**Utilizing YOLO Models for Real-World Scenarios:**

**Assessing Novel Mixed Defect Detection**

**Dataset in PCBs**

**Link [https://ieeexplore.ieee.org/document/10601640]**

**1. Introduction**

แผงวงจรพิมพ์ (PCBs) เป็นส่วนประกอบที่ขาดไม่ได้สำหรับอุปกรณ์อิเล็กทรอนิกส์ ซึ่งช่วยให้การเชื่อมต่อและความสมบูรณ์ของโครงสร้าง มีการใช้งานอย่างกว้างขวางในอุปกรณ์อิเล็กทรอนิกส์หลายประเภท โดยมีการขยายขอบเขตไปไกลกว่าแค่รุ่นพื้นฐาน มูลค่าของตลาด PCB ทั่วโลกในปี 2022 อยู่ที่ 82 พันล้านดอลลาร์ และคาดว่าจะขยายตัวไปถึงประมาณ 140.73 พันล้านดอลลาร์ในปี 2032 อย่างไรก็ตาม เมื่อขนาดของ PCB ลดลง การตรวจจับข้อบกพร่องก็ยิ่งท้าทายมากขึ้น ดังนั้น การใช้เทคนิคการตรวจจับข้อบกพร่องที่แม่นยำระหว่างการผลิตจึงเป็นสิ่งสำคัญเพื่อยกระดับคุณภาพผลิตภัณฑ์และลดค่าใช้จ่ายของบริษัท

รูปที่ 1 แสดงถึงข้อบกพร่องที่พบบ่อยใน PCB ที่พบในอุตสาหกรรมการผลิต วิธีการตรวจจับข้อบกพร่องใน PCB ประกอบด้วยการตรวจสอบด้วยสายตาด้วยมือ การทดสอบทางไฟฟ้า และการตรวจสอบด้วยแสง การตรวจสอบด้วยมือมีประสิทธิภาพต่ำเนื่องจากความแม่นยำและผลผลิตที่ต่ำ การทดสอบทางไฟฟ้าต้องใช้วงจรที่ซับซ้อนและอุปกรณ์ที่มีราคาแพง โดยมีข้อจำกัดในการตรวจจับปัญหาใน PCB หลายชั้นและความเสียหายรองที่อาจเกิดขึ้น การตรวจสอบด้วยแสงอัตโนมัติ (AOI) ใช้กล้องและการประมวลผลภาพในการตรวจจับข้อบกพร่อง เช่น การขาดหายของส่วนประกอบและปัญหาการบัดกรีใน PCB ระหว่างการผลิต ระบบ AOI ที่ใช้กันอย่างแพร่หลายในการผลิตอิเล็กทรอนิกส์ เสนอวิธีการที่ไม่ต้องสัมผัสโดยใช้เทคโนโลยีการมองเห็นของเครื่อง ซึ่งช่วยให้การตรวจจับข้อบกพร่องได้แม่นยำและรวดเร็วกว่าวิธีการอื่น ๆ อย่างไรก็ตาม ระบบเหล่านี้ค่อนข้างช้า มีความไวต่อสภาพแวดล้อม และไม่ค่อยมีประสิทธิภาพในการตรวจจับข้อบกพร่องที่ซับซ้อน

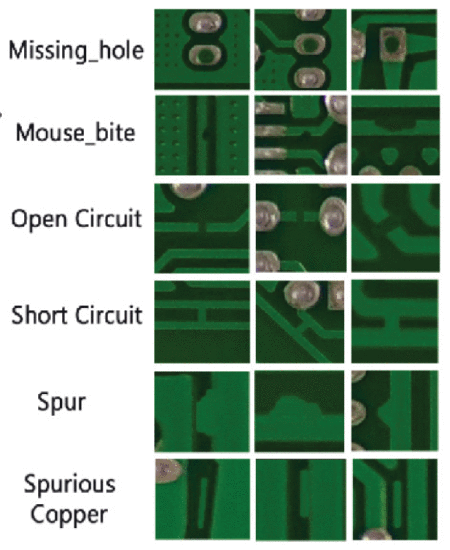


FIGURE 1.

Common PCB defects found in manufacturing industries.

การเรียนรู้เชิงลึก (Deep Learning) โดยเฉพาะอย่างยิ่งเครือข่ายประสาทเทียมแบบคอนโวลูชั่น (CNNs) ได้รับความนิยมในการตรวจจับวัตถุในคอมพิวเตอร์วิชัน เทคนิคการตรวจจับวัตถุใน deep learning แบ่งออกเป็น 2 วิธีหลัก ได้แก่ วิธีสองขั้นตอนและวิธีขั้นตอนเดียว การตรวจจับแบบสองขั้นตอนเกี่ยวข้องกับการสร้างกล่องผู้สมัครที่มีศักยภาพและจากนั้นทำการจัดประเภทโดยใช้ CNNs โดยอัลกอริธึมที่พบบ่อย เช่น RCNN และ Faster R-CNN อย่างไรก็ตาม Faster R-CNN แม้จะมีความแม่นยำ แต่ประสบปัญหาความเร็วในการประมวลผลช้าเนื่องจากส่วนประกอบของเครือข่ายข้อเสนอพื้นที่ (RPN) ซึ่งเป็นอุปสรรคต่อการใช้งานในเวลาจริง

วิธีการตรวจจับแบบขั้นตอนเดียว เช่น SSD และ YOLO แปลงการหาตำแหน่งวัตถุให้เป็นปัญหาการถดถอยโดยตรงโดยไม่ต้องใช้การสุ่มกล่องผู้สมัคร พวกมันมีความแม่นยำสูงกว่าและฝึกได้เร็วกว่าเมื่อเทียบกับวิธีหลายขั้นตอนโดยการจับคุณสมบัติที่สำคัญของภาพในขั้นตอนเดียว อย่างไรก็ตาม SSD อาจประสบปัญหากับการตรวจจับวัตถุขนาดเล็กเนื่องจากการพึ่งพาคุณสมบัติความละเอียดต่ำ ดังนั้น โมเดล YOLO โดยเฉพาะ YOLOv5 [9] จึงได้รับความนิยมสำหรับงานตรวจจับวัตถุเนื่องจากประสิทธิภาพที่สมดุลในแง่ของความเร็ว ความแม่นยำ และความทนทาน

การเข้าถึงชุดข้อมูลที่หลากหลายและครอบคลุมเป็นสิ่งสำคัญในการพัฒนาโมเดลที่มีประสิทธิภาพในการตรวจจับข้อบกพร่องใน PCB อย่างไรก็ตาม ชุดข้อมูลปัจจุบันมักมุ่งเน้นไปที่ประเภทข้อบกพร่องที่เฉพาะเจาะจงและการติดป้ายข้อมูลที่ไม่สมบูรณ์ เพื่อแก้ไขข้อจำกัดนี้ เราได้แนะนำชุดข้อมูล ‘dataset for Mixed Defect Detection in PCB’ (MDD\_PCB) ซึ่งประกอบด้วยข้อบกพร่องผสมใน PCB เพื่อให้ภาพรวมที่ครอบคลุมของสถานการณ์ข้อบกพร่องในโลกจริง ชุดข้อมูล ‘MDD\_PCB’ ประกอบด้วยข้อบกพร่อง PCB ที่ก่อให้เกิดขึ้นโดยเจตนาในพื้นที่ความละเอียด 640×640 พิกเซล (ROI) เพื่อเพิ่มประสิทธิภาพในการฝึกฝนและความสามารถในการประยุกต์ใช้จริงในสถานการณ์โลกจริง ชุดข้อมูลนี้ได้รับการประเมินด้วยโมเดล YOLO และประสบความสำเร็จในการใช้งานสำหรับการอนุมานในเวลาจริงบน Jetson Nano ซึ่งแสดงให้เห็นถึงความสามารถในการตรวจจับที่ดีขึ้น

**2. วัสดุและวิธีการ Materials and Methods**

**2.1. ชุดข้อมูลที่มีอยู่**

ชุดข้อมูล TDD\_PCB ซึ่งได้มาจาก [12] และย่อเป็นชื่อว่า Tiny Defect Detection Printed Circuit Board ประกอบด้วยภาพที่มีการติดป้ายข้อมูลข้อบกพร่องใน PCB จำนวน 693 ภาพ ซึ่งมีความละเอียดสูงถึง 2777×2138 พิกเซล โดยครอบคลุมประเภทข้อบกพร่องต่าง ๆ เช่น รูที่ขาดหาย การกัดของหนู วงจรเปิด ลัดวงจร การปูด และทองแดงผิดปกติ ชุดข้อมูลนี้มีคุณค่าในการศึกษาและพัฒนาอัลกอริธึมการตรวจจับและการจำแนกประเภทที่ออกแบบมาเพื่อการวิเคราะห์ข้อบกพร่องใน PCB ความละเอียดสูงของชุดข้อมูลและการมุ่งเน้นที่ประเภทข้อบกพร่องเดียวอาจทำให้การฝึกฝนช้าลงและจำกัดความหลากหลายของประเภทข้อบกพร่องใน PCB เนื่องจากการติดป้ายข้อมูลไม่ครบถ้วน ซึ่งอาจทำให้มีความต้องการในการคำนวณสูงขึ้นและเวลาฝึกนานขึ้น ส่งผลต่อความแม่นยำในการทดสอบและฝึกฝน

**2.2 ชุดข้อมูลที่เสนอ**

ชุดข้อมูลปัจจุบันที่มีปัญหาดังกล่าวได้รับการพิจารณาว่าไม่เหมาะสมสำหรับการใช้งานในโลกจริง แนวทางที่เสนอคือการสร้างชุดข้อมูลเฉพาะสำหรับการตรวจจับและจำแนกประเภทข้อบกพร่องใน PCB โดยมุ่งเน้นที่ ROI ด้วยการกระตุ้นข้อบกพร่องผสมหลายประเภทโดยเจตนา ข้อบกพร่องที่ถูกกระตุ้นในชุดข้อมูลนี้ช่วยปรับปรุงการฝึกฝนอัลกอริธึมโดยการให้สถานการณ์ที่หลากหลายและสมจริง ซึ่งเป็นสิ่งจำเป็นสำหรับการตรวจจับและจำแนกประเภทข้อบกพร่องใน PCB อย่างแม่นยำในกระบวนการผลิต ชุดข้อมูลนี้ประกอบด้วยภาพขนาด 640×640 พิกเซล ที่ผ่านขั้นตอนการประมวลผลหลายขั้นตอน ได้แก่ การติดป้ายข้อมูลภาพ การเตรียมข้อมูล การตรวจจับข้อบกพร่อง และการจำแนกประเภท ซึ่งช่วยเพิ่มประสิทธิภาพในการฝึกฝนและลดการใช้หน่วยความจำ หลังจากนั้นเราใช้ซอฟต์แวร์ Roboflow สำหรับการติดป้ายข้อมูลเนื่องจากอินเทอร์เฟซที่ใช้งานง่ายและมีความคุ้มค่าด้านต้นทุน ตัวอย่างเช่น รูปที่ 2 แสดงจุดที่ติดป้ายข้อมูลแต่ละจุดที่สร้างขึ้นโดยใช้ Roboflow ซึ่งมีส่วนช่วยในการเพิ่มคุณภาพโดยรวมของชุดข้อมูล การติดป้ายข้อมูลเหล่านี้ถูกบันทึกในรูปแบบ .txt เพื่อใช้งานในภายหลังในการฝึกฝนอัลกอริธึม YOLO เพื่อการประเมินผล ฟีเจอร์ ‘Bounding box annotations tool’ ช่วยเพิ่มความแม่นยำและขยายจำนวนการติดป้ายข้อมูลสำหรับวัตถุเป้าหมาย

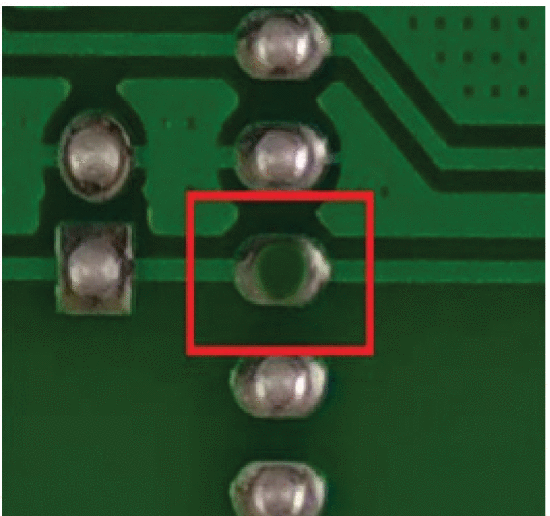


FIGURE 2.

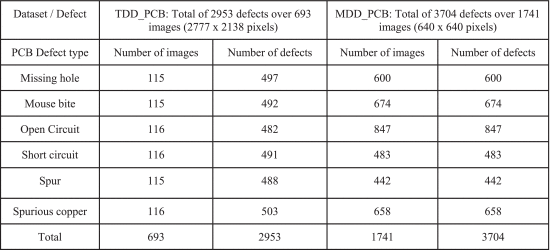
Labeling image in Roboflow bounding box tool.

ชุดข้อมูลที่ได้ประกอบด้วยภาพ PCB จำนวน 1741 ภาพ ซึ่งแต่ละภาพมีข้อบกพร่องสองถึงสามประเภท โดยเสนอชุดข้อมูลการฝึกฝนที่หลากหลายพร้อมการผสมผสานข้อบกพร่องหลายประเภท รวมทั้งหมดมีข้อบกพร่องที่แตกต่างกัน 3704 ข้อ โดยมีการติดป้ายข้อมูลอย่างแม่นยำสำหรับแต่ละภาพที่สะท้อนถึงประเภทข้อบกพร่องที่เฉพาะเจาะจง การจำแนกประเภทเหล่านี้เป็นข้อบกพร่องที่พบบ่อยในการผลิต PCB

**2.3 การเปรียบเทียบชุดข้อมูลที่มีอยู่และชุดข้อมูลที่เสนอ**

ตารางที่ 1 แสดงการกระจายของข้อบกพร่องต่าง ๆ ในชุดข้อมูลพร้อมกับจำนวนภาพที่มีประเภทข้อบกพร่องเฉพาะ ในชุดข้อมูลที่มีอยู่แต่ละภาพมีข้อบกพร่องเดียวหลายตัว ซึ่งส่งผลให้มีภาพ 693 ภาพที่มีข้อบกพร่องรวมทั้งหมด 2953 ข้อ นอกจากนี้ ชุดข้อมูลนี้ยังมีความละเอียดสูง ปัจจัยเหล่านี้ทำให้การเรียนรู้ของโมเดลท้าทายและซับซ้อน ซึ่งทำให้ไม่เหมาะสมกับสถานการณ์ในโลกจริง ในชุดข้อมูลที่เสนอ มีข้อบกพร่องที่ไม่ซ้ำกันหลายประเภทในแต่ละภาพ แต่ละข้อบกพร่องเกิดขึ้นเพียงครั้งเดียวในแต่ละภาพ ดังนั้น ชุดข้อมูลนี้จึงมีภาพ 1741 ภาพและมีข้อบกพร่องรวมทั้งหมด 3704 ข้อ รูปที่ 3 แสดงการเปรียบเทียบเชิงภาพระหว่างชุดข้อมูลที่มีอยู่และชุดข้อมูลที่เสนอ โดยแสดงตัวอย่างภาพจากแต่ละชุดข้อมูลเพื่อการอธิบาย

ตารางที่ 1 การเปรียบเทียบระหว่างชุดข้อมูลที่มีอยู่และชุดข้อมูลที่เสนอ



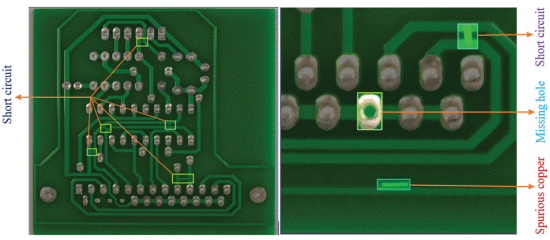


FIGURE 3.

1. TDD\_PCB Dataset (b) MDD\_PCB dataset.

**2.4 การเตรียมข้อมูล Preprocessing**

เพื่อแก้ไขปัญหาการขาดแคลนชุดข้อมูลข้อบกพร่องใน PCB ขนาดใหญ่เนื่องจากข้อจำกัดด้านความลับและค่าใช้จ่าย เราได้ใช้เทคนิคการขยายข้อมูล (Data Augmentation) เพื่อเพิ่มความหลากหลายและขยายชุดข้อมูลของเรา โดยใช้วิธีการขยายข้อมูลแบบดั้งเดิม 6 วิธี ได้แก่ การเพิ่มเสียงรบกวนแบบเกาส์เซียน (Gaussian Noise), การปรับแสง, การหมุนภาพ, การพลิกภาพ, การตัดภาพแบบสุ่ม (Random Cropping), และการย้ายภาพ (Shifting) รูปที่ 4 แสดงตัวอย่างของวิธีการขยายข้อมูลที่ใช้ในการเตรียมชุดข้อมูลนี้

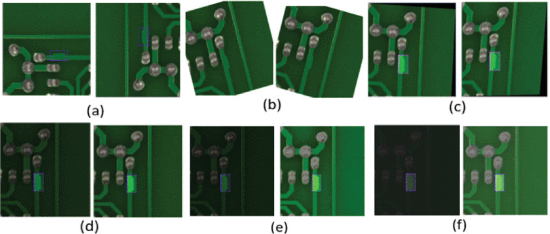


FIGURE 4.

Example of the augmentation methods used in the proposed dataset (a) Flip horizontal/vertical (b) Rotation - range [−14∘ 14∘ ] (c) Shear ±15° (d) Saturation - range [−73 73 ] (%) (e) Brightness - range [-51 51] (%) (f) Exposure - range [-25 25] (%).

เทคนิคการขยายข้อมูลพื้นฐาน เช่น การเพิ่มเสียงรบกวนแบบเกาส์เซียนและการปรับแสง ไม่จำเป็นต้องมีการเปลี่ยนแปลงการติดป้ายข้อมูลสำหรับกล่องขอบเขต (Bounding Boxes) แต่เทคนิคการขยายข้อมูลขั้นสูง เช่น การหมุนภาพ การตัดภาพ และการย้ายภาพ จำเป็นต้องมีการปรับเปลี่ยนการติดป้ายข้อมูลที่เกี่ยวข้อง โดยรวมแล้ว หลังจากการเตรียมข้อมูล ชุดข้อมูลมีภาพทั้งหมด 8705 ภาพและข้อบกพร่องทั้งหมด 18520 ข้อ

ชุดข้อมูลที่ขยายและมีการติดป้ายข้อมูลที่แม่นยำช่วยเพิ่มการนำไปใช้งานจริงโดยการรวมข้อบกพร่องที่สมจริงซึ่งพบในสถานการณ์โลกจริง ทำให้สามารถพัฒนาและประเมินผลโมเดลการตรวจจับและจำแนกข้อบกพร่องใน PCB ได้อย่างมีประสิทธิภาพ หลังจากนั้น ชุดข้อมูล MDD\_PCB ได้รับการประเมินโดยใช้โมเดล YOLO และนำไปใช้งานบน Jetson Nano เพื่อยืนยันถึงความสามารถที่เพิ่มขึ้นในการตรวจจับข้อบกพร่องใน PCB

**3. ระเบียบวิธี Methodology**

**3.1 Data**  ในระหว่างการทดลอง เพื่อให้มั่นใจในความสอดคล้องและความสามารถในการเปรียบเทียบระหว่างชุดข้อมูล TDD\_PCB และ MDD\_PCB เราได้ตัดภาพจากชุดข้อมูล TDD\_PCB โดยเลือกพื้นที่กลางขนาด 1280×1280 พิกเซล และปรับขนาดให้เป็น 640×640 พิกเซล เพื่อให้ตรงกับความละเอียดของ MDD\_PCB ขั้นตอนการเตรียมข้อมูลนี้ช่วยให้ครอบคลุมข้อบกพร่องเกือบทั้งหมดและทำให้ขนาดภาพมีความเป็นมาตรฐาน หลังจากการเตรียมข้อมูล ชุดข้อมูล MDD\_PCB ประกอบด้วยภาพทั้งหมด 8705 ภาพ และชุดข้อมูล TDD\_PCB ประกอบด้วยภาพทั้งหมด 3465 ภาพ โดยทั้งสองชุดข้อมูลมีความละเอียด 640×640 พิกเซล ชุดข้อมูลนี้ได้ถูกแบ่งออกเป็นอัตราส่วน 8:1:1 สำหรับการฝึกฝน การตรวจสอบ และการทดสอบ

**3.2 Model** โมเดล YOLO (You Only Look Once) เป็นที่นิยมสำหรับการตรวจจับวัตถุเนื่องจากความเร็ว ความแม่นยำ และความน่าเชื่อถือของมัน YOLOv5 มีประสิทธิภาพที่ดีกว่าเวอร์ชันก่อนหน้าเนื่องจากสถาปัตยกรรมที่ได้รับการปรับปรุงและการเพิ่มประสิทธิภาพ ทำให้เป็นตัวเลือกที่เหมาะสมสำหรับการตรวจจับข้อบกพร่องใน PCB ในการศึกษานี้ แม้ว่าจะมีการจำลองผลในหลายเวอร์ชัน ตั้งแต่ YOLOv5 ถึง YOLOv8 แต่บทความนี้จะเน้นที่สถาปัตยกรรมของ YOLOv5 เนื่องจากผลการจำลองแสดงว่า YOLOv5 ให้ผลลัพธ์ที่ดีที่สุด

สถาปัตยกรรมของ YOLOv5 ที่แสดงในรูปที่ 5 ประกอบด้วยสามส่วนหลัก ได้แก่ backbone, neck, และ detection heads ซึ่งมีรายละเอียดดังนี้:

1. **Backbone**: ใช้ CSPDarknet53 ซึ่งช่วยในการดึงคุณลักษณะที่ละเอียดในหลายขนาดโดยใช้เครือข่ายที่ฝึกมาแล้ว (pre-trained networks) และปรับปรุงคุณภาพของคุณลักษณะด้วยโครงสร้างที่มุ่งเน้นและเครือข่าย Cross-Stage Partial Network (CSPNet)
2. **Neck**: ใช้เทคนิค Feature Pyramid Network (FPN) และ Path Aggregation Network (PANet) ซึ่งช่วยให้การแบ่งปันฟังก์ชันเชิงพื้นที่มีความคล่องตัวและมีประสิทธิภาพ
3. **Detection Head**: รับผิดชอบในการประเมินความมั่นใจและการถดถอยของกล่องขอบเขต (bounding box) โดยอิงกับ anchor priors

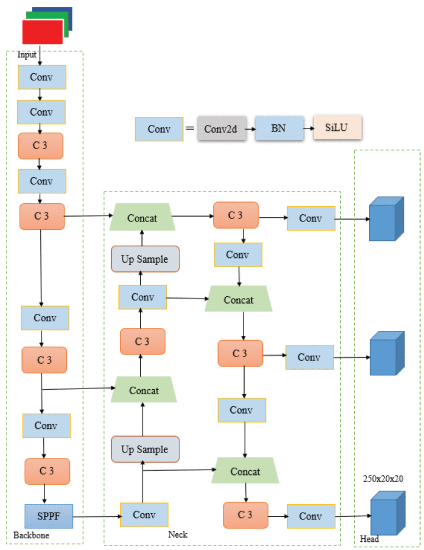


FIGURE 5.

Architecture of Yolov5 model.

โมเดล YOLOv5 ทุกเวอร์ชันมีส่วนประกอบที่เหมือนกัน ซึ่งประกอบด้วย CSP-Darknet53 สำหรับ backbone, SPP (Spatial Pyramid Pooling), PANet ใน neck และ detection head จาก YOLOv4 โมเดล YOLOv5 จะถูกจัดหมวดหมู่ตามขนาด ได้แก่ YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), และ YOLOv5x (extra large) โดยมีการปรับพารามิเตอร์เช่น depth\_multiple และ width\_multiple เพื่อเปลี่ยนแปลงความลึกและความกว้างของเครือข่าย การปรับแต่งเหล่านี้ส่งผลต่อจำนวน Bottleneck Cross-Stage Partials (BCSPs) และคอร์เนลของการคอนโวลูชัน เพื่อให้เหมาะสมกับความต้องการของแต่ละแอปพลิเคชัน

ในทุกรุ่น การเลือก YOLOv5n เป็นสิ่งสำคัญสำหรับการเพิ่มประสิทธิภาพภายใต้ข้อจำกัดของแพลตฟอร์มและการรักษาความสามารถในการประมวลผลแบบเรียลไทม์บน Jetson Nano แนวทางนี้ให้ข้อมูลเชิงลึกที่มีค่าเกี่ยวกับความเหมาะสมและประสิทธิภาพ ดังนั้น โมเดล nano ทุกเวอร์ชัน ตั้งแต่ YOLOv5n ไปจนถึง YOLOv8n จึงได้รับการประเมินเพื่อการวิเคราะห์และเปรียบเทียบเพิ่มเติม

**4. ผลการจำลอง Simulation Results**

โมเดล YOLO ทุกเวอร์ชันตั้งแต่ YOLOv5n ไปจนถึง YOLOv8n ได้รับการทดสอบโดยใช้ตัวปรับความชันสุ่ม (stochastic gradient descent optimizer) โดยมีอัตราการเรียนรู้เริ่มต้นที่ 0.001, ความเร็วโมเมนตัมที่ 0.9, และการลดน้ำหนักที่ 0.0005 สำหรับชุดข้อมูล TDD\_PCB (ชุดข้อมูลที่มีอยู่) และ MDD\_PCB (ชุดข้อมูลที่เสนอ) ชุดข้อมูลถูกแบ่งเป็นชุดการฝึกฝน การตรวจสอบ และการทดสอบในอัตราส่วน 8:1:1 สำหรับการประเมินผล เพื่อให้มั่นใจในการเรียนรู้ที่มีประสิทธิภาพ เราฝึกโมเดลทั้งในแต่ละการทดลองเป็นเวลา 300 epoch เพื่อรักษาความสอดคล้องและปรับแต่งประสิทธิภาพในชุดข้อมูล

**4.1 การฝึกฝน** การเรียนรู้แบบถ่ายทอด (Transfer Learning) ถูกนำมาใช้เพื่อเร่งกระบวนการฝึกฝนและปรับปรุงความแม่นยำในการตรวจจับวัตถุ โดยการใช้โมเดลที่ได้รับการฝึกมาแล้ว โมเดลในงานวิจัยนี้เริ่มต้นจากการฝึกฝนบนชุดข้อมูล COCO แล้วปรับให้เหมาะสมกับการตรวจจับ PCB (แสดงในรูปที่ 6) ซึ่งช่วยเสริมความสามารถในการจดจำและจำแนกข้อบกพร่องใน PCB ในสถานการณ์โลกจริง

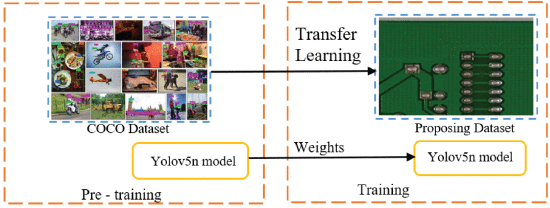
****

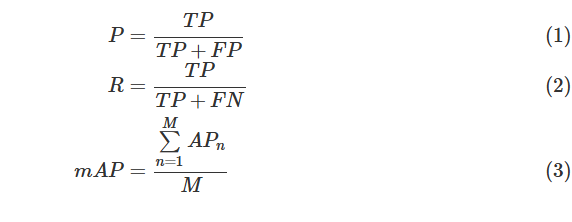
FIGURE 6.

PCB defect detection training process (Transfer learning utilization).

**4.2 เมตริกสำหรับการประเมินผล Evaluation Metrics**

เมตริกที่สำคัญในการประเมินความเหมาะสมของเครือข่ายในการตรวจจับข้อบกพร่องในโลกจริง ได้แก่ ความแม่นยำ (Precision, P), การตรวจจับครบถ้วน (Recall, R), ความเร็ว (ที่วัดโดยจำนวนเฟรมต่อวินาที (FPS)), และความซับซ้อนของเครือข่าย (ประเมินจากจำนวนพารามิเตอร์ทั้งหมด) เมตริกเหล่านี้สะท้อนถึงประสิทธิภาพและความแม่นยำของโมเดลในการใช้งานจริง

การคำนวณ P, R, และค่าเฉลี่ยความแม่นยำ (mean average precision, mAP) สามารถทำได้ดังนี้:



สมการเป็นรูปเด้อ

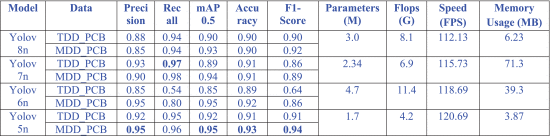
ในที่นี้ M คือจำนวนคลาสทั้งหมดที่พิจารณา และ APn คือค่าเฉลี่ยความแม่นยำ (Average Precision) ของคลาสที่ n ส่วน TP, FN, และ FP คือจำนวนที่ถูกต้อง (true positive), จำนวนที่ไม่ถูกตรวจจับ (false negative), และจำนวนที่ผิดพลาด (false positive) ตามลำดับความเร็วในการตรวจจับ (detection speed) จะวัดจาก FPS (เฟรมต่อวินาที) ซึ่งเป็นตัวบ่งชี้การประมวลผลของโมเดลในเวลาจริง สำหรับ mAP ค่า mAP ที่สูงกว่าจะบ่งชี้ถึงประสิทธิภาพในการตรวจจับที่ดีขึ้น

**4.3 ประสิทธิภาพของโมเดล YOLO Performance of YOLO Models**

การจำลองทั้งหมดได้ดำเนินการบนระบบปฏิบัติการ Ubuntu 22.04.2 LTS, Python 3.8, PyTorch 1.10.0-GPU, CUDA 11.3, และ CUDNN 8.2.2 การทดลองกับชุดข้อมูล TDD\_PCB และ MDD\_PCB ได้ถูกดำเนินการและผลลัพธ์ถูกจัดทำเป็นตารางใน ตารางที่ 2 ตารางนี้ให้ภาพรวมของประสิทธิภาพของแต่ละชุดข้อมูลกับโมเดล YOLO โดยชุดข้อมูลที่เสนอมีผลลัพธ์ที่ดีขึ้นอย่างมีนัยสำคัญเมื่อเทียบกับชุดข้อมูลที่มีอยู่ ซึ่งแสดงให้เห็นถึงประสิทธิภาพในการปรับปรุงความแม่นยำและความน่าเชื่อถือในการตรวจจับข้อบกพร่องของ PCB

นอกจากนี้ การเปรียบเทียบความแม่นยำและ mAP โดยใช้ทั้งสองชุดข้อมูลยังแสดงใน รูปที่ 7 ซึ่งเน้นถึงศักยภาพที่น่าสนใจของชุดข้อมูลที่เสนอในการเสริมสร้างระบบการตรวจจับข้อบกพร่องในโดเมนของ PCB การจัดทำป้ายและการจัดระเบียบคลาสของข้อบกพร่องในชุดข้อมูลที่เสนอมีบทบาทสำคัญในการบรรลุการปรับปรุงนี้ ทำให้ชุดข้อมูลนี้เหมาะสมกับการใช้งานในสภาพแวดล้อมที่มีข้อจำกัดด้านทรัพยากร

TABLE 2 Performance of YOLO Models Using TDD\_PCB and MDD\_PCB



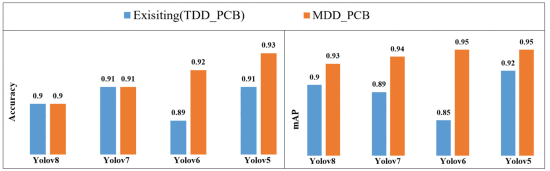


FIGURE 7.

Accuracy and mAP of using TDD\_PCB and MDD\_PCB datasets.

โมเดล YOLOv5 ถูกนำมาใช้เพื่อแสดงการตรวจจับข้อบกพร่องของ PCB โดยใช้ภาพทดสอบจากชุดข้อมูล TDD\_PCB และ MDD\_PCB ซึ่งแสดงใน รูปที่ 8 ผลลัพธ์ที่ได้แสดงให้เห็นถึงความแม่นยำในการตรวจจับที่ดีขึ้น โดยมีคะแนนความมั่นใจที่สูงกว่าคงที่สำหรับ MDD\_PCB เมื่อเทียบกับ TDD\_PCB โดยสามารถทำความเร็วในการตรวจจับได้ที่ 120.69 FPS สำหรับ MDD\_PCB การปรับปรุงในด้านความแม่นยำและความน่าเชื่อถือสามารถอธิบายได้จากการที่ MDD\_PCB มีข้อบกพร่องหลายประเภทที่ถูกเพิ่มเข้ามาโดยเจตนา ซึ่งให้สภาพแวดล้อมในการเรียนรู้ที่หลากหลายมากขึ้นสำหรับโมเดล และแสดงให้เห็นถึงความก้าวหน้าอย่างมากในการตรวจจับข้อบกพร่องของ PCB ดังนั้น ชุดข้อมูลที่เสนอ MDD\_PCB จึงเป็นมาตรฐานที่แม่นยำและครอบคลุมมากขึ้นสำหรับการประเมินแนวทางการตรวจจับข้อบกพร่องของ PCB

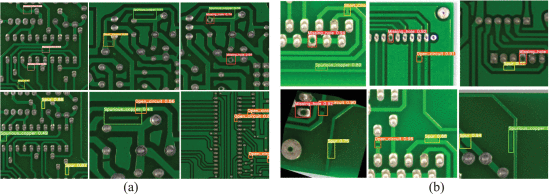


FIGURE 8.

Illustration of PCB defect detection in (a) TDD\_PCB and (b) MDD\_PCB using YOLOv5.

นอกจากนี้ เราทดสอบประสิทธิภาพและการใช้งานหน่วยความจำของโมเดล YOLO ทั้งหมด และผลลัพธ์ถูกจัดทำเป็นตารางใน ตารางที่ 2 ตารางนี้แสดงให้เห็นว่า YOLOv6n, YOLOv7n, และ YOLOv8n แสดงค่าจำนวนเฟรมต่อวินาที (FPS) ที่ลดลงเรื่อยๆ พร้อมกับเวลาในการอนุมานที่เพิ่มขึ้น โดยในกลุ่มนี้ YOLOv7n มีการใช้งานหน่วยความจำสูงสุดที่ 71.3 MB ถึงแม้ว่าเช่นนั้น โมเดลทั้งหมดยังคงรักษาความเร็วในการประมวลผลสูงไว้ได้ โดย YOLOv5n สามารถทำประสิทธิภาพรวมได้สูงสุด ทำให้เหมาะสมสำหรับการใช้งานในระบบที่ต้องการการประมวลผลแบบเรียลไทม์

ผลลัพธ์เหล่านี้แสดงให้เห็นว่า YOLOv5n มี FPS ที่เร็วที่สุดที่ 120.69 และการใช้หน่วยความจำต่ำที่สุดที่ 3.87 MB ซึ่งทำให้มันเป็นโมเดลที่มีประสิทธิภาพสูงสุดทั้งในด้านความเร็วและการใช้ทรัพยากร ทั้งนี้มาจากการออกแบบที่ได้รับการปรับแต่งให้มีประสิทธิภาพในการคำนวณพร้อมกับการจัดการทรัพยากรที่มีประสิทธิผล

จากทั้งหมดนี้ YOLOv5n แสดงประสิทธิภาพที่ดีที่สุด โดยมีพารามิเตอร์และการคำนวณ (FLOPs) ที่น้อยที่สุด ทำให้มันเหมาะสมสำหรับการใช้งานในสถานการณ์ที่ต้องการการประมวลผลแบบเรียลไทม์ ถัดไปเราจะสาธิตการใช้งานระบบตรวจจับข้อบกพร่องของ PCB ในเวลาจริง โดยการติดตั้งโมเดลที่ฝึกเสร็จแล้วบนบอร์ด Jetson Nano และการปรับแต่งโมเดลด้วยสคริปต์ Python ที่ได้รับการปรับให้เหมาะสมกับความสามารถของ CUDA บนบอร์ด

**5. ผลการทดลอง Experimental Results**

**5.1 รายละเอียดการติดตั้งระบบตรวจจับข้อบกพร่อง PCB แบบเรียลไทม์**

เราใช้บอร์ด Jetson Nano ในการติดตั้งโมเดลที่เลือก โดย Nvidia Jetson Nano เป็นบอร์ดขนาดกะทัดรัดและประหยัดพลังงาน มีโปรเซสเซอร์แบบควอดคอร์, GPU แบบ 128-core Max, และ CPU แบบ ARM Cortex-A57 พร้อมแบนด์วิธหน่วยความจำที่ 25.6 GB/s ซึ่งช่วยให้การสื่อสารระหว่าง CPU และ GPU มีความเร็วสูง [27] การใช้พลังงานที่ต่ำและแบนด์วิธหน่วยความจำที่สูงทำให้บอร์ดนี้เหมาะสมกับการติดตั้งระบบ AI

ในการติดตั้งระบบตรวจจับข้อบกพร่อง PCB แบบเรียลไทม์ เราเลือกใช้โมเดล YOLOv5n เนื่องจากมีประสิทธิภาพที่เหนือกว่าเมื่อเปรียบเทียบกับโมเดลอื่น ๆ โดยมีพารามิเตอร์และการคำนวณ (flops) น้อยที่สุด ซึ่งเหมาะสำหรับการใช้งานในสถานการณ์แบบเรียลไทม์ เราใช้ Lenovo FHD webcam ในการถ่ายภาพสตรีมสดของ PCB ที่จะแสดงผลบนจอภาพ Nvidia Jetson Nano ทำหน้าที่เป็นหน่วยประมวลผล และมีจอภาพเพิ่มเติมเพื่อแสดงผลการตรวจจับและการจำแนกข้อบกพร่องของ PCB การติดตั้งระบบตรวจจับข้อบกพร่อง PCB แบบเรียลไทม์ที่ครบถ้วนตามรายละเอียดที่กล่าวถึงนี้แสดงใน รูปที่ 9

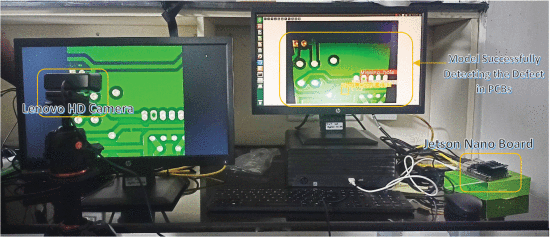


FIGURE 9.

Implementation of real-time PCB defect detection system.

**5.2 ประสิทธิภาพของระบบตรวจจับข้อบกพร่อง PCB แบบเรียลไทม์**

กระบวนการติดตั้งรวมถึงการปรับแต่งและปรับโมเดล YOLOv5n ที่เลือกให้เหมาะสมกับบอร์ด Jetson Nano โดยใช้สคริปต์ Python ที่ได้รับการปรับให้เหมาะสมกับ CUDA ภาพ PCB ที่ถ่ายโดย Lenovo FHD webcam ถูกนำมาจากการแสดงสดบนจอภาพ เรามุ่งหวังที่จะประเมินประสิทธิภาพของโมเดลเมื่อทำการติดตั้งบนบอร์ด Jetson Nanoเพื่อให้ได้การอนุมานที่ดีที่สุด เราได้แปลงโมเดลจากการฝึกใน PyTorch เป็น TensorRT การเปลี่ยนแปลงนี้ประสบความสำเร็จและทำให้โมเดลมีขนาดเล็กลงอย่างมาก ลดการใช้หน่วยความจำและความต้องการในการคำนวณได้อย่างมาก ส่งผลให้ได้โมเดลที่มีขนาดเพียง 3.87 MB และเวลาในการอนุมานที่ 33.32 ms

การตรวจจับข้อบกพร่อง PCB แบบเรียลไทม์โดยใช้ Jetson Nano และ YOLOv5 ที่ได้รับการปรับแต่ง (นำมาใช้เป็นโมเดล TensorRT) แสดงให้เห็นถึงการปรับปรุงที่สำคัญในด้านความแม่นยำในการตรวจจับและคะแนนความมั่นใจ ซึ่งแสดงให้เห็นใน รูปที่ 10 ความสามารถในการคำนวณของ Jetson Nano ช่วยให้สามารถทำการอนุมานได้อย่างรวดเร็วในการระบุข้อบกพร่อง ทำให้มันเหมาะสมกับการใช้งานในสภาพแวดล้อมอุตสาหกรรมที่ต้องการการตรวจจับข้อบกพร่องที่รวดเร็วและแม่นยำ ผลลัพธ์ยังแสดงให้เห็นถึงทางออกที่มีประสิทธิภาพในด้านต้นทุนและมีความเหมาะสมสำหรับการตรวจจับข้อบกพร่อง PCB ที่ตอบสนองความต้องการที่สำคัญในกระบวนการควบคุมคุณภาพการผลิต และแสดงให้เห็นถึงความเป็นไปได้ในการตรวจจับวัตถุแบบเรียลไทม์บนระบบคอมพิวติ้งแบบขอบ (Edge Computing) เช่น Jetson Nano

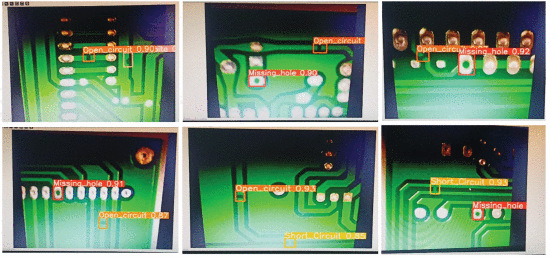


FIGURE 10.

Performance of the real-time PCB defect detection system using the MDD\_PCB dataset.

**6. Conclusion**

ในงานวิจัยนี้ เราได้แนะนำชุดข้อมูล MDD\_PCB ซึ่งเป็นชุดข้อมูลสำหรับการตรวจจับข้อบกพร่องแบบผสมใน PCB เพื่อตอบสนองข้อจำกัดที่พบในชุดข้อมูลปัจจุบันสำหรับการตรวจจับข้อบกพร่องใน PCB ชุดข้อมูลที่มีอยู่ในปัจจุบันมักเน้นประเภทข้อบกพร่องเดียวและมักขาดการทำเครื่องหมายรายละเอียดอย่างเพียงพอ ในทางตรงกันข้าม ชุดข้อมูล MDD\_PCB เอาชนะปัญหานี้โดยการแทรกข้อบกพร่องแบบผสมเข้าไปในภาพ โดยการรวมข้อบกพร่องแบบผสม ชุดข้อมูล MDD\_PCB ที่เสนอมีการแสดงผลที่ครอบคลุมมากขึ้นเกี่ยวกับสถานการณ์ข้อบกพร่องที่พบในอุตสาหกรรมการผลิต ซึ่งช่วยแก้ไขข้อบกพร่องของชุดข้อมูลข้อบกพร่อง PCB ที่มีอยู่ ชุดข้อมูลนี้ถูกประเมินโดยใช้ YOLO models โดยเฉพาะ YOLOv5 เมื่อเปรียบเทียบกับโมเดลอื่น ๆ ซึ่งแสดงให้เห็นถึงความแม่นยำสูงในการตรวจจับข้อบกพร่อง ซึ่งสามารถเห็นได้จากตัวชี้วัดต่าง ๆ เช่น precision, recall, mAP, และ F1-score ความหลากหลายที่เพิ่มขึ้นของข้อบกพร่องในชุดข้อมูลช่วยให้โมเดลสามารถเรียนรู้คุณสมบัติที่แข็งแกร่งและสามารถทั่วไปได้ดีขึ้น ในขณะที่ความละเอียดที่ได้มาตรฐานช่วยในการฝึกโมเดลและเพิ่มความสามารถในการทั่วไป

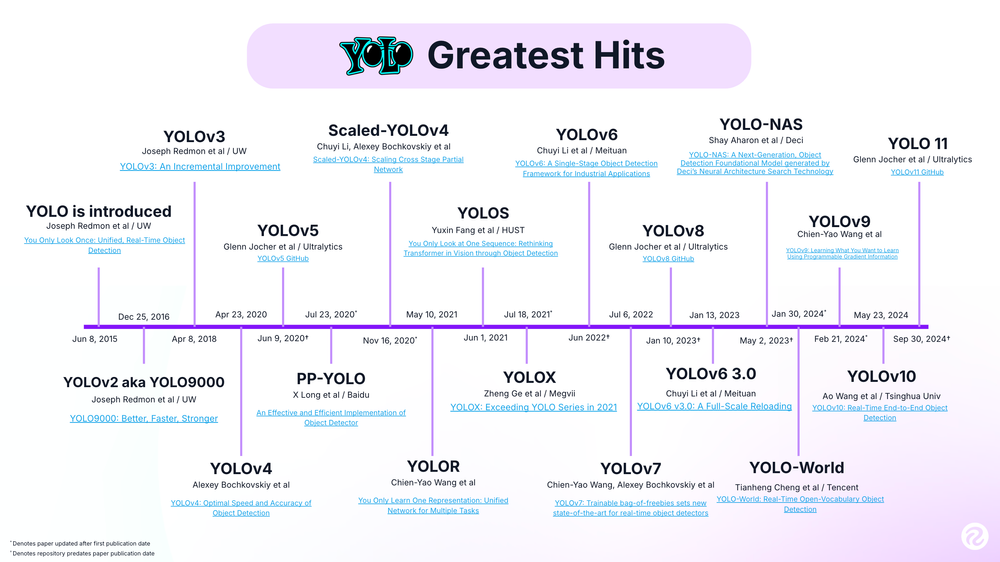
การใช้งานโมเดลที่ประสบความสำเร็จในการติดตั้งบน Jetson Nano แสดงให้เห็นถึงความสามารถในการใช้งานในสถานการณ์การใช้งานจริงได้อย่างมีประสิทธิภาพ ความก้าวหน้าที่ได้จากชุดข้อมูลที่นำเสนอช่วยพัฒนาวิธีการตรวจจับข้อบกพร่องใน PCB โดยการเสนอชุดข้อมูลที่มีการสร้างข้อบกพร่องแบบผสมอย่างจงใจ ซึ่งเป็นสิ่งสำคัญในระบบการตรวจจับข้อบกพร่องที่มีความทนทานและน่าเชื่อถือ วิธีการเหล่านี้สามารถนำไปใช้ในอุตสาหกรรมการผลิตอิเล็กทรอนิกส์ การควบคุมคุณภาพ และกระบวนการตรวจสอบอัตโนมัติ ซึ่งช่วยเพิ่มความแม่นยำและความน่าเชื่อถือในการตรวจจับข้อบกพร่องงานในอนาคตจะมุ่งเน้นที่การขยายความหลากหลายและรูปแบบของข้อบกพร่องในชุดข้อมูล เพื่อปรับปรุงโมเดลโดยการลดพารามิเตอร์และการคำนวณจุดทศนิยม ซึ่งจะช่วยให้ได้โมเดลที่มีประสิทธิภาพสูงขึ้นและมีขนาดเบากว่า นอกจากนี้ งานในอนาคตยังจะสำรวจการบูรณาการกับเทคโนโลยีที่เกิดขึ้นใหม่ เช่น ปัญญาประดิษฐ์ (AI) และการคำนวณบนขอบ (Edge Computing) การร่วมมือกับพันธมิตรในอุตสาหกรรมจะเป็นกุญแจสำคัญในการทดสอบความก้าวหน้าเหล่านี้ในสภาพแวดล้อมการผลิตจริง เพื่อส่งเสริมการยอมรับและเพิ่มผลกระทบสูงสุด

**YOLO**

YOLO (You Only Look Once) เป็นกลุ่มโมเดลคอมพิวเตอร์วิชันที่ได้รับความสนใจเป็นอย่างมากนับตั้งแต่ Joseph Redmon, Santosh Divvala, Ross Girshick และ Ali Farhadi แนะนำสถาปัตยกรรมใหม่นี้ในงาน CVPR โดยโมเดลนี้ได้รับรางวัล OpenCV's People Choice Awards

**ต้นกำเนิดของ YOLO**

YOLO (You Only Look Once) พัฒนาขึ้นขึ้นโดย Joseph Redmon บนเฟรมเวิร์กที่เขาออกแบบเองชื่อว่า Darknet ซึ่งเป็นเฟรมเวิร์กสำหรับงานวิจัยที่ยืดหยุ่นและพัฒนาด้วยภาษาโปรแกรมระดับต่ำ Darknet ได้รับการพัฒนามาอย่างต่อเนื่องและได้สร้างโมเดลตรวจจับวัตถุแบบเรียลไทม์ที่ดีที่สุดในสาย Computer Vision: YOLO, YOLOv2, YOLOv3 , YOLOv4 , YOLOv5 , YOLOV6 , YOLOV7 , YOLOv8 , YOLO - NAS , YOLO-World , YOLOv9 , YOLOv10และYOLOv11

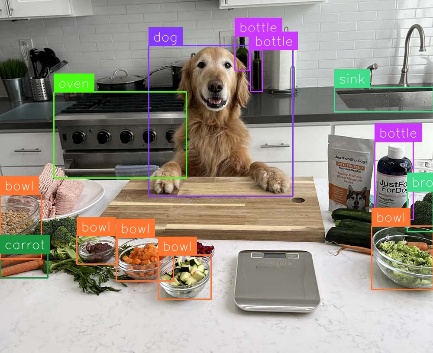


ภาพ ตระกูลโมเดล YOLO

ตระกูลโมเดล YOLO (You Only Look Once) ได้มีการพัฒนามาอย่างต่อเนื่องตั้งแต่เปิดตัวครั้งแรกในปี 2016 โดยเฉพาะ YOLOv2 และ YOLOv3 ซึ่งพัฒนาโดย Joseph Redmon ส่วนโมเดล YOLO ที่ออกมาหลังจาก YOLOv3 ได้รับการพัฒนาโดยผู้เขียนคนใหม่ และไม่ได้ถูกมองว่าเป็นการพัฒนาต่อเนื่องโดยตรงจาก YOLOv3 แต่ละเวอร์ชันมีเป้าหมายที่แตกต่างกันตามแนวคิดของผู้พัฒนา โมเดล YOLO ดั้งเดิมถือเป็นเครือข่ายตรวจจับวัตถุ (Object Detection) ตัวแรกที่สามารถรวมปัญหาการวาดกรอบ (Bounding Box) และการระบุป้ายกำกับคลาส (Class Labels) เข้าไว้ในเครือข่ายแบบ End-to-End Differentiable ได้สำเร็จในขณะที่บางโมเดลตรวจจับวัตถุจะแบ่งการทำงานออกเป็นสองส่วน ได้แก่ การระบุบริเวณที่น่าสนใจ(Bounding Box) ซึ่งเป็นที่ตั้งของวัตถุ และการจัดประเภท (Classify) บริเวณที่ระบุไว้นั้น วิธีนี้เรียกว่า Two-Stage Detector โดยโมเดลยอดนิยม เช่น Faster R-CNN ใช้วิธีการนี้ในการตรวจจับวัตถุ

**การเปรียบเทียบโมเดล YOLO**

ได้พัฒนาสร้างแพลตฟอร์มแบบอินเทอร์แอคทีฟที่สามารถเปรียบเทียบโมเดล YOLO หลายรุ่นได้ โดยแพลตฟอร์มนี้ถูกตั้งค่าให้ใช้จุดตรวจสอบพื้นฐาน (Base Checkpoints) ที่ผ่านการฝึกบนชุดข้อมูล Microsoft COCO สำหรับ YOLOv8, YOLOv9, YOLOv10 และ YOLOv11 ระบบสามารถตรวจจับคลาสจาก COCO เช่น แมว สุนัข และรถยนต์



V8 visual V9 visual



V11 visual V11 visual

(พี่มิ้นสามารถ แคป 4 รูปแล้วเขียนว่า “รูปการเปรียบเทียบโมเดล v8 – v11”)

**อัลกอริทึมและสถาปัตยกรรม YOLO**

YOLO เป็นตัวตรวจจับแบบขั้นตอนเดียว (single-stage detector) ซึ่งจัดการทั้งการระบุวัตถุและการจัดประเภทในครั้งเดียวผ่านเครือข่าย YOLO ไม่ใช่โมเดลการตรวจจับแบบขั้นตอนเดียวเพียงตัวเดียว (เช่น MobileNetSSDv2 ก็เป็นตัวตรวจจับแบบขั้นตอนเดียวที่ได้รับความนิยม) แต่โดยทั่วไป YOLO มีประสิทธิภาพที่ดีกว่าในด้านความเร็วและความแม่นยำ ด้วยการจัดการงานตรวจจับเป็นการถดถอยแบบขั้นตอนเดียวสำหรับการระบุกรอบล้อมวัตถุ (bounding boxes) โมเดล YOLO มักจะเร็วและมีขนาดเล็กมาก ซึ่งช่วยให้ฝึกได้เร็วขึ้นและนำไปใช้งานได้ง่ายขึ้น โดยเฉพาะอย่างยิ่งบนอุปกรณ์ที่มีทรัพยากรจำกัด

อัลกอริทึม YOLO ใช้สำหรับการตรวจจับวัตถุแบบเรียลไทม์ ก่อนหน้าที่จะมี YOLO, R-CNNs เป็นหนึ่งในวิธีการที่ใช้กันทั่วไปในการตรวจจับวัตถุ แต่มีความล่าช้าและไม่เหมาะสำหรับการใช้งานแบบเรียลไทม์ YOLO ให้ความเร็วที่จำเป็นสำหรับกรณีการใช้งานที่ต้องการการวิเคราะห์ที่รวดเร็ว เช่น การตรวจจับรถยนต์ การระบุสัตว์ และการตรวจสอบการละเมิดความปลอดภัย

ตัวอย่างสถานการณ์ที่ YOLO อาจมีประโยชน์:

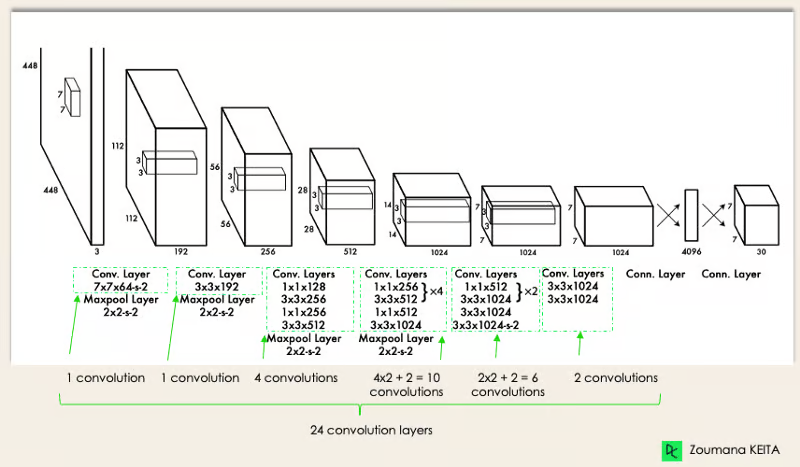
1. ระบุผู้บุกรุกในโรงงาน
2. ตรวจสอบการเคลื่อนที่ของยานพาหนะในสถานที่ก่อสร้าง
3. วิเคราะห์รูปแบบการจราจรบนถนน (เช่น หาช่วงเวลาที่ถนนมีการใช้งานมากที่สุดและน้อยที่สุด)
4. ระบุควันจากไฟในป่า
5. ตรวจสอบให้แน่ใจว่าคนงานสวมอุปกรณ์ป้องกันส่วนบุคคลที่เหมาะสมในบางสถานการณ์ (เช่น ขณะใช้เครื่องมือหรือทำงานกับสารเคมีที่ปล่อยควันอันตราย)

**-------------------------สิ้นสุด ----------------------------------------------------------------------**

**link** <https://blog.roboflow.com/guide-to-yolo-models/>

โจเซฟ เนลสัน (9 ม.ค. 2025) YOLO

สถาปัตยกรรมของ YOLO มีความคล้ายคลึงกับ GoogleNet โดยประกอบไปด้วยเลเยอร์คอนโวลูชั่น 24 ชั้น, เลเยอร์ Max-Pooling 4 ชั้น, และเลเยอร์ Fully Connected 2 ชั้น



YOLO Architecture from the original paper (Modified by Author)

**ขั้นตอนการทำงานของสถาปัตยกรรม YOLO:**

1. **การปรับขนาดภาพ**: ภาพอินพุตจะถูกปรับขนาดเป็น 448x448 ก่อนที่จะผ่านเข้าสู่เครือข่ายคอนโวลูชั่น
2. **การคอนโวลูชั่น**: เริ่มด้วยการใช้คอนโวลูชั่นขนาด 1x1 เพื่อ ลดจำนวนช่องสัญญาณ (channels) ตามด้วยคอนโวลูชั่นขนาด 3x3 เพื่อสร้างเอาต์พุตที่เป็นรูปทรงสี่เหลี่ยม
3. **ฟังก์ชันการเปิดใช้งาน**: ฟังก์ชันการเปิดใช้งานที่ใช้คือ ReLU ยกเว้นสำหรับเลเยอร์สุดท้ายที่ใช้ฟังก์ชันเปิดใช้งานแบบเชิงเส้น (linear activation function)
4. **การใช้เทคนิคเพิ่มเติม**: มีการใช้เทคนิคต่างๆ เช่น Batch Normalization และ Dropout เพื่อทำให้โมเดลไม่ overfitting

[**https://www.datacamp.com/blog/yolo-object-detection-explained**](https://www.datacamp.com/blog/yolo-object-detection-explained)

-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**YOLOv11**

YOLO เป็นหนึ่งในวิธีการที่โดดเด่นที่ใช้สำหรับการตรวจจับวัตถุ มันใช้แนวทางแบบขั้นตอนเดียว (one-stage approach) ที่ประมวลผลภาพทั้งหมดในรอบเดียวเพื่อทำนายกรอบขอบเขต (bounding boxes) และความน่าจะเป็นของคลาส (class probabilities) YOLO ได้รับการพัฒนาอย่างมาก โดยเริ่มต้นจากเวอร์ชันแรกจนถึงเวอร์ชันล่าสุด (เช่น YOLOv11) ที่สร้างโดย Ultralytics

YOLOv11 เป็นเวอร์ชันล่าสุดในตระกูล YOLO ที่นำเสนอการพัฒนาอย่างมากในด้านความเร็ว, ความแม่นยำ, และการดึงคุณลักษณะ (feature extraction) โครงสร้างของ YOLOv11 แสดงในรูปที่ 2 ซึ่งเน้นส่วนประกอบหลักของโมเดล โดยทั่วไปแล้ว ประกอบไปด้วยสามส่วนหลัก: backbone, neck, และ head ด้านล่างนี้จะอธิบายแต่ละส่วนและคุณลักษณะที่ถูกเพิ่มเข้ามาเพื่อเสริมสร้างโครงสร้างโดยรวม

**Backbone**: ส่วนประกอบหลักแรกของ YOLOv11 คือ backbone ซึ่งรับผิดชอบในการดึงคุณลักษณะสำคัญจากภาพอินพุตในหลายระดับ (scales) ส่วนนี้ประกอบด้วยหลายๆ Conv block โดยแต่ละบล็อกประกอบด้วยสามส่วนย่อย ดังแสดงในรูปที่ 2(b): Conv2D, BatchNorm2D, และฟังก์ชันการเปิดใช้งาน SiLU นอกจากนี้ YOLOv11 ยังรวมถึง C3K2 blocks ที่ใช้แทน C2f blocks ที่ใช้ใน YOLOv8 C3K2 blocks ช่วยในการทำงานที่มีประสิทธิภาพมากขึ้นในการประมวลผล Cross-Stage Partial (CSP) ดังที่แสดงในรูปที่ 2(e) สองบล็อกสุดท้ายของ backbone คือ Spatial Pyramid Pooling Fast (SPPF) และ Cross-Stage Partial with Spatial Attention (C2PSA) โดยบล็อก SPPF ใช้หลายชั้น max-pooling ตามที่แสดงในรูปที่ 2(f) เพื่อดึงคุณลักษณะหลายระดับจากภาพอินพุตอย่างมีประสิทธิภาพ ส่วน C2PSA block จะนำกลไกการให้ความสนใจ (attention mechanism) มาใช้ตามที่แสดงในรูปที่ 2(g) เพื่อเสริมความแม่นยำของโมเดล

**Neck**: ส่วนประกอบหลักที่สองของ YOLOv11 คือ neck ดังที่แสดงในรูปที่ 2 Neck ประกอบด้วยหลาย Conv layer, C3K2 blocks, การเชื่อมต่อ (Concat operations), และบล็อก upsamples พร้อมกับข้อได้เปรียบของกลไก C2PSA บทบาทหลักของ neck คือการรวมคุณลักษณะจากหลายระดับ (scales) และส่งผ่านไปยัง head blocks

**Head**: ส่วนสุดท้ายของ YOLOv11 คือ head ซึ่งเป็นโมดูลสำคัญที่รับผิดชอบในการสร้างการทำนาย มันจะกำหนดคลาสของวัตถุ คำนวณคะแนน objectness และทำนายกรอบขอบเขต (bounding boxes) ของวัตถุที่ตรวจพบได้อย่างแม่นยำ

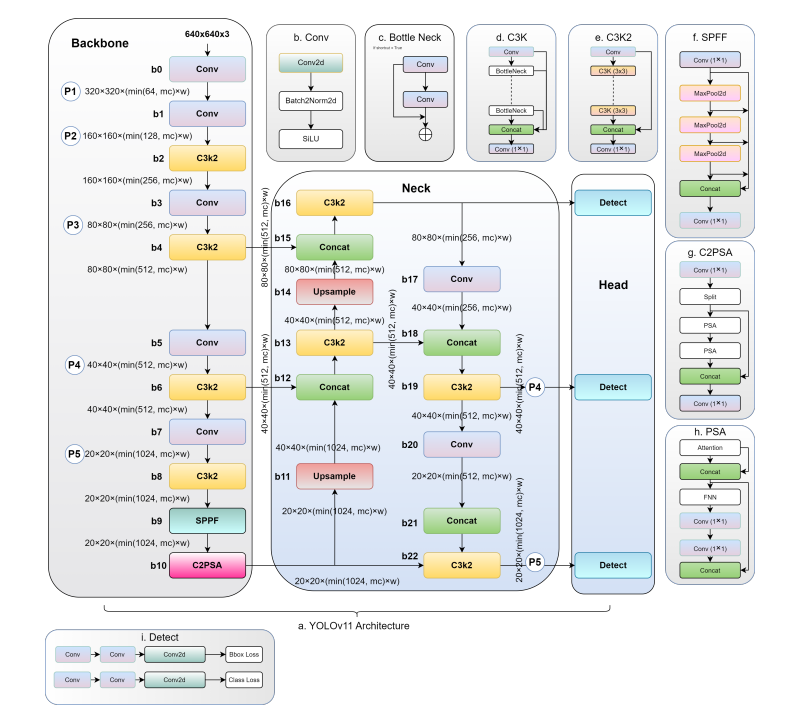


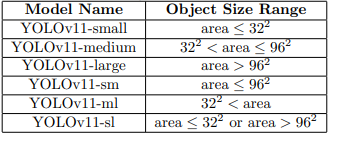
Figure 2: YOLOv11 architecture

**Modified Versions of YOLOv11**

Backbone ของสถาปัตยกรรม YOLOv11 ทำการลดขนาดของภาพอินพุตหลายรอบจนกลายเป็นหลายระดับ เช่น 2x, 4x, 8x, 16x, และ 32x ซึ่งกระบวนการนี้จะสร้างชุดคุณลักษณะห้าชุด (320x320, 160x160, 80x80, 40x40, และ 20x20) ชุดคุณลักษณะเหล่านี้ที่เรียกว่า (P1, P2, P3, P4, P5) ตามที่แสดงในรูปที่ 2 จะถูกรวมกับส่วนประกอบอื่นๆ ของโมเดล เช่น SPPF และ C2PSA แล้วส่งผ่านไปยัง head blocks ชุดคุณลักษณะที่มีขนาดใหญ่จะรับผิดชอบในการตรวจจับวัตถุขนาดใหญ่ ในขณะที่ชุดคุณลักษณะที่มีขนาดกลาง เช่น 40x40 จะใช้ในการตรวจจับวัตถุขนาดกลาง และชุดคุณลักษณะที่มีขนาดเล็ก เช่น 20x20 จะมุ่งเน้นไปที่การตรวจจับวัตถุขนาดเล็ก ตามเอกสารทางการ โมเดล YOLOv11 จะมี head ที่ประกอบไปด้วยสามบล็อกการตรวจจับ ซึ่งแต่ละบล็อกจะรับผิดชอบในการตรวจจับวัตถุในขนาดที่แตกต่างกัน เช่น วัตถุขนาดเล็กมักจะมีขนาดน้อยกว่า 32² พิกเซล วัตถุขนาดกลางมีขนาดมากกว่า 32² แต่ต่ำกว่า 96² พิกเซล และวัตถุขนาดใหญ่มีขนาดมากกว่า 96² พิกเซล ในบางกรณี แอปพลิเคชันการตรวจจับวัตถุอาจถูกออกแบบมาเพื่อมุ่งเน้นไปที่ขนาดวัตถุเฉพาะ ตัวอย่างเช่น แอปพลิเคชันทางอากาศมักจะเกี่ยวข้องกับการตรวจจับวัตถุขนาดเล็กในภาพ เพื่อเพิ่มประสิทธิภาพในการใช้ทรัพยากร แทนที่จะใช้สถาปัตยกรรม YOLOv11 มาตรฐาน เราจึงขอเสนอหกเวอร์ชันที่ปรับแต่งของ YOLOv11 ที่ออกแบบมาเพื่อตรวจจับขนาดของวัตถุที่เฉพาะเจาะจง (YOLOv11-small, YOLOv11-medium, YOLOv11-large, YOLOv11-sm, YOLOv11-ml, และ YOLOv11-sl) โดยแต่ละเวอร์ชันจะมุ่งเน้นไปที่ขนาดของวัตถุที่เฉพาะเจาะจง และจะเลือกเวอร์ชันที่เหมาะสมตามขนาดของวัตถุในชุดข้อมูล (ดูในตารางที่ 1)

เพื่อให้กระบวนการนี้ง่ายขึ้น เราได้ใช้โปรแกรมที่ง่ายในการวิเคราะห์และให้ข้อมูลรายละเอียดเกี่ยวกับขนาดของวัตถุในชุดข้อมูล ซึ่งจะได้อธิบายเพิ่มเติมในส่วนของชุดข้อมูล การใช้เวอร์ชันที่ปรับแต่งเหล่านี้แทนที่สถาปัตยกรรม YOLOv11 แบบดั้งเดิมจะช่วยลดต้นทุนการคำนวณและขนาดของโมเดลในขณะที่ยังคงความแม่นยำในหลายกรณี

Table 1: Object Size Categories for Modified YOLOv11 Models. Each model is optimized to detect specificobject sizes based on the relative area to the image.



พี่มิ้นสามารถทำตารางได้เลยเด้อ

**1 YOLOv11-small**

เวอร์ชันที่ปรับแต่งแรกของ YOLOv11 คือเวอร์ชันขนาดเล็ก ซึ่งถูกออกแบบมาเพื่อตรวจจับวัตถุที่มีขนาดพื้นที่น้อยกว่าหรือเท่ากับ 32² พิกเซล เพื่อปรับแต่ง YOLOv11

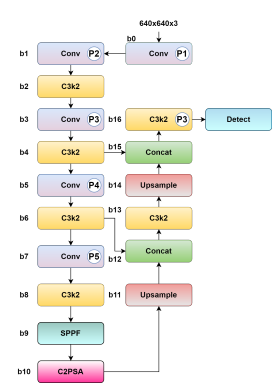


Figure 3: YOLOv11-Small: Architecture Optimized for Detecting Small Objects

ทำการติดป้ายแต่ละบล็อกในสถาปัตยกรรมดั้งเดิมโดยเริ่มต้นด้วย "b" ตั้งแต่ b0 ถึง b22 เพื่อความสะดวก ตามที่กล่าวถึงในส่วนก่อนหน้า บล็อกการตรวจจับตัวแรกถูกใช้สำหรับตรวจจับขนาดของวัตถุขนาดเล็ก สำหรับสิ่งนี้ เราจึงลบบล็อกการตรวจจับที่สองและสามออก เนื่องจากเราลบสองบล็อกนี้ออก เราจึงได้ลบบล็อกที่ให้คุณลักษณะสำหรับการตรวจจับขนาดที่ใหญ่ขึ้นออกไปด้วย ผลลัพธ์คือบล็อกตั้งแต่ b17 ถึง b22 ซึ่งเกี่ยวข้องกับวัตถุขนาดกลางและใหญ่ ถูกลบออก เวอร์ชันใหม่ของ YOLOv11 ซึ่งเรียกว่า YOLOv11-small แสดงในรูปที่ 3

**2 YOLOv11-medium**

เวอร์ชันที่ปรับแต่งที่สองของ YOLOv11 ถูกออกแบบมาเพื่อตรวจจับวัตถุขนาดกลางโดยเฉพาะ ซึ่งนิยามว่ามีขนาดใหญ่กว่า 32² พิกเซล แต่เล็กกว่า 96² พิกเซล กลับมาที่รูปที่ 2 เราได้ลบบล็อกทั้งหมดที่เกี่ยวข้องกับการตรวจจับวัตถุขนาดเล็กและขนาดใหญ่ โดยลบบล็อกที่รับผิดชอบการประมวลผลคุณลักษณะที่เกี่ยวข้องกับการตรวจจับวัตถุขนาดเล็กและขนาดใหญ่โดยเฉพาะ

สำหรับวัตถุขนาดเล็ก บล็อก b14, b15, และ b16 ถูกลบออก เนื่องจากบล็อกเหล่านี้ให้ข้อมูลกับหัวตรวจจับสำหรับวัตถุขนาดเล็ก ในทำนองเดียวกัน บล็อก b20, b21, และ b22 ถูกลบออก เนื่องจากบล็อกเหล่านี้ให้ข้อมูลกับหัวตรวจจับสำหรับวัตถุขนาดใหญ่ หลังจากลบบล็อกเหล่านี้ออกแล้ว เราได้เปลี่ยนชื่อบล็อก YOLOv11 เดิมที่เกี่ยวข้องกับวัตถุขนาดกลาง (ก่อนหน้านี้คือ b17, b18, และ b19) เป็น b14, b15, และ b16 ตามลำดับ ตามที่แสดงในรูปที่ 4

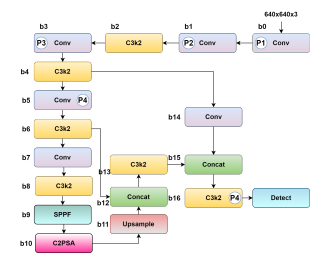


Figure 4: YOLOv11-Medium: Architecture Optimized for Detecting Medium Objects

**3 YOLOv11-large**

เวอร์ชันที่สามของ YOLOv11 ที่ปรับปรุงได้รับการออกแบบมาเพื่อมุ่งเน้นไปที่การตรวจจับวัตถุขนาดใหญ่ โดยวัตถุที่มีพื้นที่ขนาดมากกว่า 96กำลังสอง ในการสร้างโมเดล YOLOv11-large ได้มีการปรับแต่งสถาปัตยกรรมดั้งเดิมโดยการลบส่วนประกอบที่ไม่เกี่ยวข้องกับการตรวจจับวัตถุขนาดใหญ่ออก และทำการเชื่อมต่อบล็อกที่ไม่ได้เชื่อมต่อก่อนหน้านี้อีกครั้ง โดยเฉพาะบล็อก b11 ถึง b19 ได้ถูกลบออก เนื่องจากเกี่ยวข้องกับการให้ฟีเจอร์สำหรับการตรวจจับวัตถุขนาดเล็กและขนาดกลาง เพื่อให้เครือข่ายยังคงต่อเนื่อง บล็อก b19 ได้รับการเชื่อมต่อเพื่อรับฟีเจอร์จากบล็อก b6 เนื่องจากทั้งสองใช้แผนที่ฟีเจอร์เดียวกันเป็นอินพุต นอกจากนี้ บล็อก b19, b21 และ b22 ได้เปลี่ยนชื่อเป็น b11, b12 และ b13 ตามลำดับ สถาปัตยกรรมของ YOLOv11 สำหรับวัตถุขนาดใหญ่ที่ปรับปรุงแล้วแสดงในรูปที่ 5.

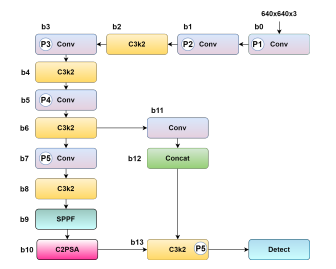


Figure 5: YOLOv11-Large: Architecture Optimized for Detecting Large Objects

**4 YOLOv11-sm**

YOLOv11-sm ซึ่งเป็นเวอร์ชันที่ปรับปรุงลำดับที่สี่ของ YOLOv11 ได้รับการออกแบบมาเพื่อเพิ่มความยืดหยุ่นในการตรวจจับวัตถุขนาดเล็กและขนาดกลาง โดยวัตถุที่ตรวจจับจะมีพื้นที่ขนาดน้อยกว่า 96² ดังที่แสดงในตารางที่ 1 สำหรับการปรับปรุงในเวอร์ชันนี้ ส่วนที่เกี่ยวข้องกับการตรวจจับวัตถุขนาดใหญ่ถูกตัดออกไปขณะที่บล็อกสำหรับการตรวจจับวัตถุขนาดเล็กและขนาดกลางยังคงอยู่ ดังที่แสดงในรูปที่ 2 การปรับปรุงนี้เกี่ยวข้องกับการลบส่วนหัวการตรวจจับที่สามและบล็อกทั้งหมดที่ป้อนข้อมูลเข้าไปในส่วนนี้ บล็อก b20, b21 และ b22 ถูกลบออก ในขณะที่บล็อกตั้งแต่ b0 ถึง b19 ยังคงไม่เปลี่ยนแปลง สถาปัตยกรรมที่ปรับปรุงสำหรับการตรวจจับวัตถุขนาดเล็กและขนาดกลางใน YOLOv11-sm แสดงในรูปที่ 6.

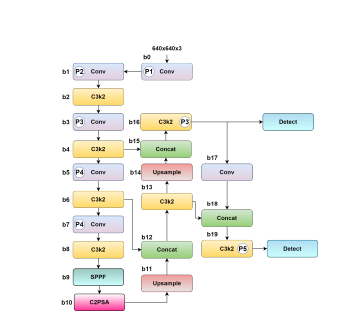


Figure 6: YOLOv11-sm: Architecture Optimized for Detecting small and medium Objects

**5 YOLOv11-ml**

YOLOv11-ml ซึ่งเป็นเวอร์ชันที่ปรับปรุงลำดับที่ห้าของ YOLOv11 ได้รับการออกแบบมาเพื่อมุ่งเน้นการตรวจจับวัตถุขนาดกลางและขนาดใหญ่ โดยวัตถุที่มีขนาดใหญ่กว่า 32² จะสามารถประมวลผลได้อย่างมีประสิทธิภาพในโมเดลนี้ การปรับปรุง YOLOv11-ml เกิดจากการปรับแต่งส่วนที่เกี่ยวข้องกับการตรวจจับวัตถุขนาดเล็กโดยเฉพาะ ซึ่งในรูปที่ 2 ได้แสดงให้เห็นว่าบล็อก b14, b15 และ b16 ถูกตัดออกไป หลังจากนั้นบล็อกที่เหลืออยู่ b17 ถึง b22 ได้รับการเปลี่ยนหมายเลขใหม่เป็น b14 ถึง b19 สุดท้ายบล็อกหมายเลข b4 ถูกเชื่อมโยงกับบล็อก b14 สถาปัตยกรรมที่ปรับปรุงใหม่สำหรับการตรวจจับวัตถุขนาดกลางและขนาดใหญ่แสดงไว้ในรูปที่ 7

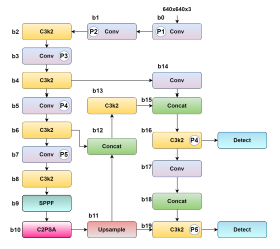


Figure 7: YOLOv11-sm: Architecture Optimized for Detecting medium and large Objects

**6 YOLOv11-sl**

YOLOv11-sl ซึ่งเป็นเวอร์ชันที่หกที่ได้รับการปรับปรุงจาก YOLOv11 ได้รับการออกแบบมาเพื่อมุ่งเน้นการตรวจจับวัตถุขนาดเล็กและขนาดใหญ่ ตามที่กำหนดในตารางที่ 1 โดยวัตถุต้องมีพื้นที่ ≤ 32² หรือ > 96² สำหรับการพัฒนาโมเดลนี้จาก YOLOv11 ส่วนที่เกี่ยวข้องกับการตรวจจับวัตถุขนาดกลางถูกตัดออกไป โดยเฉพาะบล็อก b17, b18 และ b19 ถูกลบออก และบล็อก b20 ถูกเชื่อมโยงใหม่เพื่อแทนที่บล็อก b17 หลังจากนั้นหมายเลขของบล็อกถูกปรับเปลี่ยนใหม่ โดยบล็อก b0 ถึง b16 ยังคงเดิม ขณะที่บล็อก b20, b21 และ b22 ได้รับหมายเลขใหม่เป็น b17, b18 และ b19 ตามลำดับ สถาปัตยกรรมของเวอร์ชันที่ปรับปรุงนี้แสดงอยู่ในรูปที่ 8

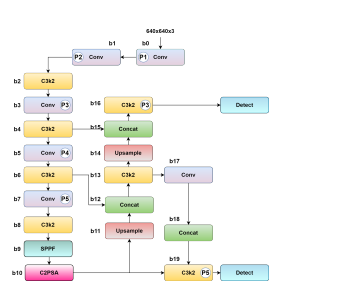


Figure 8: YOLOv11-sl: Architecture Optimized for Detecting small and large Objects

----------------------------------------------------------------------------------------------------------------------------------

<https://arxiv.org/pdf/2412.14790> มันเป็น paper ง่ะ ไม่ไหวแล้ว555555

**เครื่องมือที่ใช้**

**Fontend**

- Java Script

- HTML/CSS

- React.js

**Backend**

- Python

- node.js

**Database**

- PostgreSQL

**Web Server**

- NGINX

**ML**

- PyTorch

**ระบบจัดการ HOST**

- Docker

**JavaScript**

JavaScript เป็นภาษาโปรแกรมที่นักพัฒนาใช้ในการสร้างหน้าเว็บแบบอินเทอร์แอคทีฟ ตั้งแต่การรีเฟรชฟีดสื่อโซเชียลไปจนถึงการแสดงภาพเคลื่อนไหวและแผนที่แบบอินเทอร์แอคทีฟ ฟังก์ชันของ JavaScript สามารถปรับปรุงประสบการณ์ที่ผู้ใช้จะได้รับจากการใช้งานเว็บไซต์ และในฐานะที่เป็นภาษาในการเขียนสคริปต์ฝั่งไคลเอ็นต์ จึงเป็นหนึ่งในเทคโนโลยีหลักของ World Wide Web ยกตัวอย่างเช่น เมื่อคุณท่องเว็บแล้วเห็นภาพสไลด์ เมนูดร็อปดาวน์แบบคลิกให้แสดงผล หรือสีองค์ประกอบที่เปลี่ยนแบบไดนามิกบนหน้าเว็บ นั่นคือเอฟเฟกต์ของ JavaScript แต่ก่อนเว็บเพจเป็นแบบคงที่คล้ายกับหน้าหนังสือ โดยหลักแล้วหน้าเพจคงที่แสดงข้อมูลในเค้าโครงที่ตายตัว และไม่ได้ทำทุกอย่างที่ตอนนี้เราคาดหวังจากเว็บไซต์สมัยใหม่ JavaScript เกิดขึ้นในฐานะเทคโนโลยีฝั่งเบราว์เซอร์เพื่อทำให้เว็บแอปพลิเคชันมีความเป็นไดนามิกมากขึ้น เมื่อใช้ JavaScript เบราว์เซอร์จะสามารถตอบสนองต่อการโต้ตอบของผู้ใช้และเปลี่ยนแปลงเค้าโครงเนื้อหาบนเว็บเพจได้ เมื่อภาษาผ่านการพัฒนาอย่างเต็มที่ นักพัฒนา JavaScript ก็สร้างไลบรารี เฟรมเวิร์ก และแนวทางปฏิบัติในการเขียนโปรแกรม แล้วเริ่มนำ JavaScript ไปใช้นอกเว็บเบราว์เซอร์ สามารถใช้ JavaScript สำหรับทั่งการพัฒนาฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์

ภาษาโปรแกรมทั้งหมดทำงานด้วยการแปลไวยากรณ์ที่คล้ายภาษาอังกฤษเป็นโค้ดสำหรับเครื่อง จากนั้นระบบปฏิบัติการจะเรียกใช้โค้ดนั้น JavaScript ได้รับการจัดประเภทอย่างกว้าง ๆ ว่าเป็นภาษาเขียนสคริปต์ หรือภาษาที่แปลผลแล้ว โค้ด JavaScript ได้รับการแปลผล นั่นคือ แปลโดยตรงเป็นโค้ดภาษาสำหรับเครื่อง ด้วยกลไกล JavaScript ในขณะที่ในภาษาโปรแกรมอื่น ๆ คอมไพเลอร์จะคอมไพล์โค้ดทั้งหมดเป็นโค้ดสำหรับเครื่องในขั้นตอนที่แยกต่างหาก ดังนั้น ภาษาเขียนสคริปต์ทั้งหมดจึงเป็นภาษาโปรแกรม แต่ไม่ใช่ว่าภาษาโปรแกรมทั้งหมดจะเป็นภาษาเขียนสคริปต์เสมอไป

**กลไก JavaScript**

กลไก JavaScript คือโปรแกรมคอมพิวเตอร์ที่เรียกใช้โค้ด JavaScript กลไก JavaScript เคยเป็นเพียงตัวแปลผล แต่กลไกสมัยใหม่ทั้งหมดใช้การคอมไพล์แบบ Just-in-time หรือรันไทม์เพื่อปรับปรุงประสิทธิภาพ

**JavaScript ฝั่งไคลเอ็นต์**

JavaScript ฝั่งไคลเอ็นต์หมายถึงวิธีที่ JavaScript ทำงานในเบราว์เซอร์ของคุณ ในกรณี กลไก JavaScript จะอยู่ภายในโค้ดเบราว์เซอร์ เว็บเบราว์เซอร์เจ้าใหญ่ ๆ ทั้งหมดจะมาพร้อมกับกลไก JavaScript ในตัว นักพัฒนาแอปพลิเคชันเว็บจะเขียนโค้ด JavaScript ที่มีฟังก์ชันที่แตกต่างกันสัมพันธ์กับเหตุการณ์ต่าง ๆ เช่น การคลิกเมาส์ หรือการเลื่อนเมาส์ผ่าน ฟังก์ชันเหล่านี้จะเปลี่ยนแปลง HTML และ CSS

1 เบราว์เซอร์โหลดเว็บเพจเมื่อคุณเยี่ยมชมเว็บเพจ

2 ระหว่างการโหลด เบราว์เซอร์แปลงหน้าและองค์ประกอบทั้งหมดของหน้า เช่น ปุ่ม ป้าย และกล่องดรอปดาวน์ เป็นโครงสร้างข้อมูลที่เรียกว่าโมเดลอ็อบเจกต์เอกสาร (DOM)

3 กลไก JavaScript ของเบราว์เซอร์แปลงโค้ด JavaScript เป็นไบต์โค้ด โค้ดนี้เป็นตัวกลางระหว่างไวยากรณ์ JavaScript และเครื่อง

4 เหตุการณ์ต่าง ๆ เช่น การคลิกเมาส์บนปุ่ม จะกระตุ้นให้บล็อกโค้ด JavaScript ที่เกี่ยวข้องดำเนินการ จากนั้นกลไกจะแปลผลไบต์โค้ด และทำการเปลี่ยนแปลง DOM

5 เบราว์เซอร์แสดงผล DOM ใหม่

**JavaScript ฝั่งเซิร์ฟเวอร์**

JavaScript ฝั่งเซิร์ฟเวอร์หมายถึงการใช้ภาษาเขียนโค้ดในลอจิกของเซิร์ฟเวอร์แบ็คเอนด์ ในกรณีนี้ กลไก JavaScript จะอยู่บนเซิร์ฟเวอร์โดยตรง ฟังก์ชัน JavaScript ฝั่งเซิร์ฟเวอร์สามารถเข้าถึงฐานข้อมูล ดำเนินการทางตรรกะแบบต่าง ๆ และตอบสนองต่อเหตุการณ์ต่าง ๆ ที่ถูกกระตุ้นจากระบบปฏิบัติการของเซิร์ฟเวอร์ ข้อได้เปรียบหลักของการเขียนสคริปต์ฝั่งเซิร์ฟเวอร์คือคุณสามารถปรับแต่งการตอบสนองของเว็บไซต์โดยอ้างอิงตามข้อกำหนดของคุณ สิทธิ์เข้าถึงของคุณ และคำขอข้อมูลจากระบบได้เป็นอย่างมาก

**ฝั่งไคลเอ็นต์เทียบกับฝั่งเซิร์ฟเวอร์**

คำว่าไดนามิกอธิบาย JavaScript ทั้งฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์ พฤติกรรมไดนามิกคือความสามารถที่จะอัปเดตการแสดงของเว็บเพจเพื่อสร้างเนื้อหาใหม่ตามที่จำเป็น ความแตกต่างระหว่าง JavaScript ฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์อยู่ที่วิธีการที่ JavaScript สร้างเนื้อหาใหม่ โค้ดฝั่งเซิร์ฟเวอร์สร้างเนื้อหาใหม่แบบไดนามิกด้วยการใช้ลอจิกของแอปพลิเคชันและปรับเปลี่ยนข้อมูลจากฐานข้อมูล ส่วน JavaScript ฝั่งไคลเอ็นต์สร้างเนื้อหาใหม่แบบไดนามิกภายในเบราว์เซอร์โดยใช้ลอจิกอินเทอร์เฟซผู้ใช้ และปรับเปลี่ยนเนื้อหาเว็บเพจที่อยู่บนไคลเอ็นต์อยู่แล้ว ความหมายมีความแตกต่างกันเล็กน้อยในสองบริบทแต่ก็มีความเกี่ยวข้องกัน และทั้งสองแนวทางทำงานร่วมกันเพื่อปรับปรุงประสิทธิภาพการใช้งานของผู้ใช้

นอกเหนือจากการนำไปใช้ในคุณสมบัติไดนามิกแล้ว ความแตกต่างอีกประการหนึ่งระหว่างการใช้ JavaScript สองแบบคือทรัพยากรที่โค้ดสามารถเข้าถึงได้ ทางฝั่งไคลเอ็นต์ เบราว์เซอร์จะควบคุมสภาพแวดล้อมรันไทม์ของ JavaScript โค้ดจึงสามารถเข้าถึงได้เพียงทรัพยากรที่เบราว์เซอร์อนุญาตให้โค้ดเข้าถึงเท่านั้น ตัวอย่างเช่น โค้ดไม่สามารถเขียนเนื้อหาลงในฮาร์ดดิสก์ของคุณได้เว้นแต่คุณจะคลิกปุ่มดาวน์โหลด ในทางกลับกัน ฟังก์ชันฝั่งเซิร์ฟเวอร์สามารถเข้าถึงทรัพยากรทั้งหมดของเครื่องเซิร์ฟเวอร์ได้ตามที่จำเป็น

https://aws.amazon.com/th/what-is/javascript/

-----------------------------------------------------------------------------------------------------------------------------------

**HTML**

HTML ย่อมาจากภาษาอังกฤษในคำว่า Hypertext Markup Language ซึ่งถ้าแปลความหมายให้เข้าใจได้ง่ายมากขึ้นนั่นก็คือ ภาษาคอมพิวเตอร์ที่ใช้เขียนหน้าเว็บเพจ ถือเป็นตัวกำหนดโครงสร้างและองค์ประกอบของเว็บไซต์ที่คุณเห็นผ่าน Web Browser ซึ่งมีการสร้างรูปแบบและจัดการข้อมูลให้อยู่ในรูปแบบภาษา Markup โดยรูปแบบจะประกอบไปด้วย ‘แท็ก’ (Tags) หรือคำสั่งที่ใช้สำหรับกำหนดโครงสร้างและเนื้อหาภายในหน้าเว็บเพจ ยกตัวอย่าง เช่น ใช้คำสั่ง HTML ในรูปแบบ Tags เพื่อกำหนดลักษณะของข้อความ โดยใช้ Tags <h1> (Heading1) จะทำให้ข้อความถูกครอบคลุมด้วย Tags จากนั้นข้อความจะมีขนาดใหญ่ขึ้นและกลายเป็นหัวข้อขนาดใหญ่ในทันที ส่วน Tags <p> จะแบ่งข้อความออกเป็นย่อหน้า หรือที่เรียกว่า Paragraph เป็นต้น อย่างไรก็ตาม HTML มีประโยชน์มากมาย นอกจากการสร้างเว็บไซต์ จัดการข้อมูลหรือสร้างเอกสารดิจิทัลต่างๆ แล้ว ยังถูกจัดให้เป็นภาษาคอมพิวเตอร์ที่ใช้เขียนโปรแกรมได้อีกด้วย ซึ่งถูกกำหนดมาตรฐานโดยองค์กร World Wide Web Consortium (W3C) และ Microsoft จึงทำให้เรียกกันติดปากว่า “ภาษา HTML”

HTML หรือ HyperText Markup Language เป็นภาษามาร์กอัปที่ใช้สำหรับการสร้างและจัดการเว็บไซต์ ซึ่งจะใช้ Tags เพื่อกำหนดโครงสร้างและองค์ประกอบของเว็บไซต์ ส่งผลให้หน้าเว็บมีตัวหนังสือ ข้อความ สีสัน และลูกเล่นต่าง ๆ รวมไปถึงมีส่วนในการจัดทำรูปภาพเคลื่อนไหวได้เช่นกัน ซึ่งการเขียนเว็บ HTML ถือเป็นตัวบ่งบอกถึงคำสั่งในการจัดรูปแบบหน้าเพจหรือข้อความต่างๆ หน้าเว็บไซต์ได้เป็นอย่างดี Tags มีความสำคัญกับ HTML เพราะว่า Tags คือ การเชื่อมโยงทุกอย่างเข้าหากันอย่างง่ายดาย และผลลัพธ์ต่างๆ ที่เชื่อมโยงกันนั้น จะส่งไปยังผู้ที่เข้ามาเยี่ยมชมหน้าเว็บไซต์ ซึ่งการเชื่อมข้อมูลให้กับทุกคนที่เข้ามาที่หน้าเว็บไซต์ ช่วยเพิ่มโอกาสการดึงดูดลูกค้าให้กลายเป็นลูกค้าเราจริง ๆ หรือดึงดูดคนทั่วไปให้อยู่กับเว็บไซต์ของเราได้มากยิ่งขึ้น

https://searchstudio.co.th/th/seo/what-is-html/#elementor-toc\_\_heading-anchor-0

-------------------------------------------------------------------------------------------------------------------------------

**CSS**

Cascading Style Sheets หรือ CSS คือภาษาเว็บที่ใช้สำหรับออกแบบหน้าเว็บไซต์ โดยใช้กฎการเขียน CSS เพื่อกำหนดสไตล์และรูปแบบต่างๆ ให้กับเนื้อหาบนหน้าเว็บไซต์ เช่น สีพื้นหลัง, ขนาดตัวอักษร, รูปแบบตาราง รวมถึงการจัดตำแหน่งและการจัดรูปแบบขององค์ประกอบต่างๆ บนหน้าเว็บไซต์ และเป็นภาษาที่นิยมใช้กันอย่างแพร่หลายในการพัฒนาเว็บไซต์ และยังเป็นส่วนสำคัญของเทคโนโลยีเว็บไซต์ (Web Technology) ที่ช่วยให้เว็บไซต์มีความสวยงามและสมบูรณ์มากขึ้น

CSS เป็นส่วนสำคัญของการออกแบบเว็บไซต์ เพราะมีความสามารถในการกำหนดรูปแบบและสไตล์ต่างๆ ของเนื้อหาบนหน้าเว็บไซต์ได้อย่างอิสระ โดยไม่ต้องเปลี่ยนแปลงโครงสร้าง HTML ที่ใช้สร้างเนื้อหานั้นๆ ซึ่งจะช่วยให้สามารถออกแบบเว็บไซต์ให้สวยงามและมีความสมบูรณ์มากขึ้นได้ นอกจากนี้ CSS ยังช่วยให้เว็บไซต์มีความสม่ำเสมอในการแสดงผลบนหลายๆ Browser และช่วยลดเวลาในการออกแบบเว็บไซต์ด้วยการใช้ไฟล์ CSS สำเร็จรูปที่สามารถเรียกใช้งานได้หลายหน้าของเว็บไซต์ได้

<https://firstrank.co.th/blog-css/>

----------------------------------------------------------------------------------------------------------------

**Python**

Python คือ เป็นภาษาคอมพิวเตอร์ระดับสูง ที่ถูกปรับมาให้ใช้งานง่าย ทำงานด้วยการแปลชุดคำสั่งทีละบรรทัด หรือแบบ Interpreter เพื่อป้อนภาษาเหล่านั้นเข้าสู่หน่วยประมวลผลให้คอมพิวเตอร์เข้าใจถึงความต้องการ และทำงานได้ตรงตามจุดประสงค์ ทั้งยังลดความซับซ้อนของภาษาที่ไม่จำเป็นออกไป เพื่อให้ง่ายต่อการเรียนรู้ และใกล้เคียงกับภาษาที่เราใช้ในการสื่อสารมากที่สุด ภาษา Python เองจึงกลายเป็นภาษาโปรแกรมขั้นพื้นฐาน ที่ถูกนำไปต่อยอดและใช้งานได้หลากหลาย ไม่ได้จำกัดเฉพาะทางใดทางหนึ่ง (General-purpose language) นิยมใช้ในองค์กรทั่วไป อย่างที่รู้จักกันดี คือ YouTube, Instagram, Google ฯลฯ นับเป็นภาษาที่นักโปรแกรมเมอร์นิยมมากที่สุด

Python เป็นภาษาโปรแกรมพื้นฐานที่นำไปต่อยอดได้หลายรูปแบบ เรียกว่าอยู่ที่จะใช้ทำอะไรมากกว่า เนื่องจากมีความยืดหยุ่นคล่องตัวสูง ประยุกต์ได้หลากหลายทิศทาง ตอบโจทย์การทำเว็บไซต์ เขียนโค้ด แชทบอท ทำ Data Science ไปจนถึง Machine Learning Model ทั้งยังมีฟังก์ชันในการใช้งานเยอะ เหมาะกับโปรแกรมเมอร์มือใหม่ ทั้งยังมี Tools และ Library Support ฟรีเยอะ หาข้อมูลได้ง่าย แต่ที่นิยมนำไปใช้งานอย่างแพร่หลาย มีดังต่อไปนี้

**ภาษาไพทอนเพื่อการทำเว็บไซต์**

ถึงตรงนี้บางคนอาจจะเข้าใจว่าเว็บไซต์พัฒนาขึ้นจาก HTML และ JavaScript เพียงเท่านั้น แต่ปัจจุบันไพทอนคือภาษาหลักในการพัฒนาโปรแกรมของเว็บไซต์ดังหลากหลายแห่ง ไม่ว่าจะเป็น Spotify, Netflix, Facebook หรืออย่าง Google เอง ก็มีใช้ Python ในระบบหลังบ้านเช่นเดียวกัน

**ภาษาไพทอนใน Chatbot**

อีกหนึ่งเทรนด์เทคโนโลยีมาแรงแห่งยุค อย่างแชทบอท ตัวช่วยอำนวยความสะดวก แบบ “ถามได้ตอบได้” ที่ถูกพัฒนาจากปัญญาประดิษฐ์ เพื่อใช้เป็นเครื่องมือในการอำนวยความสะดวก โดยมีพื้นฐานการพัฒนาระบบมาจากภาษา Python นั่นเอง

<https://aws.amazon.com/th/what-is/python/>

-------------------------------------------------------------------------------------------------------------------------------

**NodeJS**

NodeJS คือ Cross Platform Runtime Environment สำหรับฝั่ง Server เป็น Open Source และ Library ที่ใช้สำหรับพัฒนาเว็บแอปพลิเคชันต่าง ๆ ด้วยภาษา JavaScript เหมาะสำหรับการสร้างแอปพลิเคชันที่ต้องการใช้ข้อมูลจำนวนมาก และนิยมใช้ในการพัฒนาแอปพลิเคชันที่ใช้ข้อมูลแบบ Realtime สามารถทำงานได้ทุกระบบปฏิบัติการ โดยถูกนำมาเป็น Web Server, IoT, Webkit, TVOS, OS และอื่น ๆ เป็นต้น NodeJS ใช้ V8 Engine ที่ถูกพัฒนาโดย The Chromium Project สำหรับเพิ่มประสิทธิภาพการทำงานของภาษา JavaScript ร่วมกับ Web Browser ให้ดีมากขึ้น โดยการใช้หลักการ Compile ก่อนประมวลผล (Just-in-time Compilation) ด้วยการเป็นตัวแปลงโค้ดภาษา JavaScript หรือ JavaScript Engine ให้เป็น Machine Code ทำให้สามารถทำงานนอก Browser อื่นได้ เนื่องจากตามปกติแล้ว JavaScript สามารถรันได้บน Client เท่านั้น

NodeJS ทำงานแบบ Single Process โดยมี Event-loop เข้ามาช่วยในการทำงานแบบ Asynchronous คือ รูปแบบการทำงานของชุดคำสั่งที่เขียนขึ้นมา โดยทำงานแบบไม่เรียงขั้นตอน เนื่องจากชุดคำสั่งทำงานพร้อมกัน และเมื่อคำสั่งไหนเสร็จเรียบร้อยจะแสดงผลลัพธ์ก่อนแบบ Non-Blocking I/O สามารถส่ง Request ของ User 1 และ User 2 พร้อมกันได้เลย ทำให้ลดการใช้ Thread ได้ โดย NodeJS ไม่เหมาะสำหรับการทำงานที่เป็น CPU Intensive เพราะทำให้ถูก Block การทำงานทั้งหมด

1. มีเครื่องมือที่สะดวกและรวดเร็วในการจัดการ Package อย่าง NPM (Node Package Manager) หรือ YARN (Dependency Management Tool) ช่วยลดเวลาในการเขียนโค้ดใหม่ทั้งหมด ทำให้สามารถทำงานได้อย่างมีประสิทธิภาพมากขึ้น
2. พัฒนาได้อย่างครอบคลุมทั้ง Frontend และ Backend โดยตัวอย่าง Framework และ Library ฝั่ง Frontend เช่น ReactJS, VueJS เป็นต้น และตัวอย่างฝั่ง Backend เช่น Express, NestJS, Meteor เป็นต้น โดยนักพัฒนาเรียนรู้แค่ภาษา Javascript สามารถเริ่มต้นพัฒนาแบบ Fullstack ได้แล้ว
3. NodeJS ใช้ภาษา JavaScript ซึ่งเป็นภาษายอดนิยมและเป็นที่ต้องการสูงของสายงงาน Programming

<https://blog.openlandscape.cloud/nodejs> ---------------------------------------------------------------------

**PostgreSQL**

PostgreSQL คือฐานข้อมูล เชิงสัมพันธ์โอเพ่นซอร์สระดับองค์กรขั้นสูงที่รองรับการค้นหาทั้งแบบ SQL (เชิงสัมพันธ์) และ JSON (แบบไม่สัมพันธ์) เป็นระบบจัดการฐานข้อมูลที่มีเสถียรภาพสูงซึ่งได้รับการสนับสนุนจากการพัฒนาชุมชนมากกว่า 20 ปี แนวทางที่ละเอียดถี่ถ้วนและร่วมมือกันนี้ช่วยให้มีระดับความยืดหยุ่น ความสมบูรณ์ และความถูกต้องสูง PostgreSQL ถูกใช้เป็นที่เก็บข้อมูลหลักหรือคลังข้อมูลสำหรับแอปพลิเคชันเว็บ มือถือ ภูมิสารสนเทศ และการวิเคราะห์มากมาย

PostgreSQL มีชุดคุณลักษณะที่แข็งแกร่ง รวมถึงพื้นที่ตาราง การจำลองแบบอะซิงโครนัส ธุรกรรมแบบซ้อน การสำรองข้อมูลออนไลน์/แบบร้อน และตัววางแผน/ตัวเพิ่มประสิทธิภาพแบบสอบถามที่ปรับแต่งแล้ว ชุมชน PostgreSQL ยังได้พัฒนาส่วนขยายที่ขยายการทำงานของฐานข้อมูล PostgreSQL PostgreSQL รองรับ:

- ชุดอักขระสากล การเข้ารหัสอักขระหลายไบต์ และ Unicode

- ประเภทข้อมูล SQL:2008 ส่วนใหญ่ ได้แก่ INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTER-VAL และ TIMESTAMP

- การจัดเก็บวัตถุไบนารีขนาดใหญ่ รวมถึงรูปภาพ เสียง วิดีโอ และแผนที่

- คีย์ต่างประเทศ การรวม มุมมอง ทริกเกอร์ และกระบวนการที่จัดเก็บไว้

- ภาษาการเขียนโปรแกรมและโปรโตคอลชั้นนำ ได้แก่ Python, Java, Perl, .Net, Go, Ruby, C/C++, Tcl และ ODBC

เซิร์ฟเวอร์ฐานข้อมูล PostgreSQL คำนึงถึงตำแหน่งสำหรับการเรียงลำดับ ความไวต่อตัวพิมพ์เล็ก-ใหญ่ และการจัดรูปแบบ เซิร์ฟเวอร์ฐานข้อมูล PostgreSQL สามารถปรับขนาดได้สูงทั้งในปริมาณข้อมูลที่สามารถจัดการได้และจำนวนผู้ใช้พร้อมกันที่สามารถรองรับได้

<https://aws.amazon.com/rds/postgresql/what-is-postgresql/>

---------------------------------------------------------------------------------------------------------------------------

**NGINX**

NGINX เป็นซอฟต์แวร์โอเพ่นซอร์สสำหรับการให้บริการเว็บ, การทำรีเวิร์สพร็อกซี, การแคช, การปรับสมดุลโหลด, การสตรีมมีเดีย และอื่น ๆ อีกมากมาย เริ่มต้นในฐานะเว็บเซิร์ฟเวอร์ที่ออกแบบมาเพื่อประสิทธิภาพและความเสถียรสูงสุด นอกเหนือจากความสามารถในการเป็นเซิร์ฟเวอร์ HTTP แล้ว NGINX ยังสามารถทำหน้าที่เป็นพร็อกซีเซิร์ฟเวอร์สำหรับอีเมล (IMAP, POP3 และ SMTP) รวมถึงเป็นรีเวิร์สพร็อกซีและตัวปรับสมดุลโหลดสำหรับเซิร์ฟเวอร์ HTTP, TCP และ UDP

**ที่มาของ NGINX**

อิกอร์ ซิสซอยฟ (Igor Sysoev) ได้เขียน NGINX ขึ้นมาเพื่อแก้ปัญหา C10K ซึ่งเป็นคำที่เกิดขึ้นในปี 1999 เพื่ออธิบายความยากลำบากที่เว็บเซิร์ฟเวอร์ในขณะนั้นเผชิญในการจัดการกับการเชื่อมต่อพร้อมกันจำนวนมาก (10,000 การเชื่อมต่อ) ด้วยสถาปัตยกรรมที่ขับเคลื่อนด้วยเหตุการณ์ (event-driven) และการประมวลผลแบบอะซิงโครนัส NGINX ได้ปฏิวัติวิธีการทำงานของเซิร์ฟเวอร์ในบริบทที่ต้องการประสิทธิภาพสูง และกลายเป็นเว็บเซิร์ฟเวอร์ที่เร็วที่สุดในยุคนั้น

หลังจากที่เปิดโครงการเป็นโอเพ่นซอร์สในปี 2004 และเห็นการใช้งานที่เติบโตอย่างมหาศาล ซิสซอยฟได้ร่วมก่อตั้งบริษัท NGINX, Inc. เพื่อสนับสนุนการพัฒนา NGINX อย่างต่อเนื่อง และทำการตลาด NGINX Plus ในฐานะผลิตภัณฑ์เชิงพาณิชย์ที่มีฟีเจอร์เพิ่มเติมสำหรับลูกค้าองค์กร ในปี 2019 NGINX, Inc. ได้กลายเป็นส่วนหนึ่งของบริษัท F5, Inc. ปัจจุบัน NGINX และ NGINX Plus สามารถรองรับการเชื่อมต่อพร้อมกันได้นับแสนครั้ง และยังเป็นพลังขับเคลื่อนเว็บไซต์ที่มีปริมาณการใช้งานสูงที่สุดบนอินเทอร์เน็ตมากกว่าเซิร์ฟเวอร์ใด ๆ

**NGINX เป็นเว็บเซิร์ฟเวอร์**

เป้าหมายของ NGINX คือการสร้างเว็บเซิร์ฟเวอร์ที่เร็วที่สุด และการรักษามาตรฐานความยอดเยี่ยมนี้ยังคงเป็นเป้าหมายหลักของโครงการเสมอมา NGINX มีประสิทธิภาพเหนือกว่า Apache และเซิร์ฟเวอร์อื่น ๆ อย่างต่อเนื่องในการวัดผลด้านประสิทธิภาพของเว็บเซิร์ฟเวอร์ ตั้งแต่การเปิดตัวครั้งแรกของ NGINX เว็บไซต์ได้พัฒนาจากหน้า HTML แบบธรรมดาไปสู่เนื้อหาแบบไดนามิกที่มีความซับซ้อนมากขึ้น NGINX ก็ได้เติบโตควบคู่ไปกับการเปลี่ยนแปลงนี้ ปัจจุบัน NGINX รองรับทุกองค์ประกอบของเว็บสมัยใหม่ รวมถึง WebSocket, HTTP/2, gRPC และการสตรีมวิดีโอในหลายรูปแบบ (HDS, HLS, RTMP และอื่น ๆ)

แม้ว่า NGINX จะมีชื่อเสียงในฐานะเว็บเซิร์ฟเวอร์ที่เร็วที่สุด แต่สถาปัตยกรรมที่สามารถขยายได้ของมันได้พิสูจน์แล้วว่าเหมาะสมสำหรับงานบนเว็บที่หลากหลายเกินกว่าการให้บริการเนื้อหา เนื่องจากสามารถรองรับการเชื่อมต่อปริมาณมากได้อย่างมีประสิทธิภาพ NGINX จึงมักถูกใช้งานเป็นรีเวิร์สพร็อกซีและตัวปรับสมดุลโหลด (load balancer) เพื่อจัดการทราฟฟิกที่เข้ามาและกระจายไปยังเซิร์ฟเวอร์ต้นทางที่ช้ากว่า เช่น เซิร์ฟเวอร์ฐานข้อมูลเดิม หรือไมโครเซอร์วิส

นอกจากนี้ NGINX ยังถูกวางไว้ระหว่างไคลเอนต์และเว็บเซิร์ฟเวอร์ตัวที่สอง เพื่อทำหน้าที่เป็นตัวจัดการ SSL/TLS (SSL/TLS terminator) หรือเว็บแอคเซลเลอเรเตอร์ โดยทำหน้าที่เป็นตัวกลาง NGINX สามารถจัดการงานที่อาจทำให้เว็บเซิร์ฟเวอร์ของคุณช้าลงได้อย่างมีประสิทธิภาพ เช่น การต่อรอง SSL/TLS การบีบอัด หรือการแคชเนื้อหาเพื่อปรับปรุงประสิทธิภาพ สำหรับเว็บไซต์ไดนามิกที่พัฒนาด้วยเทคโนโลยีต่าง ๆ ตั้งแต่ Node.js ไปจนถึง PHP มักจะใช้ NGINX เป็นตัวแคชเนื้อหาและรีเวิร์สพร็อกซี เพื่อลดภาระของเซิร์ฟเวอร์แอปพลิเคชันและเพิ่มประสิทธิภาพการใช้งานฮาร์ดแวร์ที่มีอยู่ให้ได้มากที่สุด

<https://www.f5.com/glossary/nginxc>

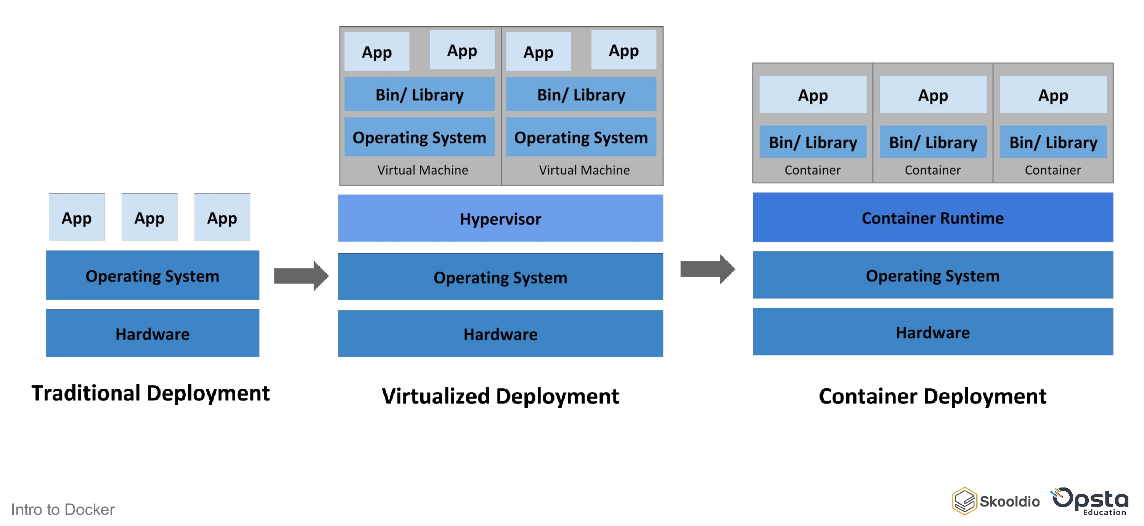
--------------------------------------------------------------------------------------------------------------------------

**Docker**

Docker คืออะไร? เครื่องมือยอดนิยมที่จะช่วย Dev ทำงานเร็วขึ้น 2 เท่า! | Skooldio Blogการ deployment เป็นหนึ่งในขั้นตอนการทำงานที่ dev ทุกคนต้องเจอไม่ว่าจะเป็นองค์กรขนาดเล็กหรือขนาดใหญ่ แต่สิ่งที่แต่ละองค์กรไม่เหมือนกันคือความยุ่งยากซับซ้อน ระยะเวลาในการ deploy ที่ต่างกัน ขึ้นอยู่กับ process และเครื่องมือที่ใช้ ซึ่ง Docker ก็เป็นหนึ่งในเครื่องมือยอดนิยมที่จะช่วยให้การ deployment รวดเร็วมากยิ่งขึ้น

Docker คือ เครื่องมือแบบ open-source ที่ช่วยจำลองสภาพแวดล้อม (environment) ในการรัน service หรือ server ตามหลักการสร้าง container เพื่อจัดการกับ library ต่างๆ อีกทั้งยังช่วยจัดการในเรื่องของ version control เพื่อง่ายต่อการจัดการกับปัญหาต่างๆ ที่เกิดขึ้น ซึ่งในปัจจุบันในโลกของการพัฒนา software มีรูปแบบการทำงานแบบ agile ที่เน้นความรวดเร็วในการส่งมอบงานในแต่ละขั้นตอน Docker จึงเป็นที่รู้จักในวงกว้างและเริ่มเข้ามามีบทบาทอย่างมากในโลกของการพัฒนา software อีกทั้งยังเป็นเครื่องมือที่จำเป็นสำหรับการทำ DevSecOps

**วิวัฒนาการของการ Deploy**

****

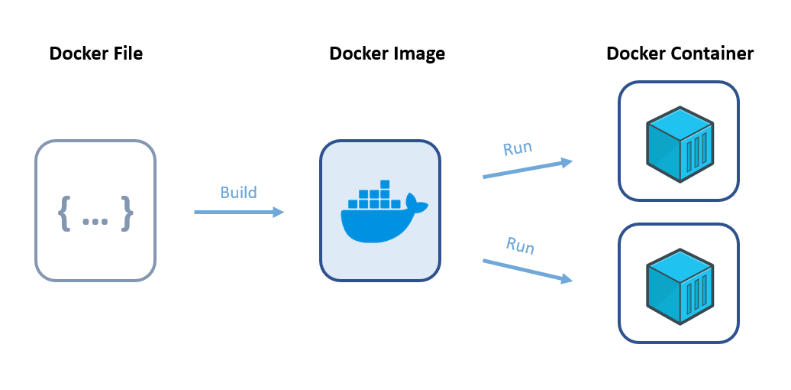
วิวัฒนาการของการ Deploy

1. Traditional Deployment ในยุคที่เราใช้ Physical server 1 เครื่อง ในการ deploy และเพื่อความคุ้มค่า Physical server ที่เรามีจะถูกใช้ในการลง application หลายๆ อันพร้อมๆ กัน ซึ่งทำให้เกิดปัญหา element ของแต่ละ application ตีกัน เช่น application แต่ละตัวมีการลง JAVA ซึ่งเป็นในเครื่องเรามี JAVA หลายๆ เวอร์ชัน 1.5, 1.6, 1.8 ทำให้เวลา run มีปัญหาเกิดขึ้น ทั้งการ maintenance และปัญหาในการเลือกเวอร์ชัน

2. Virtualized Deployment เป็นยุคที่มีการเกิดขึ้นของ software hypervisor ซึ่ง concept คือการจำลองเครื่อง Physical server ขึ้นมา เรียกว่า Virtual Machine ทั้ง CPU, memory, hard disk, hardware ต่างๆ ขึ้นมาเสมือนคอมพิวเตอร์เลย ปรับสเปคปรับความเร็วต่างๆ ได้ตามงบประมาณที่เรามี ซึ่งส่วนใหญ่ก็จะสร้าง VM ขึ้นมาหลายๆ เครื่อง ให้แต่ละเครื่องเพียงพอต่อการ run application แต่ละตัว แต่ปัญหาคือจะเกิดปัญหา Overhead ทั้งในเรื่องการจำลอง hardware ต่างๆ ทำให้ทำงานได้ช้าลง 2-5 เท่า รวมถึง Overhead ในกรณีที่เรามีการ run ใน environment ที่ใกล้เคียงกัน จะทำให้เปลือง resource ไปโดยใช่เหตุ

3. Container Deployment ซึ่งในยุคนี้จะพูดถึงการสร้าง Container ขึ้นมาเพื่อขัง resource สร้างกำแพงขึ้นมาแบ่ง resource ทำให้เราสามารถใช้ resource ได้อย่างมีประสิทธิภาพมากขึ้น เลือก container ไปใช้กับแต่ละ application ได้ดียิ่งขึ้น ทำให้ปัญหา overhead ลดลง และเครื่องมือในการสร้าง Container ที่นิยมก็คือ Docker

**องค์ประกอบพื้นฐาน Docker**

****

องค์ประกอบพื้นฐาน Docker

- Docker File คือ เอกสารบอกโค้ดคำสั่ง สำหรับสร้าง Docker Image นั้นๆ

- Docker Image เป็นแม่แบบที่ใช้ในการสร้างเป็น Docker Container ซึ่งประกอบไปด้วยแอปพลิเคชันต่างๆ ที่จะทำงานเมื่อมีการเรียกใช้งานจาก Docker Container นั้นๆ รวมทั้งการตั้งค่าจำลองสภาพแวดล้อม (Environment) ที่จำเป็นสำหรับการทำงานของมันไว้ด้วย

- Docker Container เป็นที่บรรจุรวมของแอปพลิเคชัน สภาพแวดล้อมที่จำเป็นต่อการทำงาน และองค์ประกอบต่างๆ ที่จำเป็นต่อการทำงานของมัน ซึ่งสามารถสร้างจาก Docker Image ผ่านการกำหนดโครงสร้างของมันที่ Docker File

**ข้อดี และข้อเสียของ Docker**

**ข้อดีของ docker**

- Portability ทดสอบ container ที่เดียวสามารถ deploy ได้ทุกที่ที่มี docker รันอยู่โดยไม่ต้องกลัวว่าจะไม่สามารถรันได้

- Performance เนื่องจาก container ไม่ได้มีการบรรจุ OS เข้าไปด้วย นั่นหมายความว่า docker นั่นจะมีขนาดเล็กกว่า VM ทำให้ขนาดเล็ก, build ได้เร็วกว่า รวมถึงการรันได้มีประสิทธิภาพดีกว่าด้วย

- Agility ด้วย portability และ performance ช่วยให้เหมาะสมกับการทำ agile process รวมถึงเหมาะกับการทำ CI/CD อีกด้วย ช่วยให้ compile, build, test ได้ดียิ่งขึ้น

- Scalability เราสามารถสร้าง container ใหม่ ได้ตามความต้องการของ application ที่ scale ได้ โดยใช้เวลาอันสั้น

**ข้อเสียของ docker**

- เนื่องจากการรัน docker ไม่ได้รัน OS ใหม่ทั้งหมดเป็นเพียงแค่การจำลอง env ทำให้อาจเกิดการโจมตีที่ OS หลักผ่านทาง docker ได้และอาจกระทบกับ container ตัวอื่นๆ

- ตอนเริ่มแรกที่ docker ถูกสร้าง มันถูกออกแบบมาเพื่อรองรับการรันบน linux เท่านั้น ที่เราสามารถรัน docker บน window,mac ได้นั้นเพราะ เมื่อเราลง docker ใน window,mac จะมีการสร้าง VM ที่เป็น linux เพื่อมารัน docker อีกที ทำให้ประสิทธิภาพการทำงานอาจจะไม่สามารถทำได้สูงสุดเท่ารันบน linux

- Docker ไม่เหมาะกับการจัดการ resource บนเครื่องใหญ่ๆ หรือไม่เหมาะกับโปรแกรมที่ออกแบบมาเพื่อทำงานบน VM

- Learning curve ที่สูง เนื่องจากเป็นการทำงานเกี่ยวกับ OS, network รวมถึงการจัดการทรัพยากรต่างๆ ทำให้ต้องอาศัยเวลาการเรียนรู้ที่ค่อนข้างสูง แต่ทาง docker ก็มี tool ใหม่ๆ ออกมาช่วยเหลือให้ใช้งานได้ง่ายยิ่งขึ้น แต่การที่จะใช้งาน docker ได้อย่างช่ำชองจำเป็นต้องเรียนรู้ tools อื่นเพื่อใช้ในการประกอบด้วย

https://blog.skooldio.com/what-is-docker/#Docker