113 學年度臺中市中小學科學展覽會
作品說明書

科　　別：農業與食品學科

組　　別：高級中等學校組

作品名稱：稻出病名—深度學習之水稻葉體病害辨識與分類

關 鍵 詞：水稻葉體病變辨識、卷積神經網絡模型、智慧農業

編　　號：

目錄

摘要

本研究採用跨領域研究方法，結合資訊技術與農業專業，運用深度學習提升水稻葉片病害辨識的準確度與效率。傳統病害檢測受限於人為誤差與環境影響，難以滿足大規模農地監測需求，因此本研究導入卷積神經網路（CNN），透過影像處理標註病害區域，並訓練模型自動提取病變特徵，以提升辨識效能。研究過程中，我們應用數據增強技術優化模型泛化能力，並結合農業專家回饋機制，確保系統應用的可靠性。實驗結果顯示，該方法能有效提高水稻病害識別度，降低農損，並優化農業管理流程。未來，本研究將結合物聯網（IoT）與雲端運算，實現即時監測與智慧決策，推動農業數位轉型，提升農作物生產效率與永續發展。

壹、前言

一、研究動機

農業一直是台灣經濟發展的基石，特別是在糧食安全與農業永續發展的議題上，如何提高農作物的生產效率與品質，成為當前農業科技發展的重要目標。然而，病蟲害管理仍然是農民面臨的最大挑戰之一，特別是水稻這類主要糧食作物，一旦發生病害，若無法及時診斷並採取適當防治措施，將可能導致嚴重的減產與經濟損失。傳統的病害診斷方式主要依賴農民經驗與目視判斷，但這種方式不僅受個人知識與環境因素影響，且在大規模農田中難以有效執行，導致病害控制的不確定性增加。因此，如何運用現代科技提升病害監測的準確性與效率，已成為農業科技發展的關鍵課題。

近年來，隨著人工智慧（AI）與影像辨識技術的進步，深度學習技術在農業領域的應用逐漸受到重視，特別是卷積神經網路（CNN）在影像處理上的強大能力，使其成為病害辨識研究的重要工具。許多研究已經證實，透過影像數據訓練深度學習模型，能夠有效提升作物病害的識別率，減少人為誤判。然而，目前市面上仍缺乏一款能夠讓農民實際操作、簡單易用的病害辨識系統。因此，**本研究旨在結合農業專業知識與深度學習技術，開發一套基於 CNN 的水稻葉體病變辨識與分類系統，透過 AI 技術協助農民更精確地診斷病害，以提升病害檢測的準確性與即時性，並為智慧農業應用提供更有效的技術支援。**

二、研究目的

本研究旨在開發一套基於深度學習的水稻葉片病害辨識系統，提供農民一款操作簡單、輕便實用的應用程式，以提前偵測與預防病害，降低農損風險。此外，本研究亦將技術應用於農業教育，提升學生與農民的病害辨識能力，推動智慧農業的普及與發展。透過本研究，期望能有效降低農民的種植成本，優化農藥使用策略，並透過實地試驗驗證模型於實際農業環境中的可行性，提供後續優化依據。

（一）**開發水稻葉片病害辨識模型：**訓練深度學習模型專門辨識水稻葉片病害，透過不斷優化演算法與參數調整，提升辨識準確率與反應速度，以確保應用於實務環境的可靠性。

（二）**建立病害辨識應用系統：**設計一款簡單易用的應用程式，讓農民或相關從業人員透過拍攝葉片照片，即時辨識病害並獲取預防與治理建議，提升農業管理效率。

（三）**推動智慧農業技術應用：**將研究成果應用於農業教育課程，增強學生與農民對病害辨識的知識與應用能力，提升智慧農業技術的實務應用，推動農業數位轉型。

（四）**優化病害防治策略，降低農藥使用：**透過精確的病害辨識，減少不必要的農藥使用，降低生產成本，並減輕農藥過度施用對環境與人體健康的影響。

（五）**實地測試與模型優化：**進行田間試驗，收集不同生長條件下的數據進行調整與優化，確保辨識系統能夠適應不同地區與氣候條件，提高病害診斷的可靠性與準確性。

透過本研究的實施，期望能為台灣農業帶來更高效的病害管理模式，提升智慧農業的技術發展與應用價值。

三、系統實作流程

本研究的系統實作流程將以**應用科學研究**為核心，具體來說，它結合農業科學與人工智慧領域，屬於**跨學科的綜合性研究**。透過科學方法進行規劃與開發，確保最終系統在效能與實用性上的表現。系統實作流程包含以下步驟：

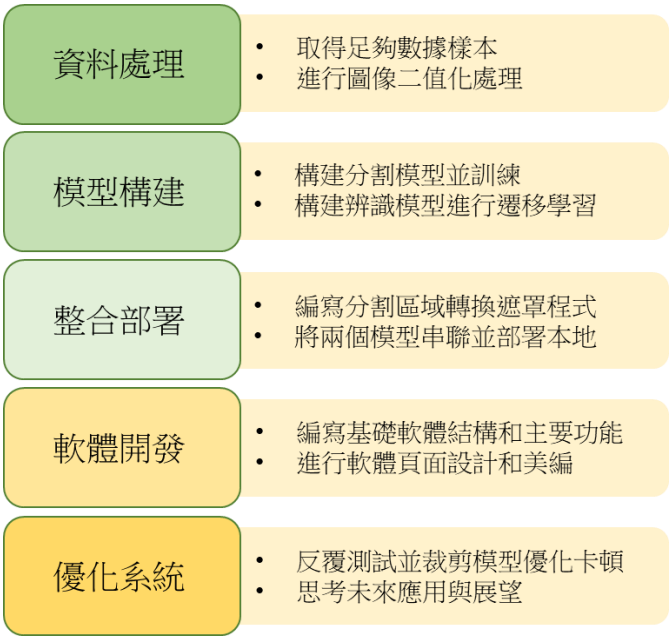


圖 1：系統實作流程（由第一作者使用 PowerPoint 繪製）

四、葉體病變模型相關文獻

表 1：葉體病變模型相關文獻（由第三作者彙整）

主題	文章標題	作者與年份	摘要	心得
卷積神經網路	基於卷積神經網路之深度學習方法之蘋果葉面病害辨識與分類	李湘渝（2022）	為提早預防、阻止病害蔓延，本論文利用 ResNet-50、DenseNet-201、MobileNet、EfficientNet-B0 四種卷積神經網路，辨識與分類四種不同類型的蘋果葉片病害。最後發現 EfficientNet-B0 於蘋果葉病害辨識表現最佳。	本篇論文是我們研究的初步啟發來源，幫助我們深入了解人工智慧在農業領域的應用，以及病蟲害對農業生產造成的重大影響，該文獻強調了精準辨識的重要性。

影像處理	應用於智慧農業之作物生長影像監測系統	李柔嫻（2020）	為有效降低務農人口，提升管理效率，結合IOT、大數據分析和人工智慧，透過人機介面判斷作物的生長面積尺寸以及生長階段，來效率的評估產期估算。	本篇文章深入探討了二值化與物件偵測的方法，雖然這些技術與我們研究中的影像切割存在一定的差異，但仍對我們的數位影像處理提供了重要的參考價值。
病變分析	專為咖啡產量預估與疾病偵測設計之智慧農業監測系統	林郁榮（2023）	本篇論文提出兩種人工智慧模型，以提升農民的咖啡產量： 一、辨識和分類咖啡葉上存在的病害。 二、估計咖啡果實產量。	在此，咖啡葉病變的辨識與分類模型與我們的研究主題具有高度的相似性，且在實作過程中面臨許多相同的挑戰，例如：樣本收集較少等等，值得參考前人的研究方法。

五、常見稻葉疾病文獻

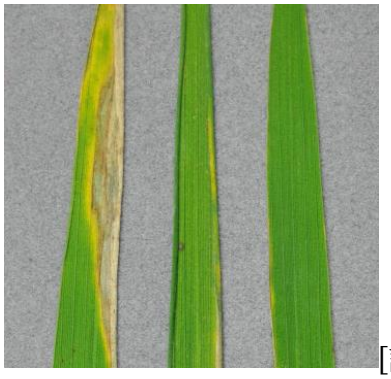
水稻病害種類繁多，包括徒長病、秧苗立枯病、稻熱病、胡麻葉枯病、線蟲白尖病、線蟲根瘤病、紋枯病以及細菌性白葉枯病等（葉俊巖，2006）。在本研究中，將聚焦於探討臺灣地區較常見的兩大病害—白葉枯病與胡麻葉枯病，並運用深度學習技術進行自動辨識。

（一）白葉枯病

白葉枯病是由細菌 Xanthomonas oryzae pv. oryzae 引起的，通常在水稻進入分蘗盛期後發現病徵，部分情況也會在幼苗期發生。這種病害主要感染水稻的葉片或葉鞘，偶爾會波及穀粒，屬於系統性病害。它的細菌通常從葉片或根部的傷口侵入，進入水稻的維管束中繁殖並擴散，也屬於維管束病害。白葉枯病對稻穀的產量、外觀及味道會造成重大影響。根據研究，白葉枯病有三種常見的病徵：典型的葉枯型病徵、急性萎凋型病徵以及淡黃化型病徵，如表 2 所示。（林駿奇，2009）

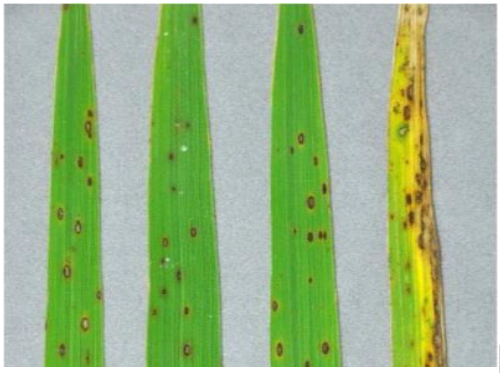
表 2：白葉枯病特徵表（由第二作者製作，參考林駿奇，2009）

病徵類型	發病時機	症狀描述
典型葉枯型病徵	插秧後 3～4 星期、分蘗盛期	1. 水浸狀小班沿葉緣往中肋蔓延逐漸延長成枯黃色，後至葉鞘呈灰白色。 2. 高溼狀態下因腐生菌感染易生黑斑。 3. 根系發育不良，植株矮小，稔實率低 4. 菌泥：乾燥的黃色小球型，易脫落
急性萎凋型病徵	幼苗期	1. 幼苗外葉捲起，呈灰綠、灰褐色 2. 植株矮小、基部腐化軟爛 3. 菌泥：呈黏性黃色 4. 出現於成株：多於孕穗或開花期，劍葉至葉鞘出現數條白色斑紋，幼穗萎凋成白色，最後枯死。
淡黃化型病徵	插秧和分蘗期	新葉呈淡黃色、黃褐色，而後凋萎，症狀似缺鐵。



[註 1]

圖 2：水稻之白葉枯病（由第三作者截圖）



[註 2]

圖 3：水稻之胡麻葉枯病（由第三作者截圖）

（二）胡麻葉枯病

胡麻葉枯病是由細菌 *Bipolaris oryzae* 引起的，水稻在各生長期都有可能感染此病。若生長於乾旱或貧瘠的土壤中，則更容易受到感染；此外，病害也可能隨著其他病蟲害引起的生育不良而發生。本病害主要會危害水稻本田期的葉片，病徵通常表現為葉斑，葉舌、穗梗及穀粒也有可能受到影響。胡麻葉枯病造成的影響除了會減少稻米的產量，還可能降低稻米的品質。胡麻葉枯病在不同生長期的病徵比較，詳見表 3（張義璋，2007）。

表 3：胡麻葉枯病各生長期病徵比較（由第二作者製作，參考張義璋，2007）

<table>
<tr><th colspan="2">感染時間</th><th>症狀描述</th></tr>
<tr><td colspan="2">萌芽時</td><td>苗枯病：
1.呈黃褐色水浸狀病變，子葉上黑色短線狀斑點，嚴重時枯萎。有墨綠色菌絲和分生孢子，子葉褐色，本葉葉鞘呈褐色病變，組織脆化，葉片生長受阻、直立黃化，嚴重則枯萎。</td></tr>
<tr><td rowspan="3">長出本葉後</td><td>老秧苗</td><td rowspan="2">1. 病斑約胡麻種子大小，墨綠色水浸狀小斑後轉褐色小斑，再擴大成紡錘型或橢圓形，周圍有黃暈，有些具同心輪紋。
2. 缺氮、鉀、矽顯感病性後病斑擴大，呈兩端圓寬之橢圓，黃暈窄但明顯。</td></tr>
<tr><td>本田期</td></tr>
<tr><td>稻孕穗期</td><td>植株矮化，葉片縐縮、變厚，抽穗慢，稻穗小結實不佳。</td></tr>
</table>

六、監督式機器學習

「在監督式學習中，可以使用一組輸入資料和一組對應的標註輸出資料來訓練模型，這些標註通常是手動完成的。」（Amazon Web Services，無日期）

相較於資料具標籤、具特徵（features）與預測目標（label）的監督式學習（Supervised Learning），非監督式學習（Unsupervised Learning）所需的訓練資料則沒有標籤，機器必須從提供的輸入範例中自行尋找規律（胡依淳，2018）。由於監督式學習需要手動標註訓練數據，這使得其具有較高的準確率。可以將標註過的資料比作模擬考的標準答案，透過比較誤差來修正學習過程，這樣可以達到更精確的預測。然而，監督式學習需要大量的前期準備，這樣的工程量龐大，當範圍擴大時，資料量會快速增加，並且可能無法標註所有特徵，這使得監督式學習模型在面對未知領域時，幾乎無法運作。如下表 4 所示，監督式學習與非監督式學習的比較表。

表 4：監督式學習與非監督式學習的比較表（由第二作者製作，參考 Andreja, 2024）

比較項目	監督式機器學習	非監督式機器學習
輸入數據需求	需要標註數據	需要未標註數據
主要目標	分類數據或進行預測	將數據分組或發現數據中的隱藏結構
模型結果	具體的預測或分類	數據分組
最常用的算法和技術	線性回歸、邏輯回歸、決策樹、神經網絡等	K-means 聚類、階層式聚類和 DBSCAN 等
最常見的應用場景	圖像分類、垃圾郵件檢測、醫療診斷、信用評分	客戶分群、異常檢測、市場分析、降維與特徵學習
人工監督需求	需要人工提供標註的訓練數據	不需要人工監督或明確指導
主要優勢	高準確性	發現數據隱藏結構
主要劣勢	需要大量標註數據	由於缺乏標籤，訓練時可能偏離方向
常用評估指標	準確率、精確率、召回率和 F1 分數	輪廓係數和 Davies-Bouldin 指數
適應性	僅適用於特定的任務	適用於探索性任務和特徵學習
預訓練中的應用	不常見，模型直接使用標註數據進行訓練	常用於預訓練模型，之後與監督式學習結合進行微調
對數據噪聲的敏感性	如果標註數據準確，則能處理噪聲	由於缺乏標籤指導學習過程，對噪聲較敏感

七、卷積神經網路

神經網路以層為單位，層與層之間彼此連接，最終構建成一個神經網路模型。卷積神經網路（Convolutional Neural Network，簡稱 CNN）是一種專門用於識別與偵測圖像或物件的深度學習模型，能夠從資料中自動辨識出具有代表性的特徵。

CNN 的基本組成單元是卷積層，這一層利用卷積核在輸入數據上滑動，進行特徵學習與提取。池化層則用來減少卷積層輸出的空間維度，並保留重要的特徵。**「而全連接層用於將卷積層和池化層的輸出映射到預測空間，在這一層，神經元與前一層的所有神經元都相連，形成全連接結構。」**（楊學智，2024）。

卷積神經網路（CNN）是一種專門用於大規模視覺識別的多層次模型，其網絡擁有足夠的神經元，能夠擬合任何複雜的數據分佈。過去的研究表明，神經網路在簡單識別任務中已顯示出極高的效能。近年來，隨著大規模訓練數據（如 ImageNet）的可用性，以及強大 GPU 計算能力的支援，使得深度 CNN 模型的訓練變得更加可行，並且深度學習的 CNN 模型在效能上顯著超越了基於詞袋模型（BoVW）的方法。

CNN 模型的結構由多層組成，每一層的功能是將上一層的特徵圖與卷積核進行卷積運算，並透過可微的非線性激活函數進行處理。因此，CNN 可以被視為一個複合函數，並透過反向傳播進行訓練。反向傳播過程是根據最上層監督信號與預測結果之間的誤差來調整網路的參數。（Xie, & Yuille, 2017）

表 5：CNN、RNN、SVM 和 KNN 的比較表（由第三作者彙整）

種類	主要用途	介紹	優缺點
卷積神經網路 CNN	圖片辨識、分類及處理（吳季桓，2010）	先進行特徵提取，再進行分類。相較多層感知器（包含輸入層、隱藏層及輸出層），多了卷積層和池化層。（林佳姿，2020）	1、精於降低頻率的變動 2、具備卓越的特徵提取能力 3、適合處理空間數據（林佳姿，2020、趙國桓，2024）
遞迴神經網路 RNN	處理具有時序性的資料、語言處理（林佳姿，2020）	中文為遞迴神經網路，每個神經元只能單獨處理一個輸入，當前輸出會受到先前輸入影響。（林佳姿，2020）	1、解決傳統類神經網路個別輸入的限制 2、增加傳統類神經網路沒有的記憶功能（林梓峰，2022）

SVM	分類問題(吳季桓，2010)	SVM 屬於一般化線性分類器，在二維空間維度中，有一個超平面將兩個類別明確分開，而兩邊超平面距離越大，代表越能將兩個類別明顯的分開。(吳季桓，2010)	1、能夠處理高維度的數據 2、具備優異的分類效果（趙國桓，2024）
KNN	分類和回歸問題（吳季桓，2010）	每個文件在多維度空間中是一個座標點，相同分類的文件會靠得很近。當有一個新文件需要分類時，它會根據距離最近的幾個已分類文件來決定分類，並選擇最多的分類作為結果。(吳季桓，2010)	1、易於理解和操作 2、能夠靈活應對數據的變化 （趙國桓，2024）

八、圖像分析

（一）圖像分割

圖像分割是指將數位影像根據不同特徵或性質劃分為多個區域，其中具有相似特徵的像素會被歸類到同一區域。**「圖像分割的目的是為了簡化圖像或改變其表示形式，使其更易於分析」**(Cheng et al., 2001)。常見的分割類型包括語義分割、實例分割和全景分割。在實際應用中，不同的分割類型各自針對不同的需求。例如，語義分割適用於辨識場景中的物件類別；實例分割則能用來區分同類別的不同個體；而全景分割則提供更全面的場景理解。

表 6：分割類型（由第二作者製作，參考黃靖程，2025）

分割類型	說明
Semantic Segmentation (語義分割)	將圖像中的所有像素點進行分類
Instance Segmentation (實例分割)	物件偵測和語義分割的結合
Panoramic Segmentation (全景分割)	語義分割和實例分割的結合

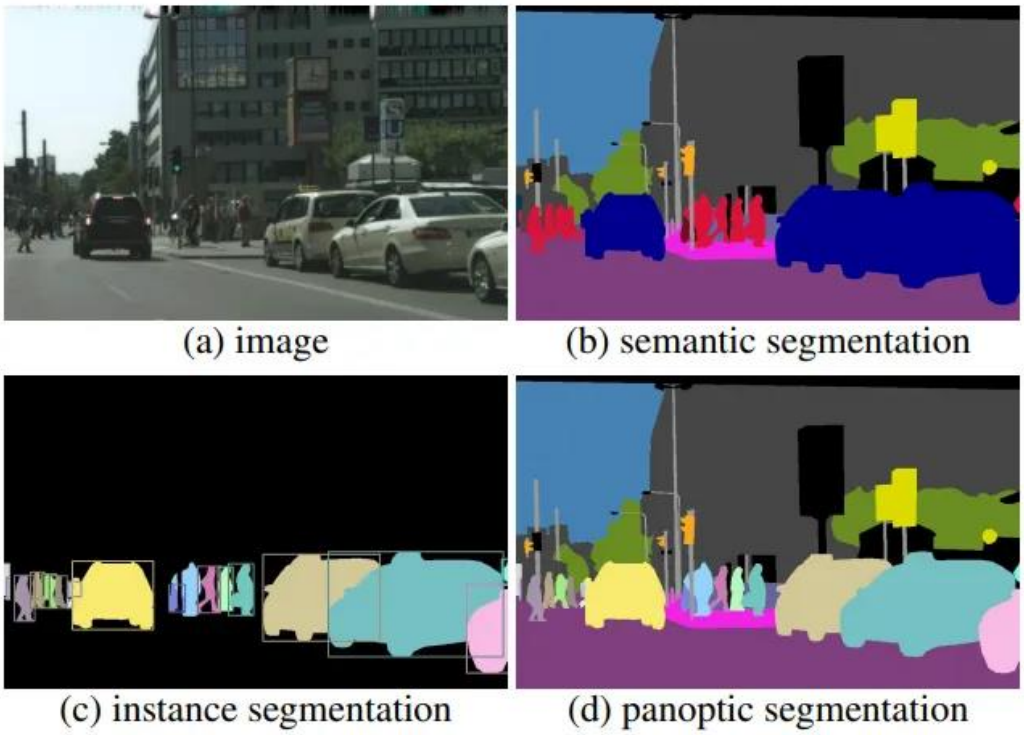


圖 4：語義分割、實例分割、全景分割比較（由第二作者截圖彙整）[註 3]

（二）二值化（閾值二進制）

二值化是圖像分割的一種方法，通常屬於語義分割。這個方法會將圖像中灰度值大於閾值（threshold）的像素轉換為黑色，而低於閾值的像素則轉換為白色。因此，**「二值化後的圖像只會留下黑色和白色，適用於簡單的前景物體和背景區分」**（Yousefi, 2011）。

相較於 256 級灰階或彩色影像資訊，二值化的雙層資訊能夠簡化並減少計算量。二值化的方式有全局二值化、局部二值化以及兩者混合的方法。全局二值化是基於整張圖像的全局灰度直方圖來確定閾值，進而將圖像分類為物件或背景；局部二值化則是將圖像劃分為小區域（如小塊或窗口），並在每個區域內根據該區域的局部灰度分佈來確定閾值，這樣可以處理圖像中局部光照變化的情況。混合方法則是結合全局和局部資訊來決定閾值標籤。（Sauvola, & PietikaKinen, 1999）

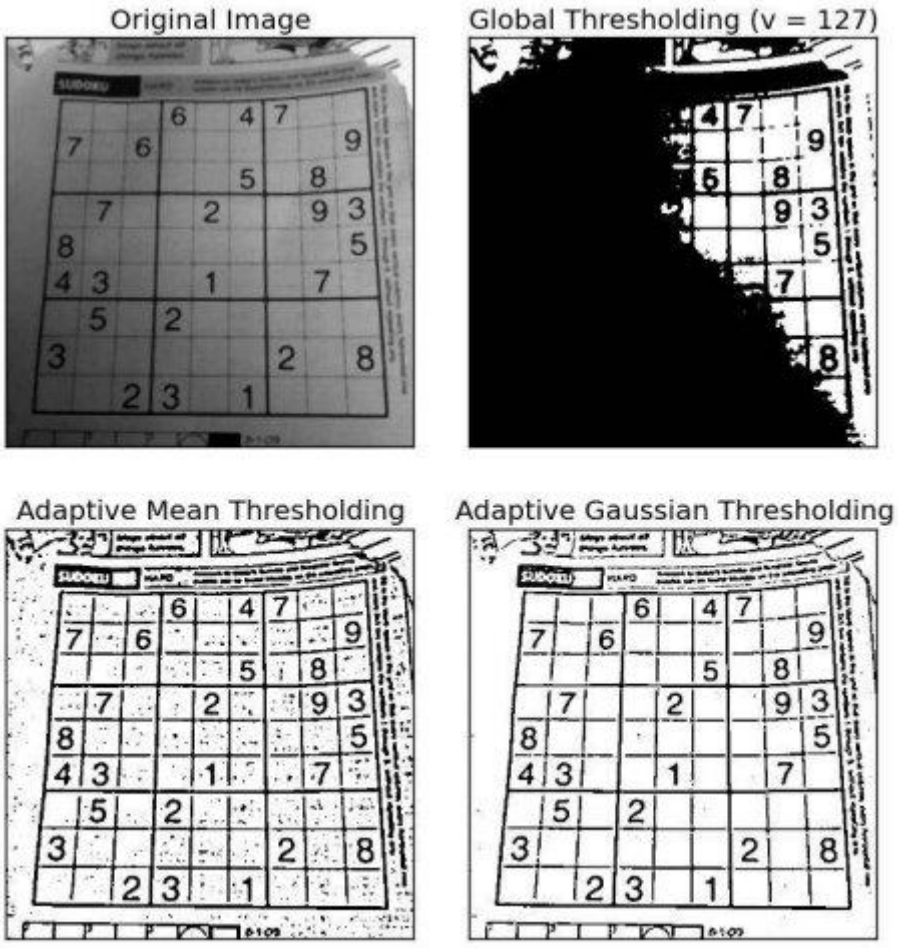


圖 5：二值化用於圖像分割之示意圖（由第二作者製作）[註 4]

表 7：二值化與各類分割技術比較（由第二作者製作，參考 Kaur & Kaur, 2014）

分割技術	簡述	優點	缺點
二值化	根據圖像直方圖的峰值來確定特定閾值	無需先前資訊，最簡單的方法	高度依賴峰值，空間細節未被考慮
聚類方法	將圖像劃分為同質群集	使用部分隸屬度，因此對於現實問題更有用	確定隸屬函數並不容易
分水嶺方法	拓撲解釋	結果更穩定，檢測到的邊界是連續的	梯度計算複雜
基於邊緣的方法	不連續性檢測	適用於物體之間對比度較高的圖像	不適用於錯誤檢測或邊緣過多的情況

基於區域的方法	將圖像劃分為同質區域	更不容易被噪聲影響，當相似性標準容易定義時很有用	消耗過多時間和記憶體
基於偏微分方程的方法	微分方程的運作	最快的方法，適合時間關鍵型的相關應用	更高的計算複雜度
基於人工神經網路的方法	根據學習過程的模擬以進行決策	無需編寫複雜的程序	訓練中浪費更多時間

為了評估圖像分割模型的表現結果，可以根據情況選擇不同的性能指標。常見的指標有 Jaccard 係數 (IoU)、準確率 (Accuracy)、Dice Coefficient 和 Boundary F1 Score 等等，這些指標各有其適用的場景和特性。例如，Jaccard 係數（IoU）是衡量預測區域與實際區域重疊程度的常用指標，其值介於 0 到 1 之間，值越大表示分割結果越準確。Dice Coefficient 強調兩者的交集部分，對於處理不平衡的類別分佈特別有效，因為它能有效降低假陽性（false positives）的影響。準確率（Accuracy）是指整體正確分類的像素比例，適合用來進行整體的評估，但當類別不平衡時，其表現可能不佳。Boundary F1 Score 則專注於分割邊界的精確性，能夠更細緻地評估分割結果在邊界處的表現，對於需要高精度邊界識別的任務特別有用。這些性能指標可以根據具體應用情境選擇，幫助研究者全面了解模型在不同維度下的表現。

表 8：性能指標（由第一作者製作）

性能指標	說明	公式
Jaccard 係數 (IoU)	衡量預測區域與實際區域重疊程度	$IoU = \frac{\lvert G \cap P \rvert}{\lvert G \cup P \rvert}$
Dice Coefficient	強調交集部分的兩倍權重，對於小目標（如少數類別）更敏感。	$Dice\ \ Coefficient = \frac{2\lvert G \cap P \rvert}{\lvert G \rvert + \lvert P \rvert}$
準確率 (Accuracy)	計算正確分類像素的比例，對於不平衡數據集可能不敏感。	$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$

Boundary F1 Score	聚焦於分割邊界的精確性，有效評估模型在細節處理上的能力。	$Boundary\ \ F1\ \ Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$ $Precision = \frac{TP}{TP + FP}$ $Recall = \frac{TP}{TP + FN}$

表 9：項目解釋（由第一作者製作）

項目	全名	說明
TP	True Positive	正確預測為正的像素數
FP	False Positive	錯誤標記為正的像素數
TN	True Negative	正確標記為負的像素數
FN	False Negative	錯誤標記為負的像素數

（三）圖像辨識

圖像辨識主要依賴深度學習技術來訓練模型，其過程模仿人類的推理過程，指導機器辨識目標對象。透過對每一個細節的深入分析，模型能夠**「針對機器學習過程進行糾錯與調整，以優化模型的精準度」**（Gu et al., 2024）。圖像辨識的運作流程通常分為四個主要步驟：預處理圖像、特徵提取、檢測並分割，以及進階處理（Fujiyoshi et al., 2019）。

1、**預處理圖像**：這一步驟主要是對原始圖像進行去噪、調整大小、對比度增強等操作，確保圖像的質量適合進行後續的特徵提取。

2、**特徵提取**：在這一階段，模型會從圖像中提取出關鍵特徵，例如邊緣、角落或顏色變化等，這些特徵能夠幫助模型識別物體的結構。

3、**檢測並分割**：此步驟涉及將圖像中感興趣的區域識別出來，並根據特徵將物體或區域分割開來，以便進行更精細的處理。

4、**進階處理**：最後，進行更加細致的處理與分析，利用已提取的特徵和分割結果進行物體識別，並進行分類或預測。

表 10：圖像辨識流程（由第二作者製作，參考 Jain et al.,2000）

運作流程	說明
預處理	1、二次取樣保證圖像座標的正確 2、平滑去噪來濾除感測器引入的裝置雜訊 3、提高對比度來保證相關資訊可以被檢測到 4、調整尺度空間使圖像結構適合局部應用
特徵提取	1、線，邊緣提取和脊部偵測 2、局部化的特徵點檢測，如邊角與斑點檢測
檢測/分割	1、篩選特徵點 2、分割一或多幅圖片中含有特定目標的部分
進階處理	1、驗證得到的資料是否符合前提要求 2、估測特定係數，比如目標的姿態和體積 3、對目標進行分類

貳、研究設備及器材

一、硬體設備表

電腦硬體名稱	規格
CPU	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz 2.11 GHz
GPU	Nvidia GeForce RTX 360 TI
記憶卡	16.0GB

二、軟體設備表

類別	軟體名稱	用途
數據蒐集	Kaggle 數據集	提供水稻葉片病害影像數據
影像處理	ImageJ 軟體	手動標記葉片病害並產生二值化掩碼圖
編碼與命名	Windows PowerShell	批量修改檔案名稱
模型訓練	TensorFlow & Keras	深度學習框架，用於構建與訓練模型
	DenseNet-121	預訓練 CNN 模型，用於特徵提取
	Unet	影像分割解碼器
	NumPy	數據處理與轉換
	Matplotlib	可視化模型訓練與預測結果
開發環境	Python	程式語言，開發與部署使用
	Visual Studio 2020	程式撰寫與測試
	Flask	後端 API 框架
部署與測試	Android 設備（APK 部署）	測試模型應用於移動端
	GPU 訓練伺服器	加速深度學習訓練
	雲端存儲(Google Drive, AWS S3)	儲存數據集與模型檔案

參、研究方法

一、研究方法

（一）**文獻分析法**：透過蒐集與分析相關的網路文章、報導、論文、期刊、以及專業書籍，深入了解並整理已有的知識體系與實作方法，為後續研究提供理論基礎與實務參考。

（二）**實作研究法**：實際編寫並執行深度學習模型，進行實際測試、調整與優化。此方法著重於模型的實際運行與改良，並對各項預測結果進行實證分析，最終評估模型的效果與準確度。

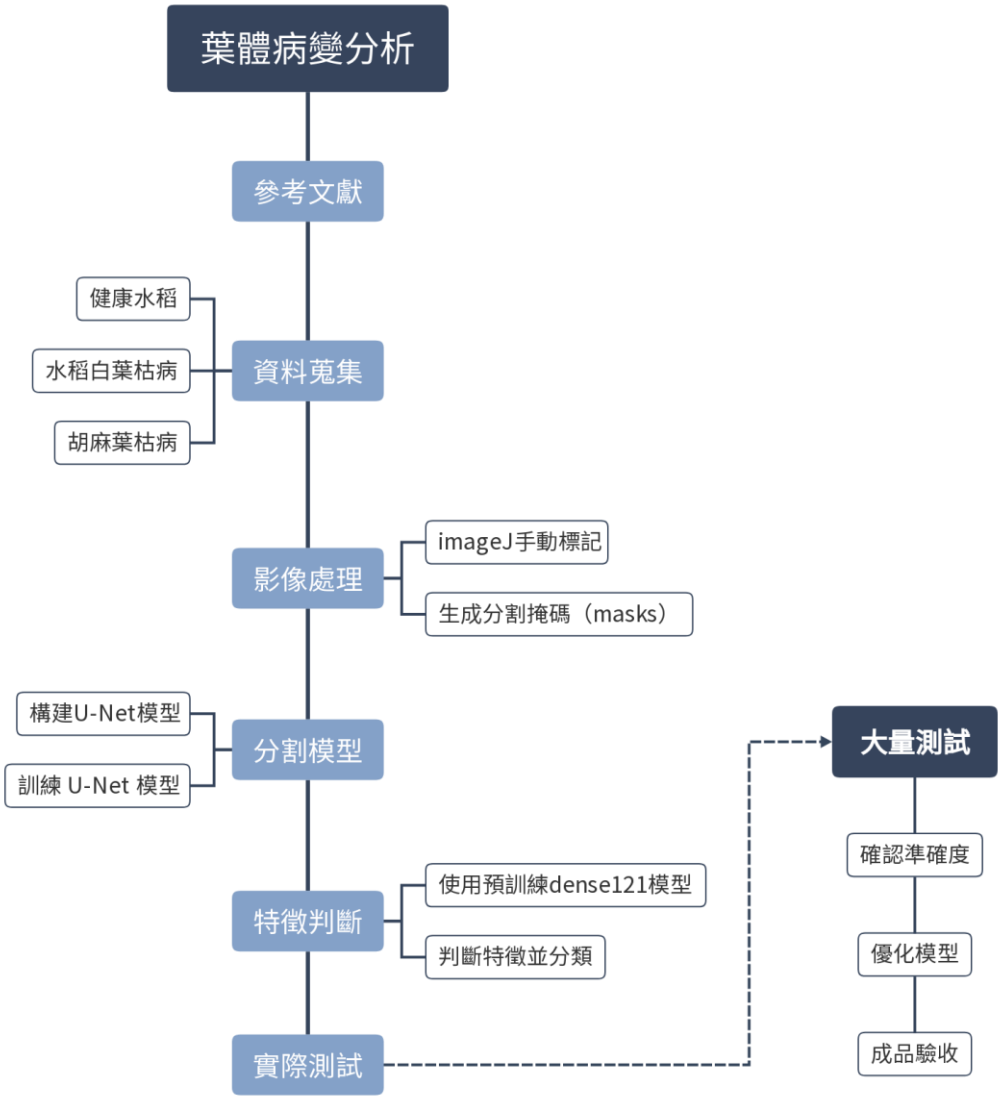


圖 6：研究流程圖（由第一作者使用 GitMind 繪製）

二、研究架構

本研究從資料蒐集開始，找到足夠數據集後，進行影像處理，取得黑白二值化的分割掩碼（masks），接著構建偕同 DenseNet-121 的 U-Net 模型，使用處理後的圖像和掩碼進行模型訓練，在分割模型訓練完成後，把數據分為三類標籤，用以訓練圖像辨識模型，使其能輸出分類結果，最後整合圖像分割和特徵判斷模型，製作簡單的應用程式來進行實際操作，研究架構如圖 7 所示。

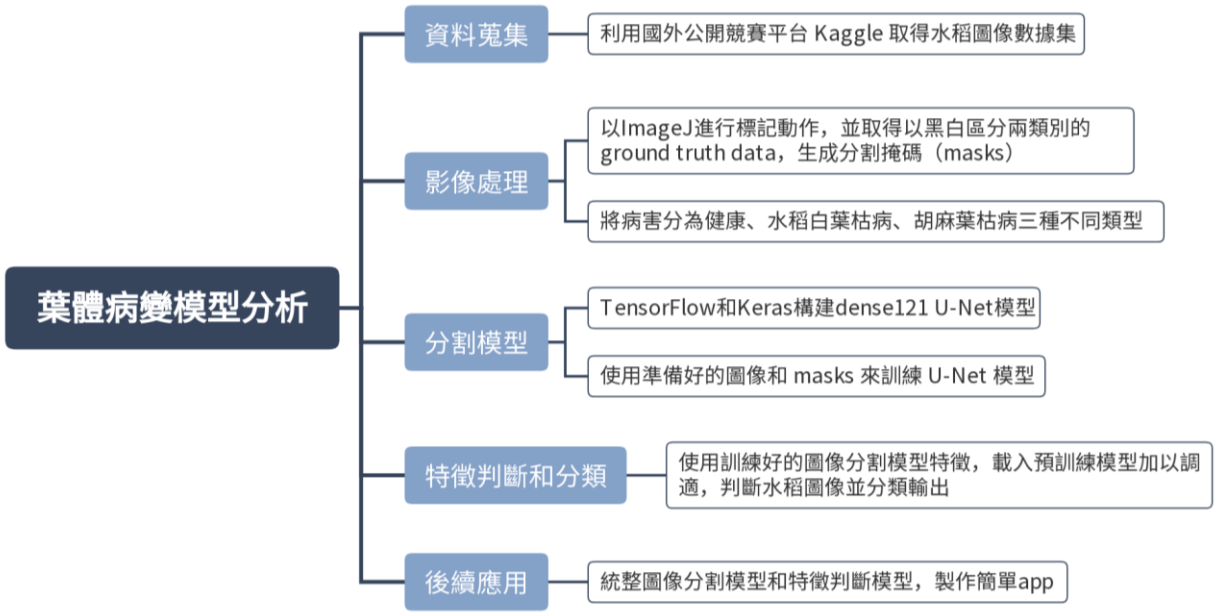


圖 7：研究架構圖（由第一作者使用 GitMind 繪製）

肆、研究過程

一、數據蒐集

我們在 Kaggle 上找到包含健康以及各疾病水稻葉面的圖片數據集 Rice Leaf Diseases Detection，挑出其中包含患有胡麻葉枯病 (brown spot) 和白葉枯病 (bacterial leaf blight) 與健康水稻的數據集。Kaggle 是全世界公認最大的資料科學社群平台，其提供了大量免費的數據及，涵蓋從生物、醫療、語言處理到電腦視覺等不同領域，用戶可以下載並使用這些數據集來進行模型訓練等項目開發。在初步取得需要的數據集後，再進行更精細的分類和劣質數據剃除，後續才能使模型學到更優質且準確的特徵。

二、影像處理

為取得分割掩碼，對於精簡過後的數據集採取二值化手動分割處理，方法為用 ImageJ 標記水稻葉圖片，轉為二值化的掩碼圖像，如下圖 9 所示，ImageJ 處理影像畫面截圖。標記過程為：將圖片的 Type 轉為 8-bit，接著手動標出水稻葉片的形狀，設為黑色，接著將 lower threshold level 設為 1，背景就會全部變成白色。最後成果為二值化掩碼圖像 707 張，作為 ground truth data 數據來訓練圖像分割模型。如下圖 8 所示，影像處理過程流程圖。

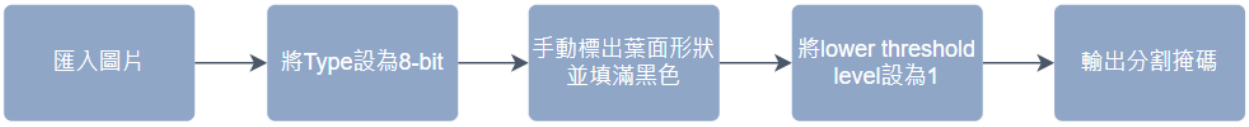


圖 8：影像處理過程流程圖（由第二作者使用 GitMind 繪製）

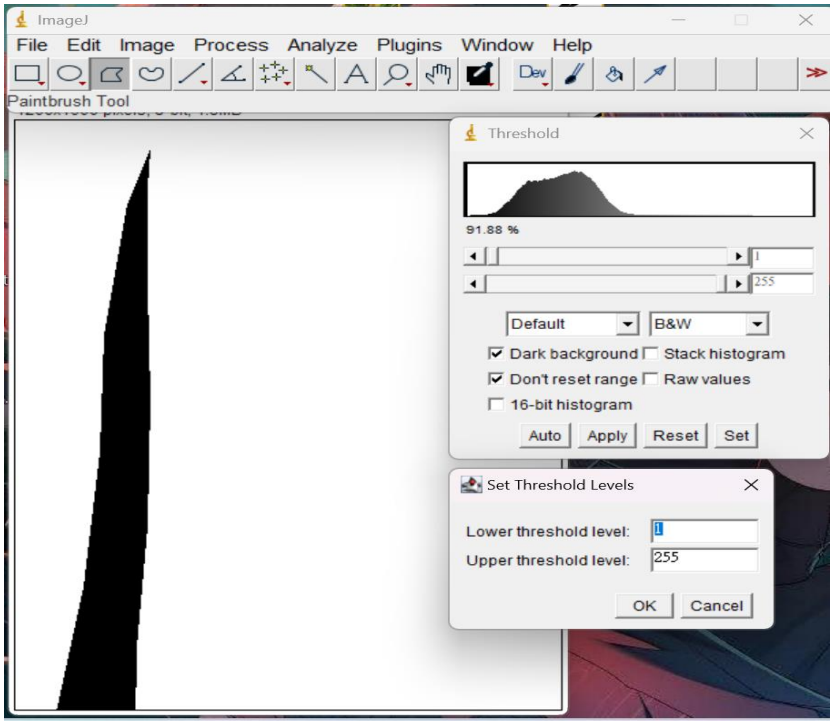


圖 9：ImageJ 處理影像畫面截圖（由第二作者自行標記並截圖）

三、檔案統一命名

在原圖及掩碼圖像資料夾中全選，按右鍵選擇重新命名，輸入 rice，按下 Enter 鍵後檔名即呈現 rice(1), rice(2), rice(3)... 以此類推。而掩碼圖像的檔名需呈現如「 rice(n)_mask」，所以在 Windows PowerShell 中輸入用於批量修改的腳本並執行，如下圖 10 所示。

```
PS C:\Users\user> $directory = "C:\Users\JIMMY\Desktop\mask"
>>
>>
>> $suffix = "_mask"
>>
>>
>> $files = Get-ChildItem -Path $directory
>>
>>
>> foreach ($file in $files) {
>>     $newName = $file.BaseName + $suffix + $file.Extension
>>     Rename-Item -Path $file.FullName -NewName $newName
>> }
```

圖 10：編碼器程式片段（由第二作者執行並截圖）

四、圖像分割模型

本模型採用了 DenseNet-121 作為編碼器，並凍結部分權重以避免對訓練結果造成過大影響。解碼器部分則使用 Unet 架構，並加入自定義的 Resize 操作來調整特徵圖尺寸，配合 Dropout 層以防止過擬合，從而提高模型的穩定性和泛化能力。如下圖 11 所示，編碼器程式片段。

```
#  使用dense121作編碼器
base_model  =  DenseNet121(weights='imagenet',  include_top=False,  input_tensor=inputs)
base_model.trainable  =  False    #  凍結權重

c1  =  base_model.get_layer('conv1_relu').output
c2  =  base_model.get_layer('pool2_relu').output
c3  =  base_model.get_layer('pool3_relu').output
c4  =  base_model.get_layer('pool4_relu').output
c5  =  base_model.get_layer('relu').output
```

圖 11：編碼器程式片段（由第一作者實作並截圖）

在深度學習領域中，DenseNet 是一種卷積神經網路 (Convolutional Neural Network, CNN) 架構，通過一種稱為「密集連接」的機制來提高模型的參數效率和信息流動性，其相當重要的一個特性是，每一層的輸出都直接傳遞給後續的所有層，這種機制確保了信息和梯度能夠在網路中更好地傳播，減少了梯度消失的問題。由於每一層接收了前面所有層的輸入 DenseNet 不需要像傳統的深度網路那樣依賴非常寬的層或非常深的網路來獲得豐富的特徵表示，因此它可以用較少的參數獲得類似或更好的性能。而 DenseNet121 是 DenseNet 家族中的一個較小模型，總共有 121 層，包含卷積層、池化層和密集連接塊，與其它較大版本的 DenseNet 相比，參數更少但仍具備良好表現，尤其適合運用在中小型數據集。

```
#  解碼器
u6  =  UpSampling2D((2,  2))(c5)
u6  =  ResizeLayer(target_size=(c4.shape[1],  c4.shape[2]))(u6)
u6  =  Concatenate()([u6,  c4])
c6  =  Conv2D(256,  (3,  3),  activation='relu',  padding='same')(u6)
c6  =  Conv2D(256,  (3,  3),  activation='relu',  padding='same')(c6)
c6  =  Conv2D(256,  (3,  3),  activation='relu',  padding='same')(c6)
d6  =  Dropout(0.4)(c6)

u7  =  UpSampling2D((2,  2))(d6)
u7  =  ResizeLayer(target_size=(c3.shape[1],  c3.shape[2]))(u7)
u7  =  Concatenate()([u7,  c3])
c7  =  Conv2D(128,  (3,  3),  activation='relu',  padding='same')(u7)
c7  =  Conv2D(128,  (3,  3),  activation='relu',  padding='same')(c7)
c7  =  Conv2D(128,  (3,  3),  activation='relu',  padding='same')(c7)
d7  =  Dropout(0.4)(c7)

u8  =  UpSampling2D((2,  2))(d7)
u8  =  ResizeLayer(target_size=(c2.shape[1],  c2.shape[2]))(u8)
u8  =  Concatenate()([u8,  c2])
c8  =  Conv2D(64,  (3,  3),  activation='relu',  padding='same')(u8)
c8  =  Conv2D(64,  (3,  3),  activation='relu',  padding='same')(c8)
c8  =  Conv2D(64,  (3,  3),  activation='relu',  padding='same')(c8)
d8  =  Dropout(0.4)(c8)

u9  =  UpSampling2D((2,  2))(d8)
u9  =  ResizeLayer(target_size=(c1.shape[1],  c1.shape[2]))(u9)
u9  =  Concatenate()([u9,  c1])
c9  =  Conv2D(32,  (3,  3),  activation='relu',  padding='same')(u9)
c9  =  Conv2D(32,  (3,  3),  activation='relu',  padding='same')(c9)
c9  =  Conv2D(32,  (3,  3),  activation='relu',  padding='same')(c9)
d9  =  Dropout(0.4)(c9)

u10  =  UpSampling2D((2,  2))(d9)
outputs  =  Conv2D(1,  (1,  1),  activation='sigmoid')(u10)
```

圖 12：解碼器程式片段（由第一作者實作並截圖）

ResizeLayer 是一個自定義的 Keras 層，用來將輸入的圖片或特徵圖調整到指定的尺寸，在此層的初始化方法中，我們設置了目標尺寸 target_size，而在接下來的前向傳播過程，call 方法會使用 TensorFlow 的 tf.image.resize 函數將輸入張量調整到指定大小，這樣當此層被用於模型中時，所有通過這個層的數據都會被自動調整為指定的尺寸，確保輸入輸出的一致性並滿足後續網路層的需求。如下圖 13 所示，自定義層程式片段。

```
@tf.keras.utils.register_keras_serializable()
class ResizeLayer(tf.keras.layers.Layer):
    def __init__(self, target_size, **kwargs):
        super(ResizeLayer, self).__init__(**kwargs)
        self.target_size = target_size

    def call(self, inputs):
        return tf.image.resize(inputs, self.target_size)
```

圖 13：自定義層程式片段（由第一作者實作並截圖）

訓練和構建模型主要基於 TensorFlow 上的 keras 來進行操作，且 numpy、Operating System Interfaces 和 matplotlib 也都會用到，TensorFlow 是由 Google Brain 開發的一個開源深度學習框架，廣泛運用於構建、訓練和部署模型，Keras 是 TensorFlow 的高級 API，提供了簡潔的接口，適合模型的快速搭建和訓練。 numpy 主要用於處理圖像數據和掩碼 (mask) ，將它們從圖像文件轉為數組，並進行二值化處理，使得掩碼值為 True 或 False，再通過 astype() 函數把 True/False 轉為 1/0，讓掩碼符合深度學習模型需要的型態。os (Operating System Interfaces) 用於處理文件和路徑操作，從指定文件夾中讀取圖像和對應的掩碼文件，並建構出完整的文件路徑。

```python
# 圖像處理
def load_images_and_masks(image_folder, mask_folder, image_size):
    images = []
    masks = []
    for filename in os.listdir(image_folder):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            img_path = os.path.join(image_folder, filename)
            img = load_img(img_path, target_size=image_size)
            img = img_to_array(img) / 255.0

            base_filename = os.path.splitext(filename)[0]
            mask_path = os.path.join(mask_folder, base_filename + '_mask.png')
            mask = load_img(mask_path, color_mode="grayscale", target_size=image_size)
            mask = img_to_array(mask) / 255.0
            mask = (mask > 0.5).astype(np.uint8)
            images.append(img)
            masks.append(mask)

    return np.array(images), np.array(masks)
```

圖 14：圖像處理部分程式片段（由第一作者實作並截圖）

完成的模型在進行 compile 之前，除了正則化 Dropout 層，為了進一步防止過擬合，我們設置早停機制並把 patience 設為 5，此外，由於數據量較少，我們也額外使用了旋轉、平移、裁剪的數據增強來確保數據豐富度能支撐起這個模型的訓練。

進行多次調適和測試後，本研究最終採用的模型設定為 lr=0.0001，epoch=100，batch_size=16，數據集劃分 8:2，作為性能指標表現最佳的模型並儲存測試，由下圖 15 中能看到雖然波動幅度稍大，但 loss 率穩定地下降，直到第 83 epoch 趨近飽和，最終數值為 0.342，而 acc 之最終數值為 0.851。雖然在訓練測試中表現亮眼，但下圖 16 顯示的預測分割測試結果相當不盡人意，除了勉強能分割出中央輪廓以外，並不能很好地切割出整條水稻，因此確定此模型的真實泛化能力稍低。

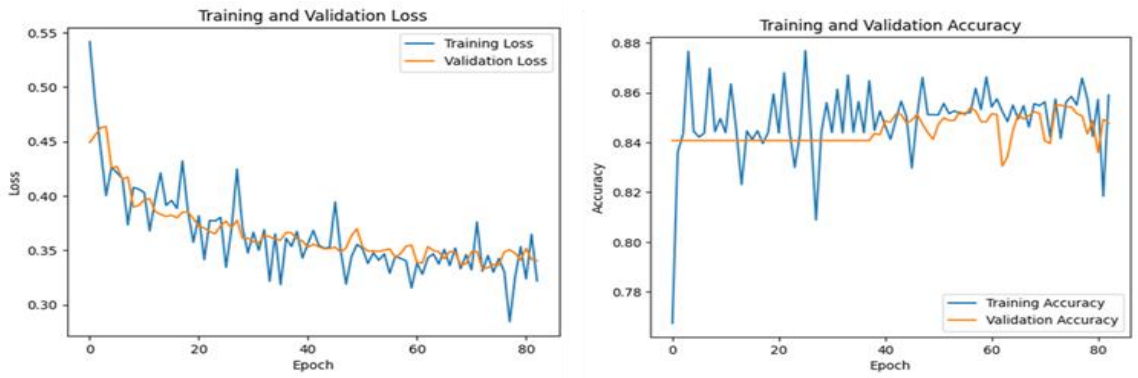


圖 15：可視化 acc 率和 loss 率圖表（由第一作者測試並截圖）

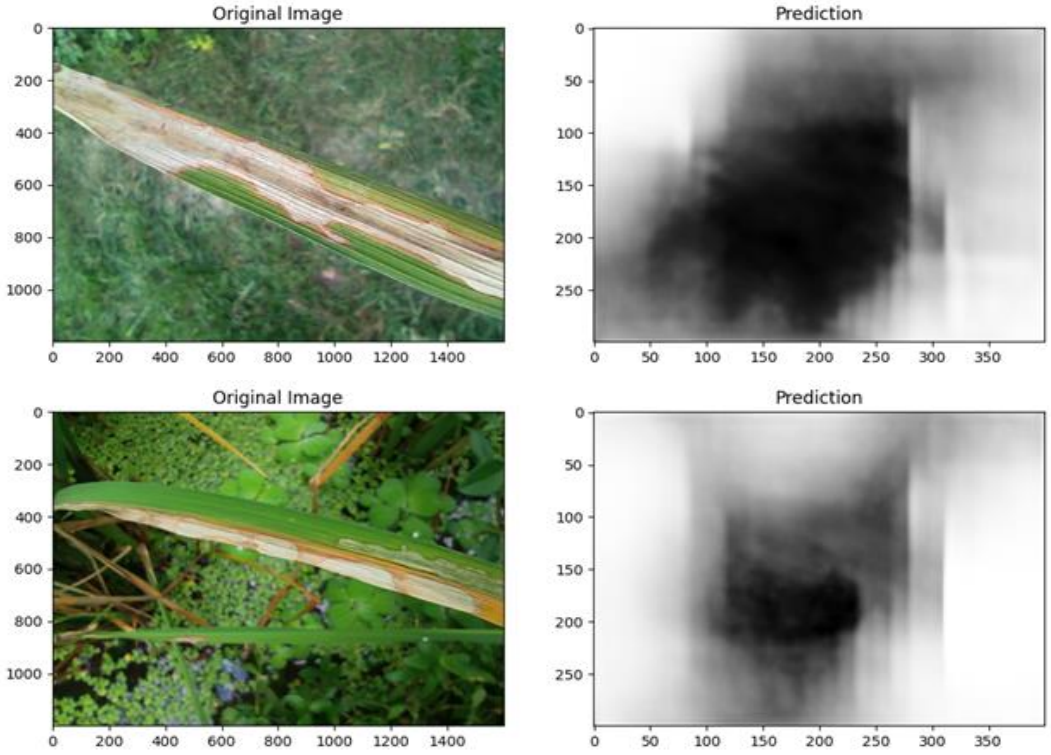


圖 16：分割模型預測結果（由第一作者測試並截圖）

五、圖像辨識模型

和圖像分割模型類似，同樣基於 TensorFlow 上的 keras，圖像辨識模型需要圖像處理、模型構建和訓練、最後預測並可視化結果。

初步圖像處理即為調整數據大小、轉數組和文件路徑操作，一樣使用 numpy 和 os，在這之後將三類 (brown, white, health) 圖像數據和標籤進行合併、預處理、打亂順序，並劃分出 80%的數據作為訓練集和 20%的數據作為測試集，再從訓練集中劃分出 10%作為驗證集。我們創建的標籤設定為如果圖像是棕色斑點，即胡麻葉枯病，模型會輸出 0，白葉枯病是 1，健康的水稻則是 2。如下圖 17 所示，圖像處理部分程式片段。

```
# 創建標籤
brown_spots_labels = np.array([0] * len(brown_spots_images))
white_spots_labels = np.array([1] * len(white_spots_images))
no_spots_labels = np.array([2] * len(no_spots_images))

# 合併標籤
images = np.concatenate((brown_spots_images, white_spots_images, no_spots_images), axis=0)
labels = np.concatenate((brown_spots_labels, white_spots_labels, no_spots_labels), axis=0)

# 歸一化
images = images / 255.0

# 轉成獨熱編碼
labels = to_categorical(labels, num_classes=3)

# 打亂數據
indices = np.arange(len(images))
np.random.shuffle(indices)
images = images[indices]
labels = labels[indices]

# 劃分數據集
from sklearn.model_selection import train_test_split
train_images, test_images, train_labels, test_labels = train_test_split(images, labels, test_size=0.2)
train_images, val_images, train_labels, val_labels = train_test_split(train_images, train_labels, test_size=0.1)
```

圖 17：圖像處理部分程式片段（由第一作者實作並截圖）

考量到再構建一個模型並訓練將消耗大量時間，性能也較為不穩定，我們選擇直接加載 DenseNet121 預訓練模型來進行遷移學習。先凍結模型中的所有層權重，防止在訓練過程中更新，並添加自訂義權連接層進行三類分類，這樣訓練完畢後，只有全連接層會被更新。經過調適和修正，最後利用 matplotlib 可視化結果，由下圖 18 確認 acc 率和 loss 率都維持在高水準的穩定狀態，並查看下圖 19 來檢查隨機預測結果的準確度。

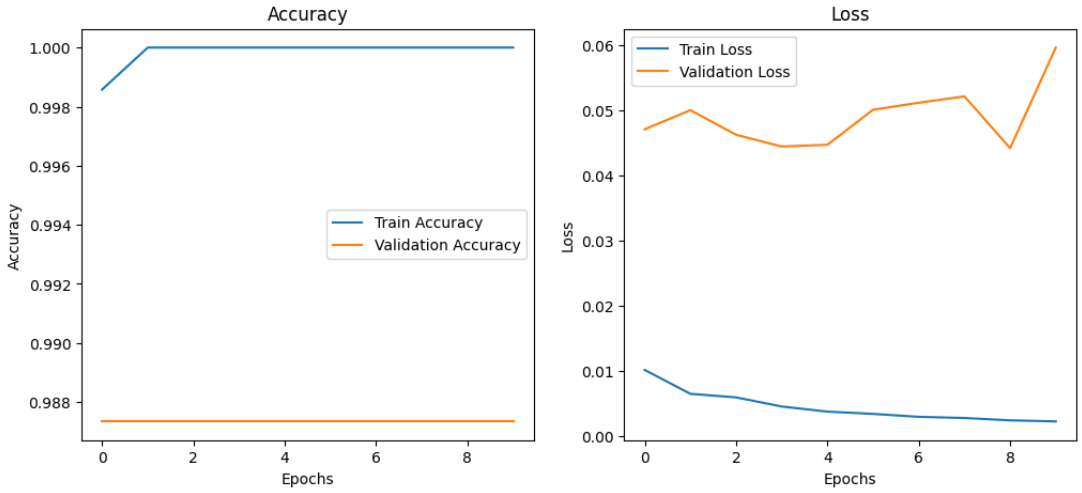


圖 18：可視化模型訓練性能結果（由第一作者測試並截圖）

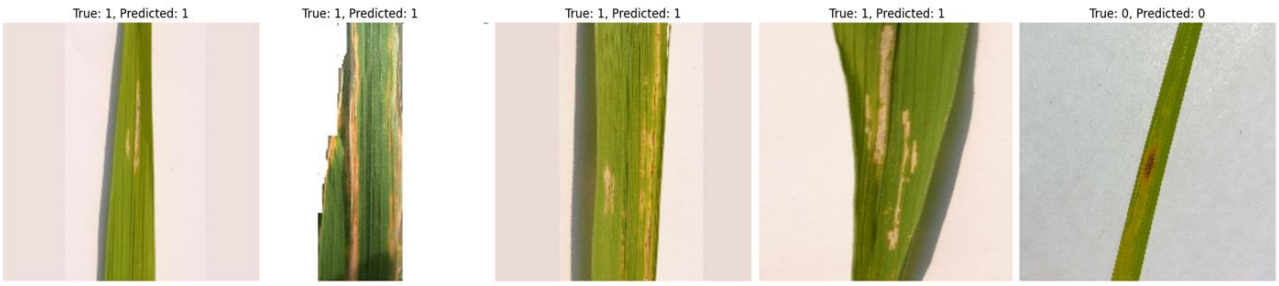


圖 19：圖像辨識預測結果（由第一作者測試並截圖）

六、模型統合及部署

完成分割模型和辨識模型後，我們在 Visual Studio 2020 為平台來進行模型整合及部署，使用的 API 統合框架是 Flask，該過程旨在將訓練好的模型轉換為可供應用程式調用的形式，最終打包為可安裝的 APK，方便在移動端設備上使用。具體步驟如下：

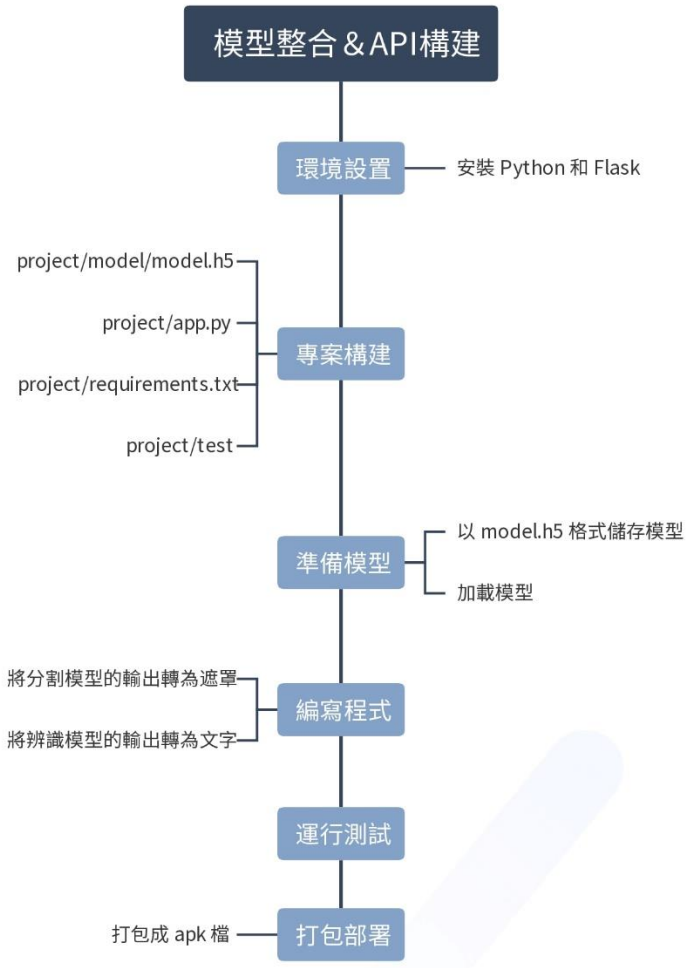


圖 20：步驟示意圖（由第一作者使用 GitMind 繪製）

在部署過程中，我們首先需要準備合適的開發環境，主要包括 Python 環境安裝及必要的函式庫，如 Flask、TensorFlow、NumPy 等，設置 Flask API 框架，作為後臺用於處理用戶請求並回傳模型預測結果，整個過程以 Visual Studio 2020 為主要開發環境，整合前端與後端系統。

而 API 的開發由載入模型開始，先使用 TensorFlow 或 Keras 讀取.h5 格式的模型權重，接著定義 API 端點，使其能接受用戶上傳的圖像，並返回預測結果，再撰寫進行圖像前處理的程式，功能為對接收到的圖像進行尺寸調整、標準化處理，以確保輸入格式與模型所需相符，處理完圖像後，將其輸入部署完畢的圖像分割模型，獲取病斑的遮罩 (mask)，再把遮罩與原始圖像結合，形成只包含病斑部分的新圖像，作為辨識模型的輸入，相較之下，圖像辨識模型的運作則簡單許多，只需輸入病斑部分的新圖像，獲取病害類別預測結果，整個後臺推論流程就結束了。最後後臺會將預測結果轉換為 JSON 格式回傳給前端，前端再將結果輸出成設定格式並顯示給用戶。

圖 21：App 首頁（由第二作者製作）

圖 22：App 預測結果（由第二作者製作）

七、針對系統進行 SWOT 分析與比較

表 11：葉體病變辨識模型之 SWOT 分析（由第三作者製作）

Strengths 優勢	Weaknesses 劣勢
1、 專注於稻葉病害分析 2、 及時預警病蟲害 3、 符合 SDG 2 - 零饑餓、SDG 12 - 負責任消費和生產、SDG 13 - 氣候行動、SDG 9 - 工業創新與基礎設施的概念 4、 減少人力成本	1、 局限於水稻疾病辨識 2、 辨識範圍限制於葉片 3、 缺乏實際應用經驗
Opportunities 機會	**Threats 威脅**
1、 結合應用程式(APP)，發展可能性高 2、 應用於食農教育領域 3、 推動 SDG 4 - 優質教育	1、 精準度受圖片清晰度影響 2、 資料量不足

伍、研究結論與建議

一、 結論

（一）由於數據量過少，導致圖像分割模型的泛化能力大幅下降。

（二）能分辨的病徵目前只有兩種，不夠多樣化。

（三）兩個模型尚未整合完畢，具體功能性不足。

二、 建議

（一）處理大量而更多元的數據集，並用來訓練圖像分割模型。

（二）於圖像辨識模型中設置更多種類標籤，導入多樣的水稻病徵數據。

（三）在兩個模型都調適完成後，透過 App 的方式來進行部署整合，增加具體功能性。

三、未來展望與應用

（一）**增加其它病因：**本研究目前僅針對水稻葉片上的白葉枯病與胡麻葉枯病進行辨識，並未涵蓋其他病害的識別，因此無法確定其他病徵的具體表現。未來，我們期望能夠將水稻葉片上的病害識別能力進一步擴展，實現對水稻各種疾病的全面辨識。此外，我們計畫將辨識範圍不僅限於葉片，還將納入水稻植物的其他重要部位，如根部、莖部等，從而實現更加全面的病害檢測。

（二）**擴充訓練數據：**鑒於本研究所使用的數據量有限，未來可以致力於擴充資料集，尤其針對其它的疾病以及其它部位的病徵做分析，以提供模型預測之準確率，並更完整我們對水稻疾病分析的研究。

（三）**結合影像監測系統：**為因應少子化問題並有效降低務農人口的需求，我們希望葉片病變檢測模型不僅侷限於教育或等待農民巡查時才發現問題。未來，系統可以結合物聯網（IoT）、大數據分析及人工智慧技術，以高精度的方式不斷更新農田即時狀態與數據，並透過農民經驗和歷史數據的累積，建立更加智能的病變監測機制，實現更高效且智慧化的農業管理。

（四）**分析成本效益：**在本研究的實作部分，由於時間與資金的限制，我們未能將系統實際應用於稻田中，並進行長期的監測與數據收集，因此未能對使用本研究所開發的應用程式所帶來的農業成本減少效果進行具體的比較與驗證。儘管如此，若未來能獲得足夠的資金與資源支持，我們計畫將本系統進一步實現並應用於實際農業生產中，進行實地測試與效益評估。透過對比有無使用此 App 的稻田，將能清楚衡量其在提升農民生產效率、提早預防病害、降低病蟲害發生率等方面的實際效果，並針對農業成本的節省與生產效能的提升進行成本效益分析，能為未來農業數位化與智慧化的發展提供寶貴的參考與依據。

陸、參考文獻

李湘渝（2022）。基於卷積神經網路之深度學習方法之蘋果葉面病害辨識與分類。〔碩士論文。國立屏東科技大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/rpf9fv。

李柔嫻（2020）。應用於智慧農業之作物生長影像監測系統。〔碩士論文。中原大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/46kr9y。

林郁榮（2023）。專為咖啡產量預估與疾病偵測設計之智慧農業監測系統。〔碩士論文。逢甲大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/58py9b。

葉俊巖（2006）。水稻育苗病害管理。農政與農情，(166)。取自：https://www.moa.gov.tw/ws.php?id=11074

林駿奇（2009 年 9 月）。水稻白葉枯病之生態與防治。花蓮區農業專訊, 69, 21-23。取自：https://www.hdares.gov.tw/upload/hdares/files/web_structure/756/bull-69_21-23.pdf

張義璋（2007 年 11 月）。稻胡麻葉枯病。植物保護圖鑑系列 8, 252-257。取自：https://www.aphia.gov.tw/Publish/plant_protect_pic_8/ricePDF/06-04.pdf

Amazon Web Services（無日期）。監督式學習與非監督式學習之間有何差異？取自：https://aws.amazon.com/tw/compare/the-difference-between-machine-learning-supervised-and-unsupervised/

胡依淳（2018）。深度卷積神經網路中卷積層之分析及比較。〔碩士論文。國立暨南國際大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/686x7u。

Andreja Velimirovic. (2024). Supervised vs. Unsupervised Machine Learning Explained. https://phoenixnap.com/blog/supervised-vs-unsupervised-learning

楊學智（2024 年 1 月 15 日）。卷積神經網絡(Convolutional Neural Network, CNN)在 python 上面的實現。https://reurl.cc/nvN0Yv。

Xie, L., & Yuille, A. (2017). Genetic cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1379-1388).

吳季桓（2010）。自動分類的實作：KNN 與 SVM。〔碩士論文。國立中正大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/vnxsgd。

林佳姿（2020）。搭配類神經 CNN、LSTM 及 DNN 方法於高混合度之母音辨識。〔碩士論文。國立中正大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/ed99mu

趙國桓（2024）。機器學習與深度學習在 RSSI 基礎的 Wi-Fi 室內定位中的效率比較研究：從 KNN 和 SVM 到 CNN+LSTM。〔碩士論文。國立臺北科技大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/p4s2w5

林梓峰（2022）。使用臉部特徵點之 RNN/LSTM 疲勞駕駛偵測。〔碩士論文。中原大學〕臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/ts9vf9。

Cheng, H. D., Jiang, X. H., Sun, Y., & Wang, J. (2001). Color image segmentation: advances and prospects. Pattern recognition, 34(12), 2259-2281.

黃靖程（2025）。應用影像分割於淹水圖像辨識。〔碩士論文。 國立臺北科技大學〕 臺灣博碩士論文知識加值系統。取自：https://hdl.handle.net/11296/5y8v55。

Yousefi, J. (2011). Image binarization using Otsu thresholding algorithm. Ontario, Canada: University of Guelph, 10.

Sauvola, J., & Pietikainen, M. (1999). Adaptive document image binarization, Pattern Recognition. Volume, 33, 00055-2.

Kaur, D., & Kaur, Y. (2014). Various image segmentation techniques: a review. International Journal of Computer Science and Mobile Computing, 3(5), 809-814.

Gu, Y., Ai, Q., Xu, Z., Yao, L., Wang, H., Huang, X., & Yuan, Y. (2024). Cost-effective image recognition of water leakage in metro tunnels using self-supervised learning. Automation in Construction, 167, 105678.

Fujiyoshi, H., Hirakawa, T., & Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. IATSS research, 43(4), 244-252.

Jain, A. K., Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. IEEE Transactions on pattern analysis and machine intelligence, 22(1), 4-37.

[註 1] 陳繹年（無日期）。水稻白葉枯病。https://reurl.cc/5dDmdz。

[註 2] 陳繹年（無日期）。水稻胡麻葉枯病。https://reurl.cc/Yq4x4n。

[註 3] Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2019). Panoptic segmentation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9404-9413).

[註 4] OpenCV.(無日期). Image Thresholding. https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html