修 平 科 技 大 學

資訊管理系

具智慧辨識之培植自動檢測設計

組長：BF109013 黃銘凱
組員：BF109023 劉耀宗
BF109025 廖翊翔
BF109022 黃筑妍

指導教師：張兆村

中 華 民 國 一 百 一 十 三 年 一 月

摘要

本研究主要探討自動腳本軟體在系統設備狀態上自動檢測的應用，並將人工智慧(Artificial Intelligence)的深度學習(Deep Learning)的技術應用於散田培植系統的設備運作檢測，研究將以散田培植系統的軟體及控制裝置為探討案例，除了設計合適的測試架構及黑箱測試個案外，亦將透過自動腳本軟體(AutoIt)的實作，進行測試個案的自動執行與結果紀錄，以提升系統檢測的效益。

在自動測試的腳本設計中，除改善先前研究在散田培植系統的個案檢測技術外，為了改善散田培植系統的 LED 燈裝置組件的檢測正確性，運用 Python 程式設計發展卷積類神經網路(Convolutional Neural Network，或稱 CNN)，用於提升影像辨識分類的準確度，此外，自動測試腳本也增加擴充功能，包含啟動 Python 發展平台、載入卷積類神經網路模型、及取得影像辨識結果等，以增加自動測試的應用範圍。

目錄

圖目錄

表目錄

第一章 前言

隨著溫室效應和極端氣候的影響下，傳統農業的作業方式慢慢地朝科技應用邁進，由於溫室效應所導致的氣溫及海平面逐漸上升，已形成地球環境的嚴重傷害，而氣候異常更直接導致農作物品質難以穩定收成。全球 2050 年預估有 95~105 億人口，糧食需求將面臨增加一倍的壓力，又臺灣以熱量為基礎之糧食自給率 2017 年僅為 32.28%，在氣候變遷導致極端氣候 日趨嚴重的困境下，糧食供應短缺與糧價上升恐無可避免[1]。

國內農業在耕地面積狹小的不利條件下，面臨農業人口老化、缺工、全球化競爭與氣候變遷等問題，以小農為主體的農業目前面臨永續發展的挑戰。近年來農業逐漸透過智慧農業的科技協助，農業重要培植因素在自動感知與智慧調控下，逐漸降低培植者對農作物的勞力負擔，也提昇了農作物的品質，期望智慧農業將創造安全又便利的從農環境，吸引更多青農人力投入，使農業邁向「效率」、「安全」、「低風險」的新時代[1]。

在智慧農業的運作下，培植者只會安排少數時間來進行設備檢查，有些裝置的損壞會影響到作物培植的品質，因此，若能透過系統自動檢測的

方式來及時瞭解控制裝置的工作狀況，讓裝置異常狀況也能進行遠端的檢測，以降低控制裝置損壞所造成的風險，並提高人員檢測的效率。

1.1 研究動機

本專題旨在優化自動檢測系統品質保證的規劃與測試流程。先前研究[2]以探討腳本軟體應用於散田培植系統的設備運作檢測，透過以軟體及控制裝置為探討案例，實作出測試個案的自動執行與結果紀錄，以提升系統檢測的效益。然而，研究[2]提到散田培植系統的 LED 燈的檢測方式，是以外掛影像辨識程式方式對燈管的工作數量進行估算，準確率大約在 70%，由於光線對於蔬菜培植至關重要，若能正確且及時發現 LED 燈管損壞，將可儘速進行維修，降低設備損害對培植的影響，更可提升遠端自動檢測功能的實用性。

人工智慧的卷積類神經網路已被公認適合用於影像辨識的分類[3]，因此，本專題嘗試將研究[2]的自動測試腳本測試功能進行擴充，包含啟動 Python 發展平台、載入卷積類神經網路(CNN)模型、及取得 CNN 影像辨識結果等，取代原先的影像辨識程式，提升燈管工作數量的估算準確率。

1.2 研究目的

本研究旨在探討智慧型系統品質保證的規劃和實施，並透過軟硬體整合的方式進行智慧型自動檢測，提早發現元件故障問題，降低蔬菜培植的風險。運用人工智慧的深度學習的影像辨識技術，並結合軟體測試過程和方法對軟體系統發展的品質具有重要影響力。本研究將以軟體自動檢測為主要方向，使用物聯網軟體(如 Arduino)和自動腳本軟體(如 AutoIt v.3)進行培植自動檢測軟體的測試規劃和實作驗證，並將深度學習的卷積類神經網路的影像辨識分類技術應用於培植自動檢測軟體的測試和驗證中。因此，本研究旨在達成以下目標：

1.通過文獻整理，瞭解培植檢測軟體的發展過程。

2.提出文獻整理的自動檢測軟體規劃和個案設計方法。

3.探討自動檢測軟體的設計規則，並進行適當的測試個案設計。

4.利用物聯網軟體(Arduino)和自動腳本軟體(AutoIt)進行自動測試腳本程式設計，並測試培植系統自動檢測個案軟體的正確性。

5.利用深度學習的 CNN 影像辨識分類技術與自動測試腳本程式設計整合，以提升植物燈管工作狀態的辨識準確性。

6. 發展自動腳本軟體啟動 Python 發展平台、載入卷積類神經網路模型、及取得 CNN 影像辨識結果等控制的設計。

藉由智慧辨識技術的自動檢測機制設計，將提供系統品質保證的方法和工具，應用於確保蔬菜培植的品質和穩定性。

第二章 文獻探討

本章將介紹專題研究的相關文獻參考，透過資料的研究瞭解相關問題的基礎知識，以便於思考如何在智慧辨識之培植系統上增加遠端自動檢測的功能。

2-1 系統品質保證與測試

軟體品質保證(Software Quality Assurance，簡稱 SQA)是監控軟體工程流程和方法以確保品質的一系列手段[4]。資訊系統的品質成本分佈中，預防成本佔 4%，鑑定成本佔 10.5%，而因應使用者抗議與抱怨的失敗成本卻高達 85.5%。為了降低整體品質成本，需加強投注於預防成本，以避免缺陷存在，從而降低失敗成本[5]。資訊系統與硬體產品存在本質差異，資訊系統的失誤主要出現於程式流程中，而非元件衰退。現有的品質保證方法難以直接套用於資訊系統，需要系統性自動檢測應用元件，以快速找出問題點並提供排除參考。

在資訊系統發展過程中，驗證和測試是重要的步驟，用於確保系統設

計符合需求並確保硬體環境的正確運作。系統通過測試後，需進行移轉給業務單位使用，並持續進行使用者疑難排除、修正和改善。本研究主要探討運用自動檢測方法偵測元件衰退或損壞所造成的事件，藉以適時降低培植系統的失效影響。

2-2 黑箱測試技術

黑箱測試是一種軟體測試技術，專注於檢查軟體的功能，而無需查看內部結構或程式碼[6]，其測試的主要依據是客戶宣告的需求規範。在這種方法中，測試人員選擇一個功能，提供輸入值並檢查其功能，測試通過的標準是該函數是否產生預期輸出，測試結果會報告給開發團隊進行修正。

黑箱測試的測試個案設計不需要程式設計知識，測試人員知道特定輸入應該產生的輸出，但不清楚內部實現細節，功能測試或非功能測試皆可適用。此類測試技術包含決策表、邊界值分析、狀態轉換、因果圖、等價分割、錯誤猜測、用例和用戶故事等方法。

2-3 自動測試工具

自動測試工具是用於自動化測試過程的軟件工具，可以幫助測試團隊快速、有效地執行測試。這些工具可以模擬用戶的行為，自動執行測試用例，檢查系統的功能和性能，並生成測試報告。軟體測試作業內，自動偵測是能夠瞄準目標程式進行測試，且將資料收集的測試軟體，能夠代替人力進行重複性但相當重要的測試工作，對於需持續改變與優化的資訊整合系統更是不可或缺的存在。

自動測試[7]擅長運用在需要大量重複，又或者需要快速比對多項結果的測試工作，且能夠在測試目標改良完後，立即的進行檢查及偵測，並且擁有快速新增測試項目的能力，一方面減低了人力上的消耗，也大大的增加了對時間的利用效率，對於需要時常收集資訊並整合的資訊系統來說非常有效。

表 2-3-1 自動化測試工具比較表

產品	發行年份	支援平台	腳本語言	優點/缺點	費用
AutoIt [2]	2011	Windows	Basic	**優點:**語法較簡單易懂，不具寫程式經驗的人亦可容易上手。 **缺點:** 與部份應用程序可能會有整合限制，尤其是需要與其他語言或工具進行整合時。	免費

Postman [9]	2012	Windows	Windows、Mac 和 Linux	**優點:**具有 Windows、Mac 和 Linux 等不同版本，開發人員可以在不同的操作系統上使用。 **缺點:** 在大量處理負擔下，可能會有性能限制，尤其是免費版。	免費
Katalon Studio [10]	2015	Windows Linux Mac	Java, C#, Groovy, PHP, Python, Ruby,Perl	**優點:**提供了直觀的用戶界面和可視化操作，使初學者能夠快速上手並開始創建測試用例。 **缺點:** 對於大型測試套件或需要處理大量數據的測試案例，性能可能會受到限制。	免費
JMeter [11]	1998	Windows Linux Mac	Java,	**優點:**提供圖形化的使用介面，便於 user 創建和管理測試腳本，並可產生較完整的測試報告。 **缺點:** 雖可以模擬多個用戶的訪問，但無法完全模擬真實瀏覽器的行為，可能導致某些情況下的測試結果與實際情況有所差異。	免費

自動測試經常被用以進行大量重覆的測試過程，尤其是回歸測試(Regression Testing)[12]，回歸是軟體測試的一種，旨在檢驗軟體原有功能在修改後是否保持完整，當軟體一有變更時，必須重新測試現有功能，以便確認修改、新加入的部分沒有影響到既有功能，且達到預期的目的。這類測試須根據測試目標來設計一系列的設計測試流程與個案，包括但不限於其依照原目的所設計的合法及非合法操作，將其製作成固定的測試流程，並交由測試工具進行自動測試，同時於測時途中，記錄下其操作所反應的

結果，判斷結果是否符合預期，透過電腦能夠不斷地運行測試案例，相較於人工測試過程往往能夠大幅縮減人工成本與精確度。表 2-3-1 說明自動化測試工具的比較[9]，表內依工具產品的年份、發行年份、支援平台、腳本語言、優點/缺點、及費用等來說明及分析個別的特性。

2-4 AutoIt 腳本軟體

AutoIt 是一個用於自動化 Windows 系統的強大工具，其主要功能包括：1.鍵盤和滑鼠操作記錄與回放，可以記錄使用者對鍵盤和滑鼠的操作，然後再次播放操作過程，這對於執行重複性任務或測試流程非常有用。2.AutoIt 提供了自己的腳本語言，允許您編寫複雜的自動化腳本，用於處理特定的任務或流程。

AutoIt 腳本文件副檔名皆為.au3，可以直接進行編寫，相當便利。執行 AutoIt 腳本前必須先安裝 AutoIt 軟體，可以雙擊腳本文件直接執行，或按右鍵選擇相關操作，如 open(開檔)、edit(編輯)或 compile script(編譯腳本)，若要在未安裝 AutoIt 的電腦上執行腳本文件，可以先將.au3 腳本文件編譯成可獨立運行的.exe 檔案，即可在電腦上執行腳本控制程序。

本專題將探討如何設計 AutoIt 腳本文件進行散田培植系統的裝置異常檢測，包含打開散田監控軟體、查詢資料庫資料、及呼叫輔助資源軟體，其中輔助資源軟體有 Jupyter Notebook (Python)、載入卷積類神經網路模型、執行辨識分類、及讀回辨識分類結果等控制，檢測個案腳本及測試結果皆以 Excel 檔案管理，以方便達到確認測試個案項目的正確性。

2-5 深度學習與智慧辨識

深度學習(Deep Learning)[13]是一種機器學習的方法，它模仿人類大腦中神經元的工作方式，通過層層堆疊的神經網絡來學習和理解複雜的模式和特徵。深度學習已被廣泛應用於智慧辯識，智慧辯識即是指機器通過學習和模式識別來識別和理解人類的語音、文字、影像等信息，並做出相應的處理和判斷，準確度普遍已達生活應用之需求。深度學習在智慧辯識領域有著廣泛的應用，例如語音識別、自然語言處理、圖像識別等方面。通過深度學習模型的訓練，機器可以不斷地從大量的數據中學習和提取特徵，進而實現對語音、文字、影像等信息的準確識別和理解，這使得智慧辯識技術在生活和工作中得到了廣泛的應用，並為人們的生活和工作帶來了更多的便利和效率。

本專題將嘗試建立卷積類神經網路模型、訓練、及測試過程，對散田培植系統的燈管工作狀態進行辨識分類，在培植自動檢測中透過回傳照片辨識燈管是否損壞，達到培植系統設備狀態的更有效掌握。

2-6 散田監控軟體介紹

散田培植系統以智農科技實現「分散式小田地」的蔬菜培植模式，具全天候、分散部署、組合式培植面積、及遠程監控之特色，並可達到數據蒐集/分析與自動培植之功能。圖 2-6-1 為系統之運作架構圖[2]，主要功能包含七個部分：培植屋、灌溉子系統、感測與控制子系統、IOT 監控伺服器子系統、監控管理子系統、養分供給子系統、及行動管理子系統等。

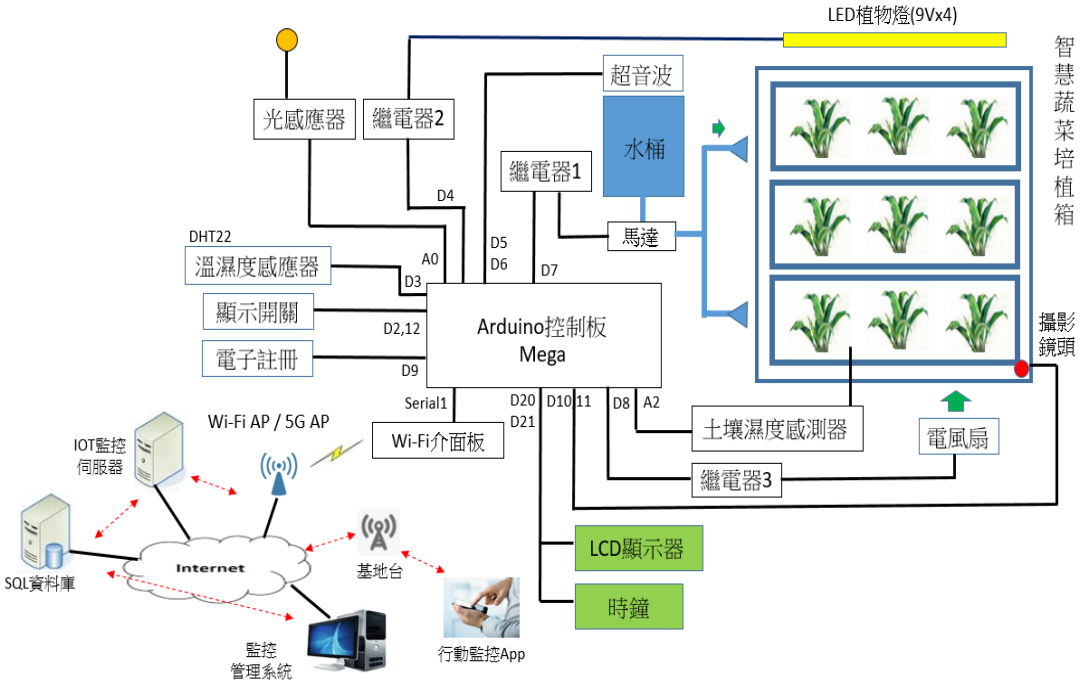


圖 2-6-1 散田培植系統架構圖

圖 2-6-2 描述培植節點的控制流程，感測與控制子系統(以下簡稱培植節點)的控制流程主要涵蓋下列四個重點：

1.節點韌體的一致性：監控伺服器可能會與服務許多培植節點，該些節點會被分區且被用於培植不同植物，若培植參數(如噴水量、光度...)皆直接設定於程式碼內，將增加節點韌體版本的複雜化，造成管理困難。

2.節點對環境的容錯性：培植節點以 Wi-Fi 無線連結與監控伺服器進行溝通，因此，可能會遇到許多環境的失誤而影響節點正常運作，例如停電、無線網路無法連結、缺水...等因素，培植節點必須正常運作，不受到環境失誤的影響。

3.節能與靈敏度的調適：培植節點以 Arduino 控制板為發展基礎，因此，在節能考量下，通常會以醒睡週期來設計節點與監控伺服器的會面時間，醒睡週期越短則越耗電，反之，醒睡週期越長則通訊越不靈敏。

4.定時控制：培植節點提供定時噴水及拍照的功能，同時為降低計時功能的成本，節點控制流程將採伺服器時間校時方式進行設計，安排在每次醒

睡週期達到時間同步，離線時則採邏輯計時，已達到節點的定時控制目標。

培植節點的控制流程大致分為兩部分，一為偵測與控制部分，另一為連線溝通部分，無法連線時，亦不會影響節點的偵測與控制功能。節能與靈敏度的調適由醒睡週期負責，管理者可以再利用監控管理系統對節點進行醒睡週期的動態調整，即不同培植蔬果種類可以設定不同的醒睡週期，使節點的節能與靈敏度得以充分符合管理者的需求。

圖 2-6-2 監控伺服器的作業畫面

圖 2-6-2 顯示監控伺服器的作業畫面，系統的培植節點會依醒睡週期長短，定期與監控伺服器以 Wi-Fi 連線，系統是以 VB.Net 2010 開發監控伺服

器。為了管理伺服器的便利，亦提供連線數、資料傳送進度、培植節點接收及控制資料的查詢介面，包含節點拍攝照片的瀏覽與存檔功能。

系統的監控管理系統主要提供系統管理者、區域管理者及培植節點管理者使用，用以觀察培植節點所收集的感測資料(含生長照片)，包含帳號管理、記錄查詢、資料統計、節點控制、生產過程及維護管理等六種功能。監控管理系統的記錄查詢畫面如圖 2-6-3 所示。記錄查詢功能可以查詢培植節點所收集的感測資料(含生長照片)，所收集的感測資料包含濕度(HMD)、溫度(TMP)、光度(LGH)、水量高(LEL)、土壤濕度(EHM)、關燈時間(LFT)、開燈時間(LOT)、關燈時間(LFT)、噴水開始(WOT)、噴水結束(WFT)、無水可噴(WNT)、影像(IMG)、連線開始(NID)及連線結束(CLS)。

圖 2-6-3 監控管理系統的記錄查詢畫面

第三章 專案規劃與設計基礎

本章介紹實務專題的相關工作，包含工作分配與進度、腳本設計基礎、設計輔助資源、與腳本架構設計等說明，以做為本專題案例實作的基礎背景。

表 3-1-1 工作分配表

專題成員	負責工作內容
黃銘凱	資料蒐集、軟體安裝、文獻資料收集、腳本撰寫及測試、CNN 程式撰寫及測試、書面資料整理、報告 PPT 撰寫
劉耀宗	資料蒐集、發展環境準備、腳本撰寫及測試、腳本撰寫與測試、CNN 模型訓練、書面資料整理、報告 PPT 撰寫
廖翊翔	資料蒐集、發展環境準備、CNN 程式撰寫及測試、散田資料集整理、腳本撰寫與測試、書面資料整理、報告 PPT 撰寫
黃筑妍	資料蒐集、文獻資料收集、散田資料集整理、CNN 模型訓練、書面資料整理、訓練課程資料整理

3-1 工作分配

本團隊為了有效達成研究目標，故分別依照組員的專長屬性以安排個人負責事項，最後再進行個別成果的統整。團隊成員的工作分配表如表 3-1-1 所示。

3-2 工作進度

本專題研究的期程自 112 年 05 月至 112 年 12 月，透過三下學期課程對 AutoIt 腳本軟體的學習與瞭解，緊接著進行研究議題的案例探討與實作，主要工作包含蒐集資料、發展環境準備、設計案例學習、程式撰寫和測試、設計案例與學習、資料集建構、專題報告書面製作、及報告 PPT 撰寫。如表 3-2-1 所示。

表 3-2-1 工作進度表

3-3 AutoIt 腳本設計基礎

學習 AutoIt 腳本軟體的基礎功能對於開發自動測試案例是相當重要的，

基礎功能包含專案目錄指定與檔案管理、腳本中斷處理、固定視窗大小與位置、測試軟體啟動、鍵盤與滑鼠控制、及輸入法切換等。本次測試案例的個案設計與結果輸出皆採用 Excel 檔案為基礎，因此，利用 AutoIt 腳本進行 Excel 檔案資料的讀寫控制相形重要。以下將介紹這些基礎功能。

1.腳本中斷處理：腳本程式在執行時，出現錯誤卻不能及時終止，會導致腳本執行測試的時間變長，所以首先要設定腳本中斷處理指令，使用者可以利用 HotKeySet( )來設定快捷熱鍵來達成目的。

2.專案目錄指定與檔案管理：

為了支持測試案例能達到批次化自動處理，測試時可以僅指定專案目錄，腳本程式可以採批次方式將測試個案的描述檔案開啟並逐一執行，因此，專案目錄名稱命名規則為「專案名_序號」，例如: pr9_001 表示專案名為 pr9 而版本序號為 001。為了方便使用者指定專案目錄，設計者可以利用 FileSelectFolder("Select a folder", "")讓使用者來選擇開啟目錄位置。

為了批次管理專案目錄內的測試個案檔案，腳本程式可以用

_FileListToArray( )設定變數，將目錄內檔案(Excel 檔)名稱分別讀取到陣列，再使用字串分割來擷取目錄名稱及檔案編號，測試個案檔案名稱命名規則為「專案名_in-序號」。專案所屬測試個案檔案會依找尋的順序來進行啟動，腳本也須將結果放置另一檔案中供使用者確認，先執行函數proc_scr( )傳入專案名稱、序號、及副檔名，再使用迴圈自動確認專案訊息，產生對應的測試個案輸出檔案名稱，檔名命名規則為「專案名_out-序號」，並且當查無測試個案後，自動結束迴圈。而為了進行 Excel 檔案的開啟、新建活頁簿、關閉等動作，會分別使用 AutoIt 所提供的_Excel_Open( )、_Excel_BookNew( )及_Excel_Close( )腳本指令。

3-4 專案腳本架構與控制

本節將介紹本專題測試個案的命令與參數架構設計，架構都是以 Excel 的 SLXS 檔案格式為基礎，每一列分別表示一個測試個案，測試個案的命令與參數將分別介紹如后。

圖 3-4-1 測試個案的架構設計說明

圖 3-4-1 描述本專題所設計的測試個案架構之設計說明，測試個案的命令與參數的表示共分為四區，

1.第一區範圍(欄位 A~W)表示為測試命令及參數，其中英文字部分描述測試命令，而數字部分表示測試的相關參數。

2.第二區範圍(欄位 X)表示為預期結果。

3.第三區範圍(欄位 Y)表示為實際結果。

4.第四區範圍(欄位 Z)表示為測試個案的是否通過，當測試影像檔時，預期結果與實際結果相同會判定為 PASS，而其他測試部分則預期結果比實際結果小時會判定為 FAIL。

本次專題以散田培植系統的 LED 燈、繼電器、及馬達等裝置組件為檢測對象，檢測的分式必須透過安裝而外的感測元件來達成，圖 3-4-2 描述散田培植系統的感應回報資料表定義，而散田培植系統的拍照影像檔則存在圖 3-4-3。散田培植系統採用遠端監控式培植蔬菜，然而部分控制元件若無法正確工作，且未能及時發現這些錯誤，蔬菜的生長勢必受到影響，其中包含下列情況:

(1) 控制 LED 燈的繼電器開啟，但繼電器未正常工作，造成 LED 燈未接通電源，無法提供蔬菜所需的光線，而無法行光合作用。

(2) 控制 LED 燈的繼電器開啟，繼電器有正常工作，但 LED 燈管有部分損壞不亮，無法提供蔬菜足夠的光線，而造成蔬菜光合作用不足，影響正常生長。

(3) 控制馬達的繼電器開啟，但繼電器未正常工作，造成馬達未接通電源，無法提供蔬菜所需的滴灌水分，而造成蔬菜缺少水分不足，影響正常生長。

(4) 控制馬達的繼電器開啟，繼電器有正常工作，但馬達損壞無法正常運作，無法提供蔬菜所需的滴灌水分，而造成蔬菜缺少水分不足，影響

正常生長。

(2) 影像資料表(Sen_img)

意義	名稱	型態	大小	說明
影像編號	Img_id	數字	int	自動編號
節點編號	Nid	文字	Varchar(20)	節點編號
影像檔	Pic	Image		不可 NULL

圖 3-4-2 散田培植系統的拍照影像資料表定義

上述的異常狀況是無法透過散田培植系統的監控軟體發現繼電器或馬達損壞的資訊，雖然可以透過拍照影像檔發現 LED 燈管是否正常，但散田培植系統的培植區域與控制點較多，培植者要利用監控軟體逐個影像檔的目視檢查，將會是件繁瑣的工作。

(21) 延伸感應表(Sen_ext)

意義	名稱	型態	大小	說明
記錄編號	Rec_id	數字	int	自動編號
類型	Dclass	數字	tinyint	感應資料類型, 如: 1:開燈繼電器光感值, 2:噴水繼電器光感值, 3:馬達震動值
資料內容	Sdata	數字	int	

圖 3-4-3 檢測延伸感應表定義

本專題為了檢測出上列系統執行時的異常狀況，將採用光線感測器及三軸加速器作為感測延伸數據，也就是在現有散田培植系統的 LED 燈控制繼

電器及馬達控制繼電器分別上加裝一個光線感測器，偵測繼電器工作時是否有亮燈，搭配系統的開燈時間控制並同時記錄繼電器工作時的亮燈感應值，圖 3-4-3 描述本專題所設計的檢測延伸感應表定義。如圖 3-4-3，每一筆紀錄將紀錄感測類型在 Dclass 欄，而感應數值將存於 Sdata 欄內，其中圖 3-4-2 的 Rec_id 欄將被用來連結 LED 燈控制繼電器、馬達控制繼電器、及馬達三軸加速器的感應值，至於 LED 燈管的亮燈數量則透過圖 3-5-2 的 Sdata 欄連結圖 3-5-3 的 Img_id 欄來取得拍照影像檔，並進一步分析燈管亮起數量。

本專題為了檢測異常狀況，設計出共 12 種腳本命令，分別是 AR(培植區域)、ND(控制器編號)、DAT(資料類型)、DS(查詢開始日期)、DE(查詢結束日期)、TS(查詢開始時間)、TE(查詢結束時間)、QRY(查詢紀錄)、GET(擷取紀錄編號)、GEG(擷取影像編號)、TSD(查詢延伸感應值)、及 TSG(檢測影像內燈管工作數量)，該些命令可能會帶有參數，會配合腳本命令一起執行，AR 後會有區域編號、ND 後會有控制器編號、DAT 後會有資料類型(如噴水時間)、DS 後會有年月日資料(如 20221204)、DE 後會有年月日資料(如 20221204)、TS 後會有時間資料(如 6)、TE 後會有時間資料(如 17)、

QRY 後無參數、GET 後無參數、GEG 後無參數、TSD 後會有查詢類型資料、及 TSG 後會有檢測特徵線的 y 軸位置，腳本命令與參數如何實現測試案例的描述，將於下一章進行實作範例說明。

圖 3-5-1 延伸感應值查詢程式

3-5 輔助設計資源

AutoIt v3 提供了一個視窗訊息工具(AutoIt Window Info Tool)獨立的工具，稱為 AU3Info。此工具允許使用者可以獲取指定視窗的相關訊息，以便於有效地實現自動化操作。視窗的訊息主要包括視窗標題(Window titles)、視窗的文本文字(包括可見部分和不可見部分)、視窗的大小和坐標位置、狀態欄的內容、滑鼠的座標位置、滑鼠座標的象素顏色值、滑鼠經過的控件(Control)的相關信息等。AU3Info 工作視窗會保持在所有視窗的最上層，以便使用者能看到擷取的信息內容。透過點擊感興趣的視窗，則 AU3Info 即可獲取該視窗的相關信息並顯示出來，該些訊息有助於實現自動化腳本對視窗元件的控制。

此外，本專題為了檢測出散田培植系統的 LED 燈、繼電器、及馬達等裝置異常狀況，檢測的方式必須透過安裝而外的感測元件來達成，圖 3-5-1 描述本專題所設計的檢測延伸感應表定義。當我們利用腳本程式啟動散田監控系統查詢紀錄時，可以透過輔助資源(延伸感應值查詢程式)來擷取對應紀錄編號的延伸感應值，而資料類型則由腳本命令的參數填入，圖 3-5-1 顯示延伸感應值查詢程式的畫面，圖 3-5-1 描述紀錄編號(15757643)的延伸感應值，即開燈繼電器光感值為 70，延伸感應值查詢程式會用於連結 LED 燈控制繼電器、馬達控制繼電器、及馬達三軸加速器的感應值，以達到輔助測試的正確性。

圖 3-5-2 燈管照片轉存程式

本專題使用另一個燈管照片轉存程式當做輔助資源，燈管照片轉存程式如圖 3-5-2 所示，當我們利用腳本程式啟動散田監控系統查詢紀錄時，可以

透過燈管照片轉存程式來擷取對應影像編號，先透過按下「載入」按鍵進行影像檔顯示，然後再填入轉存的目錄位置，此目錄位置是卷積類神經網路進行辨識的位置，採固定檔名為 test.jpg，以方便與卷積類神經網路辨識分類的程式溝通。稍後會啟動卷積類神經網路辨識分類的程式進行處理，傳回照片內的燈管亮燈數量。

第四章 實作範例說明

本章介紹如何執行自動檢測系統中的四種異常狀況，包括(1)控制 LED 燈的繼電器開啟，但繼電器未正常執行、(2)控制 LED 燈的繼電器開啟，繼電器有正常執行，燈管亮燈數是否正常、(3)控制馬達的繼電器開啟，但繼電器未正常執行、(4)控制馬達的繼電器開啟，繼電器有正常執行等。本次專題 AutoIt v3 和 Excel 進行腳本程式與個案設計，以示範 AutoIt 在實現自動檢測系統上的實際應用與成果。

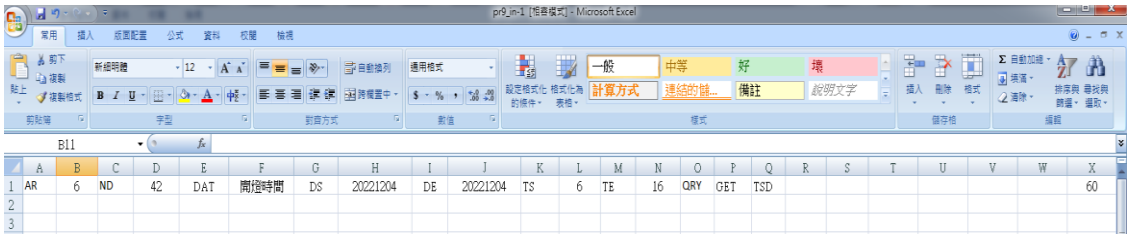


圖 4-1-1「開燈繼電器」自動檢測的測試個案範例

4.1 「開燈繼電器」自動測試

首先，說明「開燈繼電器」自動測試的設計構想。腳本程式啟動自動測試系統軟體，並以滑鼠及鍵盤控制帳號及密碼輸入。接著，開啟 Excel 測試案例檔案，逐列讀入一個測試個案，包含多個腳本命令、參數及預期結

果。「開燈繼電器」自動檢測使用 10 種腳本命令，包括：AR (培植區域)、ND (控制器編號)、DAT (資料類型)、DS (查詢開始日期)、DE (查詢結束日期)、TS (查詢開始時間)、TE (查詢結束時間)、QRY (查詢紀錄)、GET (擷取紀錄編號)、TSD (查詢延伸感應值)。

這些腳本命令用於指導自動測試系統執行不同任務，包括區域設定、控制器編號、資料類型、日期與時間查詢，以及紀錄擷取和延伸感應值查詢等。透過這些腳本命令，腳本程式模擬不同使用情境，以確保自動檢測系統的正確性和可靠性。

圖 4-1-1 顯示了本項自動檢測的一個測試個案，描述了點選第 6 號培植區的第 42 號控制器，並設定查詢資料類型為「開燈時間」。然後，選擇查詢監控記錄的開始日期及結束日期皆為 20221204，開始時間設為 6 點，結束時間設為 16 點，按下「查詢」按鍵進行查詢。當資料顯示列表後，僅點選第一筆的紀錄編號欄位，隨後擷取紀錄編號資料，如圖 4-1-2 所示。

最後，開啟「延伸感應值查詢程式」填入紀錄編號及資料類型為 1(開

燈繼電器光感值)，點選查詢後，查到感應值為 70。將此結果值填入結果檔案 Excel 的 Y 欄，此結果會與 X 欄的預期結果相較。X 欄預期結果值表示繼點器量燈的光感值的最低容許值。

編號	節點IP	Port	節點Id	類型	資料內容	時間
15757643	1. 200. 15. . .	20295	42	開燈時間	2022/12/4 7:1:33	2022/12/4 上午 07:
15757706	1. 200. 15. . .	20932	42	開燈時間	2022/12/4 7:7:26	2022/12/4 上午 07:
15757760	1. 200. 15. . .	20673	42	開燈時間	2022/12/4 7:15:40	2022/12/4 上午 07:
15757814	1. 200. 15. . .	20429	42	開燈時間	2022/12/4 7:23:53	2022/12/4 上午 07:
15757868	1. 200. 15. . .	20579	42	開燈時間	2022/12/4 7:32:6	2022/12/4 上午 07:
15757922	1. 200. 15. . .	20852	42	開燈時間	2022/12/4 7:40:19	2022/12/4 上午 07:
15757976	1. 200. 15. . .	20157	42	開燈時間	2022/12/4 7:48:32	2022/12/4 上午 07:
15758030	1. 200. 15. . .	20516	42	開燈時間	2022/12/4 7:56:46	2022/12/4 上午 07:
15758085	1. 200. 15. . .	20329	42	開燈時間	2022/12/4 8:4:59	2022/12/4 上午 08:
15758139	1. 200. 15. . .	20943	42	開燈時間	2022/12/4 8:13:12	2022/12/4 上午 08:
15758193	1. 200. 15. . .	20629	42	開燈時間	2022/12/4 8:21:25	2022/12/4 上午 08:
15758247	1. 200. 15. . .	20795	42	開燈時間	2022/12/4 8:29:39	2022/12/4 上午 08:
15758301	1. 200. 15. . .	20833	42	開燈時間	2022/12/4 8:37:52	2022/12/4 上午 08:
15758355	1. 200. 15. . .	20445	42	開燈時間	2022/12/4 8:46:5	2022/12/4 上午 08:
15758409	1. 200. 15. . .	20441	42	開燈時間	2022/12/4 8:54:18	2022/12/4 上午 08:

圖 4-1-2 「開燈繼電器」檢測的散田監控紀錄查詢

在這個範例中，「開燈繼電器」自動檢測僅透過擷取第一筆的紀錄編號，主要透過抽樣方式進行檢測。這樣的方法提高了檢測效率，因為並非每個開燈時間都需要進行測試。若在抽樣檢測後發現開燈繼電器損壞，最多只會延遲一天發現裝置異常，這種狀況並不會對植物的生長產生影響。

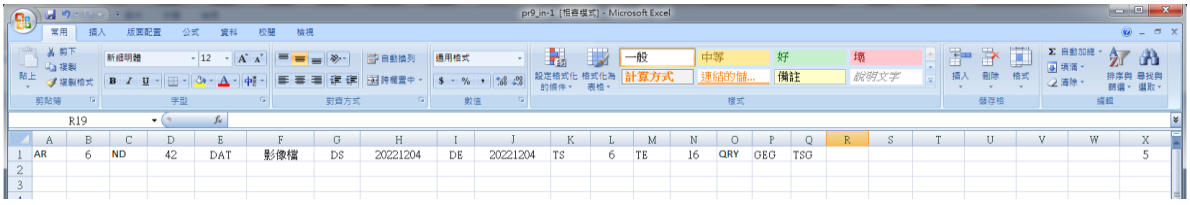


圖 4-1-3 「LED 燈亮燈」自動檢測的測試個案範例

4.2 「**LED** 燈亮燈」自動測試

本節介紹「LED 燈亮燈」自動測試的設計構想。腳本程式先啟動散田培植系統軟體，並控制滑鼠及鍵盤進行帳號及密碼的輸入。然後開啟 Excel 的測試案例檔案，逐列讀入一個測試個案，包含多個腳本命令、參數及預期結果。「LED 燈亮燈」自動檢測使用到 10 種腳本命令，分別為 AR(培植區域)、ND(控制器編號)、DAT(資料類型)、DS(查詢開始日期)、DE(查詢結束日期)、TS(查詢開始時間)、TE(查詢結束時間)、QRY(查詢紀錄)、GEG(擷取影像編號)、及 TSG(檢測影像內燈管工作數量)等。

4.2.1 腳本程式啟動辨識分類

圖 4-1-3 顯示一個測試個案，選擇第 6 號培植區的第 42 號控制器，查詢資料類型為「影像檔」，監控記錄的開始日期及結束日期皆為 20221204，開始時間 6 點，結束時間 16 點。按下「查詢」後，顯示列表，點選第一筆資料，擷取影像編號資料，如圖 4-2-1 所示。接著 GEG 腳本命令會開啟「燈管照片轉存程式」，填入影像編號及轉存的目錄位置，此目錄位置是卷積類神經網路進行辨識的位置，點選「下載」以「顯示影像」，再點選「存

檔」，採固定檔名為 test.jpg，以方便與卷積類神經網路辨識分類的程式溝通。TSG 腳本命令會檢測影像內燈管工作數量，AutoIt 會啟動 Jupyter Notebook 平台，然後載入 TF2_LEDcheck2.ipynb 程式，並啟動卷積類神經網路的燈管辨識分類。擷取比對結果欄內容，將此結果值填入結果檔案 Excel 的 Y 欄。結果與 X 欄的預期結果相較，X 欄預期結果值表示 LED 燈亮燈數應有多少隻，範例中亮燈數也是 5，表示與預期相符。

圖 4-2-1「LED 燈亮燈」檢測的散田監控紀錄查詢

在這個範例中，「LED 燈亮燈」自動檢測僅透過擷取第一筆的資料內容(影像編號)，主要透過抽樣方式進行檢測。並非每個開燈時間都能觀察到 LED 燈亮燈的影像。若在抽樣檢測後發現 LED 燈損壞，最多會延遲一天發現裝置異常。即便有部分 LED 燈損壞，也不會對植物的生長產生影響。

4.2.2 卷積類神經網路辨識分類

深度學習是人工智慧中成長最快的領域，主要是透過模擬人類神經網路的運作方式，進而學習龐大資料的內涵 [14]。本專題運用卷積類神經網路以提升研究[2]所提的燈管辨識的準確度，達到降低散田培植自動檢測燈管失效的影響。以下分別說明發展的步驟。

1. 發展環境安裝

本專題運用 Anaconda 發展平台進行卷積類神經網路辨識分類的應用，以下介紹安裝深度學習框架(Framework)的步驟，首先到 Anaconda 官方網站下載安裝軟體，並安裝 Anaconda (Windows10)虛擬開發環境，然後檢查電腦的 GPU 型號，必須依據 GPU 型號來下載適合的 Cuda 安裝套件(主要是支援到 Compute Capability 等級)，接著建立 Conda 環境，Conda 會建立及管理虛擬環境，包含 Jupyter Notebook。最後，再安裝 framework 套件，包含 tensorflow 及 keras，成功安裝 tensorflow 後，使用 tensorflow 時若機器有裝 GPU，tensorflow 會自動啟動 GPU，加速類神經網路的訓練過程。

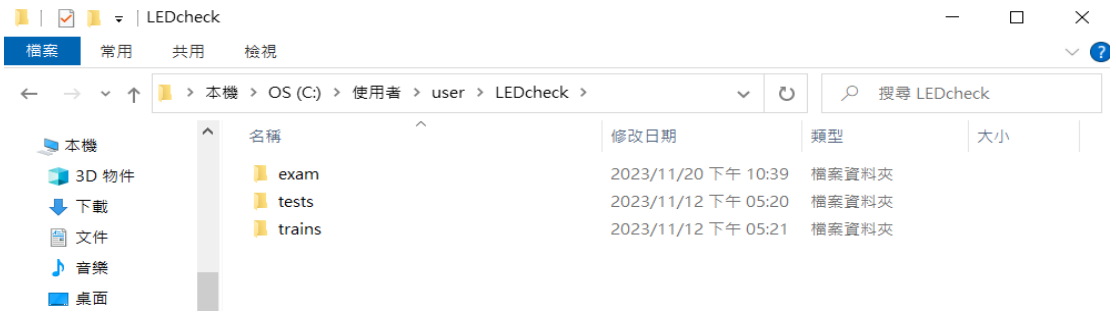


圖 4-2-2 燈管辨識分類資料集結構

2. 準備資料集

燈管辨識分類是採監督學習(Supervised learning)，模式需事先給定機器一些訓練樣本及樣本的類別(標籤)。本專題蒐集 LED 燈亮燈的影像分成兩類：四支或五支燈管兩類，因此我們在 Jupyter Notebook 的發展根目錄上建立資料集的目錄 LEDCheck，並在目錄內分別建立訓練及測試的子目錄：trains 及 tests。子目錄內分別包含四支或五支燈管兩類的照片，如圖 4-2-3 所示。子目錄 exam 則是自動燈管辨識的儲存區域。four 開頭的照片是五支燈管的檔案而 four 開頭的照片則是四支燈管的檔案，四支或五支燈管的照片類別(標籤)則分別設為 0 或 1，訓練照片及測試照片分別為 281 及 8 張。

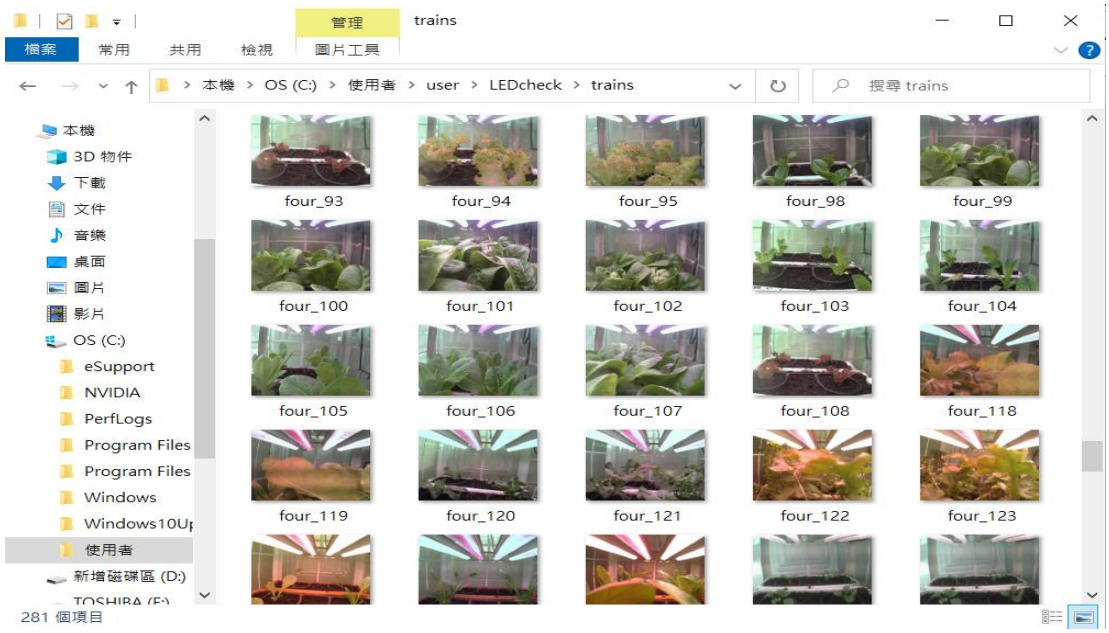


圖 4-2-3 四支燈管的辨識分類資料集內容

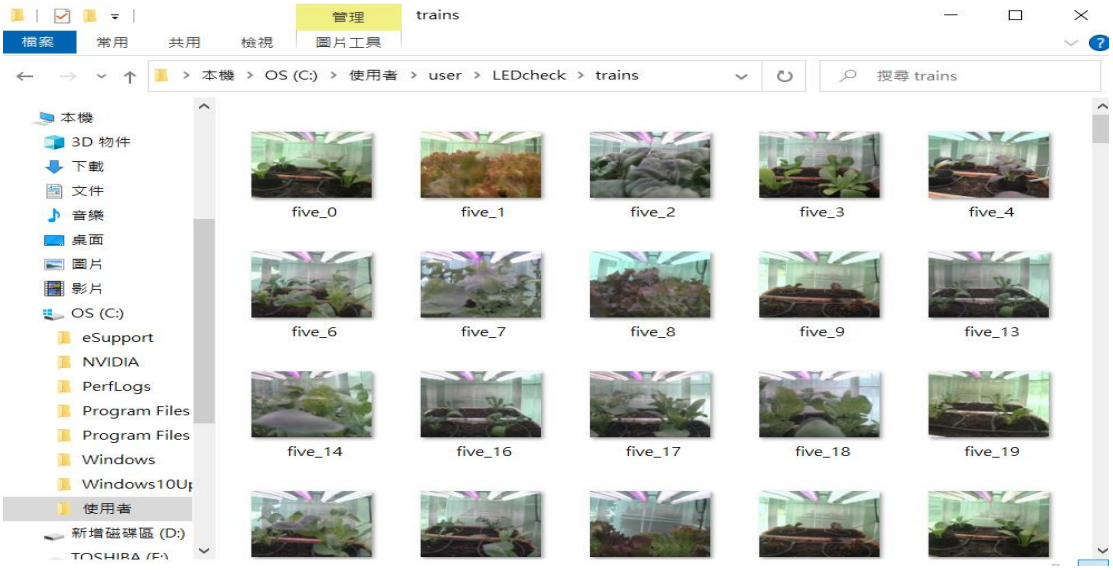


圖 4-2-4 五支燈管的辨識分類資料集內容

燈管辨識分類資料集載入時，使用自行開發的 create_test_data()將訓練資料集讀入，運用字典定義標籤(four 開頭的檔案為類別 0，four 開頭的檔

案為類別 1)識別，再使用 opencv 的 imread()將照片轉成灰階格式，及利用 resize()將照片改成 80 x 80 大小，然後分別將照片及標籤存入 x 及 y 陣列內，最後，再進行灰階正規化將照片內的 byte 值轉成 float，如圖 4-2-5 所示。

```
In [2]: #把訓練集拉進來，並且標籤完成
train_dir = "./LEDcheck/trains"
path = os.path.join(train_dir)   # ..\train
# print(path)
X = []
y = []
dict_labels = {"four":0, "five":1}
def create_test_data(path):
    for p in os.listdir(path):   #os.listdir() : 目錄內檔案或子目錄名字的列表
        # print(p)
        categy = p.split("_")[0]   #若目錄內檔案名稱是cat.xx.jpg或dog.xx.jpg
        # print(dict_labels[categy])
        img_array = cv2.imread(os.path.join(path,p),cv2.IMREAD_GRAYSCALE)   #將檔案影像轉成灰階
        new_img_array = cv2.resize(img_array, dsize=(80, 80))   #將影像縮成80x80
        X.append(new_img_array)  #檔案名稱上的編號是無關的!!
        y.append(dict_labels[categy])

In [3]: create_test_data(path)     # 傳入..\train
X = np.array(X).reshape(-1, 80,80,1)  # 將list轉成array, -1表示rows unknown, 影像格式(i,80,80,1)
y = np.array(y)
#Normalize data
X = X/255.0   # 將影像灰階值正規化
```

圖 4-2-5 燈管辨識分類資料集載入

3. 建立線性堆疊模型

燈管辨識分類的卷積類神經網路分別由輸入層、卷積層 1、池化層 1、卷積層 2、池化層 2、平坦層、隱藏層、及輸出層所組成，其中卷積層皆採用 64 個卷積核(3 x 3 的過濾器)並使用 relu 激活函數。輸入層影像為 80 x 80，經過卷積層 1 後產生 80 x 80 影像 64 張，經池化層 1 處理後變成 40 x 40 影像 64 張，再經卷積層 2 處理共產生 40 x 40 影像 64 張，隨後池化層 2 處理變成 20 x 20 影像 64 張，再連接 25600 個神經元所組成的平坦層，並

連結到 64 個神經元所組成的隱藏層，最後連接到 2 個神經元所組成的輸出層，模型建構程序及模型摘要分別如圖 4-2-6 及圖 4-2-7 所示。

```
In [5]: #[m : ] 代表列表中的第m+1?到最后一?
        #[ : n] 代表列表中的第一?到第n?
        model.add(tkl.Conv2D(64,(3,3), activation = 'relu', input_shape = X.shape[1:]))
        model.add(tkl.MaxPooling2D(pool_size = (2,2)))
        # Add another:
        model.add(tkl.Conv2D(64,(3,3), activation = 'relu'))
        model.add(tkl.MaxPooling2D(pool_size = (2,2)))
        model.add(tkl.Flatten())
        model.add(tkl.Dense(64, activation='relu'))
        # Add a sigmoid layer with 10 output units:
        model.add(tkl.Dense(1, activation='sigmoid'))
        model.summary()
        print("")

        model.compile(optimizer="adam", loss='binary_crossentropy', metrics=['accuracy'])
```

圖 4-2-6 燈管辨識分類的模型建構程序

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 78, 78, 64)        640

 max_pooling2d (MaxPooling2  (None, 39, 39, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 37, 37, 64)        36928

 max_pooling2d_1 (MaxPoolin  (None, 18, 18, 64)        0
 g2D)

 flatten (Flatten)           (None, 20736)             0

 dense (Dense)               (None, 64)                1327168

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 1364801 (5.21 MB)
Trainable params: 1364801 (5.21 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

圖 4-2-7 燈管辨識分類的模型摘要

4. 訓練及測試

燈管辨識分類的卷積類神經網路分 20 批次(epoths=20)進行訓練，每批

次載入 32 張影像(batch_size=32)，訓練歷程如圖 4-2-8 所示。燈管辨識分類的測試結果以.csv 檔案呈現如圖 4-2-9 所示，辨識結果皆正確。

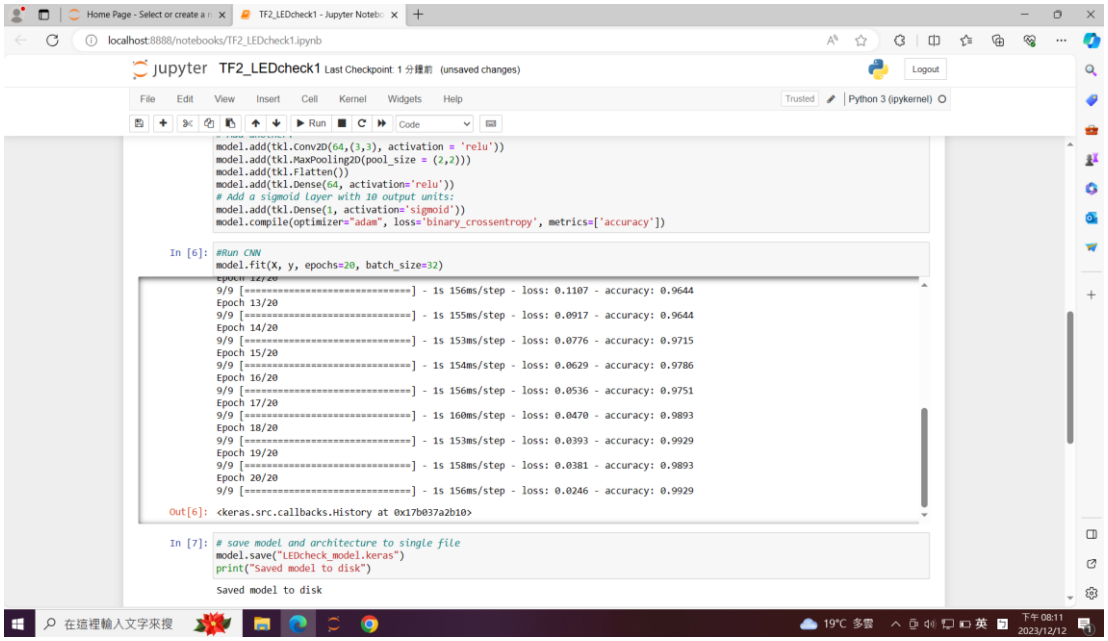


圖 4-2-8 燈管辨識分類的模型訓練歷程

	A	B	C
1	id	label	
2	five_112	1	
3	five_15	1	
4	five_5	1	
5	five_52	1	
6	four_105	0	
7	four_117	0	
8	four_201	0	
9	four_434	0	

圖 4-2-9 燈管辨識分類的測試結果

5. 載入模型及辨識分類

燈管辨識分類的卷積類神經網路模型訓練好後，會用 model.save()將模式存成 LEDcheck_model.keras，當進行燈管自動辨識時，再用 load_model()

將訓練好的模型載入，然後將 exam 子目錄內的辨識影像載入進行辨識分類，最後將辨識結果存成 LEDcheck.csv 檔案，自動檢測腳本程式再從檔案內讀入在辨識類別結果，再轉成燈管的數量，模型載入及辨識程序如圖 4-2-10 所示。

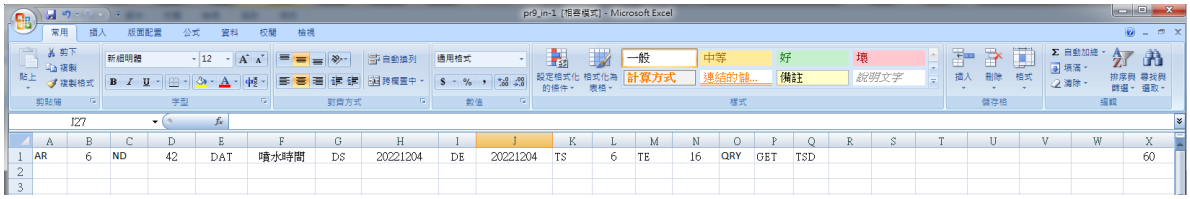
```
In [2]: dict_labels = {"four":0, "five":1}
        # load model
        model = tkm.load_model("LEDcheck_model.keras")

In [3]: #把測試集拉進來分類
        #test_dir = "./LEDcheck/tests"
        test_dir = "./LEDcheck/exam"
        path = os.path.join(test_dir)
        X_test = []
        id_line = []
        def create_test_data(path):
            # print(path)
            for p in os.listdir(path):
                categy = p.split("_")[0]
                fn = p.split(".")[0]
                # print(dict_labels[categy])
                id_line.append(fn)  #將檔名存起來
                img_array = cv2.imread(os.path.join(path,p),cv2.IMREAD_GRAYSCALE)
                new_img_array = cv2.resize(img_array, dsize=(80, 80))
                X_test.append(new_img_array)

In [4]: create_test_data(path)
        X_test = np.array(X_test).reshape(-1,80,80,1)
        X_test = X_test/255
        predictions = model.predict(X_test)
        predicted_val = [int(round(p[0])) for p in predictions]
        submission_df = pd.DataFrame({'id':id_line, 'label':predicted_val})
        submission_df.to_csv("LEDcheck.csv", index=False)
```

圖 4-2-10 燈管辨識分類的模型載入與辨識

4.3 「澆水繼電器」自動測試

本節介紹「澆水繼電器」自動測試的設計構想。腳本程式先啟動散田培植系統軟體，並控制滑鼠及鍵盤進行帳號及密碼的輸入。然後開啟 Excel 的測試案例檔案，逐列讀入一個測試個案，包含多個腳本命令、參數

及預期結果。「澆水繼電器」自動檢測使用到 10 種腳本命令，分別為 AR(培植區域)、ND(控制器編號)、DAT(資料類型)、DS(查詢開始日期)、DE(查詢結束日期)、TS(查詢開始時間)、TE(查詢結束時間)、QRY(查詢紀錄)、GET(擷取紀錄編號)、及 TSD(查詢延伸感應值)等。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	AR	6	ND	42	DAT	噴水時間	DS	20221204	DE	20221204	TS	6	TE	16	QRY	GET	TSD							60
2																								
3																								

圖 4-3-1 「澆水繼電器」自動檢測的測試個案範例

圖 4-3-1 顯示本項自動檢測的一個測試個案，描述點選第 6 號培植區的第 42 號控制器，設定查詢資料類型為「噴水時間」，選擇查詢監控記錄的開始日期及結束日期皆為 20221204，開始時間設為 6 點，結束時間設為 16 點。按下「查詢」按鍵後，資料顯示列表，點選第一筆的紀錄編號欄位，擷取紀錄編號資料，如圖 4-3-2 所示。最後開啟「延伸感應值查詢程式」，填入紀錄編號並點選「查詢」後，查到感應值為 80，將此結果值填入結果檔案 Excel 的 Y 欄。此結果與 X 欄的預期結果相較，X 欄預期結果值表示繼電器量燈的光感值的最低容許值，範例中延伸感應值查到 80，表示澆水繼

電器是正常工作狀態。

圖 4-3-2「澆水繼電器」檢測的散田監控紀錄查詢

本範例的「澆水繼電器」自動檢測透過擷取第一筆的紀錄編號，同樣是採用抽樣方式進行檢測，無需每個噴水時間都進行測試，提高檢測效率。若在抽樣檢測後發現澆水繼電器損壞，最多只會延遲一天發現裝置異常，但這種情況不會對植物的生長產生影響。

4-4「馬達運轉」自動測試

在「馬達運轉」自動測試中，腳本程式啟動散田培植系統軟體，模擬滑鼠及鍵盤進行帳號及密碼的輸入。隨後，打開 Excel 的測試案例檔案，逐列讀取一個測試個案，包含多個腳本命令、參數及預期結果。測試個案

中的腳本命令涵蓋了 AR(培植區域)、ND(控制器編號)、DAT(資料類型)、DS(查詢開始日期)、DE(查詢結束日期)、TS(查詢開始時間)、TE(查詢結束時間)、QRY(查詢紀錄)、GET(擷取紀錄編號)、及 TSD(查詢延伸感應值)等 10 種腳本命令。

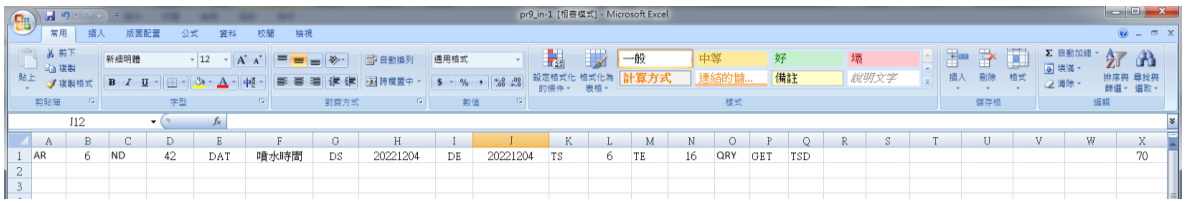


圖 4-4-1 「馬達運轉」自動檢測的測試個案範例

在「馬達運轉」自動測試的測試個案中，圖 4-4-1 描述了點選第 6 號培植區的第 42 號控制器，設定查詢資料類型為「噴水時間」的操作。選擇查詢監控記錄的開始日期及結束日期皆為 20221204，開始時間設為 6 點，結束時間設為 16 點。按下「查詢」按鍵後，資料顯示列表，只點選第一筆的紀錄編號欄位，擷取紀錄編號資料，如圖 4-4-2 所示。最後，打開「延伸感應值查詢程式」，填入紀錄編號並按下「查詢」後，查到感應值為 90，將此結果值填入結果檔案 Excel 的 Y 欄。此結果與 X 欄的預期結果相較，X 欄

預期結果值表示馬達上的三軸加速計的震動值最低容許值，範例中延伸感應值查到 90，表示馬達是正常運轉狀態(表示有較大震動)。

圖 4-4-2「馬達運轉」檢測的散田監控紀錄查詢

在「馬達運轉」自動檢測中，透過擷取第一筆的紀錄編號進行抽樣方式檢測，同樣無需針對每個噴水時間進行測試，提高檢測效率。若抽樣檢測後發現澆水馬達損壞，最多延遲一天發現裝置異常，但此狀況不會對植物的生長產生影響。

第五章　結論

透過本次專題實作，我們運用課堂學到的軟體測試觀念，並實際使用腳本語言進行散田培植系統異常狀況的自動檢測，此外，亦學習到人工智慧的深度學習技術，運用卷積類神經網路進行燈管辨識分類，以改善先前研究的辨識準確度。本章描述了在軟硬整合的自動檢測設計中遇到的問題，並分享了在腳本程式應用中的實務經驗和心得。同時提供了問題的解決方案，供有興趣使用 AutoIt 進行自動測試的人參考，節省摸索的時間。

5.1 結論

透過本次專題實作，我們的團隊成員獲得了以下具體成果：

1. 透過文獻資料，深入瞭解了軟硬整合的自動檢測設計與軟體測試之間的密切關係。

2. 學習了 AutoIt 腳本語言，並結合 Excel、散田培植系統查詢、延伸感應值查詢程式、Jupyter Notebook、tensorflow 及 keras 發展平台，進行測試個案的設計實作，跨多平台建構了自動測試的可行框架。

3. 開發了卷積類神經網路的辨識分類模型，提升先前研究的燈管辨識準確度，以支援散田培植系統異常狀況的檢測。

5.2 問題挑戰與解決方法

在本節，將探討專題發展中涉及 AutoIt 自動測試、卷積類神經網路的辨識分類的問題，以及我們所選擇的解決方案。

問題 1: 如何在便利的使用環境下，發展異常自動檢測功能並擴增卷積類神經網路的辨識分類的功能以達到檢測的目的？

解決辦法: 本專題利用 AutoIt v3 的腳本程式設計實現測試個案所需的散田監控軟體及裝置異常資訊的自動檢測功能，但使用卷積類神經網路的辨識分類的功能必須是先安裝發展套件及具備 GPU 環境，因此，目前只能遷就在預先準備完整電腦資源上執行自動檢測腳本。未來可以嘗試利用雲端人工智慧平台(如 Azure AI)或自行發展 Gateway 協助在伺服器代理執行自動檢測腳本，並結合卷積類神經網路的辨識分類的功能。

問題 2: 運用卷積類神經網路進行燈管辨識分類，需要建立散田培植系統蒐集的培植照片進行資料集，以便進行辨識分類模型的訓練及測試使用，如何蒐集或準備足夠的培植照片呢？

解決辦法**:** 散田培植系統會蒐集培植箱自動回傳的培植照片，目前共約 4300 張，由於鏡頭未拍到燈管或植物生長遮住燈管等因素，儘找出 120 張左右可資運用的照片，為了方便實現辨識分類模型的訓練及測試使用，必須進行適度的處理以增加資料集的數量，包含左右鏡射及手動製造燈管失效不亮的異常照片，實作發現效果不錯。

參考文獻

1. 陳駿季、楊智凱，推動智慧農業 - 翻轉臺灣農業，國土及公共治理，5(4)，pp. 104-111，2017。

2. 張兆村、陳宇揚、陳庭維、王京森、林昭佑，培植系統之自動檢測設計與實作，IETAC 2023。

3. 維基百科，Convolutional neural network, https://en.wikipedia.org/wiki/Convolutional_neural_network.

4. 維基百科，https://zh.wikipedia.org/zh-tw/%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F%E4%BF%9D%E8%AF%81。

5. 朱慧德、萬俊麟，資訊系統發展之品質管理，中華管理評論，2(2)，pp.113~121，1999。

6. 黑箱測試 https://www.tw511.com/24/276/9936.html。

7. 維基百科，自動化測試，https://zh.wikipedia.org/wiki/%E8%87%AA%E5%8A %A8%E5%8C%96%E6%B5%8B%E8%AF%95。

8. 2022 年全球十大最佳自動化測試工具，https://read01.com/zh-tw/38dgKyA.html#.Y6AoCnZByUk。

9. 維基百科，Postman，https://en.wikipedia.org/wiki/Postman_(software)。

10. 初識 Katalon Studio 自動化測試工具, https://www.itread01.com/content/1546633459.html。

11. 維基百科，JMeter，https://en.wikipedia.org/wiki/Apache_J Meter。

12. 維基百科，回歸測試，https://zh.wikipedia.org/wiki/%E5%9B%9E%E5%BD%92%E6%B5%8B%E8%AF%95。

13. 維基百科，深度學習，https://zh-yue.wikipedia.org/wiki/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92。

14. 林大貴，TensorFlow + Keras 深度學習人工智慧實務應用，博碩文化，2017。