 | 140 | 9 | 0.965 | 0.638 | 0.768 |
| 2 | 219 | 44 | 156 | 6 | 0.973 | 0.584 | 0.73 |
| 3 | 240 | 76 | 62 | 47 | 0.836 | 0.795 | 0.815 |
| 4 | 158 | 171 | 0 | 96 | 0.622 | 1 | 0.767 |
| 5 | 111 | 253 | 2 | 59 | 0.653 | 0.982 | 0.784 |
| 6 | 87 | 307 | 0 | 31 | 0.737 | 1 | 0.849 |
| 7 | 92 | 323 | 0 | 10 | 0.902 | 1 | 0.948 |
| 8 | 53 | 345 | 2 | 25 | 0.679 | 0.964 | 0.797 |
| 9 | 36 | 371 | 5 | 13 | 0.735 | 0.878 | 0.8 |
| 10 | 42 | 350 | 1 | 32 | 0.568 | 0.977 | 0.718 |
| 11 | 26 | 377 | 7 | 15 | 0.634 | 0.788 | 0.703 |
| 12 | 38 | 361 | 4 | 22 | 0.633 | 0.905 | 0.745 |
| 13 | 21 | 392 | 10 | 2 | 0.913 | 0.677 | 0.778 |
| 14 | 28 | 384 | 0 | 13 | 0.683 | 1 | 0.812 |
| 15 | 15 | 399 | 3 | 8 | 0.652 | 0.833 | 0.732 |
| F1 Score平均值 | | | | | | | 0.783 |

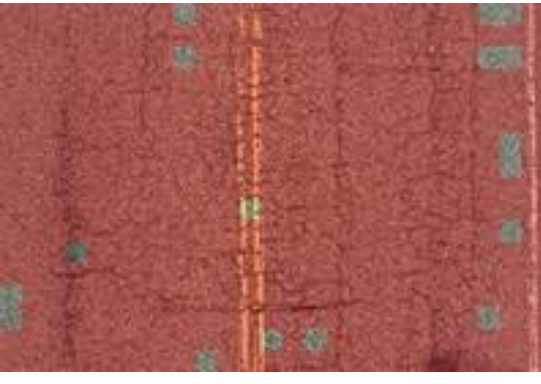
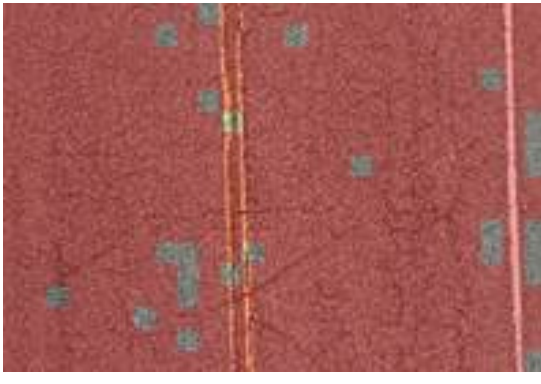

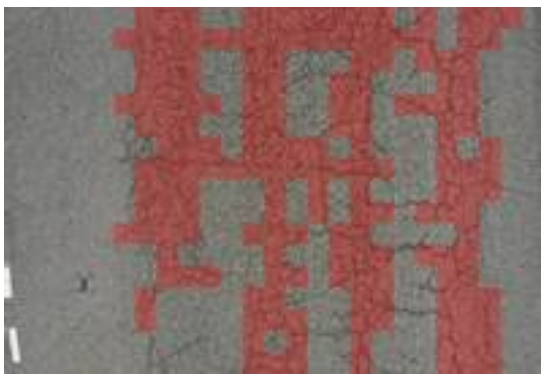
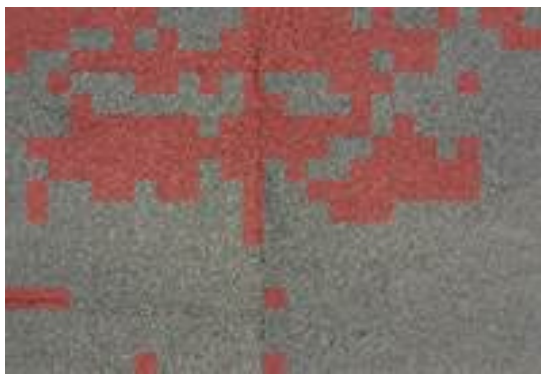


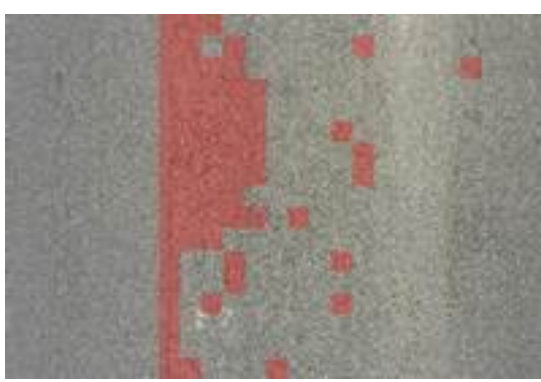
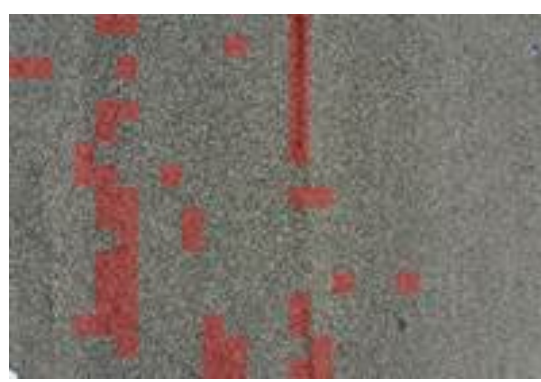





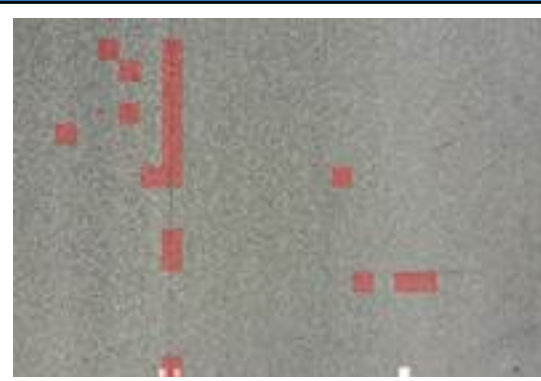
500Epoch模型

Test Accuracy為 0.9294，Test Loss為 0.2229。

模型訓練500Epoch後之評估指標^[注15]

| Folder Name | TP | TN | FP | FN | Recall | Precision | F1 Score |
|-------------|-----|-----|-----|----|--------|-----------|----------|
| 1 | 265 | 13 | 143 | 4 | 0.985 | 0.65 | 0.783 |
| 2 | 235 | 19 | 165 | 6 | 0.975 | 0.588 | 0.733 |
| 3 | 258 | 68 | 79 | 20 | 0.928 | 0.766 | 0.839 |
| 4 | 174 | 175 | 0 | 76 | 0.696 | 1 | 0.821 |
| 5 | 131 | 251 | 6 | 37 | 0.78 | 0.956 | 0.859 |
| 6 | 96 | 313 | 3 | 13 | 0.881 | 0.97 | 0.923 |
| 7 | 94 | 320 | 3 | 8 | 0.922 | 0.969 | 0.945 |
| 8 | 61 | 344 | 5 | 15 | 0.803 | 0.924 | 0.859 |
| 9 | 37 | 363 | 14 | 11 | 0.771 | 0.725 | 0.747 |
| 10 | 49 | 360 | 1 | 15 | 0.766 | 0.98 | 0.86 |
| 11 | 32 | 366 | 18 | 9 | 0.78 | 0.64 | 0.703 |
| 12 | 44 | 358 | 5 | 18 | 0.71 | 0.898 | 0.793 |
| 13 | 44 | 358 | 5 | 18 | 0.71 | 0.898 | 0.793 |
| 14 | 31 | 383 | 3 | 8 | 0.795 | 0.912 | 0.849 |
| 15 | 16 | 396 | 3 | 10 | 0.615 | 0.842 | 0.711 |
| F1 Score平均值 | | | | | | | 0.815 |

二、 滑動窗口結果^[注16]

| | | |
|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
|  |  |  |
| 1 | 2 | 3 |
|  |  |  |
| 4 | 5 | 6 |
|  |  |  |
| 7 | 8 | 9 |
|  |  |  |
| 10 | 11 | 12 |
|  |  |  |
| 13 | 14 | 15 |

三、 自動PCI值評估^[注17]

| Folder Name | jpg Count | Crack Percentage | Reduction Value1 | Reduction Value2 | Reduction Value3 | HDV | CDV | PCI | Rating |
|-------------|-----------|------------------|------------------|------------------|------------------|--------|--------|--------|-------------|
| 1 | 408 | 0.96 | 70.9 | 85.48 | 91.2 | 91.2 | 91.2 | 8.8 | 不合格 |
| 2 | 400 | 0.941 | 70.618 | 85.329 | 90.824 | 90.824 | 90.824 | 9.176 | 不合格 |
| 3 | 337 | 0.793 | 67.809 | 82.324 | 88.929 | 88.929 | 88.929 | 11.071 | 很差Very poor |
| 4 | 174 | 0.409 | 55.114 | 67.105 | 79.282 | 79.282 | 79.282 | 20.718 | 很差Very poor |
| 5 | 137 | 0.322 | 50.041 | 61.055 | 76.671 | 76.671 | 76.671 | 23.329 | 很差Very poor |
| 6 | 99 | 0.233 | 44.207 | 52.494 | 72.647 | 72.647 | 72.647 | 27.353 | 差Poor |
| 7 | 97 | 0.228 | 43.892 | 52.024 | 72.412 | 72.412 | 72.412 | 27.588 | 差Poor |
| 8 | 66 | 0.155 | 36.635 | 42.494 | 66.976 | 66.976 | 66.976 | 33.024 | 差Poor |
| 9 | 51 | 0.12 | 32.4 | 37.2 | 63.8 | 63.8 | 63.8 | 36.2 | 差Poor |
| 10 | 50 | 0.118 | 32.118 | 36.847 | 63.588 | 63.588 | 63.588 | 36.412 | 差Poor |
| 11 | 50 | 0.118 | 32.118 | 36.847 | 63.588 | 63.588 | 63.588 | 36.412 | 差Poor |
| 12 | 49 | 0.115 | 31.835 | 36.494 | 63.376 | 63.376 | 63.376 | 36.624 | 差Poor |
| 13 | 40 | 0.094 | 28.882 | 33.2 | 61.412 | 61.412 | 61.412 | 38.588 | 差Poor |
| 14 | 34 | 0.08 | 26.7 | 29.2 | 59 | 59 | 59 | 41 | 尚可Fair |
| 15 | 19 | 0.045 | 19.641 | 21.976 | 51.882 | 51.882 | 51.882 | 48.118 | 尚可Fair |

討論

一、 ResNet152模型訓練與滑窗分析討論

- 1 編號01與編號02其PCI評級為不合格，誤差百分比多達三成，編號03至05，其PCI評級為很差，誤差明顯較前述有較之低誤差值。 其餘十張照片誤差皆在正負3%以內，這些照片的裂縫誤差皆很小。 顯示路面狀況越差，造成我們的模型越容易誤判，若PCI評級在差(Poor)以上，則我們的模型有很高的準確率。
- 2 我們採用Resize方法來調整照片大小，該方法讓滑窗可以完整的檢查到整張照片有沒有裂縫。
- 3 為了簡化研究歷程，我們假設柔性鋪面裂縫與混凝土裂縫相似的前提下，採用既有的混凝土裂縫模型訓練相關參數設置，但研究過程發現其實二種是有相當程度差異的，建議有興趣的團隊對於此方面進行更為深入的研究。

二、 自動PCI值評估

- 1 我們是採用內插法的方式來計算出折減值，但內插法是有其誤差存在的，建議日後可以再針對鋪面損壞折減值開發出更精確的折減值計算法。
- 2 龜裂的值在三種裂縫中恆為最大，所以面層裂縫為重級(H)的狀態下，龜裂的形式是影響最大的。
- 3 研究證實我們的想法是實際可行的，日後只要訓練出更強大的模型，並依照先前的開發邏輯，即可讓我們的自動檢測柔性鋪面PCI值系統實際運用於業界。
- 4 我們全程採用開放框架、免費資源開發出這套系統，日後這套系統成熟時，能協助檢測員進行外業工作，節省大量時間。
- 5 近年來UAV大量使用在俄烏戰爭等戰場上，本系統除了前述可協助檢測員快速作業外。未來在戰場上當道路發生嚴重破壞，指揮單位無法使用車輛進行道路損壞評估時，使用UAV搭配本系統是一個可行的選項。

三、生成式AI的應用

團隊使用ChatGPT將GitHub上的現有程式移植到Google Colab上執行，再搭配提示工程的協助寫了三個小程式即完成整個系統，展現生成式 AI 在技術實踐中的強大輔助效益。

模型辨識裂縫與人工目視結果^[注18]

| Folder Name | A滑窗結果TP+ FP | 人工目視TP+ FN | 差值 | 誤差% | Rating |
|-------------|-------------|------------|-----|------|-------------|
| 1 | 408 | 269 | 139 | 33% | 不合格 |
| 2 | 400 | 241 | 159 | 37% | 不合格 |
| 3 | 337 | 278 | 59 | 14% | 很差Very poor |
| 4 | 174 | 250 | -76 | -18% | 很差Very poor |
| 5 | 137 | 168 | -31 | -7% | 很差Very poor |
| 6 | 99 | 109 | -10 | -2% | 差Poor |
| 7 | 97 | 102 | -5 | -1% | 差Poor |
| 8 | 66 | 76 | -10 | -2% | 差Poor |
| 9 | 51 | 48 | 3 | 1% | 差Poor |
| 10 | 50 | 64 | -14 | -3% | 差Poor |
| 11 | 50 | 41 | 9 | 2% | 差Poor |
| 12 | 49 | 62 | -13 | -3% | 差Poor |
| 13 | 40 | 26 | 14 | 3% | 差Poor |
| 14 | 34 | 39 | -5 | -1% | 尚可Fair |
| 15 | 19 | 26 | -7 | -2% | 尚可Fair |

結論

本研究以 ResNet152 模型實現道路裂縫的自動檢測，並根據內政部營建署「柔性鋪面損壞調查手冊」所訂評估方法，自動計算裂縫的 PCI 值。我們收集國內相關資料與文獻後，發覺團隊是**首先嘗試以深度學習模型結合政府規範來進行道路裂縫自動評估**，並在文章中證明了可行性，這是我們團隊最大的成果。