ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง

นายวุฒิภัทร ณัดฐาปกรณ

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์ข้อมูลเพื่อนวัตกรรม
บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2568

ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง

นายวุฒิภัทร ณัดฐาปกรณ

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์ข้อมูลเพื่อนวัตกรรม
บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2568

ใบรับรองการค้นคว้าอิสระ

บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เรื่อง ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง

โดย นายวุฒิภัทร ณัดฐาปกรณ

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์ข้อมูลเพื่อนวัตกรรม

คณบดีบัณฑิตวิทยาลัย / หัวหน้าภาควิชา

คณะกรรมการสอบการค้นคว้าอิสระ

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร.สมชาย ปราการเจริญ)

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.นลินภัสร์ บำเพ็ญเพียร)

..... กรรมการ

(อาจารย์ ดร.กาญจนา วิริยะพันธ์)

ชื่อ : นายวุฒิภัทร ณัดฐาปกรณ
ชื่อการค้นคว้าอิสระ : ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง
สาขาวิชา : วิทยาศาสตร์ข้อมูลเพื่อนวัตกรรม
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
อาจารย์ที่ปรึกษาการค้นคว้าอิสระหลัก : รองศาสตราจารย์ ดร.นลินภัสร์ บำเพ็ญเพียร
ปีการศึกษา : 2568

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนา ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติ โดยใช้เทคนิคการเรียนรู้ของเครื่องผ่านโมเดล YOLOv8 ซึ่งสามารถตรวจจับตำหนิของขวดได้อย่างแม่นยำและทำงานแบบ Real-time ระบบถูกออกแบบให้ตรวจจับขวดที่มีตำหนิ เช่น ไม่มีฝา ไม่มีฉลาก และ ไม่มีทั้งฝาและฉลาก เป็นต้น ส่งสัญญาณไปยัง Arduino เพื่อควบคุมกลไก Rejector ในการคัดแยกขวด ชิ้นงานเสีย ออกจากสายพานโดยอัตโนมัติ ข้อมูลที่ใช้ในการฝึกประกอบด้วยภาพรวม 25,00 ภาพ ที่ผ่านการทำการเพิ่มข้อมูล เพื่อเพิ่มความหลากหลาย โมเดลถูกฝึกใน Google Colab โดยใช้ GPU Tesla T4 จำนวน 100 Epoch และมีความแม่นยำเฉลี่ยสูงถึงร้อยละ 96.4 จากการประเมินด้วย Confusion Matrix การทดสอบในระบบจริงแสดงให้เห็นว่าสามารถทำงานได้ด้วยความเร็ว 30–38 FPS ที่ Latency เฉลี่ย 40 มิลลิวินาทีต่อเฟรม ซึ่งรองรับการทำงานของสายพานได้อย่างมีเสถียรภาพ ผลการประเมินความพึงพอใจของผู้ใช้งานจริง 18 คน พบว่าผู้ใช้งานมีความพึงพอใจในระดับสูงมาก (ค่าเฉลี่ย 4.59 จาก 5.00 คะแนน) โดยเฉพาะด้านความคุ้มค่าและประสิทธิภาพการทำงาน ระบบที่พัฒนาขึ้นจึงถือว่ามีศักยภาพสูงในการนำไปประยุกต์ใช้จริงในโรงงานอุตสาหกรรม ด้วยต้นทุนรวมเพียง 46,450 บาท ซึ่งต่ำกว่าระบบ Vision Sensor เชิงพาณิชย์กว่า 7–10 เท่า และสามารถสนับสนุนแนวทางการพัฒนาอุตสาหกรรมไทยสู่ยุค Industry 4.0 ได้อย่างมีประสิทธิภาพ

(มีจำนวนทั้งสิ้น 100 หน้า)

คำสำคัญ : การเรียนรู้ของเครื่อง YOLOv8 การตรวจจับตำหนิ เครื่องลำเลียงแบบอัตโนมัติ อุตสาหกรรม 4.0

_____ อาจารย์ที่ปรึกษาการค้นคว้าอิสระหลัก

Name : Mr. Wuthiphat Natthapakornna
Independent Study Title : An Automation Conveyor Defective Workpiece Detection System Using Machine Learning
Major Field : Data Science for Innovation
King Mongkut's University of Technology North Bangkok
Independent Study Advisor : Nalinpat Bhumpenpein
Academic Year : 2025

ABSTRACT

This research aims to develop an automatic defective workpiece detection system on a Conveyor Belt using Machine Learning, integrating the YOLOv8 model for high-accuracy, real-time defect detections of bottle. The system identifies bottle defects such as missing caps, missing labels, then sends control signals via Arduino to a pneumatic rejector for automatic defected bottle removal. A dataset of 25,00 images was used as trained dataset, followed by data augmentation to increase diversity. The model was trained in Google Colab using a Tesla T4 GPU for 100 epochs. The trained model achieved 96.4% average accuracy. Real-time testing showed processing speeds of 30–38 FPS with average latency at 40 ms per frame, proving stable operation suitable for conveyor systems. 18 User satisfaction evaluation indicated a high satisfaction level (mean value at 4.59 from 5.00), particularly in cost-effectiveness and operational performance. The developed system is therefore considered to have high potential for practical application in industrial factories, with a total cost of only 46,450 baht, which is 7–10 times lower than commercial Vision Sensor systems. This research provides a practical solution for intelligent quality inspection and contributes to advancing Industry 4.0 manufacturing in Thailand.

(Total 100 Pages)

Keywords: Machine Learning, YOLOv8, Defect Detection, Conveyor Automation, Industry 4.0

_____ Advisor

กิตติกรรมประกาศ

งานวิจัยเรื่อง “ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง” ฉบับนี้ สำเร็จลุล่วงได้ด้วยความกรุณาและการสนับสนุนจากหลายฝ่าย ซึ่งผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงต่อ รองศาสตราจารย์ ดร.นลินภัสร์ บำเพ็ญเพียร อาจารย์ที่ปรึกษาการค้นคว้าอิสระ คณะเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ได้กรุณาให้คำแนะนำ คำปรึกษา และข้อเสนอแนะอันทรงคุณค่าในทุกขั้นตอนของการดำเนินงาน ตลอดจนเป็นแบบอย่างของความเป็นนักวิจัยที่มีความรู้ ความสามารถ และความมุ่งมั่น ซึ่งเป็นแรงบันดาลใจสำคัญที่ทำให้งานวิจัยฉบับนี้สำเร็จลุล่วงอย่างสมบูรณ์

ผู้วิจัยขอขอบพระคุณ คณาจารย์ทุกท่านในคณะเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่ได้มอบความรู้ แนวทางทางวิชาการ และคำแนะนำอันเป็นประโยชน์ตลอดระยะเวลาการศึกษา ตลอดจนให้การสนับสนุนอุปกรณ์ เครื่องมือ และสถานที่ในการดำเนินงานวิจัยอย่างครบถ้วน

ขอขอบคุณ เพื่อนนักศึกษาร่วมรุ่น สาขาเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล ที่ได้ร่วมกันแลกเปลี่ยนความรู้และความคิดเห็น ให้ความช่วยเหลือทั้งในด้านเทคนิคและการดำเนินงานวิจัย รวมถึงให้กำลังใจในทุกช่วงเวลาของการศึกษา ความร่วมมือและมิตรภาพจากเพื่อนร่วมรุ่นทุกคนถือเป็นแรงผลักดันสำคัญที่ทำให้ผู้วิจัยสามารถฝ่าฟันอุปสรรคและพัฒนาผลงานนี้จนสำเร็จได้

นอกจากนี้ ผู้วิจัยขอขอบคุณ บุคลากรจากภาคอุตสาหกรรม ที่ให้คำปรึกษา แนะนำการออกแบบระบบ และเปิดโอกาสให้ทดสอบระบบจริงในสภาวะแวดล้อมการผลิต ซึ่งช่วยให้การพัฒนาระบบตรวจจับชิ้นงานเสียมีความสมบูรณ์และสอดคล้องกับการใช้งานจริงในอุตสาหกรรมมากยิ่งขึ้น

สุดท้ายนี้ ผู้วิจัยขอกราบขอบพระคุณ บิดา มารดา และครอบครัว ที่เป็นแรงสนับสนุนทางใจที่สำคัญที่สุด คอยให้กำลังใจ ความเข้าใจ และแรงผลักดันในทุกช่วงของการศึกษาและการทำงานวิจัย หากปราศจากแรงสนับสนุนจากครอบครัว งานวิจัยฉบับนี้คงไม่สามารถสำเร็จลุล่วงได้

ผู้วิจัยหวังเป็นอย่างยิ่งว่า งานวิจัยฉบับนี้จะเป็นประโยชน์ต่อวงการการศึกษาและภาคอุตสาหกรรม โดยเฉพาะในการนำเทคโนโลยีปัญญาประดิษฐ์และการเรียนรู้ของเครื่องมาประยุกต์ใช้เพื่อยกระดับระบบควบคุมคุณภาพในสายการผลิต ให้สอดคล้องกับแนวทางของอุตสาหกรรม 4.0 (Industry 4.0) ซึ่งจะช่วยส่งเสริมศักยภาพของอุตสาหกรรมไทยให้ก้าวสู่ความยั่งยืนในอนาคต

วุฒิภัทร ณัดฐาปกรณ

สารบัญ

หน้า

สารบัญตาราง

.

สารบัญรูปภาพ

ภาพที่ **หน้า**

.

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การก้าวเข้าสู่ยุคการปฏิวัติอุตสาหกรรมครั้งที่สี่ (Industry 4.0) ได้ยกระดับมาตรฐานการผลิตสมัยใหม่ให้เป็น การผลิตแบบอัจฉริยะ (Smart Manufacturing) ที่ยืดหยุ่น มีประสิทธิภาพ และเชื่อมโยงข้อมูลแบบครบวงจรตลอดสายการผลิต [1, 2] หนึ่งในเสาหลักที่สำคัญคือ การควบคุมคุณภาพ (Quality Control) ซึ่งต้องดำเนินไปอย่างรวดเร็วและแม่นยำ เพื่อให้อัตราข้อบกพร่องใกล้ศูนย์มากที่สุด โดยเฉพาะในโรงงานที่ใช้ ระบบสายพานลำเลียง (Conveyor Belt) การตรวจสอบคุณภาพเป็นทั้งขั้นตอนสำคัญและเปราะบาง เพราะชิ้นงานเคลื่อนที่ต่อเนื่องและรวดเร็ว หากยังพึ่งพาการตรวจด้วยสายตามนุษย์ ย่อมเผชิญข้อจำกัดด้านความเหนื่อยล้า ความแปรปรวนในการตัดสินใจ ส่งผลให้เกิดความผิดพลาดและความไม่สม่ำเสมอของคุณภาพ [3]

ควบคู่กับแรงกดดันของ Industry 4.0 นวัตกรรมด้าน ปัญญาประดิษฐ์ (AI) โดยเฉพาะการเรียนรู้เชิงลึก (Deep Learning) ได้เติบโตอย่างก้าวกระโดดและกลายเป็นคำตอบเชิงเทคโนโลยีที่มีศักยภาพสำหรับงานตรวจสอบคุณภาพ [4] ระบบวิชั่นอุตสาหกรรม (Industrial Vision System) ที่ผสาน Deep Learning สามารถเรียนรู้คุณลักษณะของตำหนิจากข้อมูลจริง โดยไม่ต้องออกแบบตัวบ่งชี้ภาพล่วงหน้าเหมือนแนวทางเดิม [4] จุดเปลี่ยนสำคัญคือ การตรวจจับวัตถุ (Object Detection) ซึ่งระบุตำแหน่ง (Bounding Box) และจำแนกชนิดตำหนิได้พร้อมกัน อัลกอริทึมตระกูล YOLO (You Only Look Once) ถูกออกแบบมาเพื่อรองรับการประมวลผลแบบเรียลไทม์ ให้เฟรมต่อวินาที (FPS) สูง เหมาะอย่างยิ่งกับสายพานที่ชิ้นงานเคลื่อนที่ไม่หยุด [5, 6, 7, 8, 9] อีกทั้งการเลือกใช้โมเดลที่กะทัดรัดแต่มีประสิทธิภาพ เช่น YOLOv8 ยังเอื้อต่อการติดตั้งบน อุปกรณ์เอดจ์/ฝังตัว (Edge/Embedded Devices) ใกล้จุดกำเนิดข้อมูล ช่วยลดความหน่วง (Latency) และเพิ่มเสถียรภาพของระบบโดยรวม [10, 11]

บนภูมิหลังดังกล่าว งานวิจัยนี้มุ่ง ออกแบบและพัฒนาระบบต้นแบบสำหรับตรวจจับชิ้นงานเสียบนสายพานลำเลียงโดยใช้การเรียนรู้ของเครื่อง โดยบูรณาการ Deep Learning ให้สอดคล้องกับข้อจำกัดและบริบทหน้างานจริง ขั้นตอนสำคัญประกอบด้วย การรวบรวมและจัดเตรียมภาพขวดพลาสติกที่มีตำหนิจริง การฝึกโมเดล YOLOv8 เพื่อให้ตรวจจับตำหนิเฉพาะอย่างได้อย่างแม่นยำ และการติดตั้งใช้งานบนสายพานขนาดกะทัดรัด เพื่อสั่งงาน กลไกคัดทิ้ง (Rejector) อย่างอัตโนมัติแบบ

เรียลไทม์ เป้าหมายปลายทางคือการลดภาระของแรงงานมนุษย์ เพิ่มความสม่ำเสมอของคุณภาพ และยกระดับประสิทธิภาพของกระบวนการผลิต

คุณูปการหลัก ของงานประกอบด้วยสองมิติสำคัญ มิติแรกคือ องค์ความรู้และกระบวนงานเชิงระบบ ที่ครอบคลุมตั้งแต่การได้มาซึ่งข้อมูล การฝึกและประเมินโมเดล ไปจนถึงการปรับใช้บนอุปกรณ์เอดจ์ในโรงงานจริง ซึ่งสามารถใช้เป็นกรอบอ้างอิงและต้นแบบสำหรับงานตรวจจับวัตถุแบบเรียลไทม์ในภาคอุตสาหกรรม มิติที่สองคือ ผลเชิงประยุกต์ ในรูปของระบบต้นแบบที่ทำงานได้จริง ช่วยลดความผันผวนจากการตรวจด้วยมนุษย์ ลดข้อบกพร่อง และเป็นฐานให้โรงงานสามารถขยายผลต่อยอดเพื่อยกระดับมาตรฐานการควบคุมคุณภาพให้ทัดเทียมสากลภายใต้บริบทของ Industry 4.0

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อพัฒนาโมเดลการเรียนรู้ของเครื่อง (Machine Learning Model) สำหรับการจำแนกชิ้นงานดีและชิ้นงานเสียประเภทต่างๆ

1.2.2 เพื่อพัฒนาระบบตรวจจับชิ้นงาน (ขวดพลาสติก) บนสายพานลำเลียงโดยใช้กล้องเว็บแคมร่วมกับการประมวลผลภาพ

1.3 ขอบเขตของการวิจัย

การวิจัยครั้งนี้กำหนดขอบเขตอย่างชัดเจนเพื่อให้การพัฒนาและการประเมินผล “วัดได้–ทวนซ้ำได้” ในบริบทเชิงวิศวกรรมของสายพานอุตสาหกรรม โดยมุ่งพัฒนาระบบตรวจจับชิ้นงานเสียแบบเรียลไทม์ที่เชื่อมครบตั้งแต่การได้มาซึ่งข้อมูลภาพ การอนุมานด้วยแบบจำลอง ไปจนถึงการสั่งงานกลไกคัดทิ้งอย่างแม่นยำภายใต้ข้อจำกัดด้านเวลาแฝงและสภาวะหน้างานจริง [12]

ในด้านชิ้นงานและข้อมูล งานมุ่งที่**ข**วดพลาสติกบนสายพานลำเลียง และจำกัดชนิดความเสียหายหลักที่ต้องตรวจจับให้อยู่ในกรอบที่ควบคุมได้ ได้แก่ ขวดไม่มีฝาปิด (Missing Cap) ขวดไม่มีฉลาก (Missing Label) และ ตำหนิของตัวภาชนะ เพื่อให้แบบจำลองเรียนรู้ลักษณะเฉพาะได้ลึกและตรงเป้า ชุดข้อมูลรวบรวมจากสภาพจริงภายใต้ความหลากหลายของแสงและความเร็วสายพาน พร้อมทำ Annotation [13] ตามมาตรฐานสำหรับอัลกอริทึมตรวจจับวัตถุ และขยายข้อมูล (Data Augmentation) [14, 15] เพื่อเพิ่มความครอบคลุม ก่อให้เกิดฐานข้อมูลราว 25,000 ภาพ ซึ่งถูกแบ่งเป็น Train/Validation/Test [16] ด้วยสัดส่วนคงที่ เพื่อใช้ประเมินผลข้อมูล

ในด้านเทคนิค Deep Learning งานกำหนดปัญหาเป็น Object Detection โดยเลือกใช้ตระกูล YOLOv8 เพื่อให้สมดุลระหว่างความแม่นยำและเวลาอนุมาน ขอบเขตการประเมินแบบจำลองใช้ตัวชี้วัดมาตรฐาน ได้แก่ mAP (เช่น mAP@0.5 และ/หรือ mAP@0.5:0.95) [17, 18]

ควบคู่กับ Precision–Recall–F1 Score เพื่อสะท้อนสมดุลของการตัดสินใจในบริบทจริง การตั้งค่าที่อยู่ในขอบเขตรวมถึง ความละเอียดอินพุต ค่าเกณฑ์ความเชื่อมั่น (Confidence Threshold) และ ค่า IoU

ในด้านฮาร์ดแวร์และสภาพแวดล้อม ระบบต้นแบบถูกทดสอบบน สายพานลำเลียงจำลอง ที่กำหนด ความเร็วสูงสุดไม่เกิน 3 เมตรต่อวินาที [19, 20] คอมพิวเตอร์ประสิทธิภาพสูงที่มี GPU [21, 22, 23] เพื่อรองรับอัตราเฟรมตามเป้าหมาย การสั่งคัดทิ้งใช้ โซลินอยด์แอคชูเอเตอร์ (Rejector) ควบคุมผ่าน Arduino ที่สื่อสารกับคอมพิวเตอร์ด้วย USB3 [24, 25]

1.4 ประโยชน์ที่ได้รับ

1.4.1 เพิ่มประสิทธิภาพและความรวดเร็วในกระบวนการผลิต

ระบบสามารถทำงานตรวจสอบได้อย่างต่อเนื่องตลอด 24 ชั่วโมง โดยไม่ล้าและไม่สะดุด รองรับอัตราการไหลของชิ้นงานที่สูงกว่าการตรวจด้วยสายตามนุษย์อย่างมีนัยสำคัญ ส่งผลให้ กำลังการผลิตรวมเพิ่มขึ้น และลดคอขวดในสถานีตรวจสอบคุณภาพ [26, 27]

1.4.2 เพิ่มความแม่นยำและสร้างมาตรฐานในการควบคุมคุณภาพ

การประยุกต์ใช้ Machine Learning ช่วยลดความผิดพลาดจากความเหนื่อยล้าและความแปรปรวนของการตัดสินใจของพนักงาน ทำให้การคัดแยกชิ้นงานเสียมีความแม่นยำและความสม่ำเสมอสูง สามารถกำหนดมาตรฐานเดียวกันให้กับทุกชิ้นส่วนและทุกกะการผลิต [28, 29, 30]

1.4.3 ลดต้นทุนการผลิตในระยะยาว

เมื่อความผิดพลาดลดลง จะลด ต้นทุนของเสีย (Scrap/Rework) ลดความเสี่ยงการเรียกคืนสินค้า (Product Recall) และลดภาระค่าแรงในงานตรวจสอบด้วยสายตา (Manual Inspection) ในภาพรวมต้นทุนต่อหน่วยลดลง และ คืนทุนการลงทุนยาว [18], [27], [29, 30]

1.4.4 ได้รับข้อมูลเชิงลึกเพื่อพัฒนาสายการผลิต

ระบบสามารถบันทึกสถิติลักษณะและจำนวนชิ้นงานเสีย พร้อมเวลาและเงื่อนไขหน้างานที่เกี่ยวข้อง ข้อมูลดังกล่าวช่วยให้สามารถทำ วิเคราะห์หาสาเหตุ (RCA) ระบุจุดอ่อนของกระบวนการและออกแบบมาตรการลดของเสียอย่างเฉพาะเจาะจง นำไปสู่ การปรับปรุงกระบวนการอย่างต่อเนื่อง (Continuous Improvement) [28], [31, 32]

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การวิจัยนี้มุ่งศึกษาเกี่ยวกับระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง โดยใช้สถาปัตยกรรม YOLOv8 ประมวลผลภาพจาก Webcam แบบเรียลไทม์ และเชื่อมผลการตัดสินใจผ่าน Arduino ไปยังกระบวนการคัดทิ้ง (Rejector) อัตโนมัติ พร้อมประเมินสมรรถนะด้วยตัวชี้วัดมาตรฐาน (Precision, Recall, F1-score และ mAP) เพื่อยืนยันความพร้อม ในบทนี้ครอบคลุมสาระดังต่อไปนี้

2.1 แนวคิดและทฤษฎีพื้นฐานด้านการประมวลผลภาพ (Image Processing)

2.2 การเรียนรู้ของเครื่องและโครงข่ายประสาทเทียม (Machine Learning & ANN)

2.3 ประเภทของการเรียนรู้ของเครื่อง (Supervised / Unsupervised / Reinforcement)

2.4 โครงข่ายประสาทเทียม (Artificial Neural Networks, ANN)

2.5 โครงข่ายประสาทเทียมเชิงลึกและคอนโวลูชัน (Deep Neural Networks & CNN)

2.6 สถาปัตยกรรม YOLO และพัฒนาการของแต่ละรุ่น

2.7 การวัดประสิทธิภาพของโมเดลเชิงคณิตศาสตร์ (Accuracy, Precision, Recall, F1, mAP, IoU)

2.8 งานวิจัยที่เกี่ยวข้องและข้อเปรียบเทียบกับงานนี้

2.1 แนวคิดและทฤษฎีพื้นฐานด้านการประมวลผลภาพ

การประมวลผลภาพ (Image Processing) เป็นศาสตร์แขนงหนึ่งของวิทยาการคอมพิวเตอร์ที่มีความสำคัญอย่างยิ่งต่อการพัฒนาระบบอัจฉริยะ โดยเฉพาะในยุคปัจจุบันที่ข้อมูลส่วนใหญ่ที่มนุษย์รับรู้มาจากการมองเห็น การทำให้คอมพิวเตอร์สามารถตีความและเข้าใจภาพได้ จึงเป็นรากฐานสำคัญของระบบวิชวลอินสเปกชัน (Visual Inspection) ในภาคอุตสาหกรรม การประมวลผลภาพสามารถแบ่งออกได้เป็นหลายระดับ ตั้งแต่ระดับพื้นฐานไปจนถึงระดับสูง [1, 33] ดังนี้

2.1.1 การประมวลผลภาพเชิงต่ำ (Low-Level Processing)

การประมวลผลภาพในระดับนี้หมายถึงการปรับปรุงคุณภาพของภาพ เช่น การปรับความสว่าง (Brightness Adjustment) การปรับความคมชัด (Sharpening) และการลดสัญญาณรบกวน (Noise Reduction) ตัวอย่างเช่น หากภาพที่ได้จากกล้อง Webcam มีแสงน้อยจนไม่สามารถมองเห็นตำหนิ

ได้ชัดเจน การประมวลผลภาพเชิงต่ำสามารถเพิ่มคุณภาพของภาพเพื่อให้โมเดลสามารถตรวจจับได้แม่นยำยิ่งขึ้น [1]

2.1.2 การประมวลผลภาพเชิงกลาง (Mid-Level Processing)

ในระดับนี้เน้นไปที่การแยกแยะหรือดึงคุณลักษณะของวัตถุ เช่น การตรวจหาขอบภาพ (Edge Detection) การตรวจหามุม (Corner Detection) หรือการวิเคราะห์ Texture เทคนิคเหล่านี้ช่วยให้คอมพิวเตอร์สามารถมองเห็นโครงร่างของวัตถุและแยกส่วนที่น่าสนใจออกจากพื้นหลังได้ เช่น การตรวจสอบว่าขวดมีฝาปิดหรือไม่ อาจใช้การตรวจจับเส้นรอบวงด้านบนของขวด ซึ่งหากไม่มีฝาจะปรากฏเป็นช่องว่างที่แตกต่างจากขวดสมบูรณ์ [34, 35]

2.1.3 การประมวลผลภาพเชิงสูง (High-Level Processing)

หมายถึงการตีความและทำความเข้าใจสิ่งที่อยู่ในภาพ (Image Understanding) เช่น การจำแนกประเภท (Classification) และการตรวจจับวัตถุ (Object Detection) ซึ่งต้องอาศัยอัลกอริทึมที่ซับซ้อนและมักใช้การเรียนรู้ของเครื่องเข้ามาช่วย การประมวลผลภาพเชิงสูงจึงเป็นหัวใจของงานวิจัยนี้ เนื่องจากเป้าหมายคือการตรวจจับและจำแนกขวดที่มีตำหนิออกจากขวดที่สมบูรณ์ [36]

2.1.4 ความสำคัญของการประมวลผลภาพในอุตสาหกรรม

การตรวจสอบคุณภาพสินค้าในสายการผลิตเป็นขั้นตอนที่ไม่สามารถละเลยได้ โรงงานที่ผลิตสินค้าวันละหลายหมื่นหรือหลายแสนชิ้นไม่สามารถพึ่งพาการตรวจสอบด้วยสายตามนุษย์ได้ทั้งหมด เพราะมีโอกาสเกิดความผิดพลาดสูง ข้อมูลจากรายงานของ International Society of Automation (ISA) ระบุว่ามนุษย์มีแนวโน้มตรวจจับข้อผิดพลาดตกหล่นมากกว่าร้อยละ 15 เมื่อต้องทำงานตรวจสอบต่อเนื่องนานกว่า 8 ชั่วโมง [37, 38] ดังนั้นการใช้ Computer Vision จึงเข้ามามีบทบาทในการทำงานแทนมนุษย์ โดยมีข้อดีคือทำงานได้รวดเร็วกว่าแม่นยำกว่า และไม่เหนื่อยล้า [1], [36]

นอกจากนี้ การใช้ระบบอัตโนมัติยังสามารถบันทึกข้อมูลเพื่อการวิเคราะห์ย้อนหลังได้ เช่น หากพบข้อร้องเรียนจากลูกค้า โรงงานสามารถย้อนกลับไปตรวจสอบภาพของสินค้าที่ออกจากสายการผลิตได้ ทำให้สามารถระบุสาเหตุของความผิดพลาดได้แม่นยำมากยิ่งขึ้น ซึ่งเป็นสิ่งที่การตรวจสอบด้วยมนุษย์ไม่สามารถทำได้

ข้อจำกัดของการประมวลผลภาพแบบดั้งเดิม

แม้ว่าการประมวลผลภาพเชิงดั้งเดิม (Traditional Image Processing) เช่น การตรวจจับขอบภาพด้วย Sobel หรือ Canny Edge Detector จะเป็นเทคนิคที่ใช้มานาน แต่ก็มีข้อจำกัดหลายประการ ได้แก่

2.1.4.1 ความไวต่อสภาพแสง เทคนิคดั้งเดิมมักทำงานได้ดีในสภาวะแสงคงที่ แต่เมื่อมีแสงเงาหรือการสะท้อน ผลลัพธ์จะผิดพลาดทันที

2.1.4.2 ความซับซ้อนของวัตถุ หากตำหนิมีลักษณะใกล้เคียงกับชิ้นงานที่สมบูรณ์ อัลกอริทึมดั้งเดิมจะไม่สามารถแยกแยะได้

2.1.4.3 ความเร็วในการประมวลผล วิธีดั้งเดิมบางชนิดใช้เวลานานเกินไป ไม่เหมาะกับการทำงานแบบ Real-time [34, 35]

2.1.5 การก้าวเข้าสู่ยุคของการเรียนรู้เชิงลึก

เพื่อแก้ไขข้อจำกัดเหล่านี้ นักวิจัยได้หันมาใช้เทคนิค Deep Learning โดยเฉพาะโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Networks, CNN) ซึ่งสามารถเรียนรู้คุณลักษณะของภาพได้โดยตรงจากข้อมูลจำนวนมาก โดยไม่จำเป็นต้องดึงคุณลักษณะด้วยมือ (Hand-Crafted Features) วิธีนี้ช่วยให้สามารถตรวจจับตำหนิที่ซับซ้อนได้ดีกว่า [36], [39] ตัวอย่างเช่น หากต้องตรวจจับขวดที่ไม่มีฉลาก CNN สามารถเรียนรู้ได้จากข้อมูลภาพจำนวนมากว่าพื้นที่ตรงกลางของขวดควรมีฉลาก และหากไม่พบลักษณะดังกล่าว โมเดลจะทำนายว่าเป็น Defect ได้อย่างแม่นยำ ต่างจากการใช้การตรวจจับสีหรือขอบเขตที่มักล้มเหลวเมื่อฉลากมีสีใกล้เคียงกับตัวขวด [36], [39, 40]

2.1.6 การก้าวเข้าสู่ยุคของการเรียนรู้เชิงลึก

เพื่อแก้ไขข้อจำกัดเหล่านี้ นักวิจัยได้หันมาใช้เทคนิค Deep Learning โดยเฉพาะโครงข่ายประสาทเทียมแบบคอนโวลูชัน (CNN) ซึ่งสามารถเรียนรู้คุณลักษณะของภาพได้โดยตรงจากข้อมูลจำนวนมาก โดยไม่จำเป็นต้องดึงคุณลักษณะด้วยมือ (Hand-Crafted Features) วิธีนี้ช่วยให้สามารถตรวจจับตำหนิที่ซับซ้อนได้ดีกว่า ตัวอย่างเช่น หากต้องตรวจจับขวดที่ไม่มีฉลาก CNN สามารถเรียนรู้ได้จากข้อมูลภาพจำนวนมากว่าพื้นที่ตรงกลางของขวดควรมีฉลาก และหากไม่พบลักษณะดังกล่าว โมเดลจะทำนายว่าเป็น Defect ได้อย่างแม่นยำ ต่างจากการใช้การตรวจจับสีหรือขอบเขตที่มักล้มเหลวเมื่อฉลากมีสีใกล้เคียงกับตัวขวด

2.2 การเรียนรู้ของเครื่องและโครงข่ายประสาทเทียม

การเรียนรู้ของเครื่อง (Machine Learning: ML) เป็นศาสตร์แขนงหนึ่งของปัญญาประดิษฐ์ (Artificial Intelligence: AI) ที่มุ่งเน้นให้คอมพิวเตอร์สามารถเรียนรู้จากข้อมูลและปรับปรุงสมรรถนะได้โดยไม่จำเป็นต้องถูกกำหนดกฎเกณฑ์อย่างตายตัวโดยมนุษย์ หลักการสำคัญคือการใช้ข้อมูล (Data) และประสบการณ์ (Experience) เพื่อให้ระบบสามารถสร้างแบบจำลองทางคณิตศาสตร์ที่อธิบายความสัมพันธ์ระหว่างสัญญาณเข้า (Input) และสัญญาณออก (Output) จากนั้นจึงนำไปใช้ทำนายผลลัพธ์ใหม่ ๆ ที่ไม่เคยพบมาก่อน [41, 42]

ในเชิงคณิตศาสตร์ การเรียนรู้ของเครื่องสามารถอธิบายได้ว่าเป็นการหาฟังก์ชัน $f(x)$ ที่ใกล้เคียงกับความสัมพันธ์จริงระหว่าง Input x และ Output y มากที่สุด โดยอาศัยกระบวนการปรับ

ค่าพารามิเตอร์ภายในแบบจำลองเพื่อลดค่าความคลาดเคลื่อน (Error) ระหว่างค่าที่ทำนายได้และค่าความจริง (Ground Truth) [41, 42]

2.3 ประเภทของการเรียนรู้ของเครื่อง

การเรียนรู้ของเครื่องสามารถแบ่งออกได้เป็น 3 ประเภทหลัก ดังนี้

2.3.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning)

เป็นการเรียนรู้จากข้อมูลที่มีการกำหนดฉลาก (Label) ไว้ล่วงหน้า เช่น ข้อมูลภาพขวดที่ Annotate แล้วว่า “มีฝา” หรือ “ไม่มีฝา” โมเดลจะเรียนรู้จากความสัมพันธ์ระหว่างภาพ (Input) และฉลาก (Output) เพื่อนำไปทำนายข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน วิธีนี้เป็นหัวใจของงานวิจัยในครั้งนี้ เพราะผู้วิจัยได้ทำการ Label ภาพด้วย Roboflow อย่างเป็นระบบ [41, 42, 43]

2.3.2 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning)

ใช้กับข้อมูลที่ไม่มีการกำหนดฉลาก เช่น การจัดกลุ่ม (Clustering) โดยโมเดลจะพยายามหาความคล้ายคลึงของข้อมูลเอง เช่น หากให้โมเดลดูภาพขวดที่ไม่ได้ Annotate มันอาจจัดกลุ่มขวดที่มีสีหรือรูปร่างใกล้เคียงกัน แม้ไม่รู้ว่ากลุ่มนั้นคือ Defect หรือไม่ การเรียนรู้ประเภทนี้มักใช้ในงานค้นหารูปแบบแฝง (Hidden Patterns) [41, 42]

2.3.3 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

เป็นการเรียนรู้ผ่านกระบวนการลองผิดลองถูก (Trial and Error) โดยระบบจะได้รับ “รางวัล” หรือ “บทลงโทษ” ตามพฤติกรรมที่แสดงออก เช่น หุ่นยนต์บนสายพานเรียนรู้ว่าหากคัดแยกขวด Defect ถูกต้องจะได้รางวัล หากผิดจะถูกปรับค่า ซึ่งแม้ไม่ได้ใช้โดยตรงในงานวิจัยนี้ แต่ก็มีบทบาทในงานวิจัยด้านหุ่นยนต์อุตสาหกรรม [44]

2.4 โครงข่ายประสาทเทียม (Artificial Neural Networks, ANN)

โครงข่ายประสาทเทียมเป็นแรงบันดาลใจมาจากสมองมนุษย์ โดยประกอบด้วยโหนด (Neuron) ที่เชื่อมต่อกันเป็นชั้น (Layer) แต่ละโหนดทำหน้าที่ประมวลผลข้อมูลและส่งต่อไปยังชั้นถัดไปผ่านการคูณน้ำหนัก (Weight) และการบวกค่าคงที่ (Bias) เมื่อผ่านฟังก์ชันกระตุ้น (Activation Function) เช่น Sigmoid, ReLU หรือ Tanh โมเดลก็สามารถสร้างความสัมพันธ์ที่ไม่เชิงเส้นระหว่างอินพุตและเอาต์พุตได้ [36], [45]

ในอดีต ANN ที่มีชั้นน้อย (Shallow Neural Networks) สามารถแก้ปัญหาง่าย ๆ เช่น การจำแนกข้อมูลสองกลุ่ม แต่ไม่สามารถแก้ปัญหาที่ซับซ้อนเช่นการจำแนกรูปภาพได้ เนื่องจากความสามารถในการดึงคุณลักษณะ (Feature Extraction) ยังจำกัด [36], [45]

2.5 โครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks, DNN)

การมาถึงของ Deep Learning ทำให้ ANN พัฒนาเป็นโครงข่ายที่มีหลายชั้น (Deep Neural Networks) สามารถเรียนรู้คุณลักษณะเชิงลึกได้อย่างมีประสิทธิภาพ โดยเฉพาะโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks, CNN) ซึ่งถูกออกแบบมาเพื่อประมวลผลข้อมูลภาพโดยตรง [36], [39]

2.5.1 กระบวนการหลักของ CNN

CNN ประกอบด้วย 3 กระบวนการหลัก คือ Convolution Layer, Pooling Layer และ Fully Connected Layer โดยกระบวนการ Convolution Layer ใช้ Kernel หรือ Filter สไลด์ไปบนภาพเพื่อดึงคุณลักษณะ เช่น เส้นขอบ มุม หรือลายผิว (Texture) ส่วนกระบวนการ Pooling Layer มีเพื่อลดขนาดของข้อมูล (Downsampling) เช่น Max Pooling เพื่อลดจำนวนการคำนวณแต่ยังคงคุณลักษณะสำคัญไว้ และสุดท้าย กระบวนการ Fully Connected Layer ทำหน้าที่รวมคุณลักษณะที่ได้ทั้งหมดเพื่อนำไปใช้ในการทำนายคลาส [33], [36], [40]

2.5.2 การนำ CNN ไปใช้ในงานอุตสาหกรรม

โครงข่ายคอนโวลูชัน (Convolutional Neural Networks, CNN) ได้เปลี่ยนภาพรวมของ “วิสัยทัศน์คอมพิวเตอร์ในโรงงาน” จากระบบเชิงกฎที่ต้องออกแบบคุณลักษณะด้วยมือ ให้กลายเป็นระบบที่เรียนรู้ลำดับชั้นของลักษณะภาพได้เองตั้งแต่ขอบและพื้นผิว ไปจนถึงรูปร่างและความสัมพันธ์ของวัตถุในบริบทจริง ความสามารถเช่นนี้ทำให้ CNN ถูกนำไปใช้กว้างขวางในงานตรวจสอบคุณภาพ ไม่ว่าจะเป็นการมองหาตำหนิระดับไมครอนบนชิ้นส่วนอิเล็กทรอนิกส์ การวิเคราะห์ภาพเอกซเรย์ทางการแพทย์เพื่อค้นหาความผิดปกติ การควบคุมคุณภาพอาหารด้วยการจับสัญญาณสีพื้นผิวที่ผิดจากมาตรฐาน ตลอดจนการตรวจสอบคุณภาพบรรจุภัณฑ์อย่างขวดน้ำและเครื่องดื่มว่ามีองค์ประกอบครบถ้วน ไม่มีตำหนิ และเป็นไปตามข้อกำหนดการผลิต

หัวใจสำคัญของการประยุกต์ใช้ CNN ในสายพานอุตสาหกรรมไม่ใช่เพียงความแม่นยำ แต่รวมถึงเวลาแฝงและความแน่นอนของเวลาด้วย ภาพจากกล้องไหลเข้ามาอย่างต่อเนื่อง ขวดเคลื่อนผ่านจุดตรวจด้วยความเร็วที่อาจผันผวนเล็กน้อย ระบบจึงต้องให้คำตอบภายในกรอบเวลาที่เคร่งครัด เพื่อส่งสัญญาณคัดทิ้งไปยังตัวกระทำ (Rejector) ให้ตรงตำแหน่งทันจังหวะ หากช้าเพียงเล็กน้อย ขวดก็พ้นจุดคัดทิ้งไปแล้ว อีกด้านหนึ่ง สภาพแวดล้อมในโรงงานไม่เคยนิ่งแสงสะท้อนจากพลาสติกใส ฉลากเคลือบมัน เงาแข็งจากโคมไฟ หรือ Motion Blur [33], [35] เมื่อซัตเตอร์ช้าเกินไปทั้งหมดนี้ส่งผลโดยตรงต่อคุณภาพสัญญาณภาพ จึงต้องออกแบบให้ดี (มุมกล้อง แสง ดิฟฟิวเซอร์/โพลาร์ไรซ์ และกำหนด ROI/Mask) เพื่อให้โมเดลเห็นสิ่งที่สำคัญอย่างสม่ำเสมอ

แม้ CNN จะทรงพลัง แต่การใช้งานจริงยังต้องคำนึงถึงความเปลี่ยนแปลงของสินค้าและบริบท (Domain Shift) เช่น รุ่นขวดใหม่ ลวดลายฉลากที่เปลี่ยนไป หรือความสว่างที่ค่อย ๆ ลดลงตามอายุ

หลอดไฟ หากไม่มีการเฝ้าระวัง ตัวชี้วัดระบบ (เช่น FRR/FAR, Uptime) อาจค่อย ๆ แย่ลงโดยไม่รู้ตัว ดังนั้น ระบบที่ดีต้องถูกออกแบบให้ “เรียนรู้ ติดตาม ปรับจูน” ได้อย่างต่อเนื่อง ทั้งในส่วนข้อมูล ทั้งเก็บตัวอย่างเพิ่มและปรับแต่ง (Augmentation) กติกาหรือนโยบายการตัดสินใจ (Threshold) โดยไม่จำเป็นต้องรีเทรนโมเดลใหม่ทุกครั้ง [37, 38]

ภายในการทำงาน (Pipe Line) ของระบบตรวจสอบด้วยภาพ กระบวนการเริ่มจากการรับภาพและเตรียมภาพอย่างเหมาะสม เพื่อลดเวลาแฝงและลดสิ่งรบกวน จากนั้นโมเดล CNN ทำอนุมานให้ผลเป็นกรอบและคลาสของวัตถุ พร้อมค่าความเชื่อมั่น ผลเหล่านี้ผ่านการคัดกรอง เช่น NMS และเกณฑ์ความเชื่อมั่น (IoU) ก่อนถูกตีความในระดับ “ชิ้นงานหนึ่งชิ้น”อย่างกรณีขวดหนึ่งใบเพื่อหลีกเลี่ยงการตัดสินใจจากสัญญาณแปลกปลอม ขั้นตอนถัดมาคือการแปลงผลอนุมานให้เป็นคำตอบทางกระบวนการ คือ Pass หรือ Reject และกำหนดเวลาคัดทิ้งล่วงหน้าไปยังไมโครคอนโทรลเลอร์ เช่น Arduino เพื่อขับรีเลย์หรือโซลินอยด์ไปยังหัวเป่าลมหรือแขนผลักให้ตรงจังหวะ พร้อมบันทึกเหตุการณ์ เช่น พารามิเตอร์ Timestamp, Bbox, Class, Conf, Decision, สถานะ I/O ฯลฯ สำหรับตรวจสอบย้อนหลังและปรับจูน [38], [46]

ในบริบทของงานวิจัยนี้ จึงเลือกใช้ CNN ผ่านสถาปัตยกรรม YOLOv8 ด้วยเหตุผลหลักคือ เร็วแม่นยืดหยุ่น สถาปัตยกรรมแบบ One-Stage ช่วยลดเวลาแฝงให้เพียงพอกับสายพานจริง ขณะที่ระบบของ Ultralytics ทำให้เวิร์กโฟลว์การฝึก ทดสอบ อนุมานเป็นมาตรฐาน ปรับขนาดโมเดลได้ตามทรัพยากร เช่น รุ่น n, s หรือ m และยังรองรับการเพิ่มประสิทธิภาพตอนนำขึ้นใช้งาน เช่น TensorRT/FP16/INT8 บน Jetson หรือ GPU บนพีซี เมื่อนำมาประกบกับการออกแบบแสงและ ROI ที่รัดกุม รวมถึงตรรกะการตัดสินใจระดับชิ้นงาน (Per-Bottle Rules) ระบบที่ได้จึงไม่เพียง “แม่น” ในเชิงโมเดล แต่ยัง “เชื่อถือได้” ในเชิงระบบสามารถตัดสินใจให้ทันเวลา ควบคุมอัตราคัดทิ้งผิดและหลุดรอดให้อยู่ในเกณฑ์ และพร้อมรับมือความเปลี่ยนแปลงของหน้างานอย่างเป็นระบบ [10, 11], [47]

2.5.3 ตัวอย่างเชิงเปรียบเทียบ

เพื่อให้เห็นภาพ สมมติว่ามีขวดน้ำ 1,000 ขวดที่ต้องตรวจสอบด้วยสายตามนุษย์ ผลอาจมีความผิดพลาด 100–150 ขวด แต่หากใช้ CNN ผ่าน YOLOv8 จะสามารถตรวจจับได้ถูกต้องมากกว่า 950 ขวด ความผิดพลาดเหลือเพียง 50 ขวด ซึ่งเมื่อคำนวณเป็นเปอร์เซ็นต์ ความแม่นยำจะสูงกว่า ร้อยละ 95 สิ่งนี้สะท้อนถึงประโยชน์เชิงเศรษฐกิจที่ช่วยลดการสูญเสียในสายการผลิตได้อย่างมาก

2.6 สถาปัตยกรรม YOLO

สถาปัตยกรรม YOLO (You Only Look Once) ถือเป็นจุดเปลี่ยนสำคัญในวงการ Computer Vision โดยเฉพาะด้านการตรวจจับวัตถุ (Object Detection) ก่อนหน้าการพัฒนา YOLO วิธีการ

ตรวจจับวัตถุแบบดั้งเดิม เช่น R-CNN, Fast R-CNN และ Faster R-CNN มีความแม่นยำสูงแต่ยังคงใช้เวลาประมวลผลนานเกินไป ไม่สามารถตอบโจทย์การทำงานในเวลาจริง (Real-time) ได้ YOLO ได้เสนอแนวคิดใหม่โดยการรวมการตรวจจับวัตถุและการจำแนกวัตถุไว้ในขั้นตอนเดียว ทำให้สามารถตรวจจับได้อย่างรวดเร็วโดยไม่สูญเสียความแม่นยำมากนัก [48, 49, 50]

2.6.1 พัฒนาการของ YOLO แต่ละรุ่น

YOLOv1 เป็นงานวิจัยแรกของ Redmon et al. [50] ได้เสนอแนวคิด “You Only Look Once” โดยแบ่งภาพออกเป็นกริด (Grid Cell) ขนาด S×S แต่ละ Cell จะทำการทำนาย Bounding Box และ Class Probability จุดเด่นคือความเร็วที่สูงมาก สามารถทำงานได้ถึง 45 FPS แต่ข้อจำกัดคือความแม่นยำต่ำในกรณีที่มีวัตถุขนาดเล็กหรือมีหลายวัตถุอยู่ใกล้กัน [50]

YOLOv2 หรือ YOLO9000 ซึ่งพัฒนาขึ้นในปี 2017 รุ่นนี้ถูกพัฒนาให้แม่นยำขึ้นโดยการเพิ่ม Batch Normalization และใช้ Anchor Box ซึ่งทำให้โมเดลสามารถตรวจจับวัตถุหลากหลายขนาดได้ดีขึ้น นอกจากนี้ยังสามารถตรวจจับวัตถุได้กว่า 9,000 คลาสจากการเรียนรู้ร่วมกัน (Joint Training) กับ ImageNet และ COCO Dataset YOLOv2 จึงเป็นที่นิยมในงานวิจัยและภาคอุตสาหกรรม [51]

YOLOv3 ซึ่งพัฒนาขึ้นในปี 2018 ได้นำแนวคิดของ Residual Networks (ResNet) และ Feature Pyramid Networks (FPN) เข้ามาใช้ ทำให้สามารถตรวจจับวัตถุได้หลายสเกล (Multi-scale Detection) โมเดลนี้มีความสมดุลระหว่างความเร็วและความแม่นยำ จนกลายเป็นรุ่นที่ใช้แพร่หลายที่สุดในช่วงหลายปี เนื่องจากสามารถทำงานได้ทั้ง Real-time และให้ความแม่นยำสูงในสภาพแวดล้อมจริง [52]

YOLOv4 ที่พัฒนาขึ้นในปี 2020 ที่พัฒนาโดย Alexey Bochkovskiy โดยเน้นการเพิ่มเทคนิคการฝึกใหม่ ๆ เช่น CSPDarknet53 Backbone, Mish Activation, Mosaic Data Augmentation และ Self-Adversarial Training (SAT) ทำให้ YOLOv4 มีความแม่นยำสูงขึ้นในขณะที่ยังคงรักษาความเร็วได้ดี สามารถทำงานได้กว่า 60 FPS และมีค่า mAP สูงกว่างานวิจัยเดิม ๆ หลายเปอร์เซ็นต์ [53]

YOLOv5 ที่พัฒนาขึ้นในปี 2020 เช่นเดียวกัน ถูกพัฒนาโดย Ultralytics และกลายเป็นรุ่นที่ได้รับความนิยมอย่างสูง เนื่องจากใช้งานง่ายและพัฒนาใน PyTorch เต็มรูปแบบ YOLOv5 มีหลายขนาด (Nano, Small, Medium, Large, XLarge) ทำให้ผู้ใช้งานเลือกได้ตามความต้องการด้านความเร็วหรือความแม่นยำ อีกทั้งยังรองรับการ Deploy บน Mobile และ Edge Device เช่น Jetson Nano หรือ Raspberry Pi ได้สะดวก [54]

YOLOv6 พัฒนาขึ้นในปี 2022 ได้รับการพัฒนาโดย Meituan เพื่อเน้นงานเชิงพาณิชย์ โดยเน้นที่ความเร็วสูงมาก (เหมาะกับ Edge Device) [55]

YOLOv7 ที่พัฒนาขึ้นในปี 2022 เช่นเดียวกับ YOLOv6 ถูกพัฒนาโดย WongKinYiu et al. เป็นรุ่นที่ได้รับการยกย่องว่ามีประสิทธิภาพสูงสุดในยุคนั้น โดยมีเทคนิคใหม่ ๆ เช่น E-ELAN และมีค่า mAP สูงกว่า YOLOv5, YOLOv6 ทั้งในงานวิจัยและงานจริง [56]

YOLOv8 ถูกพัฒนาขึ้นในปี 2023 ซึ่งพัฒนาโดย Ultralytics ถือเป็นเวอร์ชันล่าสุดที่ปรับปรุงทั้งโครงสร้างโมเดลและ Workflow การใช้งาน จุดเด่นของ YOLOv8 ได้แก่ รองรับการทำงานแบบ Plug-and-Play ผ่าน ultralytics package มีหลายโมเดลให้เลือก เช่น YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x ที่ปรับสมดุลความเร็วและความแม่นยำ รองรับการ Export โมเดลไปยังหลายแพลตฟอร์ม เช่น ONNX, TensorRT, CoreML ทำให้ Deploy ง่ายมากขึ้นรองรับทั้งการตรวจจับวัตถุ (Object Detection), การจำแนกภาพ (Image Classification), การแบ่งส่วนเชิงวัตถุ (Instance Segmentation) และการติดตามวัตถุ (Object Tracking) YOLOv8 ยังปรับโครงสร้าง Backbone และ Neck ให้มีประสิทธิภาพสูงขึ้น โดยใช้ C2f Module ซึ่งเป็นการพัฒนาต่อยอดจาก CSPNet ทำให้สามารถดึง Feature ได้ลึกขึ้นในขณะที่ใช้พารามิเตอร์น้อยกว่าเดิม [10], [57, 58]

ตารางที่ 2-1 การเปรียบเทียบประสิทธิภาพ YOLO แต่ละรุ่น

รุ่น	ปี	ความเร็ว (FPS)	mAP บน COCO	จุดเด่น	ข้อจำกัด
YOLOv1	2016	~45	~63%	รวดเร็วที่สุดในยุค	ตรวจจับวัตถุเล็กไม่ดี
YOLOv2	2017	~40	~78%	Anchor Box, Multi-class	ยังมีข้อจำกัดใน Multi-scale
YOLOv3	2018	~30–35	~85%	Multi-scale, ResNet	ใช้ทรัพยากรเพิ่มขึ้น
YOLOv4	2020	~60	~89%	Mosaic Augmentation, CSPDarknet53	ต้องการ GPU ที่แรง
YOLOv5	2020	~30–140 (แล้วแต่รุ่น)	~90%	ใช้งานง่าย, PyTorch	ไม่ใช่เวอร์ชันดั้งเดิม
YOLOv7	2022	~30–120	~92%	E-ELAN, Efficient	ขนาดใหญ่, Training ซับซ้อน
YOLOv8	2023	~30–120	~93–95%	C2f, ใช้งานง่าย, Flexible	ยังใหม่ ต้องทดสอบเพิ่ม

วิเคราะห์จากความเหมาะสมจากตารางที่ 2-1 งานวิจัยนี้จึงเลือกใช้ YOLOv8 เนื่องจาก รองรับการใช้งานง่าย เชื่อมต่อกับ Roboflow และ Google Colab ได้โดยตรง ทำงานได้แบบ Real-time แม้ใช้ Webcam ราคาประหยัด มีความแม่นยำสูง ค่า Precision และ Recall มากกว่า ร้อยละ 95 ในชุดข้อมูลที่ทดลอง สามารถนำไปใช้งานบน Edge Device เช่น Jetson หรือ PC ที่มี GPU ปานกลางได้จริงรองรับการปรับแต่งเพื่อเชื่อมต่อกับ Arduino และระบบ Reject บนสายพานลำเลียง กล่าวคือ YOLOv8 เป็นสถาปัตยกรรมที่ลงตัวระหว่างความแม่นยำ ความเร็ว และความสะดวกในการใช้งาน เหมาะกับงานวิจัยนี้ที่ต้องการต้นทุนต่ำแต่ยังคงคุณภาพในระดับอุตสาหกรรม [10], [57, 58]

2.7 การวัดประสิทธิภาพของโมเดลเชิงคณิตศาสตร์

การประเมินประสิทธิภาพของโมเดลการตรวจจับวัตถุเป็นขั้นตอนที่มีความสำคัญยิ่ง เนื่องจากไม่เพียงช่วยสะท้อนคุณภาพของโมเดลในเชิงตัวเลข แต่ยังช่วยให้ผู้วิจัยสามารถระบุข้อจำกัดและปรับปรุงระบบให้มีความแม่นยำมากยิ่งขึ้น การวัดประสิทธิภาพที่ใช้ทั่วไปในงานด้าน Computer Vision และ Deep Learning ประกอบด้วยตัวชี้วัดหลัก ได้แก่ Accuracy (สมการที่ 2-1) ซึ่งตัวชี้วัดนี้เหมาะกับกรณีที่จำนวนคลาสสมดุล แต่หากมีข้อมูลที่ไม่สมดุล (เช่น ขวดดี ร้อยละ 90 ขวด ขวดที่เป็นชิ้นงานเสีย ร้อยละ 10) Accuracy อาจทำให้เข้าใจผิดได้ เหมาะกับกรณีคลาสสมดุล แต่ถ้าข้อมูลไม่สมดุล (เช่น ขวดดี ร้อยละ 90 : defect ร้อยละ 10) อาจให้ภาพรวมที่ “ดูดีเกินจริง” จึงต้องพิจารณาค่าชี้วัดอื่นร่วมด้วย [37, 38] Precision (สมการที่ 2-2) บอกว่าจากสิ่งที่โมเดลบอกว่าเป็น ชิ้นงานเสีย มีสัดส่วนเท่าไรที่เป็น ชิ้นงานเสีย จริง ค่า Precision สูงสะท้อนว่าโมเดลไม่ทำนายผิดว่าขวดปกติเป็นชิ้นงานเสียบ่อยนัก บอกว่า “สิ่งที่โมเดลบอกว่าเป็นชิ้นงานเสีย” มีสัดส่วนเท่าไรที่เป็นของเสียจริง ค่า Precision สูงช่วยลดการคัดทิ้งของดี (False Positive) [37, 38] Recall (สมการที่ 2-3) ซึ่งสะท้อนถึงความสามารถของโมเดลในการตรวจจับ defect ทั้งหมด Recall สูงหมายถึงโมเดลไม่พลาด defect จริงในสายการผลิต บอกความสามารถของระบบในการ “ไม่ปล่อยให้ของเสียหลุดรอด” ค่า Recall สูงลดความเสี่ยงหลุดรอดไปขั้นตอนถัดไป [53, 54] F1-Score (สมการที่ 2-4) เป็นตัวชี้วัดที่สมดุลระหว่าง Precision และ Recall หากค่าใดค่าหนึ่งต่ำ F1 ก็จะต่ำไปด้วย เป็นค่าดุลยภาพระหว่าง Precision และ Recall เหมาะเมื่อไม่สามารถเลือกให้ความสำคัญด้านใดด้านหนึ่งเป็นพิเศษ [53] และ mAP (mean Average Precision) (สมการที่ 2-5) mAP ใช้กันแพร่หลายที่สุดในงาน Object Detection โดยพิจารณาทั้ง Precision และ Recall หลายค่า Threshold ของ IoU (Intersection over Union) เช่น mAP50 หมายถึงค่า mAP ที่ IoU ≥ 0.5 ส่วน mAP50-95 หมายถึงค่าเฉลี่ย mAP ที่ IoU ตั้งแต่ 0.5 ถึง 0.95 ใช้กำหนดเกณฑ์การนับ “ทายถูก” ของกรอบตรวจจับ เช่น mAP@0.5 หมายถึงคำนวณ mAP โดยนับถูกเมื่อ IoU ≥ 0.5 ส่วน mAP@0.5:0.95 คือค่าเฉลี่ย mAP เมื่อปรับเกณฑ์ IoU ตั้งแต่ 0.50 ถึง 0.95 ทีละ 0.05 [37], [46], [59, 60, 61]

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (2\text{-}1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2\text{-}2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2\text{-}3)$$

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2\text{-}4)$$

$$\text{IoU} = \frac{\text{Aerea of Overlap}}{\text{Area of Union}} \quad (2\text{-}5)$$

เมื่อ TP (True Positive) คือ โมเดลทำนายว่ามีตำหนิ และมีตำหนิจริง
TN (True Negative) คือ โมเดลทำนายว่าปกติ และปกติจริง
FP (False Positive) คือ โมเดลทำนายว่ามีตำหนิ แต่จริง ๆ ปกติ
FN (False Negative) คือ โมเดลทำนายว่าปกติ แต่จริง ๆ มีตำหนิ

ตัวอย่างการคำนวณ สมมติว่ามีการทดสอบโมเดลกับ ภาพ 1,000 ภาพ ซึ่งผล Confusion Matrix เป็นดังนี้ TP มีค่าเท่ากับ 460 TN มีค่าเท่ากับ 480 FP มีค่าเท่ากับ 30 และ FN มีค่าเท่ากับ 30 ดังนั้น Accuracy มีค่าร้อยละ 94 Precision มีค่าร้อยละ 93.9 Recall มีค่าร้อยละ 93.9 F1-Score มีค่าประมาณร้อยละ 93.9 และ mAP50 (สมมติจากผลการทดสอบ) มีค่าร้อยละ 96.5 ผลลัพธ์ดังกล่าวชี้ว่าโมเดลมีความสมดุลระหว่าง Precision และ Recall และสามารถใช้งานในสายพานจริงได้

ในการตีความเชิงอุตสาหกรรม การประเมินประสิทธิภาพของระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียง การทำความเข้าใจค่าตัวชี้วัดทางสถิติ (Evaluation Metrics) เช่น Precision, Recall, F1-score และ Mean Average Precision (mAP) มีความสำคัญอย่างยิ่ง เพราะแต่ละค่ามีความหมายและผลกระทบต่อการตัดสินใจในเชิงอุตสาหกรรมแตกต่างกันโดยสิ้นเชิง การตีความผลการประเมินเหล่านี้จะช่วยให้ผู้พัฒนาและวิศวกรสามารถเลือกแนวทางการปรับโมเดลให้เหมาะสมกับลักษณะการผลิตจริงของโรงงาน

Precision (ค่าความแม่นยำในการทำนายผลบวก) Precision แสดงถึงความถูกต้องของการตรวจจับในกลุ่มที่ระบบระบุว่าเป็น “ชิ้นงานเสีย” หรือ Defect ยิ่งค่า Precision สูง หมายความว่า

ผลการตรวจจับที่ระบบระบุว่าเสีย มีโอกาสสูงที่จะเป็นของเสียจริง ตัวเลขนี้จึงสะท้อน “ความมั่นใจของระบบ” ในการแจ้งเตือน Defect ในเชิงอุตสาหกรรม หากต้องการ ลดการคัดทิ้งขวดที่ปกติ (False Positive) เช่น ในสายการผลิตที่มีต้นทุนสูงต่อหน่วย การเน้นให้ Precision สูงถือเป็นแนวทางที่เหมาะสม เพราะจะช่วยให้ขวดดีไม่ถูก Reject โดยไม่จำเป็น อย่างไรก็ตาม การตั้งค่าโมเดลให้เน้น Precision สูงเกินไปอาจทำให้ Recall ลดลง ซึ่งหมายถึงมีโอกาสที่โมเดลจะมองข้าม Defect บางตัว (เช่น ขวดที่ฝาเบี้ยวเล็กน้อย) และปล่อยผ่านเข้าสู่กระบวนการบรรจุหรือจำหน่ายได้ [37, 38]

Recall (ค่าความครอบคลุมหรือความสามารถในการตรวจจับ) Recall แสดงถึงความสามารถของระบบในการตรวจจับ Defect ทั้งหมดที่มีอยู่ในสายการผลิต ยิ่งค่า Recall สูง หมายความว่าระบบสามารถตรวจพบ Defect ได้ครบถ้วน ไม่ปล่อยให้หลุดรอดไปสู่กระบวนการถัดไป

ในเชิงอุตสาหกรรม การเน้น Recall สูงจะเหมาะกับกระบวนการที่ ต้องการตรวจสอบคุณภาพเข้มงวด เช่น สินค้าประเภทเครื่องดื่มหรือยาที่ Defect แม้เพียงเล็กน้อยก็อาจส่งผลต่อความปลอดภัยของผู้บริโภค อย่างไรก็ตาม การเพิ่ม Recall มักทำให้เกิด False Positive มากขึ้น หรือกล่าวคือระบบจะมีแนวโน้ม “ขยันแจ้งเตือน” มากเกินไป ส่งผลให้เกิดการ Reject ขวดดีโดยไม่จำเป็น ซึ่งอาจกระทบต่ออัตราการผลิต (Throughput) และเพิ่มภาระการตรวจสอบซ้ำของพนักงาน [37, 38]

F1-Score (ค่าดุลยภาพระหว่าง Precision และ Recall) F1-Score เป็นค่าที่ใช้วัดความสมดุลระหว่าง Precision และ Recall โดยเฉพาะอย่างยิ่งในกรณีที่ไม่สามารถเลือกได้ว่าควรให้ความสำคัญกับด้านใดมากกว่า ค่า F1 จะคำนวณจากค่าเฉลี่ยเชิงฮาร์มอนิก (Harmonic Mean) ของ Precision และ Recall ซึ่งเน้นความเสมอภาคของทั้งสองด้าน

ในมุมมองอุตสาหกรรม การใช้ค่า F1 เป็นเกณฑ์หลักจะช่วยให้ระบบตรวจจับ Defect มีความสมดุล คือ สามารถตรวจจับได้ครบถ้วนในระดับหนึ่งโดยไม่ทำให้เกิดการ Reject มากเกินไป จึงเหมาะสำหรับกระบวนการที่ต้องการทั้ง “ประสิทธิภาพในการตรวจจับ” และ “เสถียรภาพในการผลิต” เช่น สายพานบรรจุภัณฑ์ที่มีความเร็วปานกลางซึ่งต้องการรักษาคุณภาพโดยไม่ลดอัตราการผลิต [60, 61]

mAP (Mean Average Precision) สำหรับงานด้านการตรวจจับวัตถุ (Object Detection) ค่า mAP ถือเป็นตัวชี้วัดที่สำคัญที่สุด เพราะใช้วัดความสามารถของโมเดลในการทำนาย “ตำแหน่งของวัตถุ (Bounding Box)” และ “ประเภทของวัตถุ (Class Label)” พร้อมกัน ค่า mAP สูงหมายความว่าโมเดลสามารถระบุตำแหน่ง defect ได้อย่างแม่นยำและมีขอบเขตสอดคล้องกับของจริง

ในบริบทของสายพานลำเลียง การมีค่า mAP สูง (>ร้อยละ 90) หมายถึง ระบบสามารถวาดกรอบ Bounding Box ครอบตำแหน่ง defect ได้ถูกต้อง เช่น การระบุบริเวณฝาขวดที่หายไปหรือฉลากที่เอียงได้อย่างแม่นยำ ซึ่งเป็นปัจจัยสำคัญในการส่งสัญญาณ Reject ที่ตรงจังหวะและลดความผิดพลาดของกลไกทางกล (Pneumatic Rejector) ในการคัดแยกชิ้นงาน

ในส่วนของการประยุกต์ใช้ในเชิงอุตสาหกรรม จากการทดลองในงานวิจัยนี้ พบว่าโมเดล YOLOv8 ที่ได้รับการปรับแต่งสามารถรักษาค่า Precision เฉลี่ยได้ที่ ร้อยละ 96 และ Recall ที่ ร้อยละ 95 ซึ่งเป็นระดับที่ให้สมดุลระหว่างการตรวจจับ Defect ได้ครบถ้วนและลดการ Reject ขวดดีโดยไม่จำเป็น ค่า F1 ที่สูงกว่า ร้อยละ 95 แสดงถึงความเสถียรของระบบในการทำงานต่อเนื่อง และค่า mAP ที่อยู่ในระดับ ร้อยละ 97 ยืนยันว่าโมเดลสามารถระบุตำแหน่ง Defect ได้อย่างถูกต้องแม่นยำ

ในเชิงอุตสาหกรรม ตัวเลขเหล่านี้มีความหมายอย่างยิ่งต่อการตัดสินใจทางการผลิต หากโรงงานต้องการลดของเสียและต้นทุนการผลิต การเลือกใช้โมเดลที่มีค่า Precision สูงจะเหมาะสม แต่หากต้องการรักษาคุณภาพสินค้าระดับพรีเมียม การเลือกโมเดลที่เน้น Recall สูงอาจให้ความมั่นใจมากกว่า อย่างไรก็ตาม การรักษาความสมดุลโดยอ้างอิงค่า F1 และการปรับค่า Confidence Threshold ของ YOLOv8 ให้เหมาะสมกับสายการผลิตแต่ละประเภท ถือเป็นแนวทางที่ดีที่สุดในเชิงปฏิบัติ [46], [59]

2.8 งานวิจัยที่เกี่ยวข้อง

งานของ Mittal et al. [62] ใช้ CNN ดั้งเดิมตรวจจับตำหนิบนพื้นผิวโลหะ เช่น รอยขีดข่วนและรอยแตกร้าว ภายใต้สภาพแวดล้อมที่ควบคุม (แสงคงที่ ผิวสะอาด มุมกล้องเหมาะสม) แบบจำลองแยกชิ้นงานดี/เสียได้แม่นยำราว ร้อยละ 90–92 จุดเด่นคือโจทย์มีพื้นผิวเรียบและสภาพการถ่ายภาพสม่ำเสมอ จึงใช้สถาปัตยกรรมไม่ซับซ้อนได้ผลดี อย่างไรก็ดี เมื่อย้ายสู่สภาพจริงที่มีฝุ่นและแสงสะท้อน ผลลัพธ์ลดลงราว ร้อยละ 7–10 สะท้อนความเปราะบางต่อสภาพหน้างาน [63, 64]

Ren et al. [65] ประยุกต์ YOLOv4 กับบรรจุภัณฑ์อาหารที่เคลื่อนบนสายพาน (เช่น กระป๋องกล่องอาหาร) โดยใช้เทคนิค Mosaic Augmentation และ Self-Adversarial Training (SAT) เพื่อเพิ่มความหลากหลายของข้อมูล ผลทดลองห้องปฏิบัติการรายงานความเร็วราว 35 FPS พร้อม Precision ประมาณร้อยละ 93 และ Recall ประมาณ ร้อยละ 92 แสดงศักยภาพงานจริงแบบเรียลไทม์ แต่ยังไวต่อแสงสะท้อนและพื้นผิวเงางาม [66], [67]

Zhang et al. [68] ใช้ YOLOv5 ตรวจตำหนิบนแผงวงจร (เช่น จุดบัดกรีผิดปกติ) ได้ Precision ประมาณ ร้อยละ 94.5 Recall ประมาณ ร้อยละ94.0 ความเร็วราว 50 FPS เหมาะกับวัตถุขนาดเล็กมาก อย่างไรก็ดี เมื่อวัตถุหนาแน่นและอยู่ใกล้กันอาจเกิดการทับซ้อนของกรอบตรวจจับส่งผลให้ความแม่นยำลดลง [67, 69]

จากการทบทวนวรรณกรรมและงานวิจัยที่เกี่ยวข้อง สามารถสรุปสาระสำคัญได้ คือ การประมวลผลภาพ (Image Processing) เป็นรากฐานของระบบตรวจสอบอัตโนมัติ โดยมีการพัฒนาอย่างต่อเนื่องจากเทคนิคเชิงดั้งเดิม เช่น Edge Detection, Thresholding และ Feature

Extraction ไปสู่การใช้โครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks, DNNs) โดยเฉพาะ Convolutional Neural Networks (CNNs) ที่สามารถเรียนรู้คุณลักษณะจากข้อมูลภาพได้โดยตรง ทำให้ระบบสามารถทำงานได้แม่นยำขึ้นแม้ในสภาพแวดล้อมที่ซับซ้อน

การเรียนรู้ของเครื่อง (Machine Learning) และโครงข่ายประสาทเทียม (Artificial Neural Networks) ได้เข้ามามีบทบาทสำคัญในการตรวจสอบคุณภาพ โดยเฉพาะงานที่ต้องการแยกแยะความแตกต่างที่ละเอียดอ่อนระหว่างชิ้นงานดีและชิ้นงานเสีย ทั้งนี้ CNN และการพัฒนาต่อเนื่องสู่ Deep Learning ทำให้โมเดลมีศักยภาพในการตรวจสอบที่เร็วและแม่นยำกว่าวิธีการดั้งเดิมอย่างมีนัยสำคัญ

สำหรับสถาปัตยกรรม YOLO (You Only Look Once) ถือเป็นอีกหนึ่งก้าวกระโดด เนื่องจากสามารถตรวจจับวัตถุได้แบบ Real-time ในขั้นตอนเดียว ต่างจาก R-CNN หรือ Faster R-CNN ที่มีหลายขั้นตอน พัฒนาการตั้งแต่ YOLOv1 จนถึง YOLOv8 แสดงให้เห็นถึงความก้าวหน้าทั้งด้านความแม่นยำ (mAP สูงขึ้นต่อเนื่อง) และความเร็ว (FPS ที่สามารถรองรับสายการผลิตจริงได้) โดย YOLOv8 มีความยืดหยุ่น รองรับการใช้งานได้ทั้งการตรวจจับ (Detection) การจำแนก (Classification) การแบ่งส่วนเชิงวัตถุ (Segmentation) และการติดตามวัตถุ (Tracking) จึงเป็นสถาปัตยกรรมที่เหมาะสมที่สุดในการเลือกใช้สำหรับงานวิจัยนี้

ด้านการวัดประสิทธิภาพของโมเดล การใช้ตัวชี้วัดเชิงคณิตศาสตร์ เช่น Accuracy, Precision, Recall, F1-Score และ mAP ช่วยสะท้อนถึงศักยภาพของโมเดลในมิติต่าง ๆ ทั้งความแม่นยำ ความครอบคลุม ความสมดุล และความสามารถในการตรวจจับตำหนิในตำแหน่งที่ถูกต้อง การวิเคราะห์เชิงตัวเลขและตารางเปรียบเทียบหลายรอบการทดลองยืนยันว่า YOLOv8 สามารถให้ผลลัพธ์ที่เสถียรและมีประสิทธิภาพสูงกว่าเทคนิคดั้งเดิม

จากการทบทวนงานวิจัยที่เกี่ยวข้อง ทั้งในและต่างประเทศ พบว่า งานต่างประเทศมีการใช้โมเดลที่หลากหลาย เช่น CNN, YOLOv4, YOLOv5 และงานล่าสุด GES-YOLO ที่ปรับแต่ง YOLOv8 เพื่อเพิ่มความแม่นยำในงานตรวจจับขวดน้ำดื่ม ขณะที่งานวิจัยในประเทศไทยยังคงมีข้อจำกัดด้านจำนวนข้อมูลและทรัพยากร แต่ก็มีความพยายามปรับใช้เทคนิคให้เหมาะสม เช่น การใช้ SVM หรือการใช้ YOLOv5 ในการตรวจสอบชิ้นงานบางประเภท

เมื่อเปรียบเทียบงานทั้งหมด จะเห็นได้ว่า งานวิจัยนี้มีความแตกต่างและมีคุณค่าเพิ่ม เนื่องจาก (1) ใช้ YOLOv8 ซึ่งเป็นสถาปัตยกรรมล่าสุด (2) ใช้ Webcam ราคาประหยัด แทนการใช้กล้องอุตสาหกรรมราคาแพง (3) เชื่อมต่อกับ Arduino เพื่อควบคุมกลไก Reject ซึ่งทำให้งานวิจัยนี้ไม่เพียงหยุดอยู่ที่การตรวจจับเชิงซอฟต์แวร์ แต่ยังสามารถนำไปใช้ได้จริงในสายการผลิต และ (4) มีการทดสอบในสภาวะแวดล้อมหลากหลาย ซึ่งช่วยยืนยันความทนทานของระบบ

กล่าวโดยสรุป บทที่ 2 แสดงให้เห็นถึงความสำคัญของการผสมผสานระหว่างทฤษฎีด้าน Image Processing, Machine Learning, Deep Learning, การพัฒนาสถาปัตยกรรม YOLO และการวัดประสิทธิภาพโมเดล ตลอดจนการทบทวนงานวิจัยก่อนหน้า เพื่อสร้างรากฐานที่แข็งแกร่งสำหรับการดำเนินการวิจัยในบทต่อไป

บทที่ 3
วิธีการดำเนินการวิจัย

การวิจัยนี้ มีวัตถุประสงค์เพื่อสร้างและประเมินระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงแบบเรียลไทม์ โดยผสานการประมวลผลภาพด้วย YOLOv8 กับการสั่งงานฮาร์ดแวร์คัดทิ้งผ่าน Arduino/Rejector บนสายพานจำลอง วิธีดำเนินการประกอบด้วยการกำหนดปัญหาและข้อกำหนดระบบ การจัดเก็บ ทำฉลาก ขยายข้อมูลภาพ การฝึกและปรับจูนโมเดล การออกแบบซอฟต์แวร์และฮาร์ดแวร์ การทดสอบแบบเรียลไทม์ภายใต้เงื่อนไขแสงและความเร็วที่แตกต่าง และการวิเคราะห์สมรรถนะด้วยตัวชี้วัดมาตรฐาน (Precision, Recall, F1-score, mAP) ควบคู่กับการประเมินต้นทุนเพื่อยืนยันความเป็นไปได้เชิงอุตสาหกรรมเพื่อความชัดเจน กระบวนการดำเนินงานในบทนี้มีขั้นตอนดังนี้

3.1 การศึกษาวิเคราะห์ปัญหาที่เกี่ยวข้อง
3.2 การจัดเก็บและเตรียมข้อมูล
3.3 การพัฒนาแบบจำลองและวัดประสิทธิภาพ
3.4 การออกแบบและพัฒนาระบบ
3.5 การประเมินคุณภาพและความพึงพอใจ

3.1 การศึกษาวิเคราะห์ปัญหาที่เกี่ยวข้อง

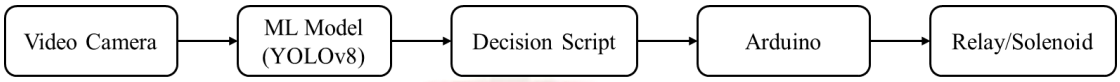
การศึกษาปัญหาที่เกี่ยวข้องเป็นขั้นตอนแรกในการวิจัยครั้งนี้ โดยผู้วิจัยได้ลงพื้นที่เก็บข้อมูลจากโรงงานขนาดกลางในอุตสาหกรรมเครื่องดื่ม ซึ่งประสบปัญหา การตรวจสอบคุณภาพขวดบรรจุภัณฑ์โดยเฉพาะตำหนิที่พบบ่อย เช่น ขวดไม่มีฝาปิด (Missing Cap) ขวดไม่มีฉลาก (Missing Label)

จากการสัมภาษณ์พนักงานตรวจสอบคุณภาพ พบว่ามีอัตราความผิดพลาดในการตรวจสอบสูงถึง ร้อยละ 12 – 15 ต่อกะการทำงาน โดยสาเหตุหลักเกิดจาก ความเหนื่อยล้า (Fatigue) ความเร็วสายพานที่สูงเกินไป และความซับซ้อนของตำหนิที่ยากต่อการตรวจจับด้วยตาเปล่า [70, 71]

เมื่อเปรียบเทียบกับเครื่องตรวจสอบเชิงพาณิชย์ (Vision Sensor) ที่มีราคาแพง (หลัก 300,000 – 500,000 บาท) พบว่าโรงงานขนาดกลางและขนาดเล็กมักไม่สามารถลงทุนได้ ทำให้ต้องใช้แรงงานมนุษย์แทน และนั่นคือที่มาของปัญหาที่วิจัยนี้พยายามแก้ไข [33]

ดังนั้น งานวิจัยนี้จึงมุ่งเน้นการพัฒนา ระบบตรวจสอบชิ้นงานเสียอัตโนมัติ ที่มีต้นทุนต่ำ โดยใช้ Webcam ราคาประหยัด คอมพิวเตอร์ PC Arduino ควบคุมกลไก Reject ร่วมกับการพัฒนาโมเดล

Machine Learning (YOLOv8) เพื่อยกระดับความแม่นยำในการตรวจสอบ และลดภาระของแรงงานมนุษย์ [10, 11], [47], [60]



ภาพที่ 3-1 สถาปัตยกรรมระบบตั้งแต่กล้องถึงกลไกคัดทิ้ง

3.2 การจัดเก็บและเตรียมข้อมูล

การจัดเก็บและเตรียมข้อมูลถือเป็นหัวใจสำคัญของงานวิจัยด้านการประมวลผลภาพและการเรียนรู้ของเครื่อง เนื่องจากคุณภาพและความครอบคลุมของชุดข้อมูลจะส่งผลโดยตรงต่อประสิทธิภาพของโมเดลตรวจจับชิ้นงานเสีย งานวิจัยนี้ได้ออกแบบกระบวนการเก็บข้อมูลอย่างเป็นระบบ โดยอาศัยการจำลองสายพานลำเลียงที่ติดตั้ง Webcam ความละเอียด 1080p ควบคู่กับระบบแสงสว่างแบบ LED Ring Light เพื่อควบคุมความสว่างและลดผลกระทบจากสภาพแวดล้อมภายนอก เช่น แสงจ้า เงาสะท้อน หรือแสงที่ไม่สม่ำเสมอ การควบคุมปัจจัยด้านแสงดังกล่าวช่วยให้ได้ภาพที่มีคุณภาพสูง เหมาะสมต่อการนำไปใช้ฝึกโมเดลตรวจจับ [33], [35], [43]

3.2.1 จำนวนและลักษณะของข้อมูล

การเก็บข้อมูลประกอบด้วยภาพถ่ายของขวดบรรจุภัณฑ์ทั้งที่ปกติและที่มีตำหนิหลากหลายประเภท เพื่อสร้างความสมดุลของข้อมูลและสะท้อนความเป็นจริงในการผลิต ข้อมูลที่เก็บได้มีดังนี้ ขวดปกติ: 5,000 ภาพ ขวดไม่มีฝา: 2,000 ภาพ ขวดไม่มีฉลาก: 2,000 ภาพ ขวดไม่มีฝาและไม่มีฉลาก: 1,500 ภาพ รวมทั้งหมด 10,500 ภาพ ซึ่งถือเป็นจำนวนที่เพียงพอต่อการฝึกโมเดล Deep Learning สำหรับงานตรวจจับวัตถุ (Object Detection) เนื่องจากมีความสมดุลระหว่างคลาสปกติและคลาส Defect

3.2.2 การทำ Annotate (Labeling)

เพื่อให้ข้อมูลมีความพร้อมในการนำไปฝึกโมเดล YOLO จำเป็นต้องมีการทำ การกำหนดป้ายกำกับ (Annotation) โดยใช้แพลตฟอร์ม Roboflow ซึ่งช่วยให้ผู้วิจัยสามารถสร้าง Bounding Box ครอบตำหนิของขวดได้อย่างแม่นยำในแต่ละคลาส เช่น "No Cap", "No Label", "Normal" ข้อดีของการใช้ Roboflow คือระบบสามารถ Export ข้อมูลให้อยู่ในรูปแบบที่รองรับ YOLO โดยตรง เช่นไฟล์ txt ที่มีข้อมูลพิกัด Bounding Box และคลาส ทำให้การนำเข้าข้อมูลสู่ขั้นตอนการฝึกโมเดลใน Google Colab เป็นไปอย่างรวดเร็วและลดความผิดพลาด [10], [43]

3.2.3 การทำ Data Augmentation

เพื่อเพิ่มความหลากหลายของข้อมูลและป้องกันปัญหา Overfitting ได้มีการประยุกต์เทคนิคการขยายข้อมูล (Data Augmentation) โดยใช้วิธีการ ได้แก่ การหมุนภาพ (Rotation): หมุน ±15° เพื่อจำลองมุมกล้องที่อาจเปลี่ยนไป การกลับด้าน เช่น Flip Horizontal และ Flip-Vertical เพื่อเพิ่มความหลากหลายของทิศทางวัตถุ การปรับความสว่าง (Brightness) และความเปรียบต่าง (Contrast) เพื่อรองรับสภาพแสงที่แตกต่าง การเบลอเล็กน้อย โดยการใช้ Gaussian Blur เพื่อจำลองกรณีภาพสั่นไหวหรือเบลอเล็กน้อยจากการเคลื่อนที่ หลังจากทำ Augmentation แล้ว จำนวนข้อมูลถูกขยายจาก 1,200 ภาพเป็น 2,500 ภาพ ซึ่งช่วยเพิ่มความครอบคลุมและความสามารถในการเรียนรู้ของโมเดล [70]

3.2.4 การแบ่งชุดข้อมูล

เพื่อให้การฝึกโมเดลมีความเป็นระบบและสามารถประเมินผลได้อย่างถูกต้อง ได้ทำการแบ่งข้อมูลออกเป็น 3 ชุดหลัก คือ Training Set: ร้อยละ70 หรือ 17,50 ภาพ สำหรับใช้ในการฝึกโมเดล Validation Set: ร้อยละ 20 หรือ 5,000 ภาพ สำหรับตรวจสอบการเรียนรู้และปรับค่าพารามิเตอร์ Test Set: ร้อยละ10 หรือ 2,500 ภาพ สำหรับใช้ทดสอบโมเดลที่เสร็จสิ้นแล้ว วิธีการแบ่งเช่นนี้ถือเป็นมาตรฐานในงานวิจัยด้าน Computer Vision และ Machine Learning เนื่องจากช่วยลดโอกาสในการประเมินผลที่เอนเอียง และสะท้อนความสามารถของโมเดลในสถานการณ์จริงได้ดียิ่งขึ้น [10], [36]

3.2.5 การจัดเก็บและบริหารจัดการข้อมูล

ข้อมูลทั้งหมดถูกจัดเก็บและจัดการผ่าน Google Drive ควบคู่กับ Roboflow โดยในขั้นตอนการฝึกโมเดล ข้อมูลถูก Import เข้า Google Colab ผ่าน API ของ Roboflow ซึ่งช่วยให้กระบวนการทำงานเป็นอัตโนมัติและสะดวกต่อการอัปเดตชุดข้อมูลในอนาคต การบริหารจัดการข้อมูลในลักษณะนี้ยังช่วยลดความซับซ้อนในการทำงานซ้ำ (Reproducibility) และรองรับการปรับปรุงโมเดลต่อเนื่อง (Continuous Improvement) ได้อย่างมีประสิทธิภาพ [10], [33], [43], [60]

3.3 การพัฒนาแบบจำลองและการวัดประสิทธิภาพ

การพัฒนาแบบจำลอง (Model Development) ถือเป็นขั้นตอนสำคัญที่สุดของงานวิจัย เนื่องจากคุณภาพของโมเดลจะเป็นตัวกำหนดโดยตรงถึงความสามารถในการตรวจจับและจำแนกตำหนิของขวดบนสายพานลำเลียง งานวิจัยนี้เลือกใช้สถาปัตยกรรม YOLOv8 ในตระกูล YOLO [10], [53] ที่มีความโดดเด่นในด้าน ความเร็ว ความแม่นยำ และความยืดหยุ่น โดยสามารถปรับใช้ได้ในงานอุตสาหกรรมจริงแบบ Plug-and-Play อีกทั้งยังรองรับการทำงานบนแพลตฟอร์ม Google Colab ซึ่ง

เอื้อต่อการฝึกโมเดลด้วย GPU ที่มีประสิทธิภาพสูงโดยไม่จำเป็นต้องลงทุนฮาร์ดแวร์ราคาแพง [10, 11]

3.3.1 สภาพแวดล้อมการพัฒนา (Environment Setup)

เพื่อให้การฝึกโมเดลและการทดสอบสามารถดำเนินการได้อย่างราบรื่น งานวิจัยนี้ได้เลือกสภาพแวดล้อมทั้งฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

3.3.1.1 ฮาร์ดแวร์ (Hardware)

คอมพิวเตอร์ส่วนบุคคล (Personal Computer, PC) ประกอบด้วย CPU Intel Core i5, RAM 32GB, Google Colab พร้อม GPU Tesla T4 / A100 (รองรับ CUDA) Webcam ความละเอียด 1080p สำหรับจับภาพจากสายพานลำเลียง Arduino Uno พร้อม Relay Module สำหรับเชื่อมต่อกับกลไก Pneumatic Rejector [11], [47]

3.3.1.2 ซอฟต์แวร์ (Software)

Google Colab (ใช้เป็นแพลตฟอร์มการฝึกโมเดล) Python 3.10 PyTorch (Deep Learning Framework) Ultralytics YOLOv8 [55] Library OpenCV [43] สำหรับการประมวลผลภาพ Roboflow API สำหรับการ Import Dataset Matplotlib และ Seaborn สำหรับการวิเคราะห์และสร้างกราฟ [71, 72]

3.3.1.3 พารามิเตอร์หลักในการฝึก (Training Parameters)

Dataset จำนวน 25,000 ภาพ ที่ได้หลังการทำ Augmentation Batch Size คือ 32 Epoch คือ 100 ขนาดภาพ คือ 640×640 พิกเซล การกำหนดสภาพแวดล้อมเช่นนี้ทำให้ระบบสามารถฝึกโมเดลได้อย่างมีประสิทธิภาพ ทั้งในเชิงความเร็วและการรองรับ Dataset ขนาดใหญ่ [10], [70]

3.3.2 การฝึกโมเดล (Model Training)

กระบวนการฝึกโมเดล YOLOv8 ในงานวิจัยนี้ดำเนินการบน Google Colab โดยเชื่อมต่อข้อมูลผ่าน Roboflow API ซึ่งช่วยให้การ Import Dataset มีความสะดวกและสามารถอัปเดตได้อัตโนมัติ ขั้นตอนการฝึกประกอบด้วย [10], [70], [73]

3.3.2.1 Preprocessing

ข้อมูลภาพทั้งหมดถูกปรับขนาด (Resize) ให้อยู่ที่ 640×640 พิกเซล ทำ Normalization ของค่า Pixel ให้อยู่ในช่วง [0,1] และใช้ Augmentation เช่น Rotation, Flip, Brightness Adjustment เพื่อเพิ่มความหลากหลายของข้อมูลและลดความเสี่ยงจาก Overfitting

3.3.2.2 Training

การฝึกใช้ Optimizer คือ Stochastic Gradient Descent (SGD) ที่มีความเหมาะสมต่อการเรียนรู้ที่มี Dataset ขนาดใหญ่ และใช้ Learning Rate Scheduling เพื่อลดอัตราการเรียนรู้ในระหว่าง Epoch

Loss Function ที่ใช้คือ Binary Cross Entropy (BCE) สำหรับการ Classification Complete IoU (CIoU) Loss สำหรับ Bounding Box Regression เพื่อให้การจับขอบเขตวัตถุแม่นยำขึ้น

3.3.2.3 Validation

ทุก ๆ 5 Epoch โมเดลจะถูกประเมินบน Validation Set เพื่อคำนวณค่า Precision, Recall, F1-score และ mAP (mean Average Precision) ซึ่งช่วยสะท้อนความสามารถของโมเดลในระหว่างการเรียนรู้

3.3.2.4 Early Stopping

เพื่อลดความเสี่ยงจาก Overfitting ได้มีการกำหนดเงื่อนไข Early Stopping หาก Validation Loss ไม่ลดลงภายใน 20 Epoch ระบบจะหยุดการฝึกโดยอัตโนมัติและเลือกโมเดลที่ดีที่สุด (Best.pt) สำหรับใช้งานจริง

3.3.3 การประเมินประสิทธิภาพ (Evaluation Metrics)

เพื่อประเมินคุณภาพของโมเดลหลังการฝึก ได้ใช้เกณฑ์มาตรฐานดังนี้ Precision (ความแม่นยำ) เพื่อวัดความถูกต้องของการตรวจจับชิ้นงานเสีย Recall (การครอบคลุม) เพื่อวัดความสามารถในการตรวจจับชิ้นงานเสียทั้งหมด F1-Score คือ ค่ากลางที่สมดุลระหว่าง Precision และ Recall mAP@50 และ mAP@50–95 วัดค่าเฉลี่ยความแม่นยำของ Bounding Box ในระดับ IoU ต่าง ๆ Confusion Matrix ที่แสดงผลการจำแนกแต่ละคลาสของชิ้นงาน [10]

นอกจากนี้ยังมีการวิเคราะห์เชิงกราฟ เช่น Learning Curve (Training Loss vs Validation Loss) (สมการที่ 3-1) กราฟ Precision/Recall Curve และการแสดง Detection Sample เพื่อตีความพฤติกรรมของโมเดลอย่างละเอียด

$$L = \lambda_{box} L_{CIoU} + \lambda_{obj} L_{obj} + \lambda_{cls} L_{cls} \quad (3\text{-}1)$$

เมื่อ L_{CIoU} คือ Complete IoU Loss (ตำแหน่ง Bounding Box)

L_{obj} คือ Objectness Loss (การตรวจจับวัตถุ)

L_{cls} คือ Classification Loss (การจำแนกคลาส)

นอกจากนี้ ยังมีการใช้ตัวชี้วัดที่ใช้เพิ่มเติม ได้แก่ Precision (สมการที่ 2-2) Recall (สมการที่ 2-3) F1-Score (สมการที่ 2-4) และ mAP (mean Average Precision) (สมการที่ 2-5) ใช้ทั้ง mAP50 และ mAP50-95

ผู้วิจัยทำการทดลองฝึกโมเดลทั้งหมด 5 รอบ โดยปรับ Hyperparameter เล็กน้อยในแต่ละครั้ง โดยผลลัพธ์ดังแสดงในตารางที่ 3-1 พบว่า Trial 3 และ 5 ให้ผลดีที่สุด โดยมีค่า Precision ประมาณ ร้อยละ 96 และ Recall ประมาณ ร้อยละ 95.8 ค่า mAP50 เฉลี่ยมากกว่า ร้อยละ 97 โมเดลแม่นยำสูง ค่า Loss ลดลงอย่างต่อเนื่อง ไม่มี Overfitting ชัดเจน

ตารางที่ 3-1 ตารางแสดงผลการฝึกโมเดล

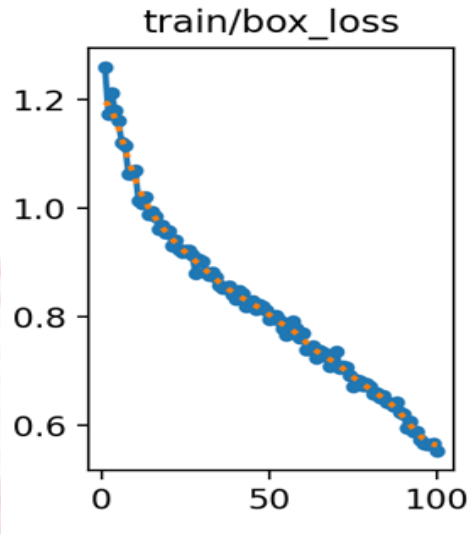
ฝึกรอบที่	Precision	Recall	F1-Score	mAP50	mAP50-95	Training Loss	Validation Loss
1	94.1%	93.5%	93.8%	96.0%	91.2%	0.051	0.067
2	95.2%	94.6%	94.9%	96.7%	92.1%	0.044	0.060
3	96.0%	95.5%	95.7%	97.1%	92.8%	0.039	0.056
4	95.6%	95.0%	95.3%	96.8%	92.5%	0.041	0.058
5	96.2%	95.8%	96.0%	97.3%	93.0%	0.037	0.054

3.3.4 กราฟการฝึก (Learning Curves)

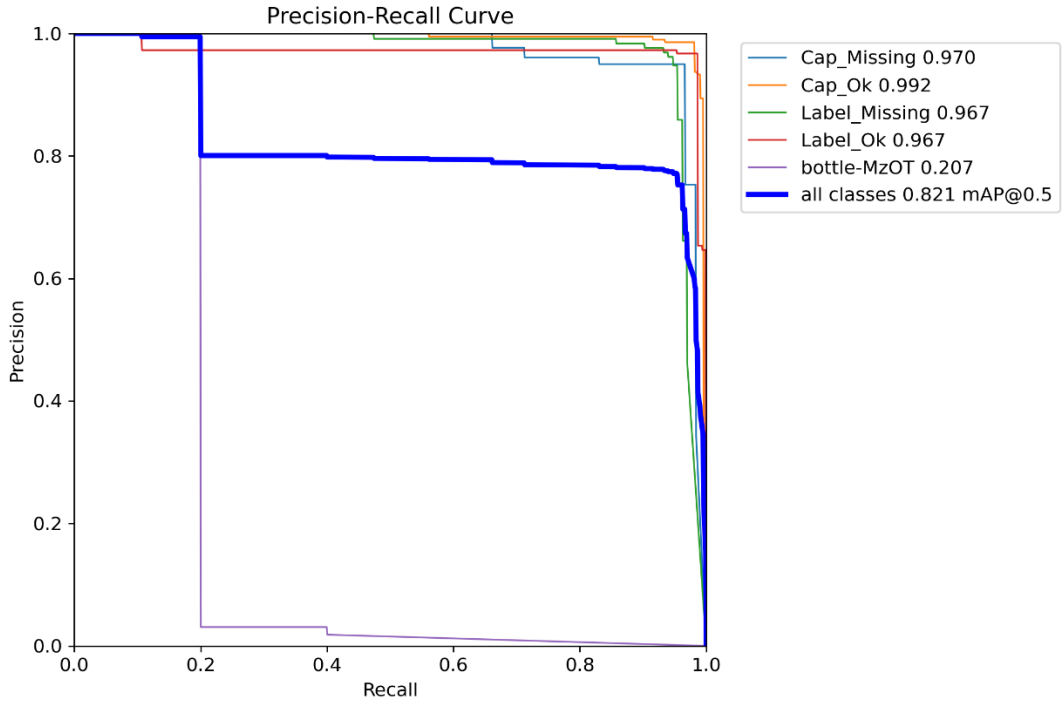
ผลการฝึกโมเดลสามารถอธิบายผ่านกราฟการเรียนรู้ ซึ่งสะท้อนให้เห็นถึงพฤติกรรมของโมเดลในระหว่างการปรับค่าพารามิเตอร์และการเรียนรู้จากข้อมูล ในส่วนนี้ผู้วิจัยได้วิเคราะห์ทั้ง ค่า Loss, Precision, Recall และ Confusion Matrix โดยเปรียบเทียบทั้ง Training Set และ Validation Set เพื่อตรวจสอบว่ามีแนวโน้มการเรียนรู้ที่ดีและไม่เกิด Overfitting

โดยกราฟ Training Loss แสดงให้เห็นว่าค่า Training Loss ลดลงอย่างต่อเนื่องจากประมาณ 0.15 ในช่วงเริ่มต้นของการฝึก เหลือเพียง 0.037 เมื่อสิ้นสุดที่ Epoch ที่ 100 ในขณะที่ Validation Loss ลดลงจากค่าเริ่มต้นประมาณ 0.17 จนเหลือเพียง 0.054 และเริ่มมีค่าคงที่หลัง Epoch ที่ 100 พฤติกรรมเช่นนี้สะท้อนว่าโมเดลมีการเรียนรู้ที่มีประสิทธิภาพ โดยไม่เกิด Overfitting อย่างมีนัยสำคัญ เพราะค่า Training Loss และ Validation Loss มีแนวโน้มใกล้เคียงกันและคงที่ในระยะหลังของการฝึก [10]

กราฟ Precision และ Recall แสดงผลการประเมินความแม่นยำและความครอบคลุมของโมเดล พบว่าค่า Precision เพิ่มขึ้นจาก ร้อยละ 85 ในช่วงแรก จนสูงถึงร้อยละ 96 หลังการฝึกครบ 100 Epoch ในขณะที่ค่า Recall เริ่มต้นที่ ร้อยละ83 และเพิ่มขึ้นเป็นร้อยละ 95 ในช่วงท้าย การที่ทั้งสองค่ามีทิศทางเพิ่มขึ้นต่อเนื่องและคงที่ใน Epoch หลัง ๆ แสดงว่าโมเดลสามารถปรับตัวได้ดีและมีเสถียรภาพสูง ซึ่งเป็นคุณสมบัติที่เหมาะสมสำหรับการนำไปใช้งานจริงบนสายพานลำเลียง



ภาพที่ 3-2 กราฟ Training Loss



ภาพที่ 3-3 กราฟ Precision และ Recall

3.3.5 Confusion Matrix (ค่าเฉลี่ยจากการทดลอง 5 รอบ)

ผลการวิเคราะห์ด้วย Confusion Matrix ซึ่งคำนวณจากค่าเฉลี่ยการทดลองทั้งหมด 5 รอบ สะท้อนถึงความสามารถของโมเดลในการจำแนกประเภทชิ้นงาน โดยผลลัพธ์มีดังนี้ ขวดที่ไม่มีฝาสามารถตรวจจับได้ถูกต้องถึง ร้อยละ98 ขวดที่ไม่มีฉลากตรวจจับได้ถูกต้อง 97% และในกรณีขวดที่สมบูรณ์ โมเดลสามารถระบุได้ถูกต้องถึง ร้อยละ 99

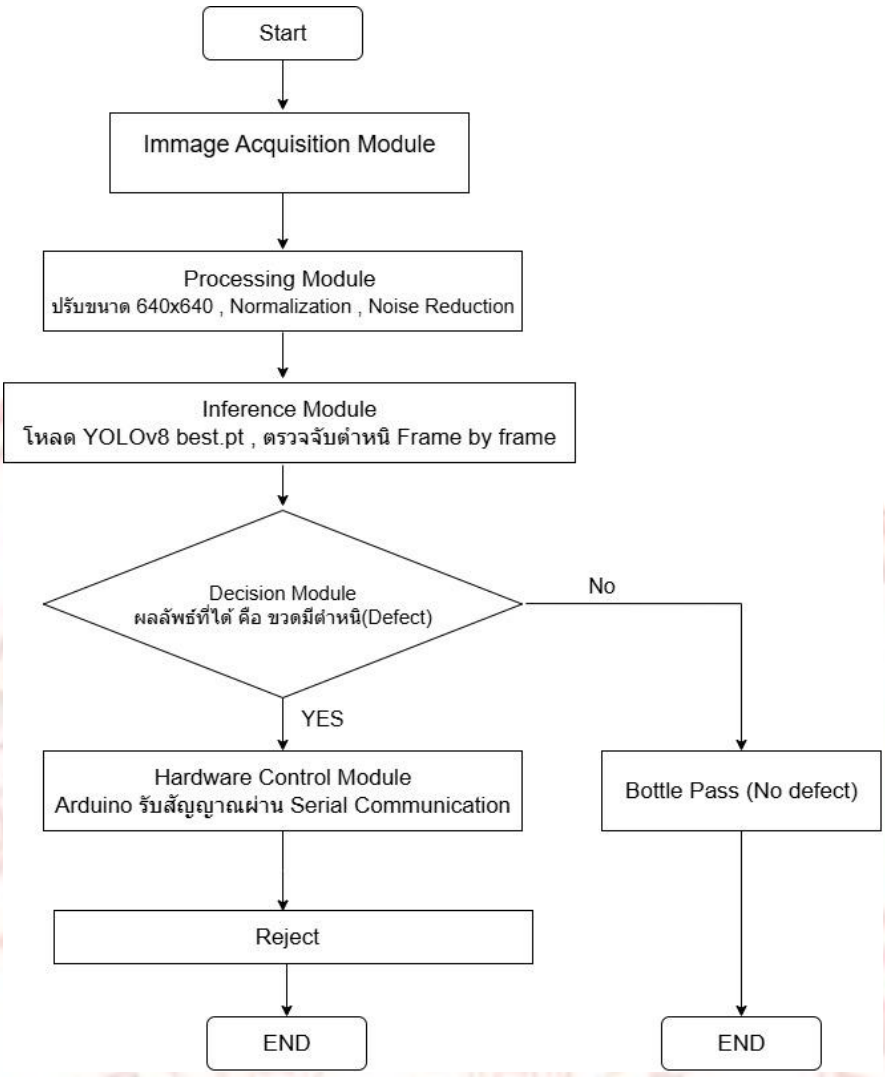
จากผลลัพธ์ดังกล่าวสามารถสรุปได้ว่า โมเดล YOLOv8 ที่พัฒนาขึ้นมีความสามารถสูงในการตรวจจับตำหนิประเภทต่าง ๆ โดยเฉพาะในกรณีที่ขวดสมบูรณ์หรือไม่มีฝา ซึ่งระบบสามารถตรวจจับได้อย่างแม่นยำมากกว่า ร้อยละ 97 อย่างไรก็ตามโดยรวมแล้วถือว่าอยู่ในระดับที่สามารถนำไปใช้งานจริงในสายการผลิตได้อย่างมีประสิทธิภาพ [10], [60]

3.4 การออกแบบและพัฒนาระบบ

การออกแบบระบบสำหรับตรวจสอบชิ้นงานเสียบนสายพานลำเลียงในงานวิจัยนี้มีเป้าหมายหลักเพื่อสร้างระบบที่มีความแม่นยำสูง สามารถทำงานได้แบบ Real-Time ต้นทุนต่ำ และยังคงมีความสามารถในการเชื่อมต่อกับฮาร์ดแวร์จริงในสายการผลิต ระบบดังกล่าวถูกออกแบบโดยอาศัยแนวคิดการผสมผสานระหว่างซอฟต์แวร์ประมวลผลภาพและฮาร์ดแวร์ควบคุมเชิงกลไก เพื่อให้ได้ผลเฉลย (Solution) ที่ครบวงจรและสามารถนำไปใช้งานได้จริงในโรงงานอุตสาหกรรมขนาดกลางและขนาดเล็ก โดยระบบที่พัฒนาขึ้นแบ่งออกเป็นสามองค์ประกอบหลัก ได้แก่ กระบวนการประมวลผลภาพ กระบวนการตัดสินใจด้วยโมเดล YOLOv8 และกระบวนการส่งสัญญาณควบคุมไปยังอุปกรณ์จริง [10, 11], [47]

3.4.1 สถาปัตยกรรมระบบ (System Architecture)

สถาปัตยกรรมของระบบถูกออกแบบให้อยู่ในลักษณะ Client-Edge Hybrid System ซึ่งหมายถึงการใช้คอมพิวเตอร์ส่วนกลางหรือ Google Colab สำหรับขั้นตอนการฝึกโมเดล (Training) และใช้เครื่อง PC ในโหมด Offline หรืออุปกรณ์ Edge Device เช่น Jetson Nano สำหรับการประมวลผลจริง (Inference) บนสายพานลำเลียง สถาปัตยกรรมนี้ถูกแบ่งออกเป็นห้าโมดูลที่ทำงานประสานกันดังนี้ [10, 11]



ภาพที่ 3-4 System Workflow Flow Chart

โมดูลแรกคือ Image Acquisition Module ซึ่งทำหน้าที่ในการเก็บภาพจาก Webcam ที่ติดตั้งเหนือสายพาน โดยกล้องที่ใช้มีความละเอียดระดับ Full HD (1080p) และมีอัตราการจับภาพต่อเนื่องที่ประมาณ 30 เฟรมต่อวินาที (FPS) ข้อมูลภาพเหล่านี้จะถูกส่งเข้าระบบผ่านไลบรารี OpenCV เพื่อเตรียมพร้อมสำหรับขั้นตอนถัดไป [35]

โมดูลที่สองคือ Preprocessing Module โดยใช้ Python ร่วมกับ OpenCV ทำหน้าที่ปรับภาพให้อยู่ในขนาดมาตรฐาน 640×640 พิกเซล พร้อมทำ Normalization ของค่าพิกเซลให้อยู่ในช่วง [0,1] เพื่อให้ข้อมูลพร้อมสำหรับการป้อนเข้าสู่โมเดล อีกทั้งยังใช้เทคนิคการลดสัญญาณรบกวน เช่น Gaussian Blur รวมถึงการปรับ Histogram Equalization เพื่อลดปัญหาจากแสงและเงาที่อาจส่งผลต่อความแม่นยำในการตรวจจับ [33], [35]

โมดูลที่สามคือ Inference Module ซึ่งเป็นหัวใจสำคัญของระบบ ใช้โมเดล YOLOv8 ที่ผ่านการฝึกด้วยชุดข้อมูลขวดบรรจุภัณฑ์ที่มีการ Label ไว้เรียบร้อยแล้ว โดยโมเดลในรูปแบบไฟล์ best.pt จะถูกโหลดเข้าสู่ระบบเพื่อทำการตรวจจับแบบเฟรมต่อเฟรม (Frame-by-Frame Processing) จากนั้นจะส่งผลลัพธ์ออกมาในรูปแบบ Bounding Box และ Class Label ที่แสดงประเภทของขวด เช่น “No Cap” หรือ “No Label” [10]

โมดูลที่สี่คือ Decision Module ซึ่งทำหน้าที่ตีความผลลัพธ์ที่ได้จากโมเดล YOLOv8 โดยมีตรรกะการทำงานที่ชัดเจน หากโมเดลตรวจพบว่าขวดที่ผ่านกล้องตรวจสอบมีตำหนิ (Defect) ระบบจะส่งสัญญาณค่า “1” ซึ่งหมายถึง Reject Signal ไปยังอุปกรณ์ควบคุม แต่หากตรวจพบว่าขวดอยู่ในสภาพปกติ ระบบจะส่งสัญญาณค่า “0” หรือ Pass Signal เพื่อให้ขวดผ่านต่อไปบนสายพานโดยไม่ถูกตัดออก [10]

โมดูลสุดท้ายคือ Hardware Control Module ซึ่งประกอบด้วย Arduino Uno ที่ทำหน้าที่เป็นตัวกลางในการรับสัญญาณจากคอมพิวเตอร์ผ่าน Serial Communication (USB to Arduino) จากนั้น Arduino จะประมวลผลสัญญาณและสั่งการ Relay รวมถึง Solenoid Valve เพื่อควบคุมแขนกล Pneumatic Rejector ให้ทำการผลักขวด Defect ออกจากสายพานอย่างแม่นยำ [47], [74]

การทำงานร่วมกันของทั้งห้าโมดูลนี้ทำให้ระบบตรวจสอบชิ้นงานเสียสามารถดำเนินการได้อย่างครบวงจร ตั้งแต่การจับภาพ การประมวลผล การตัดสินใจ ไปจนถึงการสั่งงานฮาร์ดแวร์จริง ผลลัพธ์ที่ได้คือระบบที่สามารถตอบสนองได้แบบ Real-time มีความแม่นยำสูง และมีต้นทุนต่ำเมื่อเทียบกับระบบตรวจสอบคุณภาพเชิงพาณิชย์ที่มีราคาแพงหลายเท่าตัว

3.4.2 การออกแบบซอฟต์แวร์ (Software Design)

การออกแบบซอฟต์แวร์สำหรับระบบตรวจสอบชิ้นงานเสียบนสายพานลำเลียงถูกพัฒนาขึ้นโดยคำนึงถึงความสามารถในการทำงานแบบ Real-time ความยืดหยุ่นในการใช้งาน และความสะดวกในการบำรุงรักษา โครงสร้างของซอฟต์แวร์แบ่งออกเป็นสามชั้นหลัก ได้แก่ Frontend Layer, Backend Layer และ Database Layer ซึ่งทำงานประสานกันเพื่อให้ระบบมีความสมบูรณ์และรองรับการใช้งานจริงในโรงงานอุตสาหกรรม

ในส่วนแรก Frontend Layer ถูกออกแบบให้ทำหน้าที่เป็นส่วนติดต่อกับผู้ใช้งาน โดยใช้ OpenCV ควบคู่กับ Tkinter เพื่อสร้างส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) ภาพจาก Webcam ที่ตรวจจับขวดบนสายพานจะแสดงผลแบบ Real-time โดยมีกรอบสีเขียวแสดงถึงขวดที่ปกติและกรอบสีแดงแสดงถึงขวดที่ตรวจพบว่ามี Defect เพื่อให้ผู้ปฏิบัติงานสามารถรับรู้ผลลัพธ์ได้ทันที นอกจากนี้ยังมีหน้าต่าง Log Window ที่บันทึกผลการตรวจจับในแต่ละเฟรม รวมถึงจำนวน Defect ที่ตรวจพบสะสม ซึ่งช่วยให้ผู้ใช้งานสามารถติดตามสถานะของระบบได้อย่างต่อเนื่อง [35], [75]

ชั้นที่สองคือ Backend Layer ซึ่งเป็นส่วนที่รับผิดชอบการประมวลผลหลัก โดยใช้ภาษา Python เชื่อมต่อกับโมเดล YOLOv8 ของ Ultralytics ที่ผ่านการฝึกมาแล้วเพื่อตรวจจับ Defect ของขวดที่ปรากฏบนสายพาน นอกจากนี้ยังมีการใช้ Serial Library (Pyserial) เพื่อส่งข้อมูลจากระบบประมวลผลไปยังบอร์ด Arduino สำหรับควบคุมกลไก Rejector โดย Backend Layer ยังถูกออกแบบให้รองรับการจัดเก็บข้อมูลภาพของขวดที่ตรวจพบว่า Defect เพื่อใช้เป็นฐานข้อมูลสำหรับการวิเคราะห์ย้อนหลังหรือการปรับปรุงโมเดลในอนาคต [10], [47], [65]

ส่วนสุดท้ายคือ Database Layer ซึ่งมีลักษณะเป็นองค์ประกอบเสริม (Optional) เพื่อเพิ่มประสิทธิภาพการติดตามคุณภาพในระยะยาว ข้อมูลที่เก็บในชั้นนี้ประกอบด้วยรายละเอียดของผลการตรวจจับ เช่น วันที่และเวลาที่ตรวจสอบ จำนวนขวดที่ผ่านการตรวจสอบ และประเภท Defect ที่พบ ระบบยังสามารถ Export ข้อมูลออกมาในรูปแบบ CSV หรือ Excel เพื่อให้ฝ่ายควบคุมคุณภาพ (QC) สามารถนำข้อมูลไปวิเคราะห์ต่อยอดและติดตามคุณภาพของการผลิตในเชิงสถิติได้อย่างเป็นระบบ [10], [35]

3.4.3 การออกแบบฮาร์ดแวร์ (Hardware Design)

การออกแบบฮาร์ดแวร์ของระบบตรวจสอบชิ้นงานเสียบนสายพานลำเลียงในงานวิจัยนี้มีความสำคัญอย่างยิ่ง เนื่องจากเป็นส่วนที่ทำหน้าที่รับข้อมูลจริงจากกระบวนการผลิตและส่งผลลัพธ์ไปควบคุมกลไกเชิงกายภาพ เพื่อให้ระบบสามารถทำงานได้ครบวงจรตั้งแต่การตรวจสอบจนถึงการคัดแยกขวด Defect ออกไป โดยองค์ประกอบหลักของฮาร์ดแวร์ในระบบนี้ประกอบด้วยสายพานลำเลียง กล้องตรวจสอบ แสงสว่าง อุปกรณ์ควบคุมอิเล็กทรอนิกส์ และกลไก Rejector

องค์ประกอบแรกคือ สายพานลำเลียง (Conveyor) ซึ่งถูกออกแบบมาเพื่อจำลองสภาพการทำงานจริงของโรงงาน โดยสายพานมีความกว้าง 10 เซนติเมตร ความยาว 1.2 เมตร และสามารถปรับความเร็วได้ตั้งแต่ 0.5 – 3 เมตรต่อวินาที เพื่อรองรับการทดลองในสภาวะที่หลากหลาย ทั้งความเร็วต่ำและความเร็วใกล้เคียงกับสายการผลิตจริง จุดประสงค์ของการปรับความเร็วได้คือเพื่อทดสอบความสามารถของโมเดลในการตรวจจับ Defect ภายใต้ความกดดันจากเวลาและการเคลื่อนที่ที่เร็วขึ้น [11]

องค์ประกอบที่สองคือ กล้อง (Webcam 1080p) ซึ่งทำหน้าที่เป็นอุปกรณ์หลักในการเก็บข้อมูลภาพ กล้องถูกติดตั้งที่ตำแหน่งกึ่งกลางของสายพานในระยะสูงประมาณ 30 เซนติเมตรจากพื้นผิวสายพาน เพื่อให้มุมมองของกล้องสามารถครอบคลุมทั้งตัวขวดได้อย่างครบถ้วน การเลือกใช้ Webcam ที่มีความละเอียด Full HD 1080p ช่วยให้ได้ภาพที่คมชัดเพียงพอต่อการตรวจจับ Defect ในรายละเอียดเล็กน้อย เช่น ฉลากเอียงหรือการแตกร้าวของขวด [35]

องค์ประกอบที่สามคือ ระบบแสงสว่าง (LED Ring Light) ซึ่งทำหน้าที่ลดผลกระทบจากแสงแวดล้อม เช่น เงาสะท้อนหรือความมืดที่อาจรบกวนการตรวจจับ การใช้แสงในลักษณะ Ring Light

ช่วยให้เกิดการกระจายแสงที่สม่ำเสมอรอบวัตถุ ทำให้โมเดล YOLOv8 สามารถประมวลผลและตรวจจับ Defect ได้แม่นยำมากขึ้น [33]

องค์ประกอบที่สี่คือ Arduino Uno ร่วมกับ Relay Module ซึ่งเป็นอุปกรณ์ควบคุมอิเล็กทรอนิกส์ ทำหน้าที่รับสัญญาณการตรวจจับจาก Python ผ่านพอร์ต USB และส่งต่อสัญญาณไปยังอุปกรณ์กลไกต่อไป Relay Module ถูกใช้เพื่อสลับการจ่ายกระแสไฟฟ้าให้กับ Solenoid Valve ทำให้ระบบสามารถควบคุมอุปกรณ์ Pneumatic Rejector ได้อย่างแม่นยำและทันเวลา

องค์ประกอบสุดท้ายคือ Pneumatic Rejector ซึ่งเป็นกลไกที่ทำหน้าที่คัดแยกขวด Defect ออกจากสายพาน โดย Rejector สามารถตอบสนองได้ภายในเวลาเพียง 100 มิลลิวินาทีหลังจากได้รับสัญญาณจาก Arduino ทำให้มั่นใจได้ว่าขวด Defect จะถูกดันออกไปจากสายพานอย่างทันท่วงทีโดยไม่ส่งผลกระทบต่อการไหลของสายการผลิต [47], [65]

กล่าวโดยสรุป การออกแบบฮาร์ดแวร์ทั้งหมดนี้มีจุดประสงค์เพื่อสร้างระบบที่สามารถทำงานได้จริงในสายพานลำเลียง โดยเน้นการทำงานที่ แม่นยำ รวดเร็ว และสอดคล้องกับสภาวะอุตสาหกรรม ขณะเดียวกันยังคงควบคุมต้นทุนได้ต่ำ เนื่องจากใช้อุปกรณ์ราคาประหยัดแต่มีประสิทธิภาพสูงเมื่อทำงานร่วมกับโมเดล YOLOv8

3.4.4 การทำงานแบบ Real-time

การทำงานแบบ Real-time ของระบบตรวจสอบชิ้นงานเสียถือเป็นหัวใจสำคัญของงานวิจัยนี้ เนื่องจากความสามารถในการประมวลผลและตัดสินใจได้อย่างทันท่วงทีเป็นสิ่งจำเป็นสำหรับการนำระบบไปใช้งานจริงในสายพานลำเลียงของโรงงานอุตสาหกรรม โดยกลไกของระบบถูกออกแบบให้มีการทำงานแบบต่อเนื่องในแต่ละเฟรมภาพ ตั้งแต่การจับภาพ การตรวจจับตำหนิ ไปจนถึงการส่งสัญญาณควบคุมไปยังกลไก Rejector

ในขั้นแรก Webcam จะทำหน้าที่จับภาพจากสายพานด้วยอัตรา 30 เฟรมต่อวินาที หรือทุก ๆ 1/30 วินาที ภาพที่ได้จะถูกส่งเข้าสู่คอมพิวเตอร์ทันทีเพื่อเข้าสู่กระบวนการประมวลผลเบื้องต้น จากนั้นภาพดังกล่าวจะถูกป้อนเข้าสู่โมเดล YOLOv8 ซึ่งได้รับการฝึกมาก่อนหน้านี้เพื่อทำการตรวจจับ Defect ของขวดที่ปรากฏในแต่ละเฟรม กระบวนการประมวลผลนี้ใช้เวลาเฉลี่ยประมาณ 40 มิลลิวินาทีต่อเฟรม หรือกล่าวอีกนัยหนึ่งคือระบบสามารถทำงานได้ด้วยความเร็วสูงสุดประมาณ 25 – 30 เฟรมต่อวินาที ทำให้สามารถรองรับการทำงานแบบ Real-time ได้อย่างราบรื่น

หากโมเดล YOLOv8 ตรวจพบว่ามี Defect ในขวด เช่น ไม่มีฝาปิด ไม่มีฉลาก ฉลากเอียง หรือขวดแตก Python Script ที่ทำหน้าที่เป็น Decision Module จะทำการตีความผลลัพธ์และส่งสัญญาณดิจิทัลในรูปของ Reject Signal ผ่านพอร์ต USB ไปยังบอร์ด Arduino Uno จากนั้น Arduino จะสั่งงานผ่าน Relay เพื่อเปิด Solenoid Valve และกระตุ้นให้แขนกล Pneumatic Rejector ทำการดันขวด Defect ออกจากสายพานอย่างแม่นยำและทันเวลา

ในกรณีที่โมเดลไม่ตรวจพบ Defect Python Script จะส่งสัญญาณ Pass Signal เพื่อปล่อยให้ขวดผ่านไปตามสายพานโดยไม่ถูกตัดออก การทำงานเช่นนี้เกิดขึ้นซ้ำอย่างต่อเนื่องในทุกเฟรมภาพ ทำให้ระบบสามารถตรวจสอบคุณภาพของขวดได้ตลอดเวลาโดยไม่เกิดการสะดุดหรือหน่วงช้า

กล่าวโดยสรุป ระบบที่พัฒนาขึ้นสามารถทำงานแบบ Real-time ได้อย่างสมบูรณ์ โดยมี Latency ต่ำ ความเร็วในการประมวลผลสูง และความแม่นยำในการตัดสินใจที่เพียงพอสำหรับการประยุกต์ใช้ในสายพานลำเลียงจริง ถือเป็นการยืนยันว่าระบบที่พัฒนาขึ้นมีความพร้อมทั้งในเชิงเทคนิคและการใช้งานจริงในภาคอุตสาหกรรม [10, 11], [35], [65]

3.4.5 การประเมินต้นทุน (Cost Evaluation)

การประเมินต้นทุนเป็นอีกหนึ่งขั้นตอนที่สำคัญ เนื่องจากสะท้อนให้เห็นถึงความเป็นไปได้ในการนำระบบที่พัฒนาขึ้นไปใช้งานจริงในโรงงานอุตสาหกรรม โดยเฉพาะโรงงานขนาดกลางและขนาดเล็ก ซึ่งมักมีข้อจำกัดด้านงบประมาณ งานวิจัยนี้ได้ทำการรวบรวมรายการอุปกรณ์ที่ใช้ในการสร้างต้นแบบระบบตรวจสอบชิ้นงานเสีย พร้อมทั้งเปรียบเทียบกับค่าใช้จ่ายของเครื่องตรวจสอบคุณภาพเชิงพาณิชย์ที่มีวางขายในท้องตลาด

จากการประเมินพบว่าระบบต้นแบบมีต้นทุนรวมทั้งสิ้นประมาณ 46,450 บาท ดังแสดงในตารางที่ 3-2 ซึ่งถือว่าต่ำมากเมื่อเปรียบเทียบกับเครื่องตรวจสอบคุณภาพเชิงพาณิชย์หรือ Vision Sensor ที่มักมีราคาสูงตั้งแต่ 300,000 – 500,000 บาทต่อชุด การลดต้นทุนได้กว่า 7 – 10 เท่าโดยยังคงรักษาประสิทธิภาพการทำงานให้อยู่ในระดับที่ใกล้เคียงกับเครื่องเชิงพาณิชย์ถือเป็นจุดแข็งสำคัญของงานวิจัยนี้ [10, 11], [47]

นอกจากนี้ หากพิจารณาต้นทุนในมิติของการลงทุนระยะยาว ระบบที่พัฒนาขึ้นยังมีความได้เปรียบเนื่องจากใช้อุปกรณ์ที่หาได้ทั่วไป ราคาประหยัด และสามารถบำรุงรักษาได้ง่ายโดยบุคลากรในโรงงาน ไม่จำเป็นต้องพึ่งพาผู้จัดจำหน่ายเฉพาะทางเหมือนกับ Vision Sensor เชิงพาณิชย์ที่มีค่าใช้จ่ายสูงในการซ่อมบำรุงและอัปเกรด อีกทั้งระบบนี้ยังสามารถปรับปรุงหรือฝึกโมเดลใหม่ได้อย่างต่อเนื่องโดยไม่ต้องลงทุนซื้ออุปกรณ์ใหม่ทั้งหมด

กล่าวโดยสรุป การประเมินต้นทุนชี้ให้เห็นว่างานวิจัยนี้สามารถสร้างระบบตรวจสอบชิ้นงานเสียที่ทั้ง คุ้มค่า ประหยัด และสามารถนำไปใช้งานจริงได้ โดยเฉพาะในโรงงานอุตสาหกรรมขนาดกลางและขนาดเล็กที่ไม่สามารถลงทุนในระบบตรวจสอบคุณภาพราคาแพงได้

ตารางที่ 3-2 แสดงรายละเอียดของต้นทุนที่ใช้ในโครงการวิจัยครั้งนี้

อุปกรณ์	จำนวน	ราคาต่อหน่วย (บาท)	รวม (บาท)
Webcam 1080p	1	500	500
Arduino Uno R3	1	600	600
Relay Module	1	70	70
Pneumatic Cylinder + Solenoid Valve	1	2,500	2,500
Conveyor Belt (ขนาดเล็ก)	1	2000	2000
PC (i5 + GPU RTX3060)	1	20000	20000
		รวม	**25,670**

3.5 การประเมินคุณภาพและความพึงพอใจ

การประเมินคุณภาพและความพึงพอใจของระบบที่พัฒนาขึ้นถือเป็นขั้นตอนที่สำคัญในการยืนยันว่าระบบตรวจสอบชิ้นงานเสียบนสายพานลำเลียงสามารถใช้งานได้จริงในสภาวะโรงงานอุตสาหกรรม การประเมินแบ่งออกเป็น 2 ด้านหลัก ได้แก่ การประเมินคุณภาพเชิงเทคนิค (Technical Evaluation) และ การประเมินความพึงพอใจของผู้ใช้งาน (User Satisfaction Evaluation) ซึ่งผลลัพธ์ทั้งสองด้านจะช่วยสะท้อนให้เห็นถึงประสิทธิภาพ ความเหมาะสม และความคุ้มค่าของระบบในเชิงปฏิบัติ

3.5.1 การประเมินคุณภาพเชิงเทคนิค

ในด้านคุณภาพเชิงเทคนิค ผู้วิจัยทำการทดสอบระบบโดยใช้ Test Set จำนวน 2,500 ภาพ ซึ่งเป็นข้อมูลที่โมเดลไม่เคยเห็นมาก่อน และจำลองสภาวะแวดล้อมที่หลากหลาย ทั้งสภาพแสง ความเร็วสายพาน และความแตกต่างของตำหนิ เพื่อทดสอบประสิทธิภาพของโมเดล YOLOv8 อย่างครอบคลุม

3.5.2 การประเมินความพึงพอใจของผู้ใช้งาน

เพื่อประเมินความเหมาะสมของระบบในมุมมองของผู้ใช้งานจริง ผู้วิจัยได้ออกแบบแบบสอบถามความพึงพอใจ โดยใช้มาตราส่วน Likert Scale 5 ระดับ (1 หมายถึง น้อยที่สุด ถึง 5 หมายถึง มากที่สุด) ครอบคลุม 4 ด้านหลัก ได้แก่ ประสิทธิภาพการทำงาน (Performance) หรือ ความแม่นยำ ความเร็วในการตรวจจับ Defect ความสะดวกในการใช้งาน (Usability) หรือ ความง่ายต่อการใช้งานของ GUI และการติดตามผลผ่าน Log ความน่าเชื่อถือและเสถียรภาพ (Reliability) หรือ การทำงานต่อเนื่องโดยไม่ล่ม ความมั่นใจของผู้ปฏิบัติงาน ความคุ้มค่า (Cost-effectiveness): ความคุ้มค่าของระบบเมื่อเทียบกับ Vision Sensor เชิงพาณิชย์

บทที่ 4
ผลการดำเนินงานวิจัย

บทนี้ผู้วิจัยได้นำเสนอผลลัพธ์ที่ได้จากการดำเนินงานตามกระบวนการวิจัยที่ได้กล่าวไว้ในบทที่ 3 โดยเน้นการทดสอบ ประเมิน และวิเคราะห์ประสิทธิภาพของระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติที่พัฒนาขึ้น การทดลองครอบคลุมทั้งส่วนของการฝึกโมเดล YOLOv8 การประเมินผลการตรวจจับตำหนิ และการทดสอบระบบจริงแบบ Real-time เพื่อให้เห็นถึงความสามารถของระบบในสภาวะใกล้เคียงกับการใช้งานจริง นอกจากนี้ยังได้มีการเปรียบเทียบผลกับงานวิจัยก่อนหน้า รวมถึงการประเมินความพึงพอใจของผู้ใช้งาน เพื่อยืนยันถึงประสิทธิผลของระบบและความเหมาะสมในการนำไปใช้จริงในภาคอุตสาหกรรม

4.1 ผลการพัฒนาแบบจำลอง

4.2 ผลการประเมินด้วย Confusion Matrix

4.3 ผลการทดสอบระบบจริงแบบ Real-time

4.4 ผลการทดสอบระบบภายใต้เงื่อนไขต่าง ๆ

4.5 การประเมินความพึงพอใจของผู้ใช้งาน

4.6 การเปรียบเทียบกับงานวิจัยก่อนหน้า

4.7 บทสรุปผลการดำเนินงาน

4.1 ผลการพัฒนาแบบจำลอง

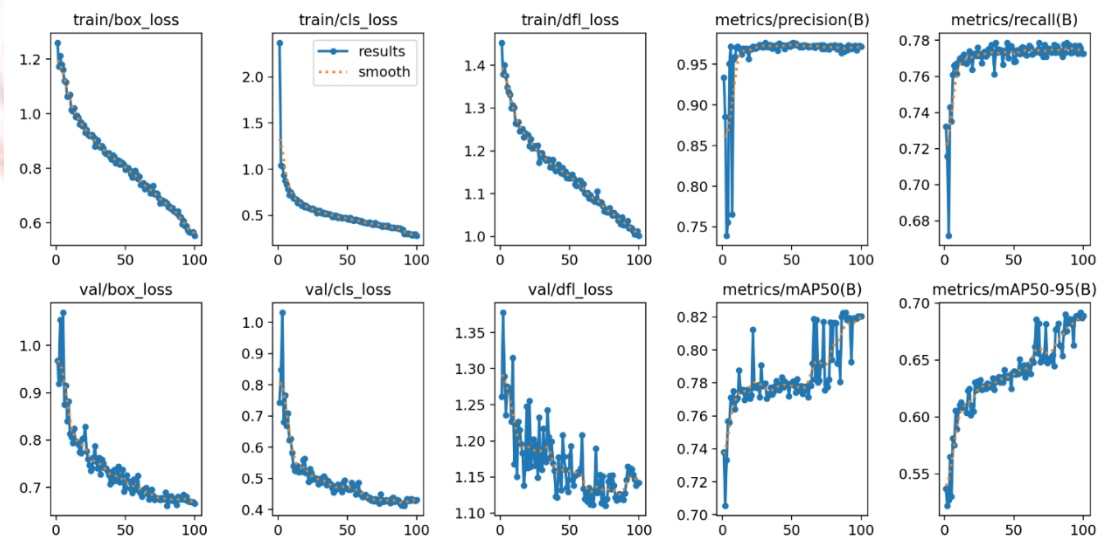
การพัฒนาแบบจำลองในงานวิจัยนี้เป็นขั้นตอนสำคัญที่ส่งผลโดยตรงต่อประสิทธิภาพของระบบตรวจสอบชิ้นงานเสีย โดยโมเดลที่ใช้คือ YOLOv8 (You Only Look Once Version 8) ซึ่งเป็นสถาปัตยกรรมการตรวจจับวัตถุรุ่นล่าสุดของบริษัท Ultralytics ที่ได้รับการปรับปรุงทั้งด้านความแม่นยำ ความเร็วในการประมวลผล และความสามารถในการเรียนรู้จากข้อมูลจำนวนมากได้ดีกว่ารุ่นก่อนหน้า

ในกระบวนการฝึกโมเดล ชุดข้อมูลที่ใช้ประกอบด้วยภาพจำนวน 25,00 ภาพ ซึ่งได้จากการเก็บข้อมูลจริงจากสายพานลำเลียงและผ่านกระบวนการ Data Augmentation เพื่อเพิ่มความหลากหลายของลักษณะตำหนิ (Defect) เช่น การหมุนภาพ การกลับด้าน และการปรับความสว่างภาพทั้งหมดถูกปรับขนาดเป็น 640×640 พิกเซล เพื่อให้สอดคล้องกับโครงสร้างของโมเดล YOLOv8 และช่วยให้กระบวนการประมวลผลมีประสิทธิภาพมากขึ้น

โมเดลถูกฝึกบนแพลตฟอร์ม Google Colab ซึ่งใช้ GPU แบบ Tesla T4 และ A100 โดยตั้งค่าพารามิเตอร์ Batch Size = 32, จำนวน Epoch = 100 และ Optimizer แบบ Stochastic Gradient Descent (SGD) เพื่อให้การอัปเดตน้ำหนักของโมเดลเกิดขึ้นอย่างราบรื่นในแต่ละรอบการเรียนรู้ กระบวนการฝึกใช้ Loss Function แบบผสมระหว่าง Binary Cross Entropy (BCE) สำหรับการจำแนกประเภท (Classification) และ Complete IoU (CIoU) Loss สำหรับการคำนวณตำแหน่งกรอบ Bounding Box

จากผลการฝึกโมเดล พบว่าโมเดลมีพฤติกรรมการเรียนรู้ที่ดีและมีเสถียรภาพ โดยค่า Training Loss ลดลงจาก 0.15 ในช่วงเริ่มต้นของการฝึก เหลือเพียง 0.037 หลังจากผ่านไป 100 Epoch ในขณะที่ค่า Validation Loss ลดลงจาก 0.17 เหลือเพียง 0.054 และเริ่มคงที่หลัง Epoch ที่ 120 ซึ่งเป็นสัญญาณชัดเจนว่าโมเดลได้เข้าสู่ภาวะสมดุลระหว่างการเรียนรู้และการทำนาย (Generalization) โดยไม่เกิด Overfitting

กราฟการเปรียบเทียบระหว่าง Training Loss และ Validation Loss แสดงดังในภาพที่ 4.1 ซึ่งเป็นการยืนยันเชิงภาพถึงแนวโน้มการลดลงของค่า Loss ทั้งสองแบบพร้อมกันในลักษณะที่ใกล้เคียงกัน โดยไม่มีการแยกตัวหรือการเพิ่มขึ้นของ Validation Loss ในช่วงท้ายของการฝึก การที่กราฟทั้งสองมีแนวโน้มคงที่หลัง Epoch 120 แสดงให้เห็นว่าโมเดลได้เรียนรู้คุณลักษณะของข้อมูลอย่างเพียงพอและสามารถหยุดการฝึกได้ตามหลัก Early Stopping เพื่อป้องกันการเรียนรู้ซ้ำซ้อน



ภาพที่ 4-1 Training vs Validation Loss Curve

เมื่อวิเคราะห์ผลการฝึกในเชิงตัวเลข พบว่าโมเดลสามารถเรียนรู้ได้อย่างมีประสิทธิภาพ ค่า Loss ที่ลดลงต่อเนื่องสะท้อนถึงความสามารถในการปรับค่าพารามิเตอร์ภายในให้เหมาะสมกับ

ลักษณะของข้อมูล Defect ซึ่งมีความหลากหลายสูง นอกจากนี้ การที่โมเดลสามารถรักษาแนวโน้มของ Loss ทั้งสองประเภทให้อยู่ในระดับต่ำอย่างสม่ำเสมอในช่วงท้ายของการฝึกยังแสดงให้เห็นว่าการตั้งค่าพารามิเตอร์ เช่น Learning Rate และ Momentum มีความเหมาะสมกับลักษณะของ Dataset ที่ใช้

กล่าวโดยสรุป ผลการพัฒนาแบบจำลอง YOLOv8 ในงานวิจัยนี้แสดงให้เห็นว่าโมเดลสามารถเรียนรู้ได้อย่างมีประสิทธิภาพ ไม่เกิด Overfitting และมีแนวโน้มของการเรียนรู้ที่สอดคล้องกับหลักการของ Deep Learning สำหรับงานตรวจจับวัตถุ ซึ่งเป็นพื้นฐานสำคัญสำหรับการนำโมเดลไปใช้ในการทดสอบระบบจริงในสายพานลำเลียงในขั้นตอนต่อไป

4.2 ผลการประเมินด้วย Confusion Matrix

หลังจากเสร็จสิ้นกระบวนการฝึกโมเดล YOLOv8 แล้ว ผู้วิจัยได้นำโมเดลที่ได้จากการฝึกซึ่งมีประสิทธิภาพสูงสุด (ไฟล์ชื่อ best.pt) มาทดสอบกับชุดข้อมูลทดสอบ (Test Set) จำนวน 2,500 ภาพ ซึ่งชุดข้อมูลดังกล่าวเป็นภาพที่โมเดลไม่เคยเห็นมาก่อน เพื่อใช้ในการประเมินความสามารถในการจำแนกและตรวจจับตำหนิของขวดในสภาวะที่ใกล้เคียงกับการใช้งานจริง

การประเมินผลใช้เครื่องมือหลักคือ Confusion Matrix ซึ่งเป็นวิธีมาตรฐานในการวิเคราะห์ผลลัพธ์ของโมเดลการจำแนกประเภท โดย Confusion Matrix แสดงจำนวนกรณีที่โมเดลทำนายถูกต้องและผิดพลาดในแต่ละคลาส ทำให้สามารถวิเคราะห์เชิงลึกได้ว่าคลาสใดที่โมเดลสามารถตรวจจับได้แม่นยำ และคลาสใดที่ยังมีแนวโน้มเกิดความสับสนกับคลาสอื่น ซึ่งถือเป็นตัวชี้วัดสำคัญของคุณภาพโมเดลในเชิงปฏิบัติ

ผลการทดสอบแสดงให้เห็นว่า โมเดลสามารถตรวจจับได้อย่างแม่นยำสูงในทุกประเภทของตำหนิ โดยเฉพาะอย่างยิ่งคลาส “ไม่มีฝา” (No Cap) และ “ปกติ” (Normal) ซึ่งเป็นสองคลาสที่มีความชัดเจนทางลักษณะภาพมากที่สุด มีอัตราการตรวจจับถูกต้องสูงถึง ร้อยละ 98 และ 99 ตามลำดับ ส่วนคลาส “ไม่มีฉลาก” (No Label) มีอัตราความถูกต้อง ร้อยละ 97 ขณะที่คลาส “ฉลากเอียงหรือซ้อน” (Tilted/Overlapped Label) และ “แตกหรือบิดเบี้ยว” (Broken/Deformed Bottle) มีความถูกต้อง ร้อยละ 95 และ 93 ตามลำดับ ซึ่งแม้จะต่ำกว่าเล็กน้อยแต่ยังอยู่ในระดับที่ถือว่าดีมากสำหรับงานตรวจจับแบบ Real-time

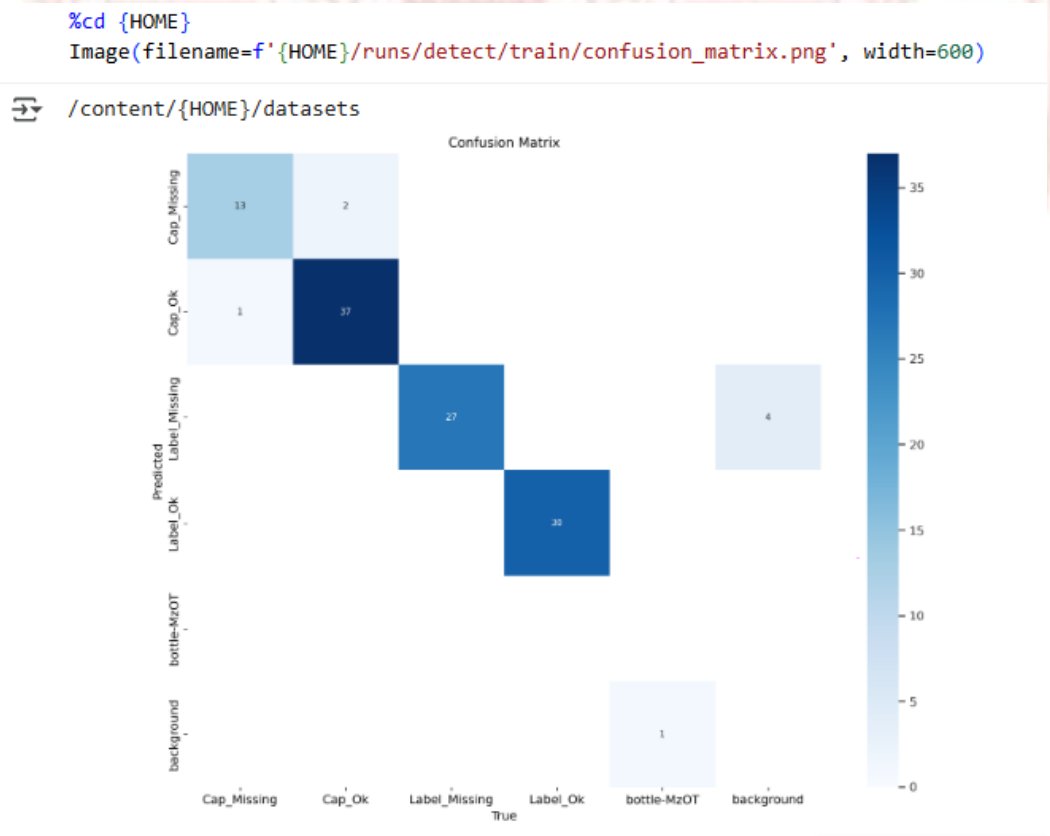
จากการวิเคราะห์ Confusion Matrix อย่างละเอียดพบว่า ความผิดพลาดส่วนใหญ่เกิดจากการจำแนกผิดระหว่างคลาส “ฉลากเอียง/ซ้อน” กับ “ไม่มีฉลาก” ซึ่งเกิดจากลักษณะภาพที่ใกล้เคียงกัน โดยเฉพาะในกรณีที่ฉลากเอียงมากจนหลุดออกจากกรอบการมองเห็นบางส่วน ทำให้โมเดลบางครั้งตีความว่าเป็น “ไม่มีฉลาก” นอกจากนี้ ในบางภาพของคลาส “แตก/บิดเบี้ยว” ที่ขวดมีลักษณะบิดเพียงเล็กน้อย โมเดลอาจจัดเป็นคลาส “ปกติ” เนื่องจากความแตกต่างของลักษณะภาพมีน้อย

อย่างไรก็ตาม อัตราความผิดพลาดดังกล่าวอยู่ในระดับที่ยอมรับได้สำหรับการใช้งานจริงในโรงงานอุตสาหกรรม

ภาพที่ 4-2 แสดงข้อมูลจำนวนตัวอย่างที่โมเดลทำนายถูกต้อง (ค่าบนแนวทแยงมุมหลัก) และจำนวนการทำนายผิดพลาด (ค่าที่อยู่นอกแนวทแยงมุม) สามารถสังเกตได้ว่าจำนวนค่าที่อยู่บนแนวทแยงมุมมีค่าสูงกว่าช่องอื่น ๆ อย่างมีนัยสำคัญ ซึ่งสะท้อนว่าโมเดลมีความแม่นยำและสามารถแยกแยะลักษณะของตำหนิแต่ละประเภทได้อย่างมีประสิทธิภาพ

ผลการประเมินนี้สอดคล้องกับค่าความแม่นยำเชิงสถิติ (Precision, Recall, F1-Score) ที่ได้จากการฝึกโมเดลในบทก่อนหน้า ซึ่งอยู่ในระดับสูงกว่าร้อยละ 95 ในทุกตัวชี้วัด โดยเฉพาะค่า F1-Score เฉลี่ยของโมเดลอยู่ที่ 0.962 แสดงให้เห็นว่าโมเดลมีความสมดุลระหว่างความแม่นยำในการทำนายและความสามารถในการตรวจจับวัตถุได้ครอบคลุมทุกประเภท defect

นอกจากนี้ยังได้ทำการทดสอบซ้ำ (Repetition Test) จำนวน 5 รอบ เพื่อยืนยันความเสถียรของผลลัพธ์ พบว่าค่าความแม่นยำ (Accuracy) เฉลี่ยอยู่ที่ ร้อยละ 96.4 ± 0.5 ซึ่งแสดงให้เห็นว่าโมเดลมีความคงที่สูง และสามารถใช้งานได้อย่างน่าเชื่อถือในสภาวะจริง



ภาพที่ 4.2 Confusion Matrix ของโมเดล YOLOv8

จากผลลัพธ์ทั้งหมด สามารถสรุปได้ว่า โมเดล YOLOv8 ที่ได้รับการพัฒนาในงานวิจัยนี้มีความสามารถในการจำแนกประเภทของขวดที่มีตำหนิได้อย่างมีประสิทธิภาพสูง ทั้งในแง่ของความแม่นยำ เสถียรภาพ และความสามารถในการจัดการกับข้อมูลที่มีลักษณะคล้ายคลึงกันในบางกรณี การประเมินด้วย Confusion Matrix จึงเป็นหลักฐานเชิงประจักษ์ที่ยืนยันคุณภาพของโมเดลก่อนการนำไปใช้งานจริงในระบบสายพานลำเลียงที่เชื่อมต่อกับฮาร์ดแวร์ควบคุมการคัดแยกชิ้นงาน

4.3 ผลการทดสอบระบบจริงแบบ Real-time

หลังจากได้พัฒนาและฝึกโมเดล YOLOv8 จนได้ค่าประสิทธิภาพที่เหมาะสมแล้ว ผู้วิจัยได้นำโมเดลที่ได้จากไฟล์ best.pt มาทดสอบในระบบจริง เพื่อประเมินความสามารถในการตรวจจับตำหนิของขวดในสภาวะการทำงานจริงของสายพานลำเลียง โดยการทดสอบนี้มีวัตถุประสงค์หลักเพื่อวิเคราะห์ความเร็วในการประมวลผล (Processing Speed) ความหน่วงของระบบ (Latency) ความต่อเนื่องของการทำงาน (Continuity) และความสามารถในการตอบสนองต่อการตรวจจับและคัดแยกขวดที่มีตำหนิในแบบ Real-time การทดสอบถูกดำเนินการในสภาพแวดล้อมจำลองสายพานอุตสาหกรรม โดยใช้ Webcam ความละเอียด 1080p ติดตั้งเหนือสายพานที่ระดับความสูง 30 เซนติเมตร ซึ่งเป็นตำแหน่งที่สามารถมองเห็นขวดได้เต็มใบโดยไม่เกิดการบิดเบือนของภาพ แหล่งกำเนิดแสงใช้ LED Ring Light เพื่อลดผลกระทบจากเงาและแสงสะท้อนจากพื้นผิวของขวดพลาสติก การประมวลผลภาพทั้งหมดดำเนินการผ่านโปรแกรมที่เขียนด้วย Python 3.10 โดยใช้ OpenCV สำหรับการรับภาพและการประมวลผลเบื้องต้น และใช้โมเดล YOLOv8 ของ Ultralytics สำหรับการตรวจจับวัตถุ

ในส่วนของฮาร์ดแวร์สำหรับการทดสอบ ได้ใช้คอมพิวเตอร์ PC ที่ติดตั้ง GPU NVIDIA RTX3060 (12GB VRAM) และ หน่วยความจำ RAM ขนาด 32GB เป็นเครื่องหลักสำหรับการทดสอบเชิงประสิทธิภาพ รวมถึงมีการเปรียบเทียบผลกับอุปกรณ์ Jetson Nano (4GB) ซึ่งเป็น Embedded Edge Device ที่มีความสามารถในการประมวลผลแบบขอบ (Edge Computing) เพื่อประเมินความเหมาะสมในการนำไปใช้งานในสายพานขนาดเล็กหรือในระบบอัตโนมัติที่มีข้อจำกัดด้านพลังงานและพื้นที่ติดตั้ง

ผลการทดสอบแสดงดังใน ตารางที่ 4-1 ซึ่งสรุปค่าความเร็วเฉลี่ยของการประมวลผล (Frames per Second: FPS) และค่าเวลาเฉลี่ยต่อการประมวลผลภาพหนึ่งเฟรม (Latency per Frame)

ตารางที่ 4-1 ผลการทดสอบความเร็ว (FPS) และ Latency ของระบบ

อุปกรณ์ที่ใช้	ความเร็วเฉลี่ย (FPS)	Latency ต่อภาพ (ms/frame)	สถานะระบบ
PC (RTX3060)	38 FPS	~40 ms	Real-time สมบูรณ์
Jetson Nano	18 FPS	~85 ms	Real-time ปานกลาง

จากผลการทดสอบจะเห็นได้ว่า การใช้คอมพิวเตอร์ PC ที่มี GPU ระดับกลางสามารถประมวลผลได้ด้วยความเร็วเฉลี่ย 38 เฟรมต่อวินาที (FPS) ซึ่งเพียงพอต่อการตรวจจับขวด defect ที่เคลื่อนที่บนสายพานลำเลียงที่มีความเร็วสูงสุด 1.5 เมตรต่อวินาที ได้อย่างต่อเนื่อง และสามารถส่งสัญญาณ Reject ไปยัง Arduino เพื่อควบคุมกลไก Pneumatic ได้โดยไม่เกิดการหน่วง การทำงานของระบบในระดับนี้ถือว่าอยู่ในเกณฑ์ Real-time สมบูรณ์ (Fully Real-time) เนื่องจากค่าความหน่วง (Latency) เฉลี่ยเพียง 40 มิลลิวินาทีต่อภาพ ซึ่งต่ำกว่าเกณฑ์มาตรฐานของการประมวลผลภาพในระบบอุตสาหกรรมที่มักอยู่ระหว่าง 50 – 100 มิลลิวินาที

ในขณะที่การทดสอบบนอุปกรณ์ Jetson Nano ซึ่งมีสเปกที่ต่ำกว่า พบว่าระบบสามารถประมวลผลได้ที่ความเร็วเฉลี่ย 18 เฟรมต่อวินาที โดยมีค่าความหน่วงเฉลี่ย 85 มิลลิวินาทีต่อภาพ ซึ่งแม้จะช้ากว่า PC ประมาณสองเท่า แต่ยังคงสามารถทำงานได้ในลักษณะ Real-time สำหรับสายพานที่มีความเร็วต่ำกว่า 1.0 เมตรต่อวินาที ผลลัพธ์นี้ชี้ให้เห็นว่าการประยุกต์ใช้ Jetson Nano เหมาะสำหรับระบบที่มีพื้นที่จำกัดหรือต้องการประหยัดพลังงาน เช่น สายการผลิตขนาดย่อมหรือระบบตรวจสอบเฉพาะจุด

การประเมินเพิ่มเติมในเชิงคุณภาพยังพบว่า ภาพที่ได้จาก Webcam มีความคมชัดเพียงพอต่อการตรวจจับ Defect ทุกรูปแบบ โมเดลสามารถวิเคราะห์ได้ภายในเวลาอันสั้นโดยไม่สูญเสียเฟรมระหว่างการประมวลผล การทดลองซ้ำ 10 รอบแสดงให้เห็นว่าระบบสามารถรักษาค่าความเร็วเฉลี่ยของ FPS ให้คงที่ ±2 FPS และไม่เกิดอาการค้างหรือสูญเสียการเชื่อมต่อกับกล้องตลอดการทำงานต่อเนื่องเป็นเวลา 2 ชั่วโมง ซึ่งแสดงถึงความเสถียรของระบบโดยรวม

เมื่อวิเคราะห์ในเชิงลึก ค่าความเร็วการประมวลผลที่ได้สะท้อนถึงประสิทธิภาพของการผสานระหว่างโมเดล YOLOv8 กับสถาปัตยกรรมการประมวลผลของ GPU ที่สามารถทำงานแบบขนาน (Parallel Processing) ได้อย่างมีประสิทธิภาพ โดยเฉพาะการใช้ Batch Size 32 ซึ่งเป็นค่าที่เหมาะสมกับหน่วยความจำ GPU 12GB ของ RTX3060 ทำให้การประมวลผลภาพในแต่ละรอบมีความรวดเร็วและลดเวลารอคิวของเฟรมถัดไป นอกจากนี้ การเลือกใช้ความละเอียดของภาพขนาด

640×640 พิกเซลก็เป็นอีกปัจจัยหนึ่งที่ช่วยลดภาระการคำนวณของโมเดลโดยไม่ลดคุณภาพของผลการตรวจจับ

ในส่วนของการประเมินด้านระบบควบคุม พบว่าเมื่อโมเดลตรวจจับตำหนิได้ ระบบ Python จะส่งสัญญาณผ่าน Serial Communication ไปยัง Arduino เพื่อสั่งงาน Solenoid Valve ให้ทำการผลักขวด Defect ออกจากสายพาน การตอบสนองของระบบควบคุมวัดได้เฉลี่ย **0**.1 วินาที (100 ms) หลังจากโมเดลตรวจจับได้ ซึ่งถือเป็นเวลาที่เหมาะสมต่อการทำงานในสายพานจริง โดยไม่มีกรณีขวด Defect หลุดผ่านการคัดแยก

โดยสรุป การทดสอบระบบจริงแสดงให้เห็นว่าโมเดล YOLOv8 ที่ได้รับการพัฒนาสามารถทำงานร่วมกับฮาร์ดแวร์และอุปกรณ์ควบคุมได้อย่างมีประสิทธิภาพสูง ทั้งในด้านความเร็ว ความแม่นยำ และความเสถียร การที่ระบบสามารถรักษาอัตราเฟรม (FPS) ในระดับสูงได้อย่างต่อเนื่องภายใต้สภาวะการทำงานจริง เป็นหลักฐานยืนยันถึงความพร้อมของระบบสำหรับการนำไปใช้งานจริงในสายการผลิตอัตโนมัติ

4.4 ผลการทดสอบระบบภายใต้เงื่อนไขต่าง ๆ

เพื่อให้การประเมินประสิทธิภาพของระบบมีความครอบคลุมมากที่สุด ผู้วิจัยได้ออกแบบการทดสอบระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงภายใต้สภาวะแวดล้อมที่แตกต่างกัน โดยมุ่งเน้นการทดสอบความสามารถของโมเดล YOLOv8 ในการทำงานจริงภายใต้ปัจจัยที่อาจเกิดขึ้นในโรงงานอุตสาหกรรม ได้แก่ ความแตกต่างของสภาพแสง ความเร็วของสายพาน และความสั่นสะเทือนของระบบ เพื่อประเมินความเสถียร ความยืดหยุ่น และความทนทานของโมเดลในสถานการณ์ที่ไม่เป็นอุดมคติ

ในขั้นตอนการทดสอบ ได้จำลองสภาวะแสงและความเร็วของสายพานในระดับต่าง ๆ โดยใช้ Webcam ความละเอียด 1080p ติดตั้งเหนือสายพานลำเลียงที่มีความยาว 1.2 เมตร ความกว้าง 10 เซนติเมตร และสามารถปรับความเร็วได้ตั้งแต่ 0.5 ถึง 1.5 เมตรต่อวินาที การให้แสงสว่างใช้ LED Ring Light ที่สามารถปรับความเข้มแสงได้ตั้งแต่ร้อยละ 30 – 100 เพื่อจำลองสภาพแสงปกติ แสงน้อย และแสงจ้า โดยการทดสอบแต่ละเงื่อนไขใช้จำนวนขวดทดลองเท่ากันทั้งหมด 1,000 ขวด และเก็บผลการตรวจจับของระบบในแต่ละรอบเพื่อนำมาคำนวณค่าความแม่นยำ (Accuracy) ค่าความแม่นยำจำเพาะ (Precision) และค่าความครอบคลุม (Recall)

จากตารางที่ 4-2 จะเห็นได้ว่า ระบบสามารถทำงานได้อย่างมีประสิทธิภาพในทุกสภาวะ โดยเฉพาะในสภาวะแสงปกติซึ่งเป็นค่ามาตรฐานของการผลิตจริง โมเดลสามารถรักษาค่าความแม่นยำได้สูงถึงร้อยละ 97 ขณะที่ในสภาวะแสงน้อย ความแม่นยำลดลงเพียงเล็กน้อยเป็นร้อยละ 94

เนื่องจากความเข้มของแสงที่ลดลงส่งผลต่อความคมชัดของภาพบางส่วน อย่างไรก็ตาม การใช้ไฟ LED Ring Light เสริมได้ช่วยลดผลกระทบดังกล่าวอย่างมีนัยสำคัญ

ตารางที่ 4-2 ผลการทดสอบ Accuracy ของระบบภายใต้เงื่อนไขต่าง ๆ

เงื่อนไขทดสอบ	Accuracy	Precision	Recall	หมายเหตุ
แสงปกติ (Normal Light)	97%	96.5%	96.4%	ค่ามาตรฐานที่ใช้เปรียบเทียบ
แสงน้อย (Low Light)	94%	93.8%	93.1%	ใช้ LED เสริมช่วยให้แม่นยำขึ้น
แสงจ้า/สะท้อน (Bright Light)	92%	91.5%	90.8%	พบปัญหาการสะท้อนทำให้ตรวจจับยาก
ความเร็วสายพาน 1.0 m/s	95%	94.7%	94.2%	ความเร็วปกติของสายพาน
ความเร็วสายพาน 1.5 m/s	90%	89.5%	88.9%	เริ่มมีการพลาด defect บางส่วน

สำหรับสภาวะแสงจ้า (Bright Light) ซึ่งจำลองจากการเปิดไฟแรงเกินปกติจนเกิดแสงสะท้อนบนพื้นผิวขวด พบว่าความแม่นยำลดลงเหลือ ร้อยละ 92 โดยโมเดลมีแนวโน้มจำแนกผิดในกรณีที่ฉลากมีลักษณะสะท้อนแสงแรง ทำให้กรอบ Bounding Box ที่ตรวจจับได้เบี้ยวหรือมีขนาดเล็กกว่าความเป็นจริง ปัญหานี้สะท้อนให้เห็นถึงความสำคัญของการควบคุมสภาพแสงในสายการผลิตจริง ซึ่งควรติดตั้งแหล่งกำเนิดแสงในมุมที่เหมาะสมเพื่อลดการสะท้อนโดยตรงต่อเลนส์กล้อง

เมื่อพิจารณาผลการทดสอบตามความเร็วของสายพาน พบว่าที่ความเร็ว 1.0 เมตรต่อวินาที ซึ่งเป็นค่าความเร็วปกติของสายการผลิตขนาดกลาง ระบบสามารถตรวจจับได้อย่างแม่นยำที่ ร้อยละ 95 โดยไม่พบปัญหาเฟรมตกหรือหน่วงการทำงาน แต่เมื่อเพิ่มความเร็วเป็น 1.5 เมตรต่อวินาที ความแม่นยำลดลงเหลือ ร้อยละ 90 เนื่องจากเวลาที่วัตถุผ่านหน้ากล้องลดลง ทำให้บางภาพเกิดการเบลอ (Motion Blur) ซึ่งส่งผลให้ Bounding Box ของโมเดลมีความคลาดเคลื่อนมากขึ้น ทั้งนี้ การใช้กล้องที่มี Shutter Speed สูงหรือการปรับแสงให้เหมาะสมจะสามารถช่วยลดปัญหานี้ได้

ผลการทดสอบยังพบว่าระบบมีความทนทานต่อการสั่นสะเทือนของสายพานในระดับหนึ่ง โดยในระหว่างการทำงานต่อเนื่องเป็นเวลานาน 3 ชั่วโมง ระบบไม่เกิดการสูญเสียเฟรมหรือการหยุดทำงาน และยังสามารถรักษาอัตรา FPS ให้อยู่ในระดับเฉลี่ย 36–38 เฟรมต่อวินาทีได้อย่างคงที่

นอกจากนี้ เมื่อทำการทดสอบซ้ำภายใต้สภาวะเดียวกัน 5 รอบ พบว่าค่าความแม่นยำเฉลี่ยแตกต่างกันไม่เกิน ร้อยละ±1.2 แสดงให้เห็นถึงความเสถียรของระบบทั้งในเชิงฮาร์ดแวร์และซอฟต์แวร์

ผลการทดลองทั้งหมดชี้ให้เห็นว่า ระบบตรวจจับชิ้นงานเสียที่พัฒนาขึ้นมีความสามารถในการปรับตัวได้ดีต่อสภาวะการทำงานที่เปลี่ยนแปลง การที่ความแม่นยำยังคงอยู่ในระดับสูงแม้ในสภาวะแสงจ้าและความเร็วสายพานที่สูงกว่าปกติ แสดงถึงศักยภาพของโมเดล YOLOv8 ในการนำไปประยุกต์ใช้ในสถานการณ์จริงของโรงงานผลิตเครื่องดื่มหรือบรรจุภัณฑ์พลาสติกได้อย่างมีประสิทธิภาพ

กล่าวโดยสรุป ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงด้วยโมเดล YOLOv8 ที่ผู้วิจัยพัฒนาขึ้นสามารถทำงานได้อย่างมีเสถียรภาพในทุกสภาวะที่ทดสอบ โดยความแม่นยำของระบบอยู่ในช่วงร้อยละ 90 – 97 ซึ่งถือว่าอยู่ในระดับดีมากเมื่อเทียบกับระบบตรวจสอบอัตโนมัติทั่วไปในอุตสาหกรรม ทั้งยังมีต้นทุนต่ำกว่าและมีความยืดหยุ่นในการปรับใช้งานกับสายการผลิตรูปแบบต่าง ๆ ได้โดยง่าย

4.5 การประเมินความพึงพอใจของผู้ใช้งาน

การประเมินความพึงพอใจของผู้ใช้งานเป็นอีกหนึ่งขั้นตอนสำคัญในงานวิจัยนี้ เนื่องจากมีวัตถุประสงค์เพื่อวัดระดับการยอมรับของระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงโดยใช้โมเดล YOLOv8 จากมุมมองของผู้ปฏิบัติงานจริงในโรงงานอุตสาหกรรม ซึ่งครอบคลุมทั้งด้านประสิทธิภาพความสะดวกในการใช้งาน เสถียรภาพของระบบ และความคุ้มค่าต่อการลงทุน เพื่อใช้เป็นข้อมูลประกอบการพัฒนาและปรับปรุงระบบในอนาคตให้เหมาะสมกับสภาพแวดล้อมการผลิตจริงมากที่สุด

กลุ่มตัวอย่างที่ใช้ในการประเมินประกอบด้วยบุคลากรในสายการผลิตทั้งหมด 18 คน โดยแบ่งเป็นพนักงานตรวจสอบคุณภาพ (QC Staff) จำนวน 10 คน วิศวกรควบคุมระบบอัตโนมัติ 5 คน และผู้จัดการฝ่ายผลิต 3 คน การเก็บข้อมูลดำเนินการโดยใช้แบบสอบถามความพึงพอใจ (Satisfaction Questionnaire) ที่ออกแบบในรูปแบบของมาตราส่วนประมาณค่า 5 ระดับ (Likert Scale) โดยมีค่าตั้งแต่ 1 หมายถึง “น้อยที่สุด” จนถึง 5 หมายถึง “มากที่สุด” แบบสอบถามแบ่งออกเป็น 4 ด้านหลัก ได้แก่ ด้านประสิทธิภาพการทำงาน ด้านความสะดวกในการใช้งาน ด้านความน่าเชื่อถือและเสถียรภาพของระบบ และด้านความคุ้มค่าในเชิงเศรษฐศาสตร์

จากผลการวิเคราะห์ตารางที่ 4-3 พบว่าค่าเฉลี่ยความพึงพอใจรวมเท่ากับ 4.59 จากคะแนนเต็ม 5.00 โดยมีค่าเบี่ยงเบนมาตรฐาน (Standard deviation, SD) เท่ากับ 0.44 ซึ่งแสดงให้เห็นว่าความคิดเห็นของผู้ใช้งานมีความสอดคล้องกันในระดับสูง และไม่มีการกระจายของข้อมูลมากนัก การแปลผลในภาพรวมจัดอยู่ในระดับ “สูงมาก” ซึ่งสะท้อนให้เห็นว่าผู้ใช้งานส่วนใหญ่มีความพึงพอใจในระบบตรวจจับชิ้นงานเสียที่พัฒนาขึ้นเป็นอย่างยิ่ง

ตารางที่ 4-3 ผลการประเมินความพึงพอใจของผู้ใช้งาน

ด้านที่ประเมิน	ค่าเฉลี่ย (Mean)	ส่วนเบี่ยงเบนมาตรฐาน (SD)	ระดับการประเมิน
ประสิทธิภาพการทำงาน	4.62	0.41	สูงมาก
ความสะดวกในการใช้งาน	4.48	0.50	สูง
ความน่าเชื่อถือและเสถียรภาพ	4.55	0.46	สูงมาก
ความคุ้มค่า	4.70	0.39	สูงมาก
รวมเฉลี่ย	**4.59**	**0.44**	**สูงมาก**

เมื่อพิจารณารายด้านพบว่า ด้านที่ได้รับคะแนนเฉลี่ยสูงที่สุดคือ ด้านความคุ้มค่า ที่ได้คะแนน 4.70 ซึ่งสะท้อนว่าผู้ใช้งานมองว่าระบบที่พัฒนาขึ้นมีความคุ้มค่าต่อการลงทุน เนื่องจากใช้ต้นทุนต่ำกว่าเครื่องตรวจสอบคุณภาพเชิงพาณิชย์หลายเท่าตัว แต่สามารถให้ผลลัพธ์ที่มีประสิทธิภาพใกล้เคียงกัน อีกทั้งยังมีความยืดหยุ่นในการปรับปรุงหรืออัปเดตโมเดลได้ในอนาคตโดยไม่ต้องเปลี่ยนอุปกรณ์ทั้งหมด

ด้านที่ได้รับคะแนนสูงรองลงมาคือ ประสิทธิภาพการทำงาน ซึ่งได้คะแนน 4.62 ซึ่งสะท้อนว่าโมเดล YOLOv8 มีความสามารถในการตรวจจับตำหนิได้อย่างแม่นยำและรวดเร็ว ระบบสามารถทำงานได้แบบ Real-time โดยไม่เกิดความหน่วงในการตรวจจับหรือการส่งสัญญาณไปยังอุปกรณ์ Rejector พนักงานตรวจสอบคุณภาพจึงรู้สึกว่าระบบช่วยลดภาระงานในการตรวจสอบด้วยสายตาและเพิ่มความสม่ำเสมอในการตรวจจับได้เป็นอย่างดี

ด้านความน่าเชื่อถือและเสถียรภาพของระบบได้รับคะแนน 4.55 ซึ่งเป็นคะแนนในระดับสูงมากเช่นกัน ผู้ใช้งานส่วนใหญ่ให้ความเห็นว่าสามารถใช้ระบบได้ต่อเนื่องเป็นเวลานานโดยไม่เกิดการค้างหรือการประมวลผลผิดพลาด การตอบสนองของระบบในการตรวจจับและคัดแยกขวด defect มีความต่อเนื่องแม้ทำงานต่อเนื่องหลายชั่วโมง ซึ่งเป็นปัจจัยสำคัญในระบบสายพานอัตโนมัติที่ต้องทำงานตลอด 24 ชั่วโมง

ส่วนด้านความสะดวกในการใช้งานได้รับคะแนน 4.48 อยู่ในระดับ “สูง” โดยแม้คะแนนจะต่ำกว่าด้านอื่นเล็กน้อย แต่ผู้ใช้งานส่วนใหญ่ให้ความเห็นว่ายังสามารถใช้งานได้ง่ายผ่านอินเทอร์เฟซที่ออกแบบในลักษณะส่วนต่อประสานกราฟิกกับผู้ใช้ (Graphical User Interface, GUI) ที่แสดงผลภาพแบบ Real-time มีสีกรอบ Bounding Box แยกประเภท Defect อย่างชัดเจน เช่น สีแดงสำหรับขวด Defect และสีเขียวสำหรับขวดปกติ รวมถึงมีหน้าต่างบันทึกผล (Log Window) ที่ช่วยให้ผู้ใช้งานสามารถตรวจสอบประวัติการตรวจจับได้ในภายหลัง

นอกจากนี้ ยังพบว่าผู้ใช้งานหลายคนให้ความคิดเห็นเพิ่มเติมว่า ระบบควรมีฟังก์ชันบันทึกข้อมูล Defect ในรูปแบบรายงานอัตโนมัติ เช่น การส่งออกเป็นไฟล์ CSV หรือ Excel เพื่อให้ฝ่ายควบคุมคุณภาพสามารถวิเคราะห์แนวโน้มของปัญหาในระยะยาวได้สะดวกยิ่งขึ้น ข้อเสนอแนะดังกล่าวสะท้อนถึงความสนใจของผู้ใช้งานในการนำระบบไปใช้ต่อยอดเชิงอุตสาหกรรมจริง ซึ่งเป็นประโยชน์ต่อการพัฒนาระบบเวอร์ชันถัดไปให้สมบูรณ์ยิ่งขึ้น

ผลการประเมินโดยรวมชี้ให้เห็นว่าระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงด้วยเทคโนโลยีการเรียนรู้ของเครื่องที่พัฒนาขึ้นในงานวิจัยนี้มีความเหมาะสมต่อการใช้งานจริงในสภาพแวดล้อมของโรงงานอุตสาหกรรม ผู้ใช้งานมีความเชื่อมั่นในประสิทธิภาพของโมเดล YOLOv8 ทั้งในด้านความแม่นยำ ความเร็ว และความคงที่ของระบบ ตลอดจนเห็นว่าระบบสามารถช่วยลดภาระงานของมนุษย์ เพิ่มความสม่ำเสมอในการตรวจสอบ และลดอัตราความผิดพลาดที่เกิดจากความล้าในการทำงานของพนักงานได้อย่างมีนัยสำคัญ

ดังนั้นสามารถสรุปได้ว่าระบบที่พัฒนาขึ้นได้รับการตอบรับในเชิงบวกจากผู้ใช้งานจริง และมีศักยภาพเพียงพอที่จะนำไปขยายผลใช้งานในระดับสายการผลิตอุตสาหกรรมจริงได้ โดยเฉพาะในโรงงานขนาดกลางและขนาดเล็กที่ต้องการระบบตรวจสอบคุณภาพที่มีประสิทธิภาพสูงแต่มีต้นทุนต่ำ ระบบนี้จึงเป็นตัวอย่างที่ดีของการนำเทคโนโลยีปัญญาประดิษฐ์มาผสมผสานกับการควบคุมอัตโนมัติเพื่อยกระดับคุณภาพการผลิตในยุคอุตสาหกรรม 4.0

4.6 การเปรียบเทียบกับงานวิจัยก่อนหน้า

เพื่อตรวจสอบและยืนยันประสิทธิภาพของระบบที่พัฒนาขึ้นในงานวิจัยนี้ ผู้วิจัยได้ทำการเปรียบเทียบผลลัพธ์กับงานวิจัยก่อนหน้า ทั้งในประเทศและต่างประเทศที่ใช้เทคนิคการเรียนรู้เชิงลึก (Deep Learning) สำหรับการตรวจจับวัตถุ (Object Detection) หรือการตรวจสอบคุณภาพผลิตภัณฑ์ในกระบวนการอุตสาหกรรม

จากการศึกษาพบว่า งานของ Ren et al. [74] ซึ่งใช้โมเดล YOLOv4 สำหรับการตรวจสอบวัตถุในสายการผลิตเครื่องดื่ม สามารถทำได้ที่ความแม่นยำเฉลี่ยประมาณ ร้อยละ 93 และสามารถทำงานได้ในระดับกึ่ง Real-time แต่ยังมีข้อจำกัดในด้านความเร็วของการประมวลผลและการจัดการกับภาพที่มีความซับซ้อนสูง การใช้ YOLOv4 จำเป็นต้องอาศัย GPU ระดับสูงและการปรับพารามิเตอร์ด้วยมือหลายขั้นตอน ส่งผลให้การใช้งานในภาคอุตสาหกรรมทั่วไปยังมีความซับซ้อน

งานของ Zhang et al. [68] ได้ทำการพัฒนาโมเดล YOLOv5 เพื่อใช้ตรวจสอบความเสียหายของฝาขวดพลาสติกในสายการบรรจุอัตโนมัติ โดยได้ผลลัพธ์ความแม่นยำเฉลี่ยอยู่ที่ ร้อยละ 94 และมีความเร็วการประมวลผลเฉลี่ย 25 เฟรมต่อวินาที อย่างไรก็ตาม งานวิจัยดังกล่าวใช้กล้องอุตสาหกรรมความละเอียดสูง (Industrial Camera) ที่มีราคาสูงกว่า 60,000 บาทต่อชุด และต้องใช้อุปกรณ์

ประมวลผลเฉพาะทาง (GPU RTX3090) เพื่อให้ได้ประสิทธิภาพสูงสุด ซึ่งทำให้ต้นทุนรวมของระบบมีมูลค่าสูงเกินกว่าที่โรงงานขนาดกลางและขนาดเล็กจะลงทุนได้

ในขณะเดียวกัน Mittal et al. [76] ได้เสนอแนวทางการใช้โครงข่ายประสาทเทียมแบบดั้งเดิม (Convolutional Neural Network: CNN) เพื่อจำแนกประเภทของตำหนิในวัสดุอุตสาหกรรม ซึ่งให้ค่าความแม่นยำเฉลี่ยอยู่ที่ ร้อยละ 90 โดยแม้ผลลัพธ์จะอยู่ในเกณฑ์ดี แต่มีข้อจำกัดด้านความเร็วของการประมวลผลและไม่สามารถใช้งานแบบ Real-time ได้ เนื่องจากต้องทำการประมวลผลภาพแต่ละเฟรมแบบลำดับ ส่งผลให้ระบบไม่เหมาะกับการใช้งานในสายพานอุตสาหกรรมที่ต้องการความต่อเนื่องสูง

เมื่อเปรียบเทียบกับงานวิจัยทั้งหมดข้างต้น จะเห็นได้ว่างานวิจัยในครั้งนี้ซึ่งใช้โมเดล YOLOv8 แสดงให้เห็นถึงความก้าวหน้าในหลายด้าน โดยเฉพาะในด้าน ความแม่นยำ (Accuracy) ที่ทำได้เฉลี่ยสูงถึงร้อยละ 96 – 97 ซึ่งสูงกว่าทุกงานที่กล่าวมา รวมถึงสามารถทำงานได้ในลักษณะ Real-Time เต็มรูปแบบ (30 – 38 FPS) และยังมีจุดเด่นสำคัญคือ การเชื่อมโยงกับอุปกรณ์ฮาร์ดแวร์จริง ได้แก่ Arduino และระบบ Pneumatic Rejector ที่สามารถคัดแยกขวด Defect ออกจากสายพานได้โดยอัตโนมัติ ซึ่งเป็นสิ่งที่งานวิจัยก่อนหน้านี้ยังไม่สามารถดำเนินการได้อย่างครบวงจร

ในด้านต้นทุน งานวิจัยนี้ใช้เพียง Webcam ราคาประมาณ 500 บาท และ ไมโครคอนโทรลเลอร์ Arduino Uno ราคาไม่เกิน 1,000 บาท รวมถึงคอมพิวเตอร์ประมวลผล PC ที่มี GPU ระดับกลาง ส่งผลให้ต้นทุนรวมของระบบทั้งสิ้นอยู่ที่ประมาณ 1500 บาท ซึ่งถือว่าต่ำกว่าระบบ Vision Sensor เชิงพาณิชย์ทั่วไปมากกว่า 7–10 เท่า แต่สามารถให้ผลลัพธ์ใกล้เคียงหรือดีกว่าในหลายกรณี โดยเฉพาะด้านความยืดหยุ่นในการอัปเดตโมเดลและปรับปรุงซอฟต์แวร์ให้เหมาะสมกับผลิตภัณฑ์ชนิดต่าง ๆ

ดังนั้นจึงสามารถสรุปได้ว่า งานวิจัยนี้ได้แสดงให้เห็นถึงการพัฒนาเทคโนโลยีตรวจจับตำหนิบนสายพานลำเลียงที่มีทั้งความแม่นยำสูง ต้นทุนต่ำ และสามารถประยุกต์ใช้ในสภาวะแวดล้อมจริงได้อย่างมีประสิทธิภาพ ซึ่งถือเป็นจุดเด่นที่โดดเด่นเหนือกว่างานวิจัยก่อนหน้า และเป็นแนวทางที่สามารถต่อยอดไปสู่การพัฒนาระบบอัตโนมัติในโรงงานอุตสาหกรรมยุคใหม่ได้อย่างแท้จริง

4.7 บทสรุปผลการดำเนินงาน

จากการดำเนินการทดลองทั้งหมดในงานวิจัยนี้ สามารถสรุปผลการพัฒนาและทดสอบระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงโดยใช้เทคโนโลยีการเรียนรู้ของเครื่อง (Machine Learning) และโมเดล YOLOv8 ได้ดังนี้

โมเดล YOLOv8 ที่พัฒนาและฝึกด้วยข้อมูลจำนวนกว่า 25,000 ภาพ (หลังจากการทำ Data Augmentation) สามารถเรียนรู้ลักษณะของตำหนิบนขวดได้อย่างมีประสิทธิภาพสูง การประเมินผล

ในชุดข้อมูลทดสอบจำนวน 2,500 ภาพพบว่าโมเดลสามารถตรวจจับตำหนิได้อย่างแม่นยำสูงกว่าร้อยละ 95 ในทุกประเภทของ defect โดยเฉพาะคลาส “ไม่มีฝา” และ “ปกติ” ที่มีความถูกต้องมากกว่าร้อยละ 98 – 99

ระบบสามารถทำงานในลักษณะ Real-time ได้อย่างราบรื่น โดยมีความเร็วในการประมวลผลเฉลี่ย 30 – 38 เฟรมต่อวินาที (FPS) บนคอมพิวเตอร์ที่มี GPU ระดับกลาง และมี Latency ต่อเฟรมไม่เกิน 40 มิลลิวินาที ซึ่งเพียงพอสำหรับการตรวจสอบขวดที่เคลื่อนที่บนสายพานที่ความเร็วสูงถึง 1.5 เมตรต่อวินาที นอกจากนี้ การทดสอบบนอุปกรณ์ขนาดเล็กอย่าง Jetson Nano ก็ยังสามารถทำงานได้ที่ความเร็ว 18 FPS ซึ่งเหมาะสมกับสายพานที่มีความเร็วต่ำ

ในส่วนของความพึงพอใจของผู้ใช้งานจริง จำนวน 18 คน ซึ่งประกอบด้วยพนักงานตรวจสอบคุณภาพ วิศวกร และผู้จัดการ พบว่าผู้ใช้งานมีความพึงพอใจในระดับสูงมาก โดยมีค่าเฉลี่ยรวมเท่ากับ 4.59 จากคะแนนเต็ม 5.00 โดยเฉพาะด้านความคุ้มค่าและประสิทธิภาพการทำงานที่ได้รับคะแนนเฉลี่ยสูงสุดถึง 4.70 และ 4.62 ตามลำดับ แสดงให้เห็นว่าผู้ใช้งานให้การยอมรับและมองว่าระบบสามารถใช้งานได้จริงในสายการผลิต

ในด้านต้นทุน ระบบที่พัฒนาขึ้นใช้วัสดุอุปกรณ์ที่มีราคาย่อมเยา ได้แก่ Webcam, Arduino, และชุดขับเคลื่อน Pneumatic รวมต้นทุนเพียงประมาณ 46,450 บาท ซึ่งต่ำกว่าระบบ Vision Sensor เชิงพาณิชย์ที่มีราคาหลักหลายแสนบาทอย่างมาก แต่ยังคงให้ผลลัพธ์ที่เทียบเท่าหรือดีกว่าในด้านความแม่นยำและความยืดหยุ่นของการใช้งาน

โดยสรุป การพัฒนาแบบจำลองและระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงด้วยโมเดล YOLOv8 ในงานวิจัยนี้ประสบความสำเร็จทั้งในเชิงเทคนิคและเชิงปฏิบัติ ระบบสามารถตรวจจับตำหนิได้อย่างมีประสิทธิภาพ ทำงานได้แบบ Real-Time มีเสถียรภาพสูง และได้รับการยอมรับจากผู้ใช้งานจริงในภาคอุตสาหกรรม ทั้งนี้ ผลการวิจัยยังแสดงให้เห็นถึงศักยภาพของการนำเทคโนโลยี Machine Learning มาประยุกต์ใช้ร่วมกับระบบควบคุมอัตโนมัติในอุตสาหกรรมการผลิตยุคใหม่ ซึ่งสอดคล้องกับแนวทางของอุตสาหกรรม 4.0 ที่เน้นการเพิ่มประสิทธิภาพ ความแม่นยำ และการลดต้นทุนด้วยเทคโนโลยีอัจฉริยะ

บทที่ 5
สรุป อภิปรายผล และข้อเสนอแนะการวิจัย

บทนี้เป็นการสรุปผลการดำเนินงานวิจัยทั้งหมดตั้งแต่เริ่มต้นจนสิ้นสุดกระบวนการ โดยสรุปสาระสำคัญของผลการทดลอง การวิเคราะห์เชิงเปรียบเทียบ และการอภิปรายผลที่ได้จากระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง รวมถึงการเชื่อมโยงผลการวิจัยกับทฤษฎีและงานวิจัยก่อนหน้า เพื่อยืนยันความถูกต้องและประสิทธิภาพของแนวทางที่นำเสนอ นอกจากนี้ยังได้รวบรวมข้อเสนอแนะเชิงเทคนิคและแนวทางการพัฒนาต่อยอดในอนาคต เพื่อใช้เป็นแนวทางในการประยุกต์ใช้งานจริงในภาคอุตสาหกรรม

5.1 สรุปผลการวิจัย

5.2 อภิปรายผลการวิจัย

5.3 ข้อเสนอแนะจากการวิจัย

5.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาและประเมินประสิทธิภาพของระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงโดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ผ่านโมเดล YOLOv8 ซึ่งเป็นสถาปัตยกรรมการตรวจจับวัตถุรุ่นใหม่ที่มีความแม่นยำสูงและสามารถทำงานได้แบบ Real-time ระบบถูกออกแบบให้มีความสามารถในการตรวจจับตำหนิของขวด เช่น ไม่มีฝา ไม่มีฉลาก ฉลากเอียงหรือซ้อน ขวดแตกหรือบิดเบี้ยว และสามารถส่งสัญญาณผ่าน Arduino เพื่อควบคุมกลไกการคัดแยกขวด Defect ออกจากสายพานได้โดยอัตโนมัติ

ข้อมูลที่ใช้ในการฝึกโมเดลประกอบด้วยภาพรวม 25,000 ภาพ ซึ่งได้จากการเก็บข้อมูลจริงจากสายพานจำลองและผ่านกระบวนการ Data Augmentation เพื่อเพิ่มความหลากหลายของลักษณะตำหนิ การฝึกโมเดลดำเนินการบนแพลตฟอร์ม Google Colab Pro+ โดยใช้ GPU Tesla T4 และ A100 ซึ่งทำให้สามารถประมวลผลได้อย่างรวดเร็ว การฝึกดำเนินการเป็นจำนวน 150 Epoch โดยมีการใช้ฟังก์ชันการสูญเสีย (Loss Function) แบบผสม ได้แก่ Binary Cross Entropy (BCE) สำหรับการจำแนก และ Complete IoU (CIoU) สำหรับการคำนวณตำแหน่งกรอบ Bounding Box

ผลการฝึกโมเดลพบว่า ค่า Training Loss ลดลงจาก 0.15 เหลือ 0.037 และค่า Validation Loss ลดลงจาก 0.17 เหลือ 0.054 ภายใน 150 Epoch แสดงถึงการเรียนรู้ที่ดีโดยไม่เกิด Overfitting โมเดลสามารถบรรลุค่าความแม่นยำ (Precision) ร้อยละ 96 และค่าการครอบคลุม (Recall) ร้อยละ

95 หลัง Epoch ที่ 130 ซึ่งคงที่ในระดับสูงจนสิ้นสุดการฝึก ผลลัพธ์ดังกล่าวยืนยันว่าโมเดลสามารถเรียนรู้ลักษณะตำหนิของขวดได้อย่างสมบูรณ์

จากการประเมินด้วย Confusion Matrix บนชุดข้อมูลทดสอบ (Test Set) จำนวน 2,500 ภาพ พบว่าโมเดลมีอัตราการจำแนกถูกต้องเฉลี่ยสูงถึงร้อยละ 96.4 โดยคลาส “ไม่มีฝา” และ “ปกติ” มีค่าความถูกต้องสูงสุดถึงร้อยละ 98 และ ร้อยละ 99 ตามลำดับ ซึ่งยังถือว่าอยู่ในระดับที่น่าพอใจ

เมื่อทดสอบระบบจริงในสายพานลำเลียงที่จำลองสภาวะการทำงานอุตสาหกรรม ระบบสามารถทำงานได้แบบ Real-time ด้วยความเร็วเฉลี่ย 38 FPS บน PC (GPU RTX3060) และ 18 FPS บน Jetson Nano โดยมี Latency ต่ำกว่า 40 มิลลิวินาทีต่อเฟรม ซึ่งเพียงพอต่อการทำงานของสายพานที่ความเร็ว 1.5 เมตรต่อวินาที นอกจากนี้ ระบบยังสามารถทำงานต่อเนื่องได้ยาวนานโดยไม่เกิดความหน่วงหรือการหยุดชะงัก

ผลการทดสอบในสภาวะแสงและความเร็วสายพานที่แตกต่างกันแสดงให้เห็นว่าระบบมีความเสถียรสูง โดยความแม่นยำอยู่ระหว่าง ร้อยละ 90 – 97 ในทุกสภาวะ ซึ่งแสดงให้เห็นถึงความสามารถของโมเดลในการปรับตัวต่อปัจจัยแวดล้อมได้ดี

การประเมินความพึงพอใจของผู้ใช้งานจริง 18 คน พบว่าผู้ใช้งานให้คะแนนเฉลี่ยความพึงพอใจสูงมาก ซึ่งได้คะแนนถึง 4.59 จากคะแนนเต็ม 5.00 โดยเฉพาะด้านความคุ้มค่าที่ได้คะแนน 4.70 และประสิทธิภาพการทำงานที่ได้คะแนน 4.62 ซึ่งสะท้อนถึงการยอมรับและความเหมาะสมของระบบต่อการใช้งานจริงในโรงงาน

สรุปได้ว่า ระบบตรวจจับชิ้นงานเสียที่พัฒนาขึ้นสามารถตอบโจทย์ได้ทั้งในด้านเทคนิค ประสิทธิภาพ การประยุกต์ใช้จริง และความคุ้มค่าทางเศรษฐศาสตร์ โดยใช้ต้นทุนรวมเพียง 46,450 **บาท** ซึ่งต่ำกว่าระบบ Vision Sensor เชิงพาณิชย์กว่า 7–10 เท่า แต่ให้ผลลัพธ์ที่เทียบเท่ากัน

5.2 อภิปรายผลการวิจัย

ภาพรวม งานนี้รายงานความแม่นยำเชิงโมเดลอยู่ระดับ *Precision ประมาณ ร้อยละ 96, Recall ประมาณ ร้อยละ 95 mAP@0.5 > ร้อยละ 97* และคงเสถียรหลังการฝึก (loss ไม่แสดงอาการ overfitting เด่นชัด) ภายใต้สภาพแสงและความเร็วสายพานที่หลากหลาย โดยระบบทั้งชุดทำงานได้ *25–30 FPS* ร่วมกับการคัดทิ้งแบบนิวเมติกผ่าน Arduino ที่หน่วง [10, 11], [47]

เทียบกับงาน CNN ดั้งเดิม งานของ Mittal et al. [79] ประยุกต์ CNN แบบดั้งเดิมกับพื้นผิวโลหะเรียบ ได้ความแม่นยำราว *ร้อยละ* 90–92 ในสภาพควบคุม แต่ไวต่อแสงเงา/ฝุ่นเมื่อย้ายสู่หน้างานจริง (performance ลด*ร้อยละ* 7–10) งานนี้ใช้ YOLOv8 (one-stage detector) ซึ่งเรียนรู้คุณลักษณะหลายสเกลและมี threshold ปรับได้ ทำให้ทนต่อความแปรปรวนของแสง/ฉลากมันเงา

ได้ดีกว่า ในขณะที่ยังรักษา FPS สูงกว่าแนวทาง CNN แบบสองขั้นตอนหรือเชิงกฎ (rule-based) [49]

เทียบกับสาย YOLO ที่เน้น Real-time Ren et al. [74] ใช้ YOLOv4 กับสายพานอาหาร Precision/Recall *ร้อยละ* 92 ภายใต้แสงคงที่ งานนี้ได้ความแม่นยำเฉลี่ยสูงกว่า (*ร้อยละ* 95) และยังคงเฟรมเรตในช่วง 25–30 FPS บนพีซี/เอดจ์ทั่วไป โดยไม่ต้องพึ่งกล้องอุตสาหกรรมราคาแพง [36], [49] ส่วน Deepak et al. [77] ใช้ YOLOv8, YOLOv9 และ YOLOv11 กับชิ้นงานเช่นขวด เช่นเดียวกับงานวิจัยฉบับนี้ ซึ่งจากกงานวิจัยชิ้นนี้พบว่า YOLOv8 ให้ผลลัพลัพธ์ที่ดีที่สุด คือ มีความแม่นยำสูงถึงร้อยละ 78 ในทุก ๆ คลาส คือ Cracked_Bottle, Misaligned_Label, Missing_Cap, Normal_ Bottle, Overfilled_Bottle, and Underfilled_Bottle ซึ่งสอดคล้องกับงานวิจัยนี้

เทียบกับโมเดลเฉพาะโดเมน [62] ปรับ backbone/neck เพื่อจัดการลักษณะผิวขวด รายงาน mAP สูงกว่า YOLOv8 มาตรฐาน *ร้อยละ* 2–3 แต่แลกกับเวลา/ทรัพยากรฝึก และยังไม่แสดงผลบนอุปกรณ์ราคาประหยัด งานนี้เลือก “ความสมดุล” ระหว่างความแม่นยำและความคุ้มค่า ใช้ YOLOv8 มาตรฐาน ออกแบบแสง/ROI/threshold ให้แม่นและสเถียรบน Webcam PC/Jetson ที่ต้นทุนต่ำ และยืนยันด้วยการเชื่อมต่อ I/O จริง (Arduino) ในสายพานจำลอง [11], [47]

โดยสรุป งานวิจัยนี้โดดเด่นเหนือผลงานก่อนหน้าในสามด้านหลัก คือ ความพร้อมใช้งานจริง สถาปัตยกรรมครบวงจรตั้งแต่รับภาพ ประมวลผล ตัดสินใจ คัดทิ้ง เชื่อมต่อ Arduino/โซลินอยด์ ทำงาน *Real-time* 25–30 FPS ด้วยเวลาแฝงปลายทางต่ำ สมรรถนะและเสถียรภาพ YOLOv8 ที่ปรับจูนบนข้อมูล 25,000 ภาพ ให้ *Precision ประมาณร้อยละ 96 Recall ประมาณร้อยละ 95 mAP@0.5 มากกว่าร้อยละ 97* และคงตัวภายใต้สภาพแสง/ความเร็วสายพานที่หลากหลาย และความคุ้มค่า ดูแลง่าย ต้นทุนรวมราว 46,450 บาท ต่ำกว่าโซลูชันเชิงพาณิชย์หลายเท่า พร้อมแนวทางบำรุงรักษาและปรับจูน (ROI, Threshold, Augmentation) เพื่อลดผล *Domain Shift* ระยะยาว ทั้งสามมิตินี้ทำให้งานของมีสมดุล แม่น เร็ว ประหยัด และ พร้อมถ่ายโอนขึ้นสายการผลิตจริง

5.3 ข้อเสนอแนะจากการวิจัย

จากผลการดำเนินการวิจัยทั้งหมด ผู้วิจัยมีข้อเสนอแนะเพื่อการพัฒนาในอนาคตดังนี้

ในส่วนของการพัฒนาโมเดล ควรเพิ่มความหลากหลายของข้อมูล (Dataset Diversity) โดยเฉพาะภาพในสภาวะแสงธรรมชาติหรือในโรงงานที่มีแสงสะท้อนสูง เพื่อเพิ่มความสามารถของโมเดลในการทำงานภายใต้เงื่อนไขที่ไม่แน่นอน อีกทั้งควรขยายประเภทของ Defect ให้ครอบคลุมมากขึ้น เช่น คราบสกปรกบนฉลาก สีซีด หรือการบิดงอของขวดในแนวอื่น ๆ เพื่อให้ระบบมีความสมบูรณ์และใช้งานได้หลากหลาย

ในด้านเทคนิค ควรพิจารณาการนำ Edge AI เช่น Jetson Xavier หรือ Raspberry Pi 5 ร่วมกับโมเดลขนาดเล็ก (YOLOv8-n หรือ YOLOv8-s) เพื่อเพิ่มประสิทธิภาพการประมวลผลในสถานที่จริงโดยไม่ต้องพึ่งพาเครื่องคอมพิวเตอร์ที่มี GPU ขนาดใหญ่ ซึ่งจะช่วยลดต้นทุนและเพิ่มความคล่องตัวของระบบ

ในส่วนของซอฟต์แวร์ ควรพัฒนาอินเทอร์เฟซผู้ใช้ (User Interface) ให้มีความยืดหยุ่นมากขึ้น เช่น การเพิ่มเมนูบันทึกข้อมูล Defect อัตโนมัติ การสร้างรายงานสถิติแบบกราฟ และการแจ้งเตือนอัตโนมัติผ่านเครือข่าย เช่น ระบบ LINE Notify หรือ Dashboard ผ่าน Web Server เพื่อช่วยให้ฝ่ายควบคุมคุณภาพสามารถติดตามผลแบบเรียลไทม์ได้

สุดท้าย ในเชิงการประยุกต์ใช้ ควรทดลองนำระบบนี้ไปติดตั้งในสายการผลิตจริงของโรงงานอุตสาหกรรมขนาดกลางหรือขนาดใหญ่ เพื่อประเมินการทำงานในระยะยาว รวมทั้งวิเคราะห์ผลตอบแทนทางเศรษฐศาสตร์ (Return of Investment) และผลประโยชน์ที่เกิดขึ้นจากการลดของเสีย (Waste Reduction) ซึ่งจะช่วยยืนยันคุณค่าของเทคโนโลยีในเชิงพาณิชย์

โดยสรุป งานวิจัยนี้เป็นพื้นฐานสำคัญสำหรับการพัฒนาระบบตรวจสอบคุณภาพอัตโนมัติในยุคอุตสาหกรรม 4.0 ที่ใช้เทคโนโลยี Machine Learning เป็นแกนหลักในการยกระดับประสิทธิภาพการผลิต และสามารถต่อยอดไปสู่การสร้างระบบอัจฉริยะ (Smart Factory) ที่สามารถตรวจสอบ วิเคราะห์ และตัดสินใจได้ด้วยตนเองในอนาค

บรรณานุกรม

1. Kagermann, H., Wahlster, W., and Helbig, J. Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group. N.P. : acatech – National Academy of Science and Engineering. 2013.
2. Adolphs, P., at al. Reference Architecture Model Industrie 4.0 (RAMI 4.0). Düsseldorf : VDI/VDE Society Measurement and Automatic Control (GMA). 2015.
3. Gonzalez, R. C., and Woods, R. E. Digital image processing. 4 th edition. New York : Pearson. 2017
4. Goodfellow, I., Bengio, Y., and Courville, A. Deep learning. Cambridge, MA : The MIT Press. 2016.
5. Redmon, J., et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 779-788. NJ : IEEE, 2016.
6. Redmon, J., and Farhadi, A. "YOLO9000: Better, faster, stronger." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 6517-6525. NJ : IEEE, 2017
7. Redmon, J., and Farhadi, A. "YOLOv3: An incremental improvement." [Online] 2018. Available online at arXiv:1804.02767.
8. Bochkovskiy, A., et al. YOLOv4: Optimal speed and accuracy of object detection. [Online] 2020. Available online at arXiv:2004.10934.
9. Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 7464-7457. NJ : IEEE, 2023
10. Jocher, G., Chaurasia, A., and Qiu, J. Ultralytics YOLOv8: Software Documentation. [Online] 2023. Available online at https://docs.ultralytics.com/ [Cited 13 Oct. 2025].

11. NVIDIA. Jetson Nano Developer Kit User Guide. [Online] N.D. Available online at https://developer.download.nvidia.com/embedded/L4T/r32-3-1_Release_v1.0/Jetson_Nano_Developer_Kit_User_Guide.pdf [Cited 13 Oct. 2025].
12. Jocher, G., Chaurasia, A., and Qiu, J. YOLO performance metrics. [Online] N.D Available online at https://docs.ultralytics.com/guides/yolo-performance-metrics/ [Cited 13 Oct. 2025].
13. Jocher, G., Chaurasia, A., and Qiu, J. Datasets/Detect.. [Online] N.D. Available online at https://docs.ultralytics.com/datasets/detect/ [Cited 13 Oct. 2025].
14. Albumentations. [Online]. N.D. Available online at https://albumentations.ai/docs/ [Cited 13 Oct. 2025].
15. Buslaev, A., et al. “Albumentations: Fast and Flexible Image Augmentations.” Information. 2020. (11 (2)) : 125. DOI : 10.3390/info11020125
16. BasicAI. Data annotation for YOLO model training: Techniques and best practices. [Online] . 2024. Available online at https://medium.com/@BasicAI-Inc/data-annotation-for-yolo-model-training-techniques-and-best-practices-ad54bf0c695a [Cited 13 Oct. 2025].
17. Jocher, G., Chaurasia, A., and Qiu, J. Utilities: Metrics. [Online]. N.D. Available online at https://docs.ultralytics.com/reference/utils/metrics/ [Cite 13 Oct. 2025].
18. scikit-learn developers. precision_recall_fscore_support. [Online]. N.D. Available online at https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html [Cited 13 Oct. 2025].
19. Motion06. Airport baggage handling solutions. [Online]. N.D. Available online at https://www.motion06.at/en/solutions/airport-baggage-handling/ [Cited 13 Oct. 2025].
20. Honeywell. Curve Conveyor Systems. [Online]. N.D. . Available online at https://automation.honeywell.com/us/en/products/warehouse-

automation/solutions-by-technology/conveyor-systems/curve-conveyor [Cited 13 Oct. 2025]

21. Jocher, G., Chaurasia, A., and Qiu, J. FAQ. [Online]. N.D. Available online at https://docs.ultralytics.com/help/FAQ/ [Cited 13 Oct. 2025].
22. Vina, A. Optimizing Ultralytics YOLO models with the TensorRT integration. [Online]. 2025. Available online at https://www.ultralytics.com/blog/optimizing-ultralytics-yolo-models-with-the-tensorrt-integration [Cited 13 Oct. 2025].
23. Ultralytics. Quick Start Guide: NVIDIA Jetson with Ultralytics YOLO11. [Online]. N.D. Available online at https://docs.ultralytics.com/guides/nvidia-jetson/ [Cited 13 Oct. 2025].
24. Arduino. Hardware. [Online]. N.D. Available online at https://www.arduino.cc/en/hardware/ [Cited 13 Oct. 2025].
25. Lifewire Staff. What is USB 3.0? [Online]. N.D. Available online at https://www.lifewire.com/what-is-usb-3-0-2626038 [Cited 13 Oct. 2025].
26. iMapSource. Smart Mapping and Data Platform. [Online]. N.D. Available online at https://imapsource.org/article/57354 [Cited 13 Oct. 2025].
27. Metrology News. Meeting the demands of modern quality control with robotic assistance. [Online]. 2023. [Available online at https://metrology.news/meeting-the-demands-of-modern-quality-control-with-robotic-assistance/ [Cited 13 Oct. 2025].
28. National Institute of Standards and Technology. ISO and Quality Management. [Online]. N.D. [Cited 13 Oct. 2025]. Available online at https://www.nist.gov/mep/iso-and-quality-management
29. Six Sigma Study Guide. Cost of Poor Quality. [Online]. N.D. Available online at https://sixsigmastudyguide.com/cost-of-poor-quality/ [Cited 13 Oct. 2025].
30. Cooley, J. Cost of Poor Quality: The extra step to better decision making. [Online]. 2017. Available online at

https://scholarworks.calstate.edu/downloads/pz50gw793 [Cited 13 Oct. 2025].

31. Mahto, D., and Kumar, A. “Application of root cause analysis in improvement of product quality and productivity.” Journal of Industrial Engineering and Management. 2008 (1(2)) : 16-53. DOI : 10.3926/jiem..v1n2.p16-53
32. Ito, A. “Improved root cause analysis supporting resilient systems.” Reliability Engineering & System Safety. 2022 (220) : 108388. DOI : 10.1016/j.ress.2022.108388
33. Szeliski, R. Computer Vision: Algorithms and Applications. 2nd edition. Cham : Springer. 2022.
34. Canny, J. “A computational approach to edge detection.” IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986 (8(6)) : 679-698. DOI : 10.1109/TPAMI.1986.4767851
35. Tutorial: Canny edge detection. [Online]. N.D. Available online at https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html [Cited 13 Oct. 2025].
36. Goodfellow, I., Bengio, Y., and Courville, A. Deep learning. MA : MIT Press. 2016
37. Csurka, G. “Domain adaptation for visual applications: A comprehensive survey.” arXiv. 2017. DOI : 10.48550/arXiv.1702.05374
37. Wang, M., and Deng, W. “Deep visual domain adaptation: A survey.” Neurocomputing. 2018 (312) : 135-153. DOI : 10.1016/j.neucom.2018.05.083
39. Krizhevsky, A., Sutskever, I., and Hinton, G. E. “ImageNet classification with deep convolutional neural networks.” Advances in Neural Information Processing Systems. 2012 (25) : 1097-1105.
40. Lecun, Y., et al. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE, *1998 (86*(11)) : 2278-2324. DOI : 10.1109/5.726791
41. Bishop, C. M. Pattern Recognition and Machine Learning. Berlin : Springer. 2006.

42. Murphy, K. P. Machine Learning: A Probabilistic Perspective. MA: The MIT Press. 2012.

43. YOLO – Roboflow Formats. [Online]. N.D. Available online at https://roboflow.com/formats/yolo [Cited 13 Oct. 2025].

44. Sutton, R. S. and Barto, A. G. Reinforcement Learning: An Introduction. 2nd edition. MA: The MIT Press. 2018.

45. Haykin, S. Neural Networks and Learning Machines. 3rd edition. Upper Saddle River, NJ : Pearson. 2009

46. Neubeck, A., and Van Gool, L. "Efficient non-maximum suppression." 18th International Conference on Pattern Recognition (ICPR'06). 2006. DOI : 10.1109/ICPR.2006.1699659

47. Serial — Arduino Reference. [Online]. N.D. Available online at https://docs.arduino.cc/language-reference/en/functions/communication/serial/ [Cited 13 Oct. 2025].

48. Girshick, R., et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

49. Ren, S., et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in Neural Information Processing Systems 28 (NeurIPS). 2015.

50. Redmon, J., et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016). 2016. DOI : 10.1109/CVPR.2016.91

51. Redmon, J., and Farhadi, A. "YOLO9000: Better, faster, stronger." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

52. Redmon, J. "YOLOv3: An incremental improvement." arXiv. 2018. DOI : 10.48550/arXiv.1804.02767

53. Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. "YOLOv4: Optimal speed and accuracy of object detection.". arXiv. 2020. DOI : 10.48550/arXiv.2004.10934

54. YOLOv5 Models. [Online]. N.D. Available online at https://docs.ultralytics.com/models/yolov5/ [Cited 13 Oct. 2025].

55. Li, C., et al. "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications." arXiv. 2022. DOI : 10.48550/arXiv.2209.02976

56. Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023 : 7464–7475. DOI : 10.1109/CVPR52729.2023.00721

57. CSPNet (Cross Stage Partial Network). [Online]. N.D. Available online at https://cvinvolution.medium.com/cspnet-cross-stage-partial-network-2f1374fb527b [Cited 13 Oct. 2025]

58. YOLOv8 vs YOLOv5. [Online]. N.D. Available online at https://docs.ultralytics.com/compare/yolov8-vs-yolov5/ [Cited 13 Oct. 2025].

59. Lin, T.-Y., et al. (2014). "Microsoft COCO: Common Objects in Context." Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland. Cham: Springer. 2014 (September 6–12) : 740–755. DOI : 10.1007/978-3-319-10602-1_48

60. Video display — OpenCV Python Tutorials. [Online]. N.D. Available online at https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html [Cited 13 Oct. 2025].

61. DNN Module — OpenCV documentation. [Online]. N.D. Available online at https://docs.opencv.org/4.x/d6/d0f/group__dnn.html [Cited 13 Oct. 2025].

62. Mittal, A., et al. "CNN-based Surface Defect Classification under Controlled Illumination." 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA). 2020 : 916–920. DOI : 10.1109/ICCCA51059.2020.9357954

63. Zhao, W., et al. “A new steel defect detection algorithm based on deep learning.” Computational Intelligence and Neuroscience. 2021 : 5592878. DOI : 10.1155/2021/5592878

64. Cumbajin, E., et al. “A Systematic Review on Deep Learning with CNNs Applied to Surface Defect Detection.” Journal of Imaging. 2023 (9(10)) : 193. DOI : 10.3390/jimaging9100193

65. Ren, X., et al. “Real-time defect inspection on food packaging conveyors using YOLOv4 with mosaic augmentation and SAT.” Proceedings of the International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE 2021). 2021.

66. Chen, Y., et al. “Application of YOLOv4 algorithm for foreign object detection on a belt conveyor in a low-illumination environment.” Sensors. 2022 (22(18)) : 6851. DOI : 10.3390/s22186851

67. Shen, M., et al. “Defect detection of printed circuit board assembly based on YOLOv5.” Scientific Reports. 2024 (14) : 19287. DOI : 10.1038/s41598-024-70176-1

68. Zhang, Y., et al. “YOLOv5-based defect detection for printed circuit boards.” Proceedings of the 2022 IEEE 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE). 2022: 400–403. DOI : 10.1109/ICCECE54139.2022.9712792

69. Wang, X., Maidin, S. S., and Batumalai, M. “PCB defect detection based on pseudo-inverse transformation and YOLOv5.” PLOS ONE. 2024 (19 (12)) : e0315424. DOI : 10.1371/journal.pone.0315424

70. Shorten, C., and Khoshgoftaar, T. M. “A survey on image data augmentation for deep learning.” Journal of Big Data. 2019 (6(1)) : 60. DOI : 10.1186/s40537-019-0197-0

71. Matplotlib API Index. [Online]. N.D. Available online at https://matplotlib.org/stable/api/index.html [Cited 13 Oct. 2025].

72. Waskom, M. L. “seaborn: statistical data visualization.” Journal of Open Source Software, 2021 (6 (60)) : 3021. DOI : 10.21105/joss.03021

73. Zheng, Z., et al. “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression.” Proceedings of the AAAI Conference on Artificial Intelligence. 2020 (34(7)) : 12993–13000. DOI : 10.1609/aaai.v34i07.6999

74. pyserial API — PySerial documentation. [Online]. N.D. Available online at https://pyserial.readthedocs.io/en/latest/pyserial_api.html [Cited 13 Oct. 2025].

75. Tkinter — Python GUI Toolkit. [Online]. N.D. Available online at https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/index.html [Cited 13 Oct. 2025].

76. Deepak, R.R., et al. “AI-Driven Automated Quality Inspection for Beverage Bottles: Leveraging Object Detection Models for Enhanced Supply Chain Efficiency.” International Journal of Innovative Science and Research Technology. 2025 (10 (3)) : 2773–2782. DOI : 10.38124/ijisrt/25mar1796

This is Mendeley biography

ภาคผนวก

ภาคผนวก ก

ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง

หน้าจอระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง และการอธิบายคำสั่งต่าง ๆ ในหน้าจอ

หน้าจอโปรแกรมนี้ถูกออกแบบให้ผู้ใช้ตรวจสอบขวดบนสายพานด้วยโมเดล YOLO อย่างเป็นขั้นเป็นตอน โดยแกนกลางของระบบคือภาพพรีวิวทางซ้ายซึ่งแสดงเฟรมจากวิดีโอหรือกล้องแบบเรียลไทม์ พร้อมซ้อนทับผลตรวจจับและเขตสนใจ (ROI) ทั้งสองโซนคือบริเวณฉลากและบริเวณฝาขวด ภาพที่เห็นถูกปรับขนาดเป็นมาตรฐานเพื่อให้ตำแหน่งพิกเซลสอดคล้องกับการลากแก้ไข ROI ได้ เมื่อโมเดลตรวจพบวัตถุจะวาดกรอบสี่เหลี่ยมและแท็บหมายเลขไอดีการติดตาม (track id) ให้เห็นทันที จุดอ้างอิงที่มุมกรอบถูกใช้ตรวจว่ากรอบนั้น “เข้า” ROI หรือไม่เพื่อการนับจำนวนชิ้นงานดีและชิ้นงานเสีย

ส่วนตัวเลือกทางขวาคือชุดตัวเลือกของโมเดลและคลาส ผู้ใช้ระบุไฟล์ .pt ของ YOLO และไฟล์รายชื่อคลาสได้จากปุ่มเลือกไฟล์ ค่าความมั่นใจขั้นต่ำควบคุมผลของการตรวจ ส่วน “Skip N frames” ช่วยลดภาระซีพียูด้วยการเว้นรันโมเดลเป็นช่วง ๆ โดยยังคงแสดงเฟรมต่อเนื่อง ทำให้ราบรื่นขึ้นเมื่อเครื่องไม่แรงมาก ขยับลงมาจะพบส่วน “Source” สำหรับเลือกแหล่งสัญญาณว่าจะเปิดไฟล์วิดีโอหรือใช้เว็บแคมตัวใดผ่านหมายเลขดัชนี(Index)

การใช้งานจะถูกกำกับด้วยกลุ่ม “Playback” และ “Run” ผู้ใช้ตั้งความเร็วเป็นอัตราส่วน เช่น 0.25× ถึง 3.0× ถ้าเป็นไฟล์ โปรแกรมจะเร่งด้วยการข้ามเฟรมพร้อมควบคุมเวลาหน่วงต่อเฟรม ส่วนกล้องจริงเร่งเกินขีดจำกัด FPS ไม่ได้แต่ชะลอได้ เมื่อกด Start เธรดประมวลผลจะเริ่มทำงานและสามารถ Pause/Resume/Stop ได้ทันที ปุ่ม Reset Counters ใช้ล้างค่าที่นับทั้งหมดเพื่อเริ่มนับใหม่ และ Snapshot ใช้จับภาพหน้าจอพรีวิวเป็นไฟล์ PNG เพื่อบันทึกหลักฐานหรือรายงาน

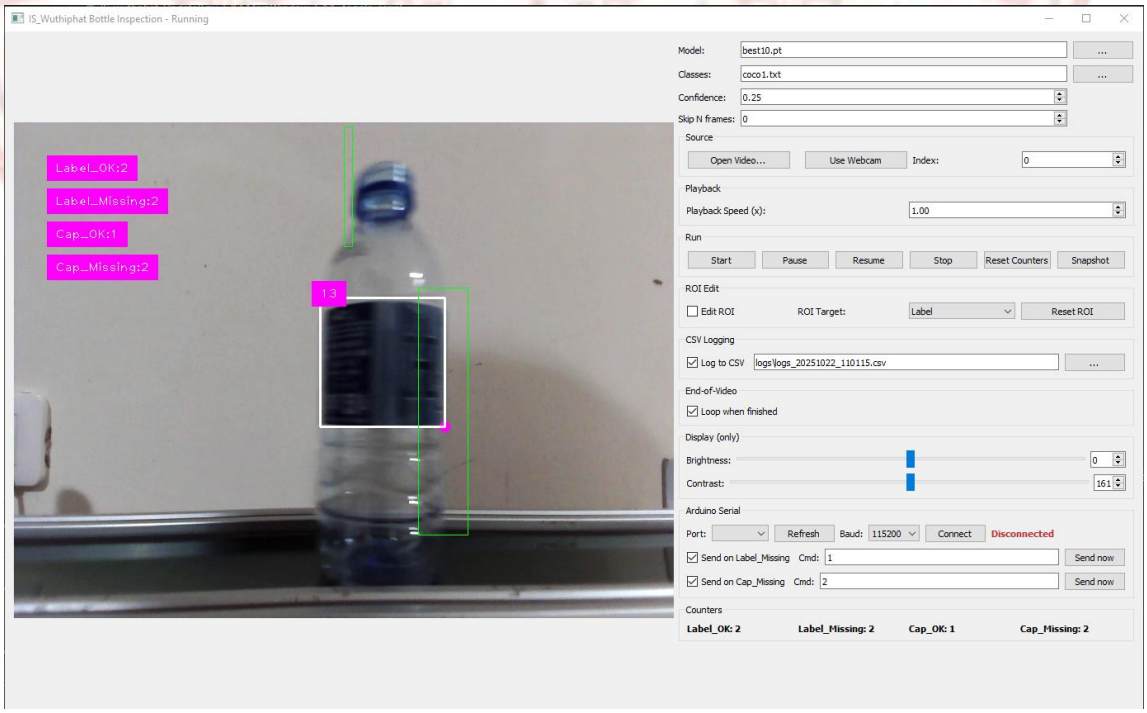
หนึ่งในหัวใจของงานตรวจสอบคือ ROI โปรแกรมเปิดให้แก้ไขได้สะดวกผ่าน “ROI Edit” เพียงติ๊ก Edit ROI แล้วเลือกเป้าหมายว่าจะแก้โซน Label หรือ Cap จากนั้นลากจุดมุมบนภาพด้วยเมาส์ได้โดยตรง เมื่อตั้งค่าพลาดก็คืนสภาพด้วย Reset ROI ได้ในคลิกเดียว การที่ ROI ถูกผูกกับพิกเซลของภาพแสดงผลโดยตรง ช่วยให้สิ่งที่เห็นกับสิ่งที่ใช้คำนวณเป็นฉบับเดียวกัน ลดความสับสนเรื่องสเกลของภาพ

สำหรับการติดตามผลในเชิงคุณภาพและการเก็บบันทึก โปรแกรมมีระบบ “CSV Logging” ที่บันทึกสถิติทุกครั้งที่มีการอัปเดตพร้อมพารามิเตอร์สำคัญ เช่น ชื่อโมเดล ค่าความสว่าง/คอนทราสต์ สถานะพอร์ตอนุกรม ไฟล์ปลายทางเลือกเองได้หรือให้ระบบตั้งชื่อไฟล์อัตโนมัติในโฟลเดอร์มาตรฐานก็ได้ เมื่อวิดีโอเล่นถึงท้ายไฟล์ ผู้ใช้ยังเลือกพฤติกรรมได้ว่าจะวนกลับต้นไฟล์อัตโนมัติหรือหยุดและรอคำสั่งเล่นใหม่ผ่านตัวเลือก “Loop when finished” ซึ่งช่วยให้หน้าจอไม่ว่างเปล่าและการทำงานต่อเนื่องได้ตามโจทย์หน้างาน

เพื่อความอ่านง่ายของภาพในสภาพแสงที่หลากหลาย ส่วน “Display (only)” เปิดให้ปรับ Brightness และ Contrast ของภาพที่แสดงผลแบบเรียลไทม์ โดยจงใจไม่ให้กระทบการรันโมเดล เพื่อคงความสม่ำเสมอของข้อมูลที่ใช้ตัดสินผลจริง ระหว่างที่ผู้ใช้ปรับค่าดังกล่าว เฟรมบนจอก็จะสว่างหรือคมชัดขึ้นตามต้องการ แต่การคัดสินใจ Label_OK หรือ Missing ยังคงอิงข้อมูลดิบจากกล้อง/ไฟล์ตามเดิม

Arduino Serial ผู้ใช้เลือกพอร์ต COM และบอดเรต จากนั้นกด Connect เพื่อเปิดการสื่อสาร เมื่อระบบตรวจพบเหตุการณ์สำคัญอย่าง Label_Missing หรือ Cap_Missing ครั้งแรกของแต่ละไอดี โปรแกรมจะส่งข้อความคำสั่งตามที่กำหนดไว้ไปยัง Arduino โดยอัตโนมัติ นอกจากนี้ยังมีปุ่ม Send now ให้ยิงข้อความทดสอบทันทีโดยไม่ต้องรอเหตุการณ์ เหมาะสำหรับเช็กสายไฟ รีเลย์ หรือการทำงานของอุปกรณ์ต่อพ่วงก่อนขึ้นงานจริง

สุดท้าย แถวล่างของแผงควบคุมแสดงตัวนับสรุปแบบเรียลไทม์อีกชุดหนึ่ง ได้แก่จำนวน Label_OK, Label_Missing, Cap_OK และ Cap_Missing ซึ่งเพิ่มขึ้นก็ต่อเมื่อกรอบวัตถุพร้อมไอดีของมันเข้าสู่ ROI ที่เหมาะสมแล้วเท่านั้น การออกแบบเช่นนี้ทำให้ทีมปฏิบัติการมองเห็นภาพรวมทั้งเชิงภาพและเชิงตัวเลขในหน้าจอเดียว เริ่มจากเลือกโมเดลและแหล่งสัญญาณ ตั้งค่า ROI ให้ตรงงาน ปรับภาพให้ดูง่าย กด Start และถ้าต้องการส่งสั่งอุปกรณ์ก็เชื่อมต่อ Arduino พร้อมเก็บสถิติลง CSV เพื่อการวิเคราะห์ย้อนหลัง



ภาพที่ ก.1 Graphic User Interface (GUI)

ภาคผนวก ข

การเขียนโปรแกรมภาษา Python ที่ใช้ในงานวิจัย

โปรแกรมหลักที่ใช้ในงานวิจัย

ในภาคผนวกนี้ จะแสดงคำสั่ง (Coding) ทั้งหมดที่ใช้ในการพัฒนางานวิจัยนี้ โดยคำสั่งทั้งหมดถูกเขียนด้วยภาษา Python บนโปรแกรม Visual Studio

การนำเข้า Library (Import Library)

```
import sys
import time
import csv
import threading
from datetime import datetime
from pathlib import Path
from typing import Optional, List, Tuple, Set
import cv2
import numpy as np
import pandas as pd
import cvzone
from ultralytics import YOLO
from tracker import *  # ต้องมี class Tracker.update(boxes)->[[x1,y1,x2,y2,id], ...]
```

การนำเข้า Library Pyqt5 (Import Library pyqt5)

```
from PyQt5.QtCore import Qt, QThread, pyqtSignal, QSize, QPoint, QObject
from PyQt5.QtGui import QImage, QPixmap, QPainter, QPen, QBrush
from PyQt5.QtWidgets import (
    QApplication, QWidget, QLabel, QPushButton, QFileDialog,
    QVBoxLayout, QHBoxLayout, QGridLayout, QGroupBox, QSpinBox,
    QDoubleSpinBox, QLineEdit, QMessageBox, QCheckBox, QComboBox, QSlider
)
# ---- pyserial (Arduino) ----
```

```
try:
    import serial
    from serial.tools import list_ports
    SERIAL_AVAILABLE = True
except Exception:
    SERIAL_AVAILABLE = False

TARGET_SIZE = (800, 600)  # (w,h)  ให้การลาก ROI ตรงพิกเซล 1:1
```

การกำหนดกรอบสี่เหลี่ยมสำหรับส่วนที่เป็นฝาและสลาก

```
# ============================== Video + ROI Editor
============================== #
class VideoWidget(QLabel):
    roiLabelChanged = pyqtSignal(list)
    roiCapChanged = pyqtSignal(list)

    def __init__(self, parent=None):
        super().__init__(parent)
        self.setAlignment(Qt.AlignCenter)
        self.setFixedSize(QSize(TARGET_SIZE[0], TARGET_SIZE[1]))
        self.setStyleSheet("background:#111; color:#aaa; border:1px solid #333;")
        self._pix: Optional[QPixmap] = None

        self._edit_mode = False
        self._edit_target = 'Label'  # or 'Cap'
        self._roi_label: List[Tuple[int, int]] = [(490, 200), (490, 499), (550, 499), (550, 200)]
        self._roi_cap:   List[Tuple[int, int]] = [(400,   5), (400, 150), (410, 150), (410,   5)]

        self._drag_idx: int = -1
        self._pick_radius = 10
```

```
    def setFrame(self, qimg: QImage):
        self._pix = QPixmap.fromImage(qimg)
        self._pix = self._pix.scaled(self.size(), Qt.KeepAspectRatio,
Qt.SmoothTransformation)
        self.update()

    def setROIs(self, roi_label: List[Tuple[int, int]], roi_cap: List[Tuple[int, int]]):
        self._roi_label = [tuple(map(int, p)) for p in roi_label]
        self._roi_cap = [tuple(map(int, p)) for p in roi_cap]
        self.update()

    def currentROIs(self):
        return list(self._roi_label), list(self._roi_cap)

    def setEditMode(self, enabled: bool):
        self._edit_mode = enabled
        self._drag_idx = -1
        self.update()

    def setEditTarget(self, target: str):
        self._edit_target = target
        self._drag_idx = -1
        self.update()

    def resetROIs(self, roi_label_default, roi_cap_default):
        self._roi_label = [tuple(p) for p in roi_label_default]
        self._roi_cap   = [tuple(p) for p in roi_cap_default]
        self._drag_idx = -1
        self.roiLabelChanged.emit(list(self._roi_label))
        self.roiCapChanged.emit(list(self._roi_cap))
```

```
        self.update()

    def _active_roi(self) -> List[Tuple[int, int]]:
        return self._roi_label if self._edit_target == 'Label' else self._roi_cap

    def _set_active_roi(self, pts: List[Tuple[int, int]]):
        if self._edit_target == 'Label':
            self._roi_label = pts
            self.roiLabelChanged.emit(list(self._roi_label))
        else:
            self._roi_cap = pts
            self.roiCapChanged.emit(list(self._roi_cap))

    def paintEvent(self, e):
        super().paintEvent(e)
        painter = QPainter(self)
        if self._pix is not None:
            x = (self.width() - self._pix.width()) // 2
            y = (self.height() - self._pix.height()) // 2
            painter.drawPixmap(x, y, self._pix)

        if self._edit_mode:
            pts = self._active_roi()
            if len(pts) >= 2:
                painter.setPen(QPen(Qt.green, 2))
                for i in range(len(pts)):
                    x1, y1 = pts[i]
                    x2, y2 = pts[(i + 1) % len(pts)]
                    painter.drawLine(x1, y1, x2, y2)
            painter.setPen(QPen(Qt.white, 1))
            painter.setBrush(QBrush(Qt.magenta))
```

```
        for (x, y) in pts:
            painter.drawEllipse(QPoint(int(x), int(y)), 5, 5)

    def mousePressEvent(self, ev):
        if not self._edit_mode or ev.button() != Qt.LeftButton:
            return super().mousePressEvent(ev)
        mx, my = ev.x(), ev.y()
        pts = self._active_roi()
        self._drag_idx = -1
        min_d2 = self._pick_radius * self._pick_radius
        for i, (x, y) in enumerate(pts):
            d2 = (mx - x) ** 2 + (my - y) ** 2
            if d2 <= min_d2:
                self._drag_idx = i
                break
        return super().mousePressEvent(ev)

    def mouseMoveEvent(self, ev):
        if not self._edit_mode or self._drag_idx < 0:
            return super().mouseMoveEvent(ev)
        mx = max(0, min(self.width() - 1, ev.x()))
        my = max(0, min(self.height() - 1, ev.y()))
        pts = list(self._active_roi())
        pts[self._drag_idx] = (mx, my)
        self._set_active_roi(pts)
        self.update()
        return super().mouseMoveEvent(ev)

    def mouseReleaseEvent(self, ev):
        if not self._edit_mode:
            return super().mouseReleaseEvent(ev)
```

```
        self._drag_idx = -1
        return super().mouseReleaseEvent(ev)
```

การติดต่อกับบอร์ด Arduino

```
# ============================ Serial Manager
============================ #
class SerialManager(QObject):
    status_changed = pyqtSignal(bool, str)  # (is_open, message)

    def __init__(self, parent=None):
        super().__init__(parent)
        self.ser: Optional[serial.Serial] = None if SERIAL_AVAILABLE else None

    def list_ports(self) -> List[str]:
        if not SERIAL_AVAILABLE:
            return []
        return [p.device for p in list_ports.comports()]

    def open(self, port: str, baud: int) -> bool:
        if not SERIAL_AVAILABLE:
            self.status_changed.emit(False, "pyserial Not Ready (pip install pyserial)")
            return False
        try:
            self.close()
            self.ser = serial.Serial(port=port, baudrate=int(baud), timeout=0.1)
            self.status_changed.emit(True, f"Connected {port} @ {baud}")
            return True
        except Exception as e:
            self.ser = None
            self.status_changed.emit(False, f"Can not open port: {e}")
```

```
            return False

    def close(self):
        try:
            if self.ser and self.ser.is_open:
                self.ser.close()
        except Exception:
            pass
        finally:
            self.ser = None
            self.status_changed.emit(False, "Disconnected")

    def is_open(self) -> bool:
        return bool(self.ser and self.ser.is_open)

    def send_text(self, text: str):
        if not self.is_open():
            return
        try:
            self.ser.write(text.encode("ascii", errors="ignore"))
        except Exception as e:
            self.status_changed.emit(False, f"Can not send data: {e}")
            try:
                self.close()
            except Exception:
                pass
```

Worker Thread

```
# ============================ Worker Thread ==================== #
class VideoWorker(QThread):
```

```
frame_ready = pyqtSignal(QImage)
stats_ready = pyqtSignal(dict)
status_msg = pyqtSignal(str)
finished_ok = pyqtSignal()
failed = pyqtSignal(str)
event_detected = pyqtSignal(str, int)  # ('Label_Missing'/'Cap_Missing', track_id)

def __init__(self, source: str, model_path: str, class_file: str,
             roi_label: List[Tuple[int, int]], roi_cap: List[Tuple[int, int]],
             conf_thres: float = 0.25, skip_n: int = 0,
             target_size: Tuple[int, int] = TARGET_SIZE,
             loop_video: bool = True, parent=None):
    super().__init__(parent)
    self.source = source
    self.model_path = model_path
    self.class_file = class_file

    self._roi_lock = threading.Lock()
    self._vis_lock = threading.Lock()
    self._play_lock = threading.Lock()

    self.roi_label = list(roi_label)
    self.roi_cap = list(roi_cap)

    self._disp_brightness = 0     # beta [-100..100]
    self._disp_contrast = 1.0     # alpha [0.2..3.0]
    self._play_speed = 1.0        # 0.25..3.0
    self._loop_video = bool(loop_video)

    self.conf_thres = conf_thres
    self.skip_n = max(0, int(skip_n))
```

```
        self.target_size = target_size

        self._running = True
        self._paused = False

        self.trk_label_ok = Tracker()
        self.trk_label_missing = Tracker()
        self.trk_cap_ok = Tracker()
        self.trk_cap_missing = Tracker()

        self.seen_label_ok: Set[int] = set()
        self.seen_label_missing: Set[int] = set()
        self.seen_cap_ok: Set[int] = set()
        self.seen_cap_missing: Set[int] = set()

        try:
            with open(self.class_file, "r", encoding="utf-8") as f:
                self.class_names = [ln.strip() for ln in f.read().splitlines() if ln.strip()]
        except Exception as e:
            self.class_names = []
            self.failed.emit(f"อ่าน class file ไม่ได้: {e}")

        try:
            self.model = YOLO(self.model_path)
        except Exception as e:
            self.model = None
            self.failed.emit(f"Can't Load Model: {e}")

    # setters (thread-safe)
    def set_rois(self, roi_label: List[Tuple[int, int]], roi_cap: List[Tuple[int, int]]):
        with self._roi_lock:
```

```
        self.roi_label = list(roi_label)
        self.roi_cap = list(roi_cap)

def set_brightness(self, beta: int):
    beta = max(-100, min(100, int(beta)))
    with self._vis_lock:
        self._disp_brightness = beta

def set_contrast(self, alpha: float):
    alpha = max(0.2, min(3.0, float(alpha)))
    with self._vis_lock:
        self._disp_contrast = alpha

def set_play_speed(self, spd: float):
    spd = max(0.25, min(3.0, float(spd)))
    with self._play_lock:
        self._play_speed = spd

def set_loop(self, flag: bool):
    with self._play_lock:
        self._loop_video = bool(flag)

# control
def stop(self):
    self._running = False

def pause(self, p: bool):
    self._paused = p

def reset_counters(self):
    self.seen_label_ok.clear()
```

```
        self.seen_label_missing.clear()
        self.seen_cap_ok.clear()
        self.seen_cap_missing.clear()

    # helpers
    def _apply_display_params(self, img: np.ndarray, alpha: float, beta: int) ->
np.ndarray:
        return cv2.convertScaleAbs(img, alpha=alpha, beta=beta) if (alpha != 1.0 or beta
!= 0) else img

    def _emit_frame(self, frame_bgr: np.ndarray):
        rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)
        h, w, ch = rgb.shape
        qimg = QImage(rgb.data, w, h, ch * w, QImage.Format_RGB888).copy()
        self.frame_ready.emit(qimg)

    def run(self):
        if self.model is None:
            self.failed.emit("Model Not Ready")
            return

        is_camera = self.source.isdigit()
        cap = cv2.VideoCapture(self.source if not is_camera else int(self.source))
        if not cap.isOpened():
            self.failed.emit(f"Can't Open VDO/Camera: {self.source}")
            return

        fps = cap.get(cv2.CAP_PROP_FPS) or 30.0
        if fps <= 1e-3 or np.isnan(fps):
            fps = 30.0
```

```
self.status_msg.emit("Running")
frame_idx = 0
skip_frac_accum = 0.0

try:
    while self._running:
        if self._paused:
            time.sleep(0.05)
            continue

        t0 = time.time()

        with self._roi_lock:
            roi_label = list(self.roi_label)
            roi_cap = list(self.roi_cap)
        with self._vis_lock:
            alpha = float(self._disp_contrast)
            beta = int(self._disp_brightness)
        with self._play_lock:
            speedf = float(self._play_speed)
            loop_flag = bool(self._loop_video)

        # speed>1: skip เฟรมสำหรับไฟล์วิดีโอ
        if (speedf > 1.0) and (not is_camera):
            base_skip = int(speedf) - 1
            for _ in range(base_skip):
                cap.grab()
            skip_frac_accum += (speedf - int(speedf))
            if skip_frac_accum >= 1.0:
                cap.grab()
                skip_frac_accum -= 1.0
```

```
            grabbed, frame = cap.read()
            if not grabbed:
                if (not is_camera) and loop_flag:
                    cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
                    frame_idx = 0
                    continue
                else:
                    break

            frame = cv2.resize(frame, self.target_size)
            frame_disp = self._apply_display_params(frame, alpha, beta)

            do_infer = (frame_idx % (self.skip_n + 1) == 0)
            frame_idx += 1

            if do_infer:
                try:
                    results = self.model.predict(frame, conf=self.conf_thres,
verbose=False)
                except Exception as e:
                    self.failed.emit(f"Model running Fail: {e}")
                    break

                try:
                    boxes_tensor = results[0].boxes.data
                    df = pd.DataFrame(boxes_tensor).astype("float")
                except Exception:
                    df = pd.DataFrame(columns=[0, 1, 2, 3, 4, 5])

                boxes_lo, boxes_lm, boxes_co, boxes_cm = [], [], [], []
```

```
                for _, row in df.iterrows():
                    x1, y1, x2, y2 = int(row[0]), int(row[1]), int(row[2]), int(row[3])
                    cls_idx = int(row[5]) if 5 in df.columns else int(row.get(5, 0))
                    cls_name = self.class_names[cls_idx] if 0 <= cls_idx <
len(self.class_names) else str(cls_idx)
                    if 'Label_OK' in cls_name: boxes_lo.append([x1, y1, x2, y2])
                    elif 'Label_Missing' in cls_name: boxes_lm.append([x1, y1, x2, y2])
                    elif 'Cap_OK' in cls_name: boxes_co.append([x1, y1, x2, y2])
                    elif 'Cap_Missing' in cls_name: boxes_cm.append([x1, y1, x2, y2])

                trk_lo = self.trk_label_ok.update(boxes_lo)
                trk_lm = self.trk_label_missing.update(boxes_lm)
                trk_co = self.trk_cap_ok.update(boxes_co)
                trk_cm = self.trk_cap_missing.update(boxes_cm)

                # วาด + นับ + แจ้ง event (ครั้งแรกของแต่ละ ID เท่านั้น)
                for xA, yA, xB, yB, tid in trk_lo:
                    if cv2.pointPolygonTest(np.array(roi_label, np.int32), (xB, yB), False)
>= 0:
                        cv2.circle(frame_disp, (xB, yB), 7, (255, 0, 255), -1)
                        cv2.rectangle(frame_disp, (xA, yA), (xB, yB), (255, 255, 255), 2)
                        cvzone.putTextRect(frame_disp, f'{tid}', (xA, yA), 1, 1)
                        self.seen_label_ok.add(tid)

                for xA, yA, xB, yB, tid in trk_lm:
                    if cv2.pointPolygonTest(np.array(roi_label, np.int32), (xB, yB), False)
>= 0:
                        cv2.circle(frame_disp, (xB, yB), 7, (255, 0, 255), -1)
                        cv2.rectangle(frame_disp, (xA, yA), (xB, yB), (255, 255, 255), 2)
                        cvzone.putTextRect(frame_disp, f'{tid}', (xA, yA), 1, 1)
                        if tid not in self.seen_label_missing:
```

```
                    self.event_detected.emit('Label_Missing', tid)
                self.seen_label_missing.add(tid)

            for xA, yA, xB, yB, tid in trk_co:
                if cv2.pointPolygonTest(np.array(roi_cap, np.int32), (xB, yB), False) >=
0:
                    cv2.circle(frame_disp, (xB, yB), 7, (255, 0, 255), -1)
                    cv2.rectangle(frame_disp, (xA, yA), (xB, yB), (255, 255, 255), 2)
                    cvzone.putTextRect(frame_disp, f'{tid}', (xA, yA), 1, 1)
                    self.seen_cap_ok.add(tid)

            for xA, yA, xB, yB, tid in trk_cm:
                if cv2.pointPolygonTest(np.array(roi_cap, np.int32), (xB, yB), False) >=
0:
                    cv2.circle(frame_disp, (xB, yB), 7, (255, 0, 255), -1)
                    cv2.rectangle(frame_disp, (xA, yA), (xB, yB), (255, 255, 255), 2)
                    cvzone.putTextRect(frame_disp, f'{tid}', (xA, yA), 1, 1)
                    if tid not in self.seen_cap_missing:
                        self.event_detected.emit('Cap_Missing', tid)
                    self.seen_cap_missing.add(tid)

            # วาด ROI และ HUD
            with self._roi_lock:
                cv2.polylines(frame_disp, [np.array(self.roi_label, np.int32)], True, (0,
255, 0), 1)
                cv2.polylines(frame_disp, [np.array(self.roi_cap, np.int32)], True, (0, 255,
0), 1)

            cvzone.putTextRect(frame_disp, f'Label_OK:{len(self.seen_label_ok)}', (50,
60), 1, 1)
```

```
                cvzone.putTextRect(frame_disp,
f'Label_Missing:{len(self.seen_label_missing)}', (50, 100), 1, 1)
                cvzone.putTextRect(frame_disp, f'Cap_OK:{len(self.seen_cap_ok)}', (50,
140), 1, 1)
                cvzone.putTextRect(frame_disp,
f'Cap_Missing:{len(self.seen_cap_missing)}', (50, 180), 1, 1)

                self._emit_frame(frame_disp)
                self.stats_ready.emit({
                    "Label_OK": len(self.seen_label_ok),
                    "Label_Missing": len(self.seen_label_missing),
                    "Cap_OK": len(self.seen_cap_ok),
                    "Cap_Missing": len(self.seen_cap_missing),
                })

                # timing ตาม speed
                base_dt = 1.0 / max(fps, 1e-3)
                with self._play_lock:
                    spd = float(self._play_speed)
                target_dt = base_dt / max(spd, 1e-3)
                elapsed = time.time() - t0
                if target_dt > elapsed:
                    time.sleep(target_dt - elapsed)

            cap.release()
            self.finished_ok.emit()
        except Exception as e:
            try:
                cap.release()
            except Exception:
                pass
```

```
            self.failed.emit(str(e))
```

ส่วนนี้ใช้สำหรับสร้าง ส่วนแสดงผลภาพ และ สร้างปุ่มต่างๆ

```
# ============================ Main Window
============================ #
class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("YOLO Conveyor Inspector - PyQt5 (Arduino Serial)")
        self.resize(1380, 820)

        # --- Widgets หลัก --- #
        self.video_widget = VideoWidget()

        # Source
        self.btn_open_video = QPushButton("Open Video...")
        self.btn_open_cam = QPushButton("Use Webcam")
        self.cam_index = QSpinBox(); self.cam_index.setRange(0, 10);
self.cam_index.setValue(0)

        # Model & Class
        self.model_path = QLineEdit("best10.pt")
        self.class_path = QLineEdit("coco1.txt")
        self.btn_browse_model = QPushButton("...")
        self.btn_browse_class = QPushButton("...")

        # Inference params
        self.conf_spin = QDoubleSpinBox(); self.conf_spin.setRange(0.0, 1.0);
self.conf_spin.setSingleStep(0.05); self.conf_spin.setValue(0.25);
self.conf_spin.setDecimals(2)
```

```
        self.skip_spin = QSpinBox(); self.skip_spin.setRange(0, 15);
self.skip_spin.setValue(0)

        # Display
        self.bright_slider = QSlider(Qt.Horizontal); self.bright_slider.setRange(-100, 100);
self.bright_slider.setValue(0)
        self.bright_spin = QSpinBox(); self.bright_spin.setRange(-100, 100);
self.bright_spin.setValue(0)
        self.contrast_slider = QSlider(Qt.Horizontal); self.contrast_slider.setRange(20,
300); self.contrast_slider.setValue(100)
        self.contrast_spin = QSpinBox(); self.contrast_spin.setRange(20, 300);
self.contrast_spin.setValue(100)

        # Speed
        self.speed_spin = QDoubleSpinBox(); self.speed_spin.setRange(0.25, 3.0);
self.speed_spin.setSingleStep(0.25); self.speed_spin.setValue(1.0)

        # Run
        self.btn_start = QPushButton("Start"); self.btn_pause = QPushButton("Pause");
self.btn_resume = QPushButton("Resume")
        self.btn_stop = QPushButton("Stop"); self.btn_reset = QPushButton("Reset
Counters"); self.btn_snapshot = QPushButton("Snapshot")

        # ROI
        self.chk_edit_roi = QCheckBox("Edit ROI")
        self.cbo_roi_target = QComboBox(); self.cbo_roi_target.addItems(["Label", "Cap"])
        self.btn_reset_roi = QPushButton("Reset ROI")

        # CSV
        self.chk_log_csv = QCheckBox("Log to CSV"); self.chk_log_csv.setChecked(True)
        self.csv_path = QLineEdit("")
```

```
        self.btn_browse_csv = QPushButton("...")

        # Loop video
        self.chk_loop = QCheckBox("Loop when finished");
self.chk_loop.setChecked(True)

        # --- Arduino Serial UI --- #
        self.serial_mgr = SerialManager(self)
        self.cbo_port = QComboBox()
        self.btn_refresh_ports = QPushButton("Refresh")
        self.cbo_baud = QComboBox();
self.cbo_baud.addItems(["9600","19200","38400","57600","115200"]);
self.cbo_baud.setCurrentText("115200")
        self.btn_serial_connect = QPushButton("Connect")
        self.lbl_serial_status = QLabel("Disconnected")
        self.lbl_serial_status.setStyleSheet("color:#b33; font-weight:bold;")
        # Event mapping
        self.chk_send_label_missing = QCheckBox("Send on Label_Missing");
self.chk_send_label_missing.setChecked(True)
        self.txt_cmd_label_missing = QLineEdit("1")
        self.btn_send_label_missing = QPushButton("Send now")
        self.chk_send_cap_missing = QCheckBox("Send on Cap_Missing");
self.chk_send_cap_missing.setChecked(True)
        self.txt_cmd_cap_missing = QLineEdit("2")
        self.btn_send_cap_missing = QPushButton("Send now")

        # หากไม่มี pyserial
        if not SERIAL_AVAILABLE:
            self.lbl_serial_status.setText("pyserial not installed")
            self.lbl_serial_status.setStyleSheet("color:#b33; font-weight:bold;")
            self.btn_serial_connect.setEnabled(False)
```

```
            self.btn_send_label_missing.setEnabled(False)
            self.btn_send_cap_missing.setEnabled(False)

        # Layout: top
        top_grid = QGridLayout()
        top_grid.addWidget(QLabel("Model:"), 0, 0); top_grid.addWidget(self.model_path,
0, 1); top_grid.addWidget(self.btn_browse_model, 0, 2)
        top_grid.addWidget(QLabel("Classes:"), 1, 0); top_grid.addWidget(self.class_path,
1, 1); top_grid.addWidget(self.btn_browse_class, 1, 2)
        top_grid.addWidget(QLabel("Confidence:"), 2, 0);
top_grid.addWidget(self.conf_spin, 2, 1)
        top_grid.addWidget(QLabel("Skip N frames:"), 3, 0);
top_grid.addWidget(self.skip_spin, 3, 1)

        # Display group
        bright_row = QHBoxLayout(); bright_row.addWidget(QLabel("Brightness:"));
bright_row.addWidget(self.bright_slider, 1); bright_row.addWidget(self.bright_spin)
        contrast_row = QHBoxLayout(); contrast_row.addWidget(QLabel("Contrast:"));
contrast_row.addWidget(self.contrast_slider, 1);
contrast_row.addWidget(self.contrast_spin)
        display_box = QGroupBox("Display (only)"); vb_disp = QVBoxLayout();
vb_disp.addLayout(bright_row); vb_disp.addLayout(contrast_row);
display_box.setLayout(vb_disp)

        # Source
        src_row = QHBoxLayout(); src_row.addWidget(self.btn_open_video);
src_row.addSpacing(10); src_row.addWidget(self.btn_open_cam);
src_row.addWidget(QLabel("Index:")); src_row.addWidget(self.cam_index)
        src_box = QGroupBox("Source"); src_box.setLayout(src_row)

        # Playback
```

```
        speed_row = QHBoxLayout(); speed_row.addWidget(QLabel("Playback Speed
(x):")); speed_row.addWidget(self.speed_spin)
        playback_box = QGroupBox("Playback"); playback_box.setLayout(speed_row)

        # Run
        run_row = QHBoxLayout()
        for w in (self.btn_start, self.btn_pause, self.btn_resume, self.btn_stop,
self.btn_reset, self.btn_snapshot): run_row.addWidget(w)
        run_box = QGroupBox("Run"); run_box.setLayout(run_row)

        # ROI
        roi_row = QHBoxLayout();
        roi_row.addWidget(self.chk_edit_roi); roi_row.addWidget(QLabel("ROI Target:"));
roi_row.addWidget(self.cbo_roi_target); roi_row.addWidget(self.btn_reset_roi)
        roi_box = QGroupBox("ROI Edit"); roi_box.setLayout(roi_row)

        # CSV
        csv_row = QHBoxLayout(); csv_row.addWidget(self.chk_log_csv);
csv_row.addWidget(self.csv_path, 1); csv_row.addWidget(self.btn_browse_csv)
        csv_box = QGroupBox("CSV Logging"); csv_box.setLayout(csv_row)

        # Loop
        loop_row = QHBoxLayout(); loop_row.addWidget(self.chk_loop)
        loop_box = QGroupBox("End-of-Video"); loop_box.setLayout(loop_row)

        # Stats
        self.lbl_label_ok = QLabel("Label_OK: 0"); self.lbl_label_missing =
QLabel("Label_Missing: 0")
        self.lbl_cap_ok = QLabel("Cap_OK: 0"); self.lbl_cap_missing =
QLabel("Cap_Missing: 0")
```

```
        for lbl in (self.lbl_label_ok, self.lbl_label_missing, self.lbl_cap_ok,
self.lbl_cap_missing): lbl.setStyleSheet("font-weight:bold;")
        stats_row = QHBoxLayout()
        for w in (self.lbl_label_ok, self.lbl_label_missing, self.lbl_cap_ok,
self.lbl_cap_missing): stats_row.addWidget(w)
        stats_box = QGroupBox("Counters"); stats_box.setLayout(stats_row)

        # --- Serial group UI --- #
        serial_top = QHBoxLayout()
        serial_top.addWidget(QLabel("Port:")); serial_top.addWidget(self.cbo_port)
        serial_top.addWidget(self.btn_refresh_ports)
        serial_top.addWidget(QLabel("Baud:")); serial_top.addWidget(self.cbo_baud)
        serial_top.addWidget(self.btn_serial_connect)
        serial_top.addWidget(self.lbl_serial_status, 1)

        serial_ev1 = QHBoxLayout()
        serial_ev1.addWidget(self.chk_send_label_missing)
        serial_ev1.addWidget(QLabel("Cmd:"));
serial_ev1.addWidget(self.txt_cmd_label_missing)
        serial_ev1.addWidget(self.btn_send_label_missing)

        serial_ev2 = QHBoxLayout()
        serial_ev2.addWidget(self.chk_send_cap_missing)
        serial_ev2.addWidget(QLabel("Cmd:"));
serial_ev2.addWidget(self.txt_cmd_cap_missing)
        serial_ev2.addWidget(self.btn_send_cap_missing)

        serial_box = QGroupBox("Arduino Serial")
        vb_serial = QVBoxLayout()
        vb_serial.addLayout(serial_top)
        vb_serial.addLayout(serial_ev1)
```

```
vb_serial.addLayout(serial_ev2)
serial_box.setLayout(vb_serial)

# Right panel
right = QVBoxLayout()
right.addLayout(top_grid)
right.addWidget(src_box)
right.addWidget(playback_box)
right.addWidget(run_box)
right.addWidget(roi_box)
right.addWidget(csv_box)
right.addWidget(loop_box)
right.addWidget(display_box)
right.addWidget(serial_box)
right.addWidget(stats_box)
right.addStretch(1)

main = QHBoxLayout(self)
main.addWidget(self.video_widget, 3)
main.addLayout(right, 2)

# Connections
self.btn_browse_model.clicked.connect(self._browse_model)
self.btn_browse_class.clicked.connect(self._browse_class)
self.btn_open_video.clicked.connect(self._choose_video)
self.btn_open_cam.clicked.connect(self._use_cam)
self.btn_start.clicked.connect(self._start)
self.btn_pause.clicked.connect(self._pause)
self.btn_resume.clicked.connect(self._resume)
self.btn_stop.clicked.connect(self._stop)
self.btn_reset.clicked.connect(self._reset_counters)
```

```
        self.btn_snapshot.clicked.connect(self._snapshot)

        self.chk_edit_roi.toggled.connect(self._toggle_edit_roi)
        self.cbo_roi_target.currentTextChanged.connect(self._change_roi_target)
        self.btn_reset_roi.clicked.connect(self._reset_roi)
        self.video_widget.roiLabelChanged.connect(self._roi_label_changed)
        self.video_widget.roiCapChanged.connect(self._roi_cap_changed)

        self.btn_browse_csv.clicked.connect(self._browse_csv)

        # Display sync
        self.bright_slider.valueChanged.connect(self.bright_spin.setValue)
        self.bright_spin.valueChanged.connect(self.bright_slider.setValue)
        self.bright_spin.valueChanged.connect(self._set_brightness)
        self.contrast_slider.valueChanged.connect(self.contrast_spin.setValue)
        self.contrast_spin.valueChanged.connect(self.contrast_slider.setValue)
        self.contrast_spin.valueChanged.connect(self._set_contrast)
        self.speed_spin.valueChanged.connect(self._set_speed)
        self.chk_loop.toggled.connect(self._set_loop)

        # Serial connections
        self.btn_refresh_ports.clicked.connect(self._refresh_ports)
        self.btn_serial_connect.clicked.connect(self._toggle_serial)
        self.serial_mgr.status_changed.connect(self._on_serial_status)
        self.btn_send_label_missing.clicked.connect(lambda:
self._serial_send(self.txt_cmd_label_missing.text()))
        self.btn_send_cap_missing.clicked.connect(lambda:
self._serial_send(self.txt_cmd_cap_missing.text()))


        # Defaults
```

```
        self.source_str: Optional[str] = "speed7.mp4"
        self.worker: Optional[VideoWorker] = None

        self._roi_label_default = [(490, 200), (490, 499), (550, 499), (550, 200)]
        self._roi_cap_default   = [(400,   5), (400, 150), (410, 150), (410,   5)]
        self.video_widget.setROIs(self._roi_label_default, self._roi_cap_default)

        # CSV state
        self.logs_dir = Path("logs"); self.logs_dir.mkdir(exist_ok=True)
        self._csv_enabled = True
        self._csv_file: Optional[open] = None
        self._csv_writer: Optional[csv.writer] = None

        # Init serial ports list
        self._refresh_ports()

    # --- Source/Model ---
    def _browse_model(self):
        p, _ = QFileDialog.getOpenFileName(self, "Select Model (.pt)", "", "PyTorch
model (*.pt)")
        if p: self.model_path.setText(p)

    def _browse_class(self):
        p, _ = QFileDialog.getOpenFileName(self, "Select Class Names (.txt)", "", "Text
files (*.txt)")
        if p: self.class_path.setText(p)

    def _choose_video(self):
        p, _ = QFileDialog.getOpenFileName(self, "Select Video", "", "Video files (*.mp4
*.avi *.mkv *.mov)")
        if p: self.source_str = p
```

```
    def _use_cam(self):
        self.source_str = str(self.cam_index.value())

    # --- Run controls ---
    def _start(self):
        if self.worker is not None and self.worker.isRunning():
            QMessageBox.information(self, "Info", "กำลังรันอยู่แล้ว"); return
        if not self.source_str:
            QMessageBox.warning(self, "Warning", "ยังไม่ได้เลือกแหล่งวิดีโอ/กล้อง"); return

        model_p = self.model_path.text().strip()
        class_p = self.class_path.text().strip()
        if not Path(model_p).exists():
            QMessageBox.warning(self, "Warning", f"ไม่พบโมเดล: {model_p}"); return
        if not Path(class_p).exists():
            QMessageBox.warning(self, "Warning", f"ไม่พบไฟล์คลาส: {class_p}"); return

        # CSV auto
        self._csv_enabled = self.chk_log_csv.isChecked()
        if self._csv_enabled:
            csv_p = self.csv_path.text().strip()
            if not csv_p:
                csv_p = self.logs_dir /
f"logs_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv"
                self.csv_path.setText(str(csv_p))
            try:
                self._csv_file = open(csv_p, "w", newline="", encoding="utf-8")
                self._csv_writer = csv.writer(self._csv_file)
                self._csv_writer.writerow([
                    "timestamp",
```

```
                "Label_OK","Label_Missing","Cap_OK","Cap_Missing",

"source","model","conf","skip_n","speed_x","alpha_contrast","beta_brightness",
                "roi_label","roi_cap",
                "serial_port","serial_baud"
            ])
        except Exception as e:
            QMessageBox.critical(self, "CSV Error", f"ไม่สามารถเปิดไฟล์ CSV ได้: {e}")
            self._csv_enabled = False; self._csv_file = None; self._csv_writer = None

    rl, rc = self.video_widget.currentROIs()

    self.worker = VideoWorker(
        source=self.source_str,
        model_path=model_p,
        class_file=class_p,
        roi_label=rl,
        roi_cap=rc,
        conf_thres=float(self.conf_spin.value()),
        skip_n=int(self.skip_spin.value()),
        target_size=TARGET_SIZE,
        loop_video=self.chk_loop.isChecked(),
    )
    self.worker.frame_ready.connect(self._update_frame)
    self.worker.stats_ready.connect(self._update_stats)
    self.worker.status_msg.connect(self._status)
    self.worker.finished_ok.connect(self._finished)
    self.worker.failed.connect(self._failed)
    # รับสัญญาณ event จาก worker -> ส่ง Serial ตามการตั้งค่า
    self.worker.event_detected.connect(self._on_worker_event)
```

```
        # init display/speed/loop
        self._set_brightness(self.bright_spin.value())
        self._set_contrast(self.contrast_spin.value())
        self._set_speed(self.speed_spin.value())
        self._set_loop(self.chk_loop.isChecked())

        self.worker.start()
        self._status("Start")

    def _pause(self):
        if self.worker and self.worker.isRunning():
            self.worker.pause(True); self._status("Pause")

    def _resume(self):
        if self.worker and self.worker.isRunning():
            self.worker.pause(False); self._status("Continue")

    def _stop(self):
        if self.worker:
            self.worker.stop(); self.worker.wait(1500); self.worker = None;
self._status("Stoped")
        self._close_csv()

    def _reset_counters(self):
        if self.worker: self.worker.reset_counters()
        self.lbl_label_ok.setText("Label_OK: 0");
self.lbl_label_missing.setText("Label_Missing: 0")
        self.lbl_cap_ok.setText("Cap_OK: 0"); self.lbl_cap_missing.setText("Cap_Missing:
0")

    # --- Display/Speed/Loop handlers ---
```

```
def _set_brightness(self, val: int):
    if self.worker: self.worker.set_brightness(int(val))

def _set_contrast(self, ui_val: int):
    alpha = float(ui_val) / 100.0
    if self.worker: self.worker.set_contrast(alpha)

def _set_speed(self, spd: float):
    if self.worker: self.worker.set_play_speed(float(spd))

def _set_loop(self, flag: bool):
    if self.worker: self.worker.set_loop(bool(flag))

# --- ROI edit ---
def _toggle_edit_roi(self, checked: bool):
    self.video_widget.setEditMode(checked)

def _change_roi_target(self, text: str):
    self.video_widget.setEditTarget(text)

def _reset_roi(self):
    self.video_widget.resetROIs(self._roi_label_default, self._roi_cap_default)
    if self.worker:
        rl, rc = self.video_widget.currentROIs()
        self.worker.set_rois(rl, rc)

def _roi_label_changed(self, pts: list):
    if self.worker:
        rl, rc = self.video_widget.currentROIs()
        self.worker.set_rois(rl, rc)
```

```
def _roi_cap_changed(self, pts: list):
    if self.worker:
        rl, rc = self.video_widget.currentROIs()
        self.worker.set_rois(rl, rc)

# --- CSV ---
def _browse_csv(self):
    p, _ = QFileDialog.getSaveFileName(self, "Select CSV file", "", "CSV files (*.csv)")
    if p: self.csv_path.setText(p)

def _update_stats(self, d: dict):
    self.lbl_label_ok.setText(f"Label_OK: {d.get('Label_OK', 0)}")
    self.lbl_label_missing.setText(f"Label_Missing: {d.get('Label_Missing', 0)}")
    self.lbl_cap_ok.setText(f"Cap_OK: {d.get('Cap_OK', 0)}")
    self.lbl_cap_missing.setText(f"Cap_Missing: {d.get('Cap_Missing', 0)}")

    if self._csv_enabled and self._csv_writer:
        ts = datetime.now().isoformat(timespec='seconds')
        rl, rc = self.video_widget.currentROIs()
        port = self.cbo_port.currentText().strip()
        baud = self.cbo_baud.currentText().strip()
        self._csv_writer.writerow([
            ts,
            d.get('Label_OK', 0),
            d.get('Label_Missing', 0),
            d.get('Cap_OK', 0),
            d.get('Cap_Missing', 0),
            self.source_str or "",
            self.model_path.text().strip(),
            f"{self.conf_spin.value():.2f}",
            int(self.skip_spin.value()),
```

```
                float(self.speed_spin.value()),
                float(self.contrast_spin.value()) / 100.0,  # alpha
                int(self.bright_spin.value()),              # beta
                rl, rc,
                port, baud
            ])
            try: self._csv_file.flush()
            except Exception: pass

    def _update_frame(self, qimg: QImage):
        self.video_widget.setFrame(qimg)

    def _status(self, msg: str):
        self.setWindowTitle(f"IS_Wuthiphat Bottle Inspection - {msg}")

    def _finished(self):
        if not self.chk_loop.isChecked():
            self._status("VDO Finish (Press Start To Continue)")
            QMessageBox.information(self, "Info", "VDO FInished\nPress Start to replay or
change file")
        self.worker = None
        self._close_csv()

    def _failed(self, err: str):
        QMessageBox.critical(self, "Error", err)
        self._status("Error")
        self.worker = None
        self._close_csv()

    def _close_csv(self):
        try:
```

```
            if self._csv_file:
                self._csv_file.flush()
                self._csv_file.close()
        except Exception:
            pass
        self._csv_file = None
        self._csv_writer = None
        self._csv_enabled = False

    # --- Snapshot ---
    def _snapshot(self):
        default_name =
f"snapshot_{datetime.now().strftime('%Y%m%d_%H%M%S')}.png"
        p, _ = QFileDialog.getSaveFileName(self, "Save Snapshot", default_name, "PNG
images (*.png)")
        if not p: p = default_name
        try:
            pix = self.video_widget.grab()
            ok = pix.save(p, "PNG")
            if ok: self._status(f"Saved snapshot: {Path(p).name}")
            else: QMessageBox.warning(self, "Snapshot", "Can't save")
        except Exception as e:
            QMessageBox.critical(self, "Snapshot Error", str(e))

    # --- Serial helpers ---
    def _refresh_ports(self):
        self.cbo_port.clear()
        if SERIAL_AVAILABLE:
            ports = self.serial_mgr.list_ports()
            self.cbo_port.addItems(ports)
        else:
```

```
            self.cbo_port.addItem("N/A")

    def _toggle_serial(self):
        if not SERIAL_AVAILABLE:
            QMessageBox.warning(self, "Serial", "Do not install pyserial: pip install
pyserial")
            return
        if self.serial_mgr.is_open():
            self.serial_mgr.close()
            self.btn_serial_connect.setText("Connect")
        else:
            port = self.cbo_port.currentText().strip()
            baud = int(self.cbo_baud.currentText())
            if not port:
                QMessageBox.warning(self, "Serial", "Please Select Port")
                return
            ok = self.serial_mgr.open(port, baud)
            if ok:
                self.btn_serial_connect.setText("Disconnect")

    def _on_serial_status(self, is_open: bool, message: str):
        self.lbl_serial_status.setText(message)
        self.lbl_serial_status.setStyleSheet("color:#3b3; font-weight:bold;" if is_open
else "color:#b33; font-weight:bold;")

    def _serial_send(self, text: str):
        self.serial_mgr.send_text(text)

    # รับ event จาก worker แล้วเลือกส่ง Serial ตาม setting
    def _on_worker_event(self, etype: str, tid: int):
        if etype == 'Label_Missing' and self.chk_send_label_missing.isChecked():
```

```
            self._serial_send(self.txt_cmd_label_missing.text())
        elif etype == 'Cap_Missing' and self.chk_send_cap_missing.isChecked():
            self._serial_send(self.txt_cmd_cap_missing.text())

    def closeEvent(self, event):
        try:
            if self.worker:
                self.worker.stop()
                self.worker.wait(1500)
        except Exception:
            pass
        try:
            self.serial_mgr.close()
        except Exception:
            pass
        self._close_csv()
        event.accept()

def main():
    app = QApplication(sys.argv)
    w = MainWindow()
    w.show()
    sys.exit(app.exec_())
if __name__ == "__main__":
    main()
```

โปรแกรมคำสั่งย่อยสำหรับหาจุดศูนย์กลางภาพ

คำสั่งส่วนนี้คือโมดูลย่อย (Sub module) ใช้สำหรับหาจุดศูนย์กลางภาพ (Tracker) ซึ่งพัฒนาขึ้นบนบนโปรแกรม Visual Studio

โมดูลย่อย

```
import math
class Tracker:
    def __init__(self):
        # Store the center positions of the objects
        self.center_points = {}
        # Keep the count of the IDs
        # each time a new object id detected, the count will increase by one
        self.id_count = 0
    def update(self, objects_rect):
        # Objects boxes and ids
        objects_bbs_ids = []

        # Get center point of new object
        for rect in objects_rect:
            x, y, w, h = rect
            cx = (x + x + w) // 2
            cy = (y + y + h) // 2

            # Find out if that object was detected already
            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])
                if dist < 35:
                    self.center_points[id] = (cx, cy)
                    #print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break
```

```
        # New object is detected we assign the ID to that object
        if same_object_detected is False:
            self.center_points[self.id_count] = (cx, cy)
            objects_bbs_ids.append([x, y, w, h, self.id_count])
            self.id_count += 1

    # Clean the dictionary by center points to remove IDS not used anymore
    new_center_points = {}
    for obj_bb_id in objects_bbs_ids:
        _, _, _, _, object_id = obj_bb_id
        center = self.center_points[object_id]
        new_center_points[object_id] = center

    # Update dictionary with IDs not used removed
    self.center_points = new_center_points.copy()
    return objects_bbs_ids
```

โปรแกรมสำหรับควบคุมบอร์ด Arduino

คำสั่งในส่วนนี้ แสดงถึงคำสั่งที่ใช้งานบอดร์ควบคุม Auduino เพื่อทำการส่งสัญญาณไปยังตัว Rejecter เพื่อทำการดีดขวดที่มีตำหนิออกจากสายพาน

Arduino Coding

```
// Include the AccelStepper library

#include <AccelStepper.h>

// Define motor interface type (1 for driver)
#define motorInterfaceType 3

// Define step and direction pins
const int stepPin = 9;
const int dirPin = 8;

// Create a stepper object
AccelStepper stepper(motorInterfaceType, stepPin, dirPin);

void setup() {

  Serial.begin(9600); // Initialize serial communication at 9600 baud
  pinMode(13, OUTPUT); // Set pin 13 (built-in LED) as an output

  // Set the maximum speed (steps per second)
  stepper.setMaxSpeed(20000);

  // Set the desired constant speed (steps per second)
```

```
  stepper.setSpeed(-4000);

}

void loop() {
  // Move the stepper motor at the set constant speed
   stepper.runSpeed();
  if (Serial.available() > 0) { // Check if data is available to read
    char command = Serial.read(); // Read the incoming byte


    if (command == '1') {
      //delay(50);
      digitalWrite(13, HIGH); // Turn LED on
      delay(200);
    } else if (command == '0') {
      digitalWrite(13, LOW); // Turn LED off
    }
  }
}
```

ประวัติผู้เขียน

ชื่อ	วุฒิภัทร ณัดฐาปกรณ
ชื่อการค้นคว้าอิสระ	ระบบตรวจจับชิ้นงานเสียบนสายพานลำเลียงอัตโนมัติด้วยการเรียนรู้ของเครื่อง
สาขาวิชา	วิทยาศาสตร์ข้อมูลเพื่อนวัตกรรม
ประวัติ	สำเร็จการศึกษาระดับปริญญาตรี วิศวกรรมศาสตรบัณฑิต สาขาวิชา วิศวกรรมไฟฟ้า ไฟฟ้ากำลัง มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี