逢 甲 大 學
資訊工程學系
專題研究報告

基於量化與剪枝優化**PCB**瑕疵辨識之**YOLO**模型

指導教授:林哲維 老師

學生:林慧婷 D1104724
段昱如 D1104843
吳珮瑜 D1112149
吳姁恩 D1273851

中 華 民 國 一 一 三 年 十 一月

誌 謝

感謝指導教授林哲維老師一直以來對我們的細心指導與無私奉獻。每當我們在學術研究或實驗過程中遇到困難，老師總是能在繁忙的工作中抽出寶貴時間，耐心地傾聽我們的疑惑，並一步步引導我們找到問題的根源，進而提出有效的解決方案。林哲維教授不僅具備深厚的專業知識，還展現出對學生無比的關懷與責任感，這份專注與敬業精神，讓我們深感敬佩，也激勵我們在學習與研究的道路上不斷精進。他的諄諄教誨與無私奉獻，將永遠是我們學習與成長的重要典範。

感謝中國大陸北京大學智能機器人開放實驗室提供PKU-Market-PCB開放資源，讓我們得以使用這個高品質的資料集進行研究與開發。該資料集對於我們的研究工作具有重要的支持作用，幫助我們在技術創新和實驗推進方面取得了顯著的進展。

摘要

在現代工業生產中，產品品質的檢測是確保最終出品符合標準的關鍵環節之一。傳統上，這些檢測工作往往依賴大量人工來逐一檢查每件產品，確保其無瑕疵。然而，這種方式不僅耗費了大量的人力和時間，還容易因為長時間、高強度的工作導致檢測員的視覺疲勞，從而增加誤判和漏檢的風險。此外，隨著生產規模的擴大，人工檢測的效率也無法滿足現代大規模、高速生產線的需求。因此，如何通過技術手段提升檢測效率和準確度，並減少對人工檢測的依賴，成為工業生產中的一個重要課題。

隨著自動化技術的飛速發展，特別是自動光學檢測（AOI）技術的應用，取代人工檢測已成為工業領域的趨勢。AOI技術能夠利用攝像機、光學傳感器和算法來實現對產品的高效檢測，並能在極短時間內完成大量產品的瑕疵辨識。然而，為了進一步提升自動檢測系統的智能化水平，應用深度學習技術來開發高準確度的瑕疵檢測模型成為目前的研究熱點之一。

本專題應用YOLOv8n模型。我們選擇了PCB資料集作為模型訓練的基礎，通過深度學習技術對資料中的各類瑕疵進行分類與檢測。在模型訓練完成後，為了進一步提升模型的應用性能，我們對模型進行了剪枝和靜態量化。縮減模型的推理時間、減少模型的層數和參數量，從而降低運算資源的消耗，使其在保持準確度的同時大幅提升運行效率。這些優化過程使得模型更加輕量化，並其應用於自動光學檢測（AOI）流程中，幫助企業提高生產良率、減少人工檢測成本。

關鍵詞**:** 自動光學檢測、**YOLO**、瑕疵辨識、模型剪枝、靜態量化

目 錄

第一章 緒論 1
1.1 研究背景 1
1.2 研究動機 1
1.3 研究目的 1
1.4 研究範圍 1
1.5 研究方法 2
1.6 研究流程 2

第二章 相關知識及文獻探討 3
2.1 缺陷種類及定義 3
2.1.1 標誌分類0 Missing_hole 3
2.1.2 標誌分類1 Mouse_bite 3
2.1.3 標誌分類2 Open_circuit 3
2.1.4 標誌分類3 Short 4
2.1.5 標誌分類4 Spur 4
2.1.6 標誌分類5 Spurious_copper 4
2.2 YOLO (You Only Look Once) 模型 5
2.2.1 Backbone 5
2.2.2 Neck 6
2.2.3 Head 6
2.2.4 卷積 6
2.2.5 正規化 6
2.2.6 SiLU 6
2.2.7 2D conv 6
2.2.8 C2f 6
2.2.9 SPPF 7
2.2.10 全局平均池化層 7
2.2.11 殘差連接 7
2.3 剪枝 7
2.4 靜態量化 8
2.5 混淆矩陣 8
2.6 mAP50 8

第三章 研究方法 10
3.1 **資料預處理及分類** 10
3.2 **影像標註** 10
3.3 **模型選擇與構建** 10
3.4 **模型訓練** 10
3.5 **模型驗證**(val) 11
3.6 **模型優化** 11
3.7 **模型測試與評估** 11
3.8 **模型結果展現** 12

第四章 實證分析 13
4.1 **資料集來源** 13
4.1.1 **料集名稱** 13
4.1.2 **資料集來源** 13
4.1.3 內容分成 13
4.1.4 瑕疵分類 13
4.2 **影像標記** 13
4.3 **影像擴充** 14
4.4 **模型訓練** 14
4.5 **模型剪枝和刪除模型層** 14
4.6 **模型量化** 15
4.7 剪枝後的模型進行量化 16
4.8 結果 16
4.9 模型成果展現 26

第五章 結論與建議 27

第六章 參考文獻 28

圖目錄

圖 2.1 標誌分類0 Missing_hole 3
圖 2.2 標誌分類1 Mouse_bite 3
圖 2.3 標誌分類2 Open_circuit 4
圖 2.4 標誌分類3 Short 4
圖 2.5 標誌分類4 Spur 4
圖 2.6 標誌分類5 Spurious_copper 5
圖 2.7 YOLO(You Only Look Once) 模型 5
圖 4.1 專題完整架構圖 13
圖 4.2 經過擴充訓練後的混淆矩陣結果 14
圖 4.3 刪除模型層前 15
圖 4.4 刪除模型層後 15
圖 4.5 先剪參數再刪除層-模型層數比較 17
圖 4.6 先剪參數再刪除層-模型參數量比較 18
圖 4.7 先剪參數再刪除層-模型檔案大小比較 18
圖 4.8 先剪參數再刪除層-模型速度比較_GPU 18
圖 4.9 先剪參數再刪除層-模型速度比較_CPU 19
圖 4.10 先刪除層再剪參數-模型層數比較 19
圖 4.11 先刪除層再剪參數-模型參數量比較 20
圖 4.12 先刪除層再剪參數-模型檔案大小比較 21
圖 4.13 先刪除層再剪參數-模型速度比較_GPU 21
圖 4.14 先刪除層再剪參數-模型速度比較_CPU 21
圖 4.15 各個模型的f1 score 22
圖 4.16 last.pt原始模型訓練結果混淆矩陣 23
圖 4.17 先剪參數再剪層的最小模型預測結果混淆矩陣 23
圖 4.18 先剪層再剪參數的最小模型預測結果混淆矩陣 23
圖 4.19 先剪參數再剪層的最小模型量化為fp16預測結果混淆矩陣 24
圖 4.20 先剪層再剪參數的最小模型量化為fp16預測結果混淆矩陣 24
圖 4.21 先剪層再剪參數的最小模型量化為int8預測結果混淆矩陣 25
圖 4.22 先剪參數再剪層的最小模型量化為int8預測結果混淆矩陣 25
圖 4.24 網頁呈現流程圖 26

第一章 緒論

1.1 研究背景

印刷電路板（Printed Circuit Board, PCB）是各類電子產品中不可或缺的關鍵零組件，無論是電子錶、手機、相機、液晶螢幕還是電腦組件等產品，都必須依賴PCB來實現其功能。隨著消費性電子產業的蓬勃發展，PCB的生產越來越精密且複雜，因此為了確保產品品質，必須高效進行檢測。然而，傳統的人工目檢方法存在耗時長、誤判率高等問題，特別是隨著電路板的日益複雜，檢測的難度和誤判率也隨之上升。因此，自動光學檢查系統（Automated Optical Inspection, AOI）逐漸被引入，以提升檢測效率和準確性。AOI系統利用機器視覺技術，結合光學、電控、機構以及檢測軟體，能夠快速篩檢成品和半成品的瑕疵。

1.2 研究動機

雖然 AOI 與傳統人工目檢相比效率以及精確度都有大幅提升，瑕疵判定標準也較為一致，但還是有不足的地方，AOI 採用 Aule-based 判斷機制，利用程式語言撰寫檢測邏輯，僅能用定義好的參數作為基準檢測樣本瑕疵，舉例而言，若將檢測邏輯定義為圓形，非圓形的瑕疵便無法被檢測出來，因此常產生漏檢問題。且因為自動光學檢測系統是透過傳統演算法進行檢測，所以參數設定往往較為嚴格，為了達到百分百檢測率，誤判率也跟著提高，所以通常需要再耗費人力以及時間進行二次檢查。最後與 AI 相比，AOI 需要的樣本數以及調整設定時間較長，若需要條機、更換零組件，需要重新設定參數、對位，再加上軟硬體調整過程複雜，維修時間從幾日到數週都有，相當耗時。
隨著零件生產環境漸漸由大規模生產演變成少量多樣，傳統 AOI 難以適應產線的多樣化需求，因此我們將利用 YOLO 製作 AI 辨識系統，AI 在經過影像辨識訓練後，可以依據瑕疵特徵做偵測，模型建立速度也更快，並且後續只要以增加樣本的方式就能提高檢測率或降低漏檢率。若以後需要新增新的瑕疵檢測目標，AOI 需要重新編寫程式以及調整參數，而 AI 只需要重新訓練新的目標檢測模型，能投入生產線的時間比 AOI 快很多。

1.3 研究目的

利用YOLO模型開發一套基於AI的瑕疵辨識系統，該系統將能夠快速、準確地檢測PCB中的瑕疵。透過深度學習技術，AI系統在經過影像辨識訓練後，能夠根據瑕疵特徵進行檢測，相較於傳統的AOI系統，模型建立速度更快。當需檢測的新瑕疵目標出現時，只需重新訓練檢測模型，而不必重新編寫程序和調整參數，這將顯著提高投入生產線的效率。最終，研究期望能實現一個高效且靈活的PCB檢測解決方案，以滿足現代化生產線對多樣化檢測需求的挑戰。

1.4 研究範圍

本研究專注於開發一套基於YOLO模型的自動化瑕疵檢測系統，主要針對印刷電路板（PCB）的瑕疵辨識。
資料集選擇：使用多種類型的PCB圖像作為訓練和測試資料集，這些圖像涵蓋了六種不同類型的瑕疵。分別為: 缺孔、鼠咬、斷路、短路、雜散、雜銅。
模型開發：應用YOLOv8中的YOLOv8n進行深度學習模型的訓練，以提高瑕疵檢測的準確性和效率。

1.5 研究方法

資料收集與預處理：將北京大學智能機器人開放實驗室提供PKU-Market-PCB開放資源作為資料集，並進行標註，以提高資料集的質量。
模型訓練：利用YOLOv8n進行深度學習模型的訓練，並將訓練後的模型進行量化及剪枝。
模型評估：通過混淆矩陣對模型進行性能評估，並分析其在檢測不同類型瑕疵時的表現。
實驗驗證：使用經過剪枝和量化的模型預測其他PCB的資料集。

1.6 研究流程

文獻回顧：調查和分析目前在PCB瑕疵檢測領域的研究進展和技術。
資料集建構：收集、整理和標註PCB瑕疵圖像資料集。
模型訓練：利用YOLOv8n進行模型的訓練和優化。
性能評估：對訓練後的模型進行測試，並分析其檢測效果。
實證分析：在生產環境中進行應用驗證，評估系統的實際效果。
結果分析與總結：整理研究結果，進行討論與總結，提出未來的研究方向。

第二章 相關知識及文獻探討

隨著深度學習技術的快速發展，物體檢測在計算機視覺領域中得到了廣泛應用。特別是YOLO（You Only Look Once）系列模型，因其高效性和精度，在物體偵測、瑕疵檢測等領域被廣泛採用。然而，為了進一步提升模型的實用性，剪枝和量化技術逐漸成為焦點研究方向。現有研究雖然在物體檢測和瑕疵識別中取得了顯著進展，但仍存在模型體積過大、運算效率較低的問題。本研究將通過應用剪枝和量化技術，優化YOLOv8模型，從而提高瑕疵檢測系統的實時性與可用性，為資源受限環境下的工業應用提供更有效的解決方案。

2.1 缺陷種類及定義

使用開源的 PCB 數據集，此數據及一共有 6 種不同的缺陷種類，分別為 0: 缺孔、1: 鼠咬、2: 斷路、3: 短路、4: 雜散、5: 雜銅。

2.1.1 標誌分類0 Missing_hole

造成瑕疵原因為蝕刻不足，孔洞沒有塞孔，會導致元件無法正確安裝或沒有足夠的電氣連接。

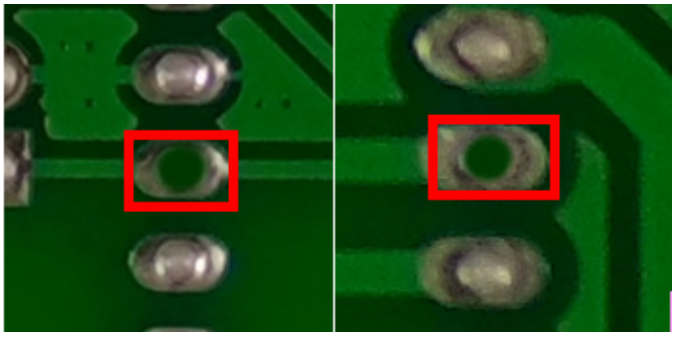

圖 2.1 標誌分類0 Missing_hole

2.1.2 標誌分類1 Mouse_bite

造成瑕疵原因為蝕刻過量，在PCB的邊緣或連接處留下的痕跡，會導致PCB邊緣的結構強度下降，使邊緣更容易破裂或分層。

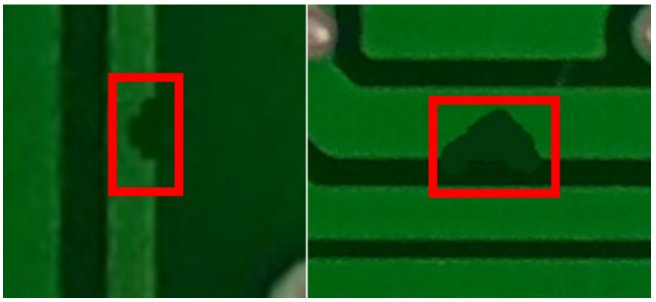

圖 2.2　標誌分類1 Mouse_bite

2.1.3 標誌分類2 Open_circuit

造成瑕疵原因為本應連接的兩個電路點之間出現斷開，會導致電路無法正常工作。

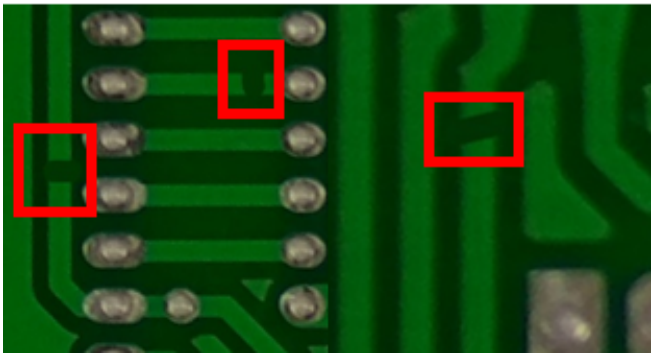

圖 2.3 標誌分類2 Open_circuit

2.1.4 標誌分類3 Short

造成瑕疵原因為電路中不應該連接的兩個電路點之間存在導電連接，會導致元件損壞或電路板燒毀。

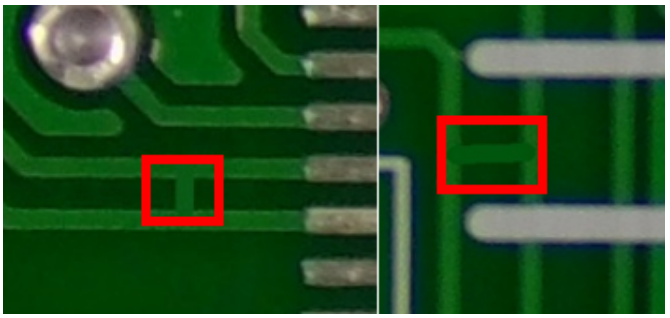

圖 2.4 標誌分類3 Short

2.1.5 標誌分類4 Spur

造成瑕疵原因為蝕刻不足，出現不需要的細小銅枝或突起，會導致導致短路或元件受損。

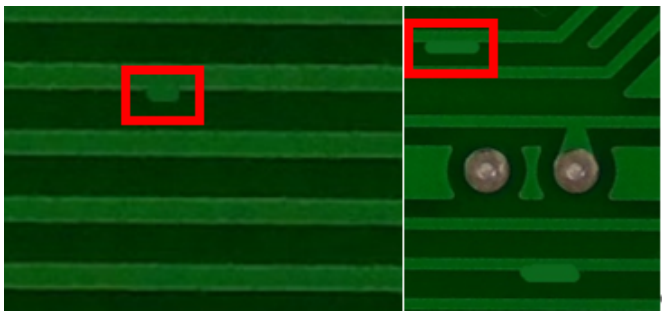

圖 2.5 標誌分類4 Spur

2.1.6 標誌分類5 Spurious_copper

造成瑕疵原因為電鍍槽或電極孔中含有過多的污染物，會導致銅鹽結晶不均勻，產生的不需要的銅箔或銅塊，會導致PCB短路。

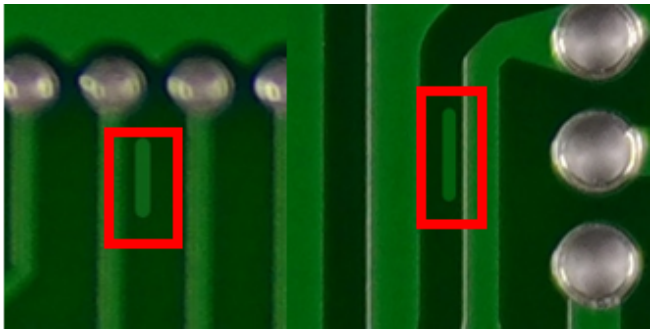

圖 2.6 標誌分類5 Spurious_copper

2.2 YOLO (You Only Look Once) 模型

Ultralytics 沒有直接將 open source 命名為 YOLOv8 而是使用 ultralytics，原是 ultralytics 把它定義為一個框架，而非特定一個算法。其最大的特點是可擴展性，讓使用者可以輕易的去更換自己想要的 Backbone，Neck 等等。

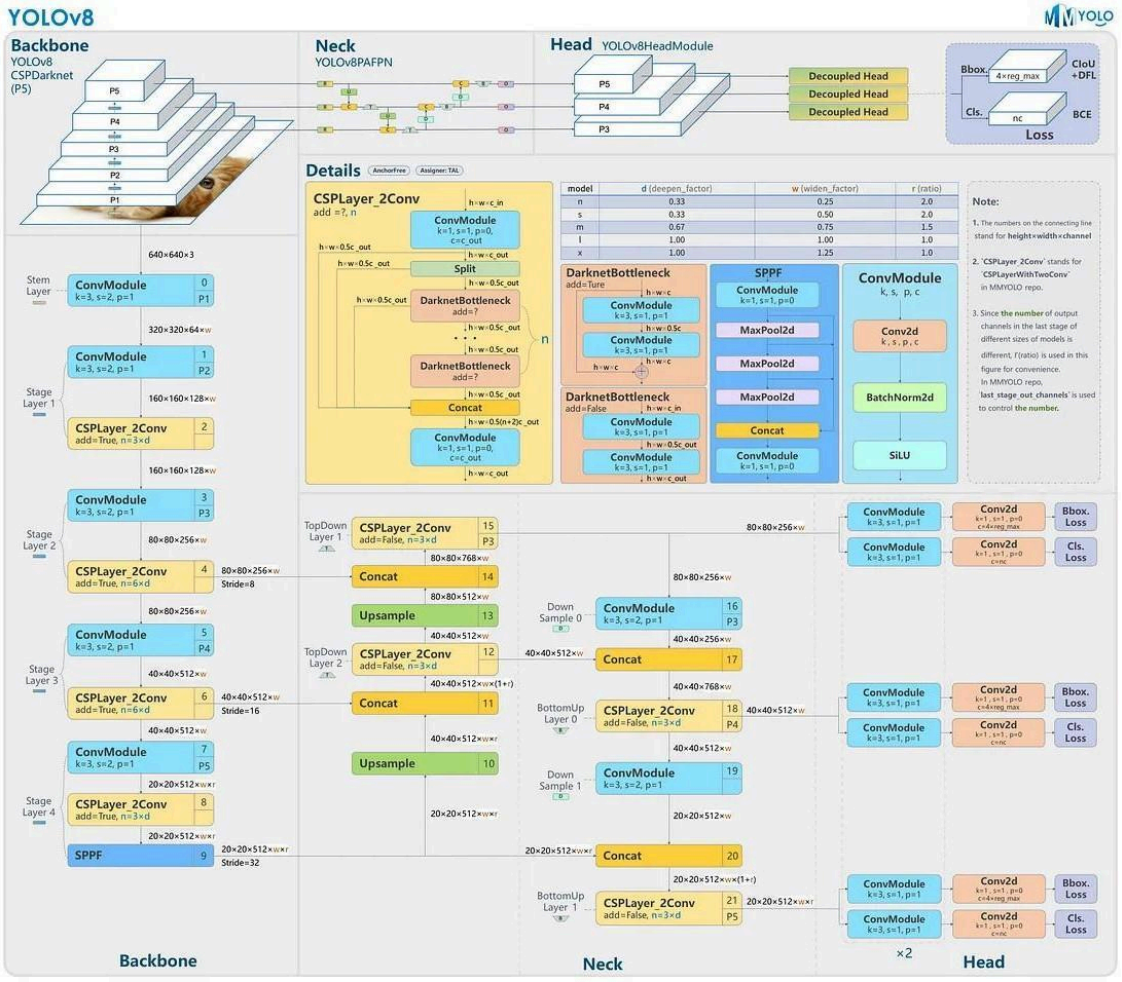

圖 2.7 YOLO(You Only Look Once) 模型

以上是 Yolov8 網路結構圖，主要由三個部分組成：

2.2.1 Backbone

模型基礎，從輸入的圖片提取特徵，而提取的特徵為目標檢測的基礎。

2.2.2 Neck

讓Backbone及Head更好融合以及提取特徵以提升特徵多樣性及穩健性。

2.2.3 Head

改成了Decoupled-Head，將分類與檢測頭分離，對特徵圖進行處理已產生模型的輸出結果的過程

2.2.4 卷積

卷積是一個包含多個層的組合，用於提升模型的特徵提取能力。首先，它使用 2D 卷積（Conv2D）來捕捉圖片中的局部特徵，接著透過 BatchNorm2D 正規化數據以加快訓練速度並穩定模型。接下來，使用 SiLU（Sigmoid Linear Unit）作為激活函數，以增強模型的非線性表示能力。該模塊還透過降採樣（即池化操作）縮小圖片尺寸，這有助於減少計算量，同時保留重要的特徵資訊。這個結構旨在有效提取圖片中的關鍵特徵，並提高模型的整體性能。

2.2.5 正規化

資料縮放技術將數據值限制在固定區間內，通常是 0 到 1 之間，這樣能夠提升模型的收斂速度和精度。當資料的分佈不符合常態分佈時，縮放技術尤其有效，因為它能將數據轉換到平均值為 0、方差為 1 的正態分佈，使數據分佈更加一致。同時，這樣的處理方式還能減少梯度消失問題，幫助模型在訓練過程中保持穩定，從而提高最終的性能表現。

2.2.6 SiLU

當輸入的 x 值越大時，輸出會逐漸趨近於 x，而當輸入的值較小時，輸出則會接近於 0。這種函數具備良好的平滑性和非單調性，能夠處理不同範圍的數值變化。此外，它具有零中心性，這意味著輸出的平均值會接近零，有助於加速模型的收斂，並提升梯度的更新效率。這些特性使其在深度學習中非常適合用於激活函數。

2.2.7 2D conv

在圖片處理中，卷積核（kernel）會沿著圖片的二維方向移動，這個過程稱為卷積操作。卷積核是一個小尺寸的矩陣，當它逐步滑動經過圖片的每個區域時，會計算該區域的加權和，從而提取局部特徵。這種操作能夠捕捉圖片中的邊緣、紋理等細節，並逐步構建出更高層次的特徵表示，進而用於圖像識別、分類等任務。

2.2.8 C2f

這種結構具有更少的參數並且具備更強的特徵提取能力。輸入首先通過一個卷積操作，該卷積的 kernel=1、stride=1、padding=0，並且輸出的通道數為 cout。接著，輸入會被分割（split）並經過 n 個 Bottleneck 結構，這些結構用於進一步提取特徵。隨後，將 Bottleneck 的殘差結果與主幹網絡（backbone）的輸出進行 Concatenate（拼接）。最後，拼接的結果再經過一次卷積操作輸出，這種設計能夠在減少參數的同時，保留有效的特徵表示，提升模型性能。

2.2.9 SPPF

這種技術旨在避免圖片在裁剪和縮放時出現失真問題，並且能有效解決卷積神經網路中重複特徵提取的問題。相較於 SPP（Spatial Pyramid Pooling），它具有更快的處理速度，能夠在保證特徵提取精度的同時提高運算效率，從而實現更高效的圖像處理和識別。

2.2.10 全局平均池化層

對每張特徵圖進行平均操作，將特徵圖所有元素的平均值作為輸出，舉例若是輸入特徵圖 H x W x C(高度 H，寬度 W，通道數 C)，GAP 將輸出一個 1 x 1 x C 的特徵圖，也就是每個通道輸出一個數值，數值是一個通道的所有元素的平均值，這樣有助於減少參數和防止過擬和。

2.2.11 殘差連接

這種技術稱為殘差網路（Residual Network, ResNet），是深層神經網路中的一種創新方法。它通過跳過層級連接將殘差或誤差直接傳遞到後續層級，從而避免梯度消失或梯度爆炸問題。這種跨層級的連接使得輸入的原始資訊可以直接傳遞到後面的層級，增加了網路的輸入與中間層的輸出。這樣的捷徑結構讓梯度在網路中更容易傳遞，使得網路能夠訓練得更深，並提升模型的效果和性能。

2.3 剪枝

突觸剪枝和神經元剪枝是深度學習模型壓縮中的兩種核心技術，廣泛應用於減少模型的大小和運算複雜度，同時保持或僅略微影響模型的效能。突觸剪枝的原理是透過識別神經網絡中不重要或冗餘的權重，將這些權重設為零，以減少不必要的連接。這不僅使模型結構更加簡化，還有效降低了存儲和計算負擔。雖然在剪枝後模型效能可能略有下降，但透過重新訓練，模型通常能恢復原本的效能，甚至有時會出現效能提升的情況。突觸剪枝的關鍵在於如何正確識別哪些權重對模型性能的影響最小，以確保保留下來的部分能夠維持模型的核心功能。

神經元剪枝則更進一步，直接移除整個神經元，而不僅是單一權重的剪枝。這種方法會計算每個神經元輸入和輸出權重的總重要性，並根據這些指標對神經元進行排序，剪除影響最小的神經元。在模型運行中帶來顯著的資源節約效果，尤其是在需要將大型模型部署到資源有限的硬件設備上時，神經元剪枝能顯著提升運行效率。同時，這種方法也有助於減少推理過程中的計算資源消耗，使得模型能夠在更小的硬件設備上運行，而不犧牲準確性或核心功能。

此外，factor（因素）通常是指影響剪枝過程中的一個關鍵參數，通常代表保留模型權重或通道的比例。它決定了剪枝後模型中會保留多少神經元、通道或參數，從而控制模型的壓縮程度。這個值通常介於 0 和 1 之間。

這兩種剪枝技術的綜合應用，特別適合於資源受限的應用場景，如嵌入式系統或移動設備上的大規模語言模型或物件檢測模型。通過剪除冗餘的權重和神經元，模型變得更加輕量化，運行效率顯著提升，這對於需要即時運算的應用尤

為重要。最終，剪枝技術不僅在模型壓縮中發揮了關鍵作用，還為資源節約型深度學習應用開闢了新的可能性。

2.4 靜態量化

量化技術是深度學習模型壓縮中常用的方法，透過將表示模型參數的浮點數轉換為整數或其他離散形式，從而達到降低儲存空間和模型大小的目的。具體來說，量化將模型中的浮點權重數值從較大的集合映射到較小的集合，在這個過程中，目標是最大限度地減少資訊量損失。雖然量化不可避免地會導致一定程度的精度損失，但其關鍵在於在保持模型效能的前提下，實現模型的高效壓縮。這種技術在現代人工智慧應用中廣泛使用，因其能顯著降低推理計算資源的需求，同時提升運行效率，尤其在資源受限的設備上，如嵌入式系統或移動設備中，尤為重要。

目前，傳統的深度學習模型在訓練階段通常使用32位浮點數（Float32）來表示每個權重，但隨著模型在推理階段對運算效率的需求不斷增高，業界的趨勢已經逐步向使用更低位數的參數轉移。主流的方法是採用8位整數（Int8）來替代浮點數，以此來減少模型的計算量和儲存需求，達到更高效的運行。然而，隨著技術的進一步發展，一些研究已經開始探索使用4位、3位、2位甚至1位的權重來進行模型量化，這些低位數的量化方法能夠進一步減少模型的大小和計算複雜度，尤其適合於需要即時處理的應用場景中。

靜態量化是目前廣泛應用的一種技術，它將訓練完成的浮點數模型參數轉換為低精度的整數，在這個過程中通常會使用校準數據集來確保量化的準確性。透過靜態量化，我們能夠顯著降低模型的儲存空間需求，並在一定程度上提升推理的速度，這對於大規模應用或需要快速反應的系統至關重要。雖然量化過程可能會引入一些精度損失，但隨著技術的進步，這種損失已經被有效控制在可以接受的範圍內。未來的研究可能會進一步探索更加極端的量化技術，如使用更低位數的量化方法來實現更高效的模型壓縮，進一步推動深度學習模型在各類設備上的廣泛應用。

2.5 混淆矩陣

混淆矩陣，又稱為誤差矩陣，是監督式學習中常用的工具，特別在機器學習的分類任務中尤為重要。它以一個二維的列聯表形式呈現，行和列分別代表實際標籤和模型的預測標籤。具體來說，每一列對應模型對各類別的預測結果，而每一行則代表數據集中的實際類別。這樣的排列方式使得我們能夠直觀地看到模型在區分不同類別時的表現，並揭示出模型是否會將某些類別混淆。例如，通過查看矩陣中的非對角元素，我們可以發現模型是否經常將某一特定類別誤分類為另一類別。這種分析方式可以幫助評估模型的準確性，識別分類錯誤的類型，並為後續改進模型提供具體方向。

2.6 mAP50

mAP50(B) 指的是 "Mean Average Precision at IoU 0.50 for Large Objects"，它的

意思是在 IoU（重疊率，即 Intersection over Union）設為 0.50 的條件下，針對大型目標物體計算出來的平均精度 (AP) 的平均值。IoU 是衡量預測框與真實框之間重疊程度的重要指標，而 0.50 表示當預測框與真實框的重疊度至少達到 50% 時，預測才會被視為正確的檢測結果。

mAP 是一個反映模型整體性能的指標，用來評估模型在所有類別中的檢測精度。具體來說，mAP 是對每個類別的平均精度 (AP) 值進行求和並取平均後得到的指標，因此它可以展示模型在多類別物件檢測中的表現。而 mAP50(B) 進一步細化了這個概念，它專門針對較大目標物體進行精度評估，在 IoU 0.50 的閾值下，mAP50(B) 只關注大物體的檢測結果，這樣可以更準確地評估模型在檢測大型物體時的性能。

這個指標特別適合應用於場景中存在大目標的任務，因為大物體的邊界較容易與預測框重疊，計算出的 mAP50(B) 也會反映出模型是否在處理大型物體時有更高的準確性。因此，mAP50(B) 既是對模型精度的總體評估，又是針對特定目標大小的細化指標，為評估大型物體檢測性能提供了更精確的衡量依據。

第三章 研究方法

3.1 資料預處理及分類

在資料預處理階段，首先對輸入圖像進行縮放和標準化，以適應網絡的輸入需求，同時保持目標物體的比例和特徵。接著，使用數據增強技術，如隨機裁剪、旋轉、翻轉等，擴充訓練資料的多樣性，以提高模型的泛化能力。在分類階段，YOLOv8 將圖像中的物體定位於邊界框內，並基於特徵圖進行分類，最終輸出每個物體的類別標籤及其對應的置信度分數，從而實現精確的物體檢測與分類。

3.2 影像標註

目的是讓模型學會識別圖像中的物體並準確進行分類。在影像標記過程中，每張圖像中的物體被手動標註為邊界框，並分配相應的類別標籤。這些邊界框標記物體的位置與大小，而類別標籤則指明物體的類型。這些標註資料會用來訓練 YOLOv8模型，使其能夠學習如何自動檢測並分類圖像中的物體，從而在實際應用中實現精準的目標識別與定位。

3.3 模型選擇與構建

模型選擇與構建階段是決定模型性能的關鍵之一，尤其是在瑕疵檢測這類高精度要求的任務中。首先，我們需要根據任務的具體需求選擇適當的 YOLOv8 模型架構，這涉及在速度、精度和資源消耗之間做出平衡。YOLOv8 提供多種模型變體，如 YOLOv8n（輕量快速）、YOLOv8m（平衡速度與精度）及 YOLOv8l（高精度），每種模型在層數與參數量上各有不同。為了適應不同硬體環境與實際應用場景，我們將選擇適當的模型作為瑕疵檢測的基礎。

接下來，我們深入了解模型的完整結構，包括其各層的組成和作用。這通常包括卷積層、Batch Normalization 層、激活函數層（如 Leaky ReLU）等。我們會詳細查看每一個模型層的參數設定，例如卷積的大小、步伐、填充方式，以及通道數的變化。這些層次與參數設定直接影響到模型的特徵提取能力和最終的檢測精度。此外，我們也會檢查模型的總層數以及總參數量，了解模型的計算複雜度，這有助於預測模型的推理速度和資源消耗。

在這個過程中，我們還會對一些關鍵參數進行微調，例如學習率、批量大小、優化器等，這些參數的設定將直接影響模型訓練的穩定性與收斂速度。最後，我們會充分理解各層和參數的作用，從而確保模型的設計與應用需求高度匹配，以達到最佳的瑕疵檢測效果並提升整體系統的運行效率。

3.4 模型訓練

這一階段直接決定了模型能否準確地完成瑕疵檢測任務。在開始訓練前，我們會對模型進行初始化設置，包括選擇適當的超參數，這些設定會影響模型的學習效率與最終性能。訓練過程中，模型會對資料集中的影像進行多輪次的學習，通過反向傳播調整每一層的權重和偏置，逐漸提升模型對瑕疵檢測的準確度。我們會使用驗證集來檢查模型在每個訓練周期後的表現，以確保模型能夠在新數據上保持良好的泛化能力，避免過度擬合。在訓練完成後，我們會保存最後一次訓練的模型權重，這個最終的權重檔案通常被稱為 last.pt，它將作為後續優化過程的原始模型。該模型不僅包含了從數據中學到的特徵，也為接下來的模型剪枝、量化等優化步驟提供了基礎。整個訓練過程不僅是讓模型學習如何檢測瑕疵，更是為模型後續優化奠定了穩固的基礎，確保最終模型能在保持高精度的同時，達到運行效率和資源消耗的最佳平衡。

3.5 模型驗證(val)

該模型具備自動設置功能，能夠記住其訓練配置，使用戶在後續驗證過程中無需重複設置，直接進行評估。此外，模型支持多指標評估，可以根據一系列準確度指標來全面評估性能。在資料相容性方面，該模型能夠無縫相容於訓練階段使用的數據集以及自定義數據集，確保其靈活性和適應性，無論是在開發還是測試階段，都能輕鬆應用。

3.6 模型優化

優化的目標不僅是讓模型更加輕量化，更使其在推理階段的速度更快、資源消耗更少。為了同時兼顧模型的精度和推理速度，我們決定採用剪枝、刪除模型層及量化三種技術。我們會對不論是剪枝前刪除模型層後或是刪除模型層前剪枝後的模型進行量化，包含將原始模型的浮點數32位元轉為浮點數16位元以及靜態量化（static quantization）。

剪枝和刪除模型層的目的是減少模型中的冗餘參數，通過刪除對性能貢獻較小的通道或神經元，使模型更加輕量化。這樣不僅可以降低運算複雜度，還能縮短推理時間。同時，剪枝能有效減少模型大小，這對於存儲和傳輸也有一定的優勢。

靜態量化技術將模型的浮點運算轉換為低精度的整數運算，如將權重和激活值從 32 位的浮點數轉換為 8 位的整數。這不僅逐步減少了模型的存儲需求，還顯著提高了運算速度，特別是在硬體支援低精度運算的情況下，量化能夠極大地加速推理過程。

選擇這種優化策略的原因在於通過結合剪枝、刪除模型層與量化，使模型的體積得以顯著縮小，運行效率得到大幅提升，而預測精度則依然能夠保持在可接受的範圍內。最終，我們的優化模型在推理階段的性能將顯著優於未優化的原始模型，實現了在資源受限的環境中快速且準確的瑕疵檢測。

3.7 模型測試與評估

模型測試與評估是檢驗優化後模型性能的關鍵步驟。通過對優化後的模型進行預測，我們將檢查模型的體積是否縮小，運行效率是否明顯提升，以及預測精度是否能夠保持在可接受的範圍內。我們會對比優化前後的模型體積大小、推理速度，並使用測試數據生成混淆矩陣，以評估模型在不同類別上的預測能力。同時，將優化模型的預測結果與未經優化的模型進行比較，確認其在提升效率的同時，仍然保持足夠的準確度和穩定性。

3.8 模型結果展現

利用Streamlit結合Python來建置一個簡易的網頁平台，能夠讓使用者輕鬆選擇要使用的模型進行預測。該網頁包含一個直觀的介面，使用者能夠從可用的多個預測模型中進行選擇，根據不同的需求或模型精度選擇最適合的預測模型。此外，網頁還提供一個可信度篩選功能，讓使用者設定預測結果的可信度門檻，確保輸出的結果符合需求。透過這個平台，使用者只需上傳圖片，即可在後端自動處理該影像，並顯示模型的預測結果及置信度標籤。該設計不僅提供了簡單的預測功能，還透過互動式的設定，讓使用者可以快速調整參數以獲取最適合的預測結果，是一個便捷且功能完善的圖像預測網頁應用。

第四章 實證分析

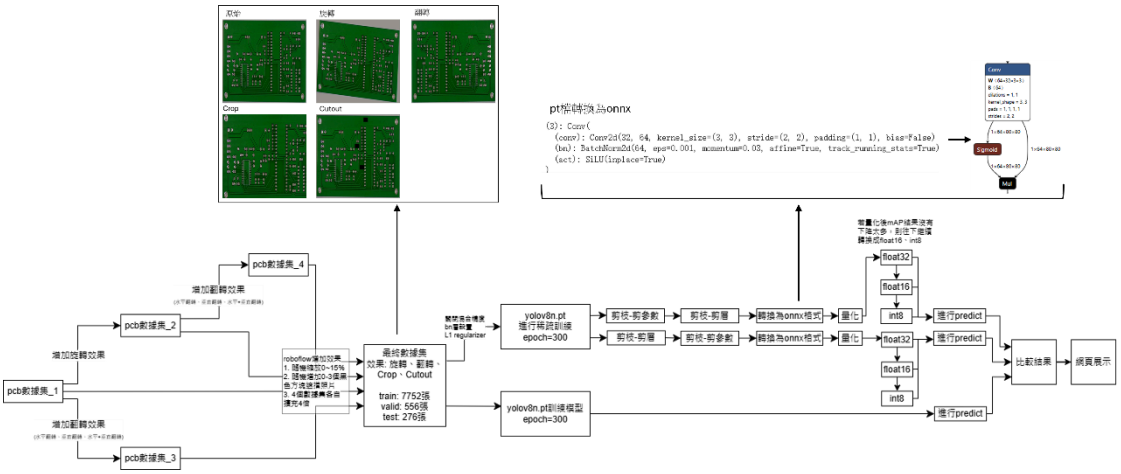

圖 4.1 專題完整架構圖

4.1 資料集來源

4.1.1 料集名稱

PKU-Market-PCB

4.1.2 資料集來源

北京大學智能機器人開放實驗室、PCB_DATASET - Google 雲端硬碟, 第二個為一開始所下載的來源, 後來發現第二個的資料集是引用第一個。

4.1.3 內容分成

Annotations、Images、PCB_USED 和 Rotation四個資料夾, 旋轉照片的python檔。

4.1.4 瑕疵分類

Missing_hole: 115張、Mouse_bite: 115張、Open_circuit: 116張、Short: 116張、Spur: 115張、Spurious_copper: 116張

4.2 影像標記

在本研究中，我們針對總數為693張的印刷電路板照片進行了逐張標記，對每張照片中的瑕疵進行精確註記。這些照片中的瑕疵數量各不相同，每張照片中的瑕疵數量從3個到7個不等。為了確保標記的準確性，我們根據瑕疵的類型、形狀、大小和位置進行了詳細的分類與標註，我們使用了標註工具來進行手動標記，為每個瑕疵框定邊界框，並為其分配相應的標籤。這些標記數據將作為模型訓練的基礎，確保模型能夠學習如何自動識別不同類型的瑕疵。

4.3 影像擴充

將原始數據集中的圖片進行旋轉處理，並在此基礎上進行以下4種擴充操作：原始數據集、旋轉後數據集、翻轉後的原始數據集：包括水平翻轉、垂直翻轉及水平 + 垂直翻轉和翻轉後的旋轉數據集：包括水平翻轉、垂直翻轉及水平 + 垂直翻轉。最終，將這4個數據集整合至網路平台Roboflow中進行擴充。運用的增強效果: crop 0%~15%、cutout 3boxes with 5% sizes each
擴充後的數據集總共有：訓練集 (train): 7752張、驗證集 (valid): 556張和測試集 (predict): 276張。下圖是經過擴充後訓練完的混淆矩陣結果。

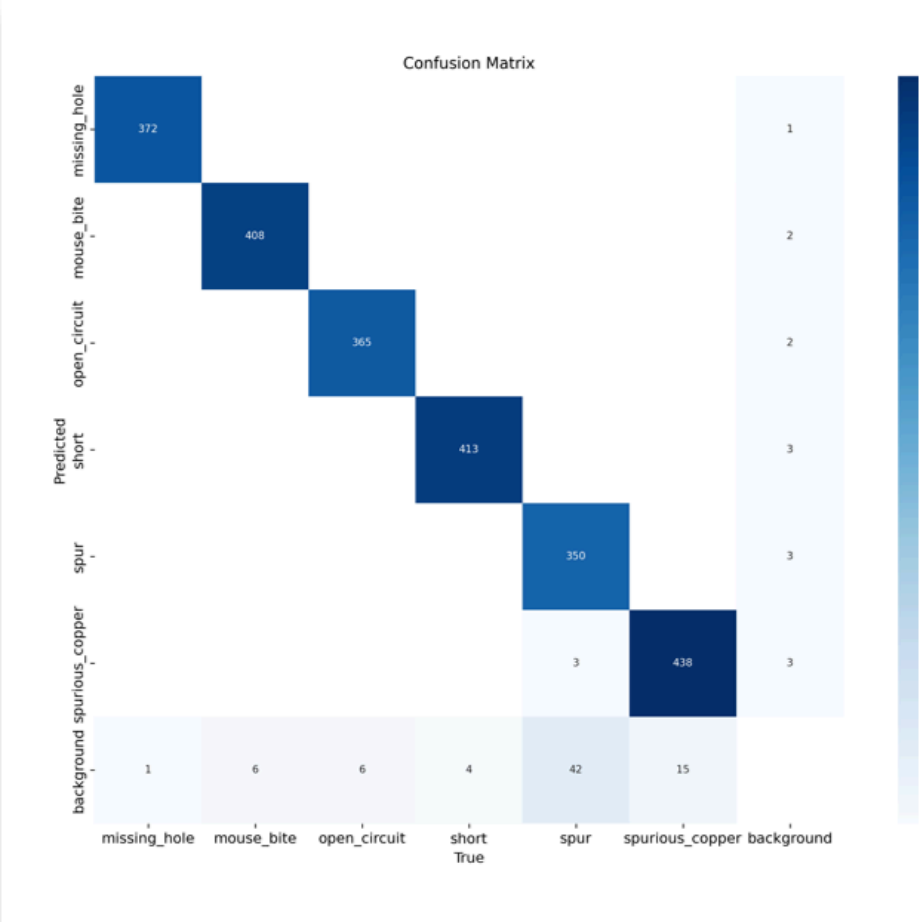

圖 4.2 經過擴充訓練後的混淆矩陣結果

4.4 模型訓練

先透過YOLOv8的YOLOv8n將資料集進行300epochs的訓練，訓練結果會有一個last.pt檔案，該檔案是在訓練的300次中，最後的第300次訓練出來的pt檔。

4.5 模型剪枝和刪除模型層

我們選擇使用模型訓練的第300次生成的模型檔案（last.pt）作為量化的基礎模型。首先，將自動混合精度訓練（AMP）設置為False，以確保模型在進行剪枝和量化過程中使用單一精度進行運算。接著，在批次標準化（Batch Normalization，BN）層中添加L1正則化，並進行稀疏訓練，這一過程有助於強化模型對不重要參數的抑制，從而使BN層中的大部分偏置值（bias）趨近於0，為後續的剪枝操作奠定基礎。

剪枝分為兩種方法，1.只剪參數而不修改模型結構 2.修剪模型結構，刪除後參數量會減少，我們分成1.先剪參數再剪層 2.先剪層再剪參數，查看調換先後順序是否能將模型壓縮更小。

在只剪參數部分，我們首先對要刪除的通道以及其對應的權重進行分析，並將這些數據存入陣列。剪枝的具體策略是根據L1正則化的結果來刪除不重要的

通道，保持率設在0.2~0.5之間。當保留的通道數少於8時，我們將根據權重的絕對值進行排序，優先保留權重較大的通道。例如，若保留的通道數量為5（len(keep_idxs)=5），我們會從準備刪除的通道中選擇權重最大的三個通道予以保留，從而避免對模型性能造成過度影響。

在修剪模型層部分，我們是先取得每層bn層權重後排序，若假設剪枝率為0.5，所有權種有1000個，那便會取weight[500]作為臨界值，之後遍歷模型，找出bn層並計算該層小於臨界值的權重有多少，若超過一半則會嘗試刪除此層。某一層輸入為例，原始輸入為128×3=384個通道，當其中一個輸入被刪除後，下一層的輸入必須相應調整為256個通道。為此，我們新增了一個中間層，將輸入通道數修改為256，並將剪枝後的前256個權重及偏置值存入這個新建立的層中，最後將該層與其他模型層連接，進行反向傳播以完成整個剪枝過程。以下為部分模型刪除前(圖4.2)和部分模型刪除後(圖4.3)

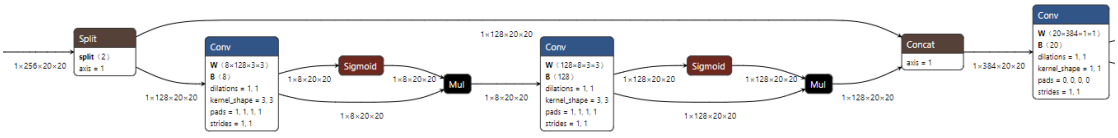

圖 4.3 刪除模型層前

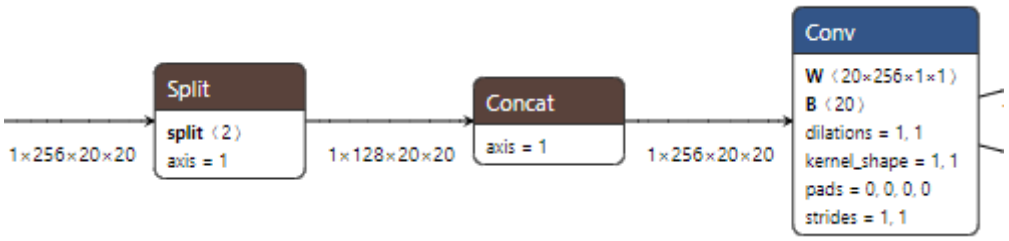

圖 4.4 刪除模型層後

通過這一剪枝方法，我們成功地減少了模型的冗餘參數，提升了運算效率，並為後續的模型量化與部署打下了堅實的基礎。這樣的優化能確保模型在精度和運行速度之間達到平衡，特別適合應用於資源受限的環境中。

4.6 模型量化

選擇使用模型訓練的第300次完成後生成的權重檔案(last.pt)作為進行量化的基礎模型。首先，我們將該last.pt檔案轉換為ONNX格式的檔案(last.onnx)，這樣可以利用ONNX框架進行進一步的優化和處理。

將原本的浮點數從32位元轉16位元的使用方法為先將模型的每個參數做有效位數的調整，設置允許設置的最大最小範圍，將超過最大上限的參數值設為最大上限的值，低於最低下限的參數值設為最低下限的值，以防止參數值經過量化後變成0或無限大。將參數值截斷後，讓參數值的浮點數四捨五入到浮點數16位元能夠表示的最接近數值。之後進行測試模型可以將原本的浮點數從32位元轉16位元後，我們檢查浮點數16位元的模型評分(val)的MAP50和預測結果的推理(influence)。接下來，我們對last.onnx檔案進行靜態量化。靜態量化過程需要準確的校準，以確保量化後模型的表現能夠接近原始模型。因此，我們決定使用我們在預測時所用的PCB資料集作為校準數據，這樣可以提供與實際應用場景更為相符的統計信息。

在進行靜態量化之前，我們需要排除那些無法進行量化的模型層。這些層包括Sigmoid、Mul、Add、Split、Softmax和Div等，因為這些層的量化會導致在進行

預測時出現多重框的瑕疵結果，影響模型的檢測準確性和可靠性。

完成排除後，我們利用ONNX Runtime的量化工具進行最終的量化步驟。首先準備一組校準數據，用於統計模型各層的激活範圍，然後透過 ONNX Runtime 的 quantize_static 工具來進行靜態量化。此過程中會將模型的權重和激活值量化至 int8 格式，並依據校準數據自動調整參數範圍。最後，量化後的模型將儲存至指定位置，並經過推理測試以確保精度和性能保持在可接受的範圍內。這樣可以顯著降低模型大小和計算量，同時保留原有的檢測準確性。

4.7 剪枝後的模型進行量化

將模型訓練的第300次完成後生成的權重檔案(last.pt)，分別有進行剪枝完畢後，刪除部分模型層，再將pt檔轉換為onnx檔，對模型進行浮點數從32位元轉16位元確保模型有縮減存儲空間和運行時間，再拿onnx檔做靜態量化，產生出一個有先剪枝後刪除層的靜態量化模型。以及進行刪除層完畢後，進行剪枝，再將pt檔轉換為onnx檔，對模型進行浮點數從32位元轉16位元確保模型有縮減存儲空間和運行時間，再拿onnx檔做靜態量化，產生出一個有先刪除模型層後剪枝的靜態量化模型。最後和原始的last.onnx檔比對，得到最佳的模型，使模型不僅具備更小的存儲空間和更高的運行效率，同時也能保持較高的檢測準確性。

4.8 結果

1.先剪參數再剪層: 剪枝率=0.5時進行驗證，mAP無明顯下降，之後查看小於臨界值的參數的比率是否大於50%，若是的話則刪除該層並重新驗證檢查mAP是否下降，在剪除部分層厚最小模型層數為162，參數量914,444

2.先剪參數再剪層: 先將第一個方法中有剪去的層全部剪除後再剪參數，經過測試後，剪枝率=0.42時，可達到最好結果，模型層數162，參數量802740，比第一個方法多刪除111,704個參數

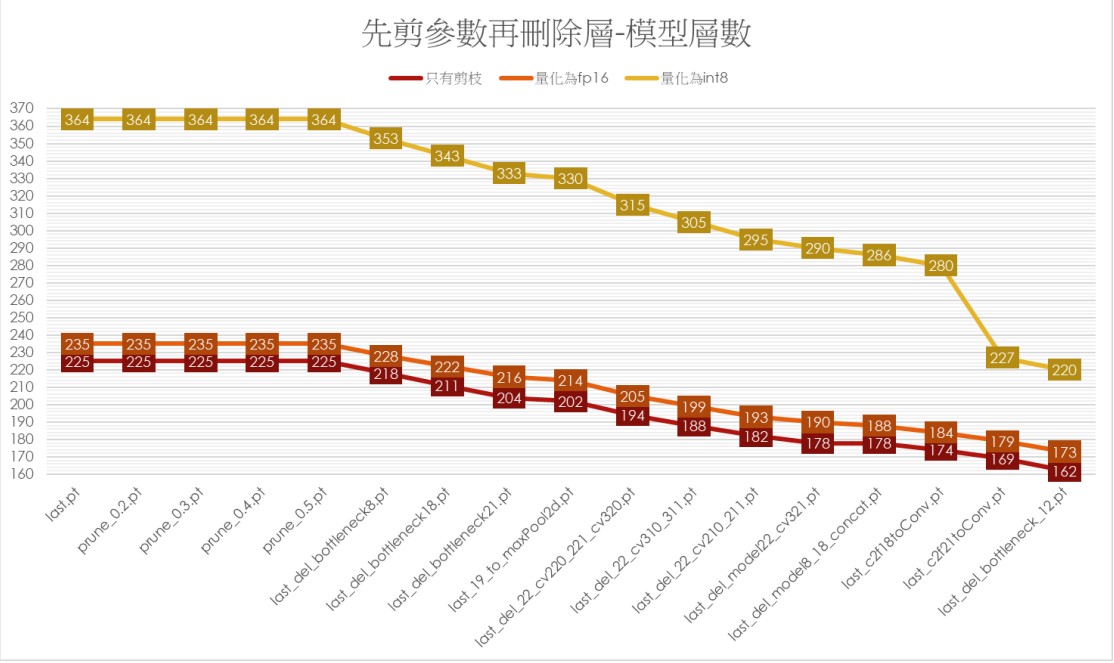

圖 4.5 先剪參數再刪除層-模型層數比較

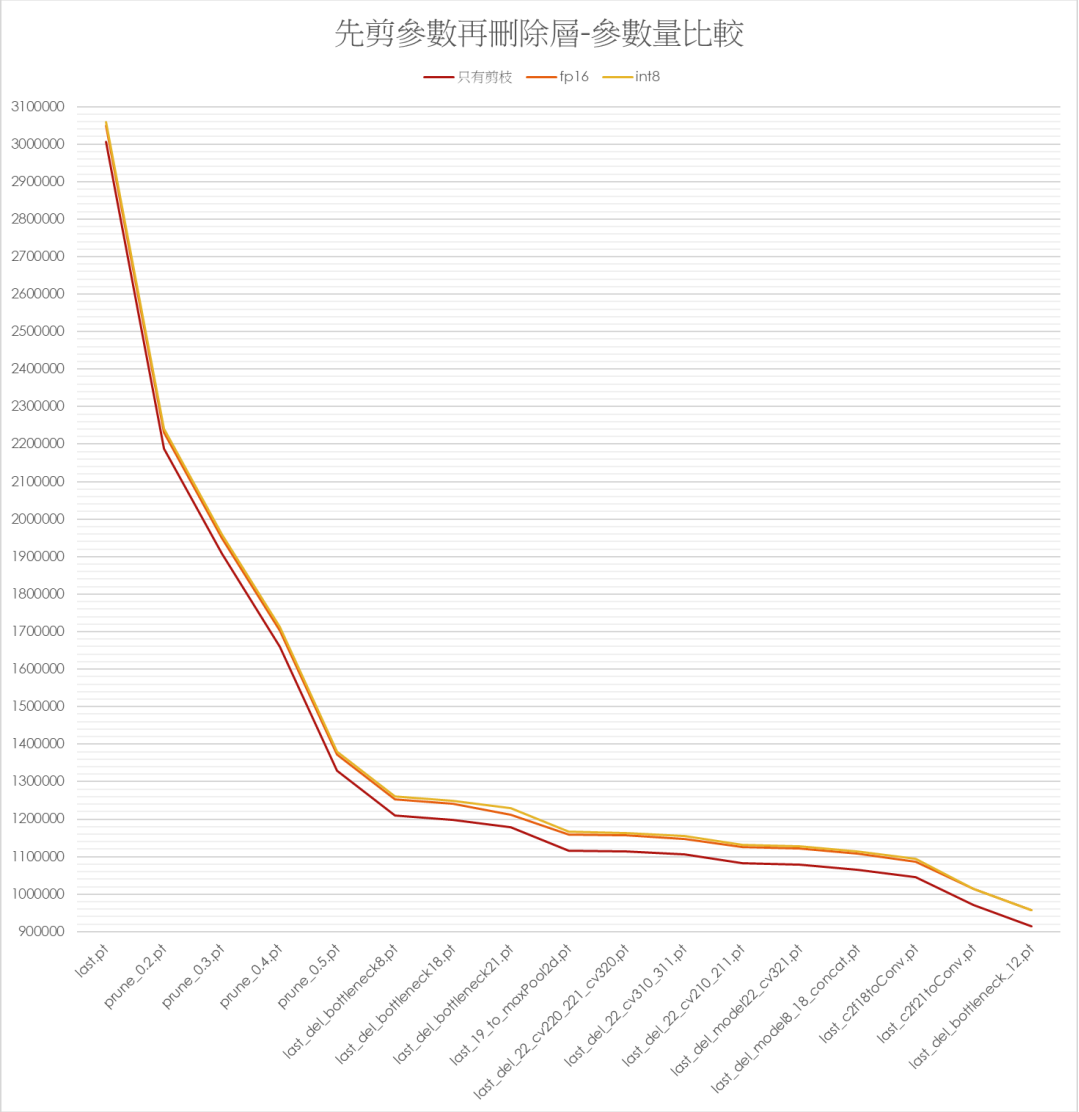

圖 4.6 先剪參數再刪除層-模型參數量比較

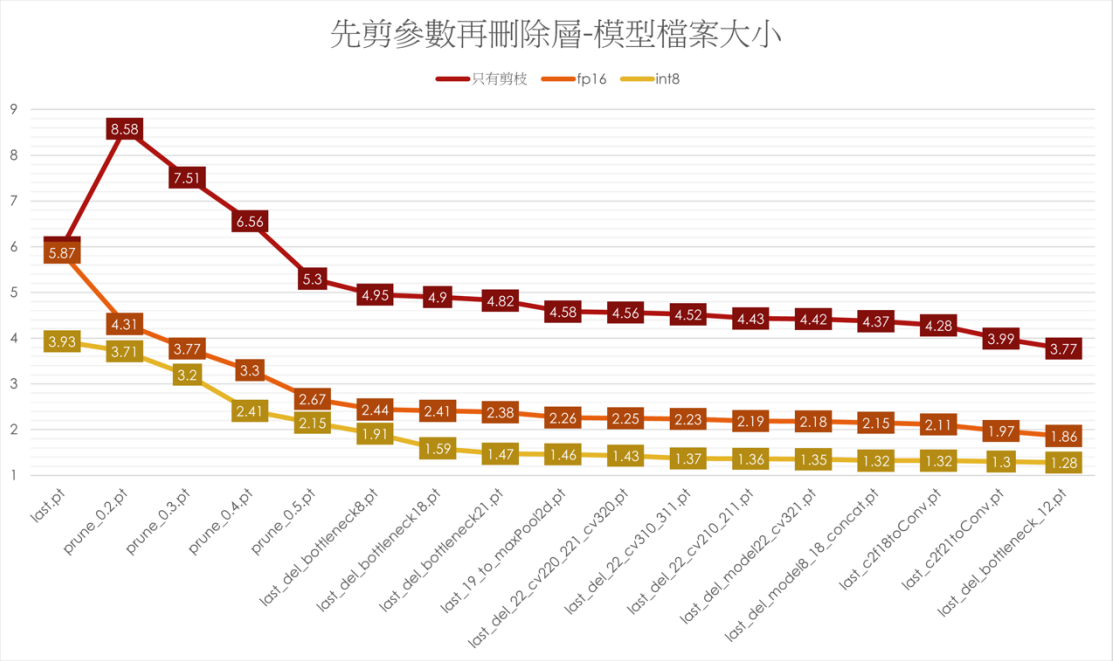

圖 4.7 先剪參數再刪除層-模型檔案大小比較

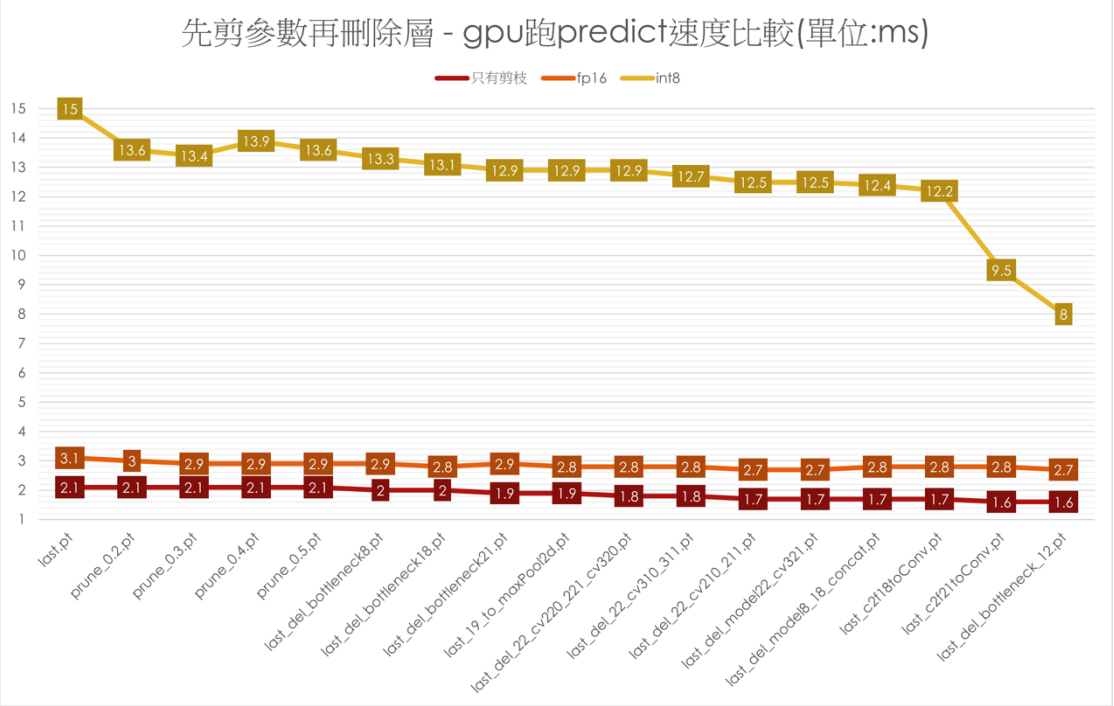

圖 4.8 先剪參數再刪除層-模型速度比較_GPU

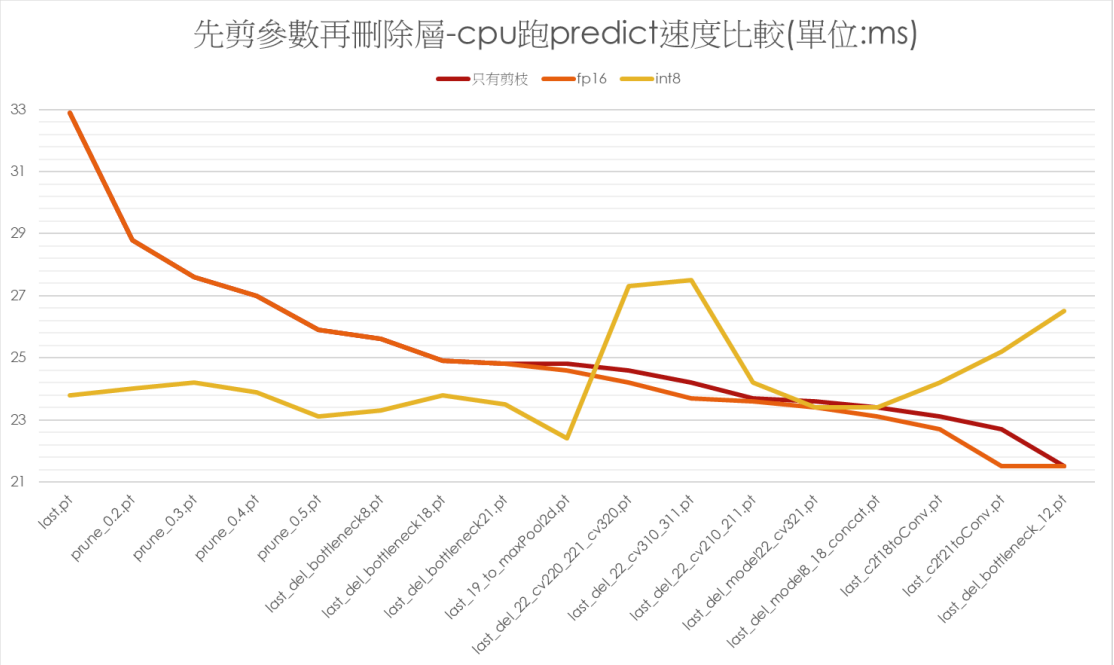

圖 4.9 先剪參數再刪除層-模型速度比較_CPU

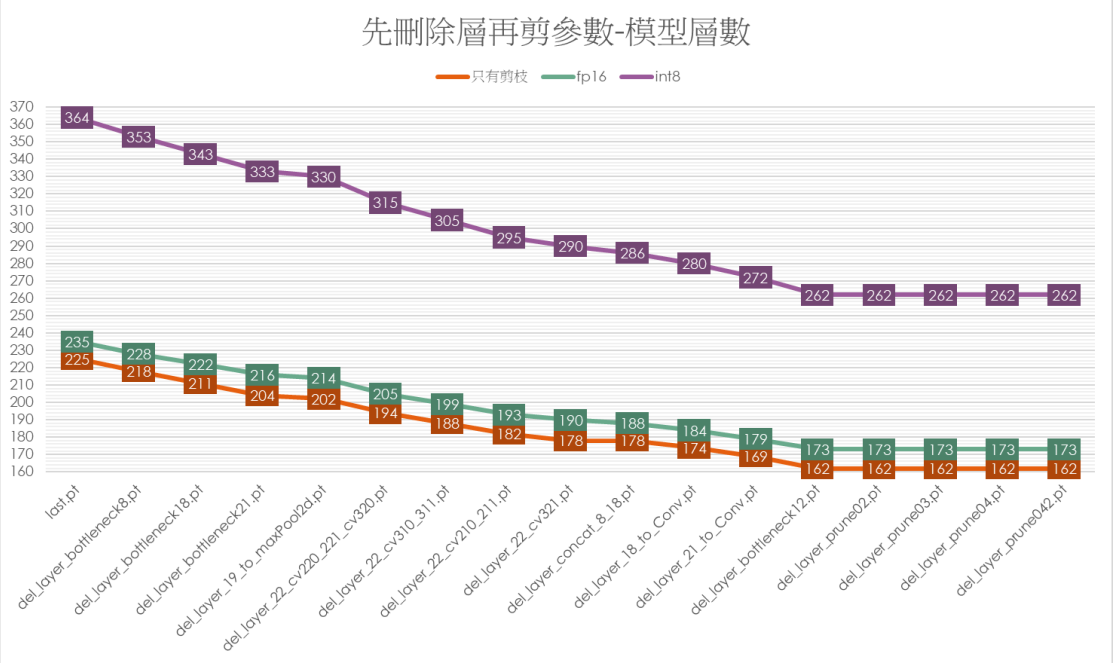

圖 4.10 先刪除層再剪參數-模型層數比較

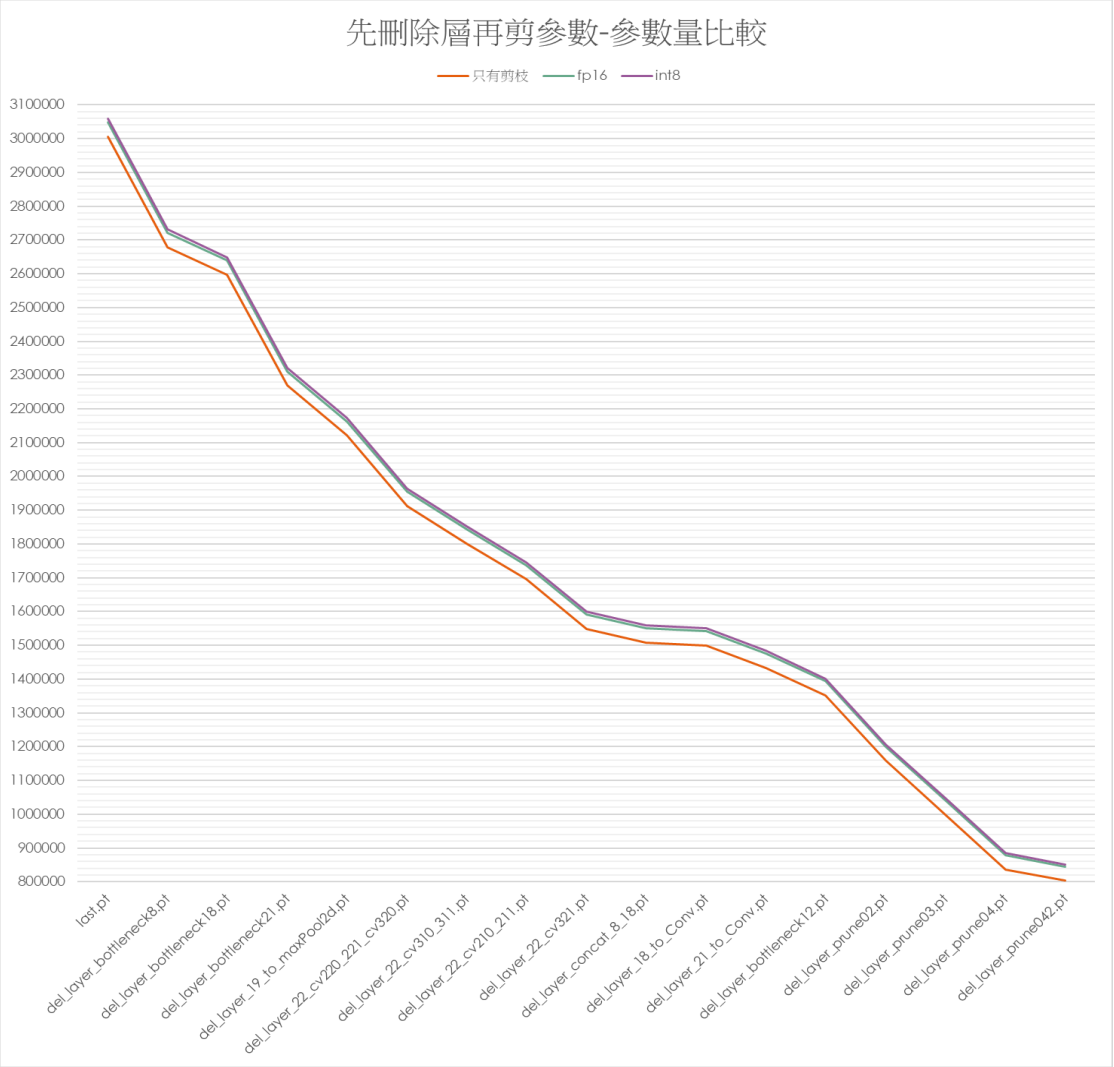

圖 4.11 先刪除層再剪參數-模型參數量比較

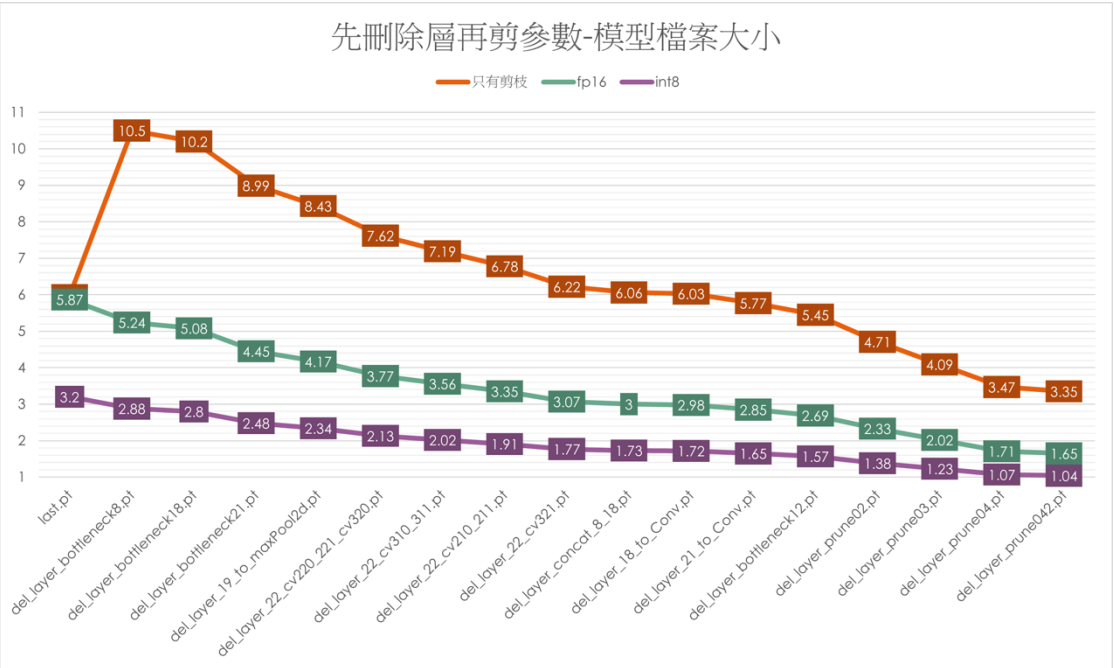

圖 4.12 先刪除層再剪參數-模型檔案大小比較

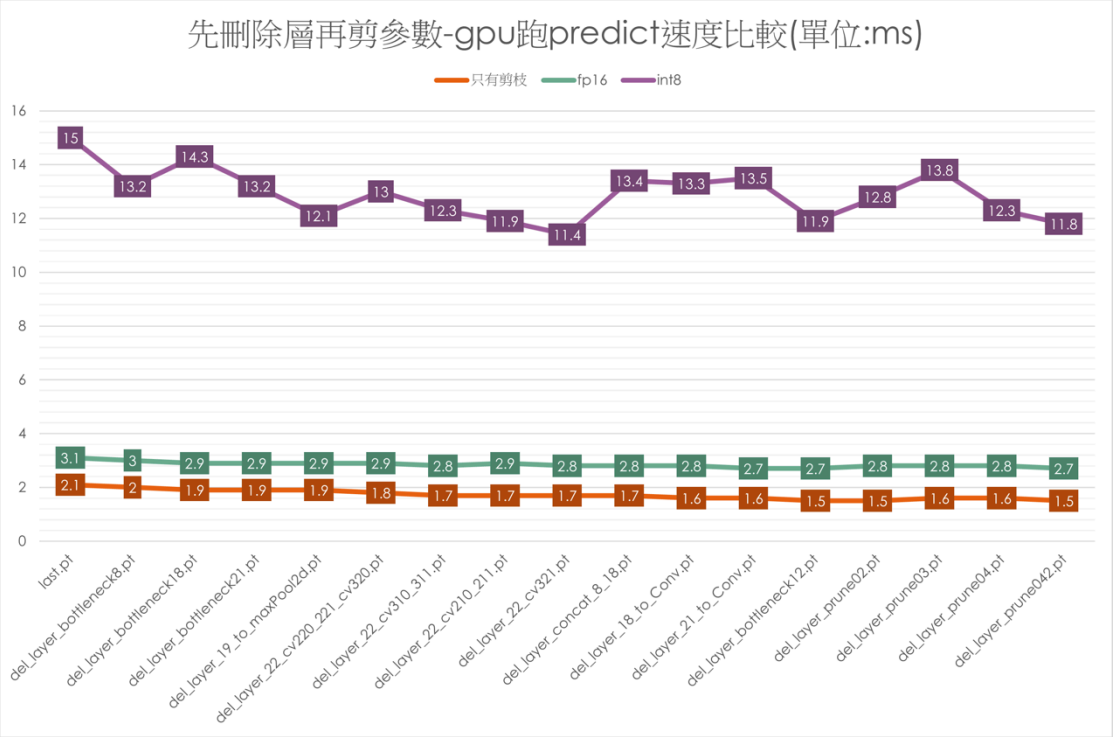

圖 4.13 先刪除層再剪參數-模型速度比較_GPU

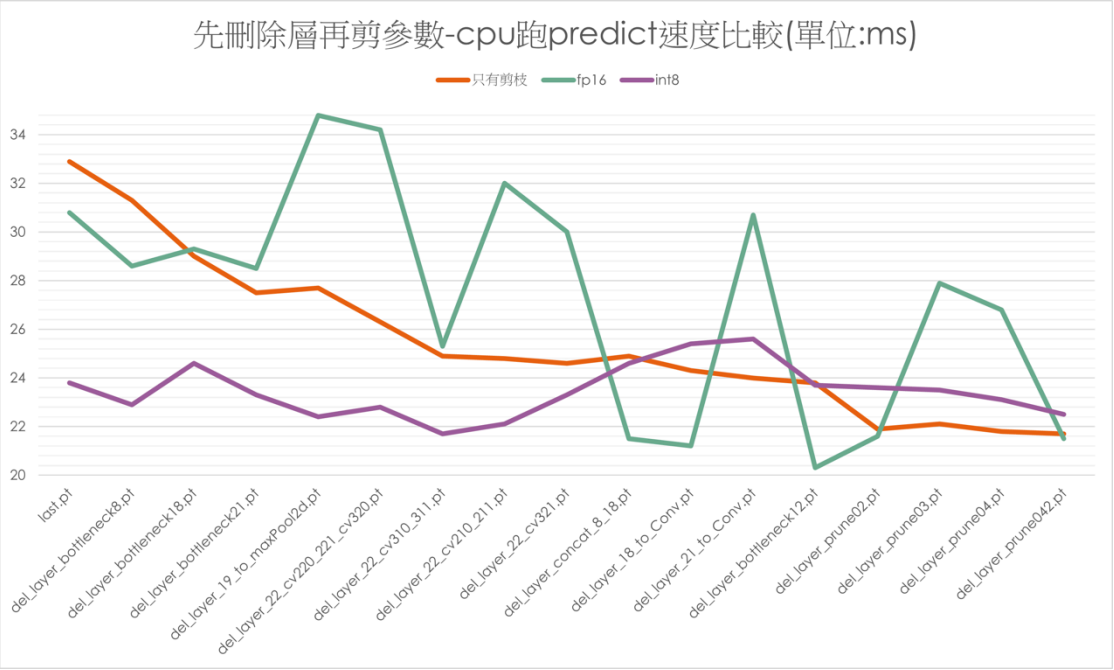

圖 4.14 先刪除層再剪參數-模型速度比較_CPU

	刪層先參數後	參數先刪層後	fp16	int8
del_layer_bottleneck8	0.968845173		0.968110257	0.962001031
del_layer_bottleneck18	0.969349166		0.967365455	0.962170916
del_layer_bottleneck21	0.969349166		0.967365455	0.962170916
del_layer_19_to_maxPool2d	0.969349166		0.967365455	0.961974576
del_layer_22_cv220_221_cv320	0.969349166		0.967365455	0.961974576
del_layer_22_cv310_311	0.963417961		0.961446269	0.95689686
del_layer_22_cv210_211	0.963417961		0.961446269	0.95689686
del_layer_22_cv321	0.963417961		0.961446269	0.95689686
del_layer_concat_8_18	0.959366574		0.958894381	0.94968244
del_layer_18_to_Conv	0.96092971		0.960850074	0.953206669
del_layer_21_to_Conv	0.960469692		0.962113612	0.952822349
del_layer_bottleneck12	0.955641141		0.95571747	0.949502756
del_layer_prune02	0.955240476		0.95689211	0.948680582
del_layer_prune03	0.959531141		0.959705051	0.953396208
del_layer_prune04	0.960595592		0.959544887	0.95206272
del_layer_prune042	0.956321011		0.956501239	0.949817665
prune_0.2		0.972252538	0.967340895	0.964409897
prune_0.3		0.97224889	0.967646433	0.964153246
prune_0.4		0.972252552	0.967633238	0.962294708
prune_0.5		0.972249245	0.967649403	0.964507656
ast_del_bottleneck8		0.968843251	0.96784456	0.96266258
last_del_bottleneck18		0.969349065	0.967362382	0.964121873
last_del_bottleneck21		0.969349065	0.967362382	0.964121873
last_19_to_maxPool2d		0.969349065	0.967362382	0.963793325
last_del_22_cv220_221_cv320		0.969349065	0.967362382	0.963793325
last_del_22_cv310_311		0.963422721	0.961594369	0.959039031
last_del_22_cv210_211		0.963422721	0.961594369	0.959039031
last_del_model22_cv321		0.963422721	0.961594369	0.959039031
last_del_model8_18_concat		0.960772337	0.960212574	0.953613755
last_c2f18toConv		0.960500879	0.960805588	0.951989093
last_c2f21toConv		0.956683651	0.95939845	0.958286722
last_del_bottleneck_12		0.954501773	0.958906803	0.960607404

圖 4.15 各個模型的f1 score

以下是預測出來的最終模型混淆矩陣:

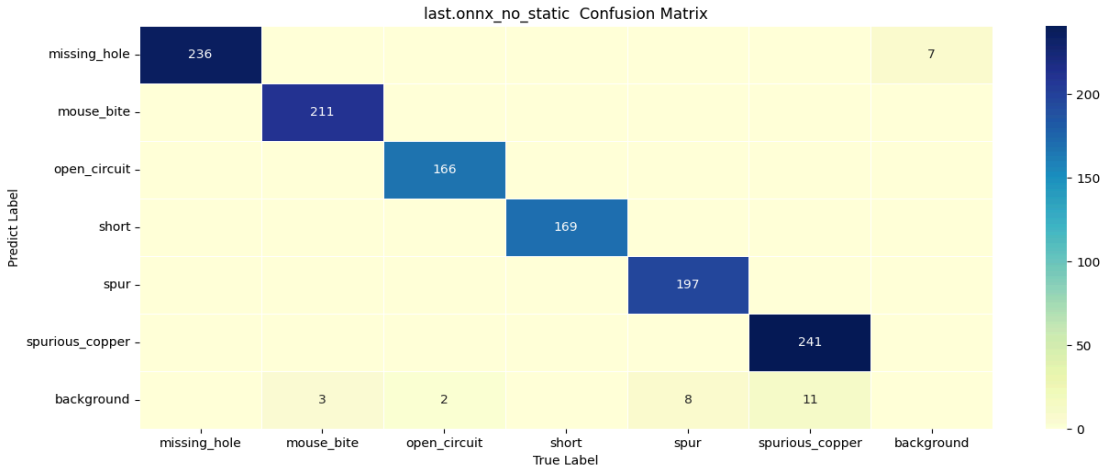

圖 4.16 last.pt原始模型訓練結果混淆矩陣

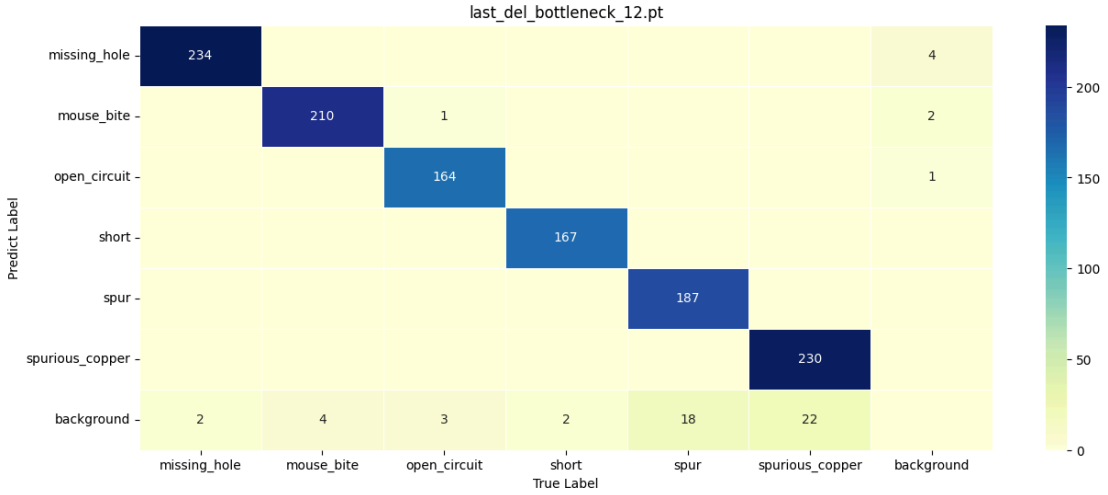

圖 4.17 先剪參數再剪層的最小模型預測結果混淆矩陣

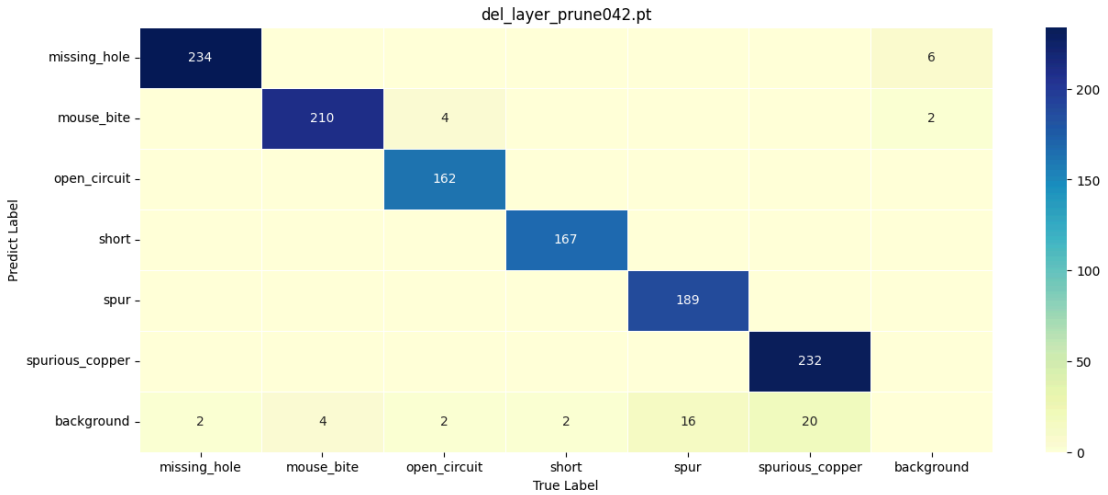

圖 4.18 先剪層再剪參數的最小模型預測結果混淆矩陣

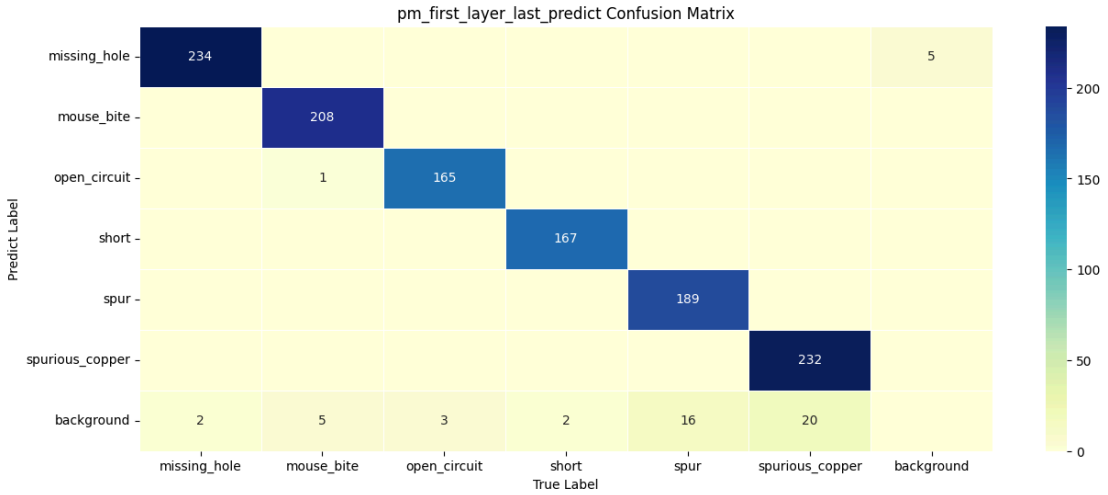

圖 4.19 先剪參數再剪層的最小模型量化為fp16預測結果混淆矩陣

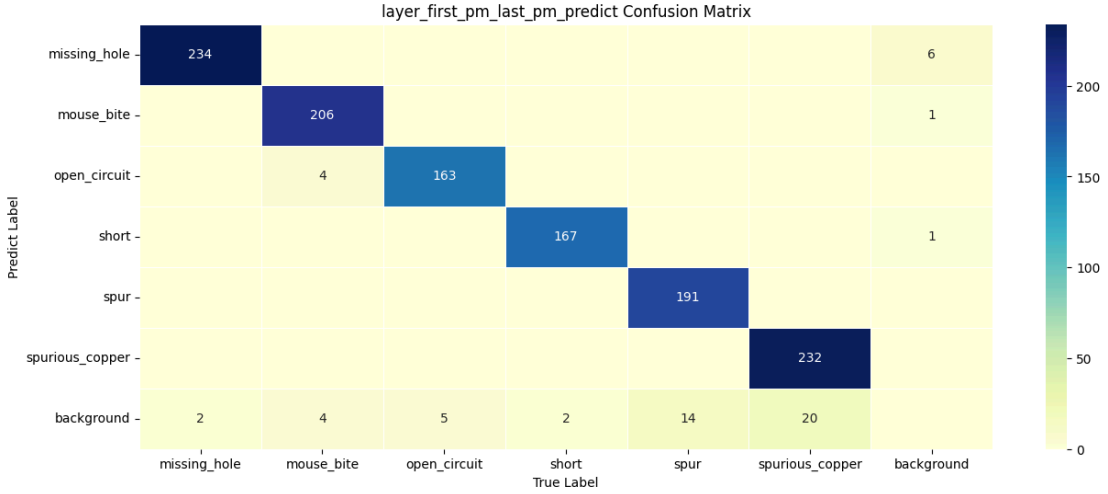

圖 4.20 先剪層再剪參數的最小模型量化為fp16預測結果混淆矩陣

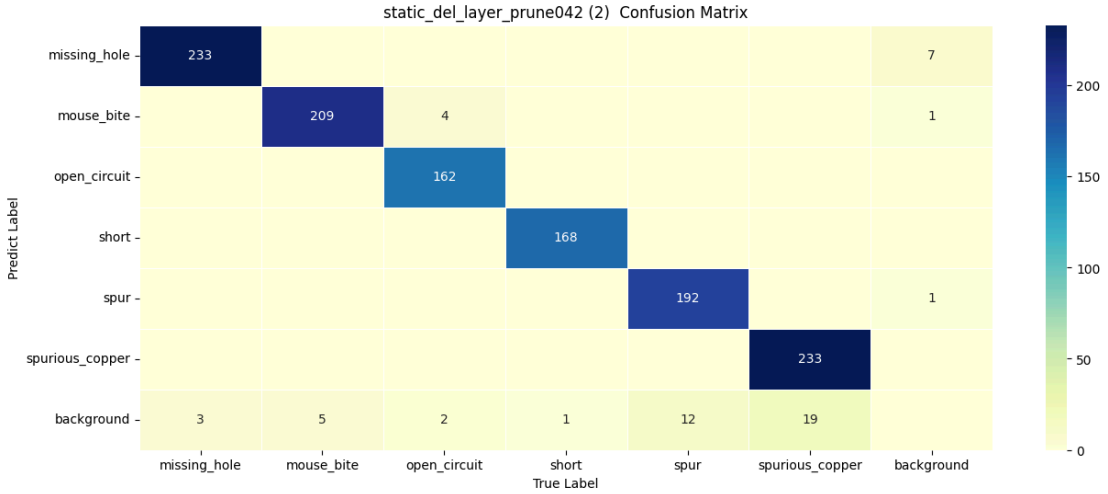

圖 4.21 先剪層再剪參數的最小模型量化為int8預測結果混淆矩陣

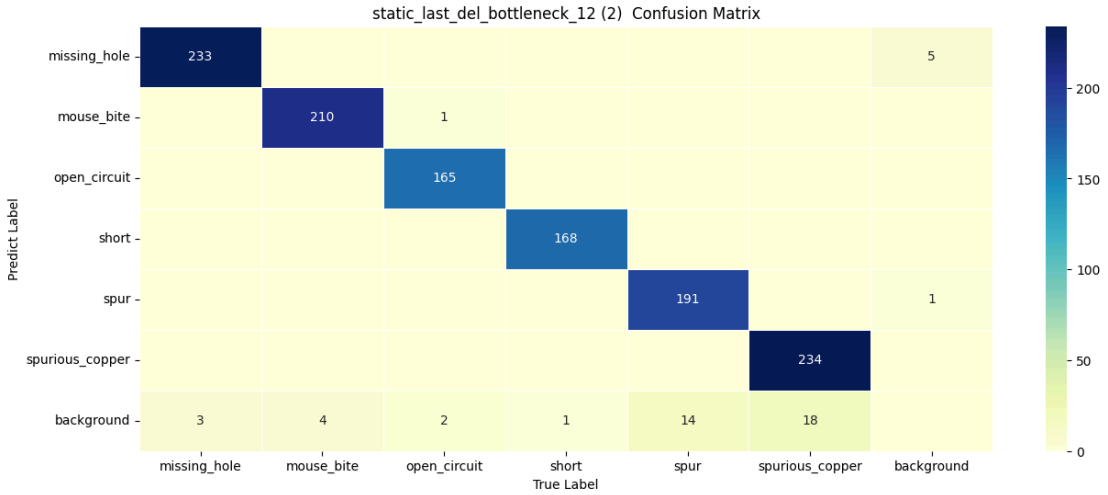

圖 4.22 先剪參數再剪層的最小模型量化為int8預測結果混淆矩陣

透過對模型進行剪枝、刪除多餘層次、並採用靜態量化等技術，我們不僅成功減少了模型的複雜度，同時也大幅減少了其儲存空間。這些優化策略在保持模型準確度的基礎上，顯著減少了運算資源的消耗，從而在模型推理時間和系統響應速度上表現更好。剪枝通過移除不必要的權重和神經元，降低了模型的冗餘度，使得運算過程更加精簡。而刪除多餘的模型層則進一步縮短了前向傳播的路徑，從結構上減少了計算量。

同時，量化技術針對模型節點進行將浮點運算轉換為定點運算，極大地減輕了硬體運算負擔，特別適合於在邊緣設備或資源受限的環境下部署模型。這樣的處理不僅減少了模型的內存占用和存儲空間需求，還在推理階段加快了處理速度，實現了低延遲的高效運行。此外，這些優化技術在不影響模型準確度的情況下，有效控制了過度擬合和過於複雜的模型結構，進而提升了模型的泛化能力。

透過這樣的設計和調整，我們的模型不僅在性能表現上達到了理想的效果，還具備了更強的可擴展性和可部署性，能夠應用於更廣泛的場景中，滿足各類應用的需求。這樣的結果證明了我們在模型優化過程中的策略是成功且有效的，未來我們還可以進一步針對特定的應用場景進行更深層次的優化，以達到更高的效能和精度。

4.9 模型成果展現

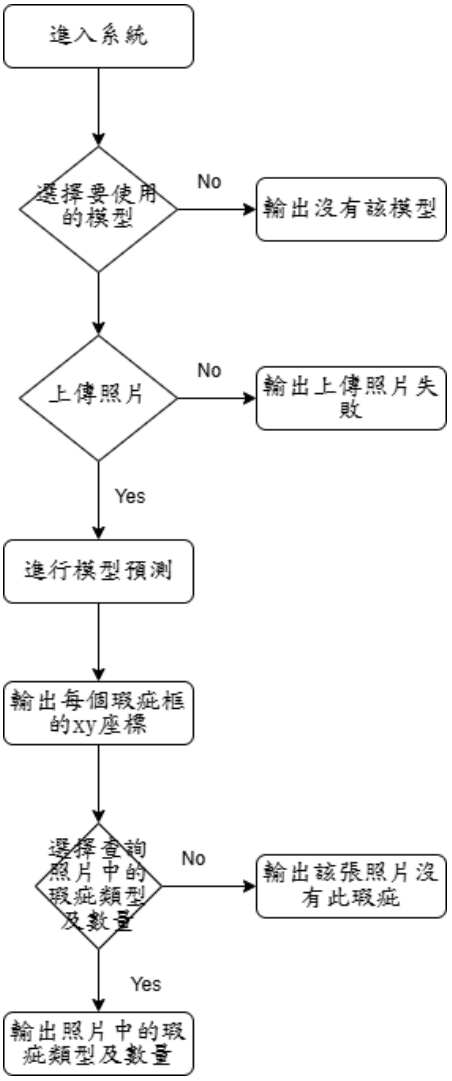

圖 4.23 網頁呈現流程圖

第五章 結論與建議

本專題著重於 AOI 瑕疵檢查系統的建置與測試，通過多次優化和技術應用提升模型的準確度與運行效率。

首先，為了提高訓練準確率，對數據集標註進行了優化和擴充，並在訓練過程中引入稀疏化技術，以促進模型的穩定性和泛化能力，增強了對瑕疵的識別準確性。接著，採用了剪枝和靜態量化技術來進行模型壓縮，去除冗余層以縮小模型體積和減少推理時間。然而，靜態量化導致模型層數和參數量的增加，顯示出 ONNX 格式在量化過程中可能引入非預期的架構變化。針對此情況，未來將優化模型結構以符合輕量化需求。

模型經過剪枝和量化後，進行了 ONNXRuntime 平台上的推理測試，並將其部署於 Streamlit 平台開發用戶端網頁，允許用戶上傳圖片進行檢測，這大大提升了系統的實用性。過程中遇到模型層數增加和參數異常等問題，經過多次測試和層結構比對，掌握了剪枝與量化對模型結構的影響。為改善 CPU 和 GPU 環境下的推理速度，將持續優化剪枝層數和量化參數範疇。

針對未來的優化，老師建議在剪枝時優先刪除結構層，再對參數進行裁剪，以提升優化效果。此外，應進一步研究浮點數精度轉換（如 float32 到 float16）對推理速度的影響，並在簡報呈現中加強圖表和結果解釋的邏輯性。未來將在 YOLO 模型基礎上進行更多剪枝和量化測試，以持續提升系統的準確度、效率及應用範圍，並積極解決技術瓶頸，為不同場景的瑕疵檢測應用提供更高效的解決方案。

第六章 參考文獻

[1] Yaseen, Muhammad. (2024). What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector. 10.48550/arXiv.2408.15857.

[2] onnxruntime,"Model Training with Ultralytics YOLO". URL:https://docs.ultralytics.com/modes/train/

[3] onnxruntime,"Quantize ONNX Models". URL:https://pse.is/6kn7ys

[4] 泓宇," [Object detection] YOLOv8詳解". URL: https://pse.is/6kn6e2

[5] 西笑生," 目標檢測 YOLOv5 - 如何提高模型的指標，提高精確率，召回率，mAP 等". URL:https://blog.csdn.net/flyfish1986/article/details/120704968

[6] 穿着帆布鞋也能走貓步," 超詳解| Yolov8 模型手把手調參 | 配置 | 模型訓練| 驗證 | 推理". URL:https://blog.csdn.net/xu1129005165/article/details/132720801

[7] Hzp666," 設置 YOLO-V8 的參數". URL:https://blog.csdn.net/hzp666/article/details/133337953

[8] Pan_peter," windows 使用 YOLOv8 訓練自己的模型(0 基礎保姆級教學)". URL:https://blog.csdn.net/Pan_peter/article/details/129907710

[9] Zhijun.li@Studio," YOLOv8 教程系列:一、使用自定義數據集訓練 YOLOv8 模型(詳細版教程，你只看一篇->調參攻略)，包含環境搭建/數據準備/模型訓練/預測/驗證/導出等". URL:https://blog.csdn.net/weixin_45921929/article/details/128673338

[10] 芒果汁没有芒果," 手把手調參YOLOv8 模型之 訓練｜驗證｜推理配置-詳解". URL:https://pse.is/6kn5ut

[11] githubcurry," **如何優化 yolov8 模型，壓縮模型大小，部署到邊緣設備上**". URL: https://blog.csdn.net/weixin_45921929/article/details/128673338

[12] **海_纳百川**, "yolov8**的模型剪枝教程**". URL:https://www.cnblogs.com/chentiao/p/18342742

[13] Tommy Huang, " AI**模型壓縮技術-量化**(Quantization)". URL: https://pse.is/6kn75l

[14] **「已注銷」**, "**【重磅盤點】62種PCB板不良實例的原因分析及規避措施！必收藏干貨！**". URL:https://pse.is/6kn7pchttps://pse.is/6kn6r5

[15] **小北的北**, " YOLOv8 **模型量化:動態量化&靜態量化**". URL:https://blog.csdn.net/weixin_38739735/article/details/140727654

[16] AIF Editor, "**模型部署前哨站！模型壓縮的原理與方法**" URL: https://edge.aif.tw/aicafe-1129-model-compression/

[17] Jeremy Pai, "mean Average Precision (mAP) — **評估物體偵測模型好壞的指標**" URL: https://pse.is/6l2wlt

[18] Streamlit, "Streamlit documentation" URL: https://pse.is/6m25td

[19] Mr.Luyao, "使用Streamlit開發YOLOv8的可視化交互界面"
URL: https://zhuanlan.zhihu.com/p/630108152