การตรวจจับเปลวไฟจากการเรียนรู้เชิงลึก

วิทยานิพนธ์

ของ

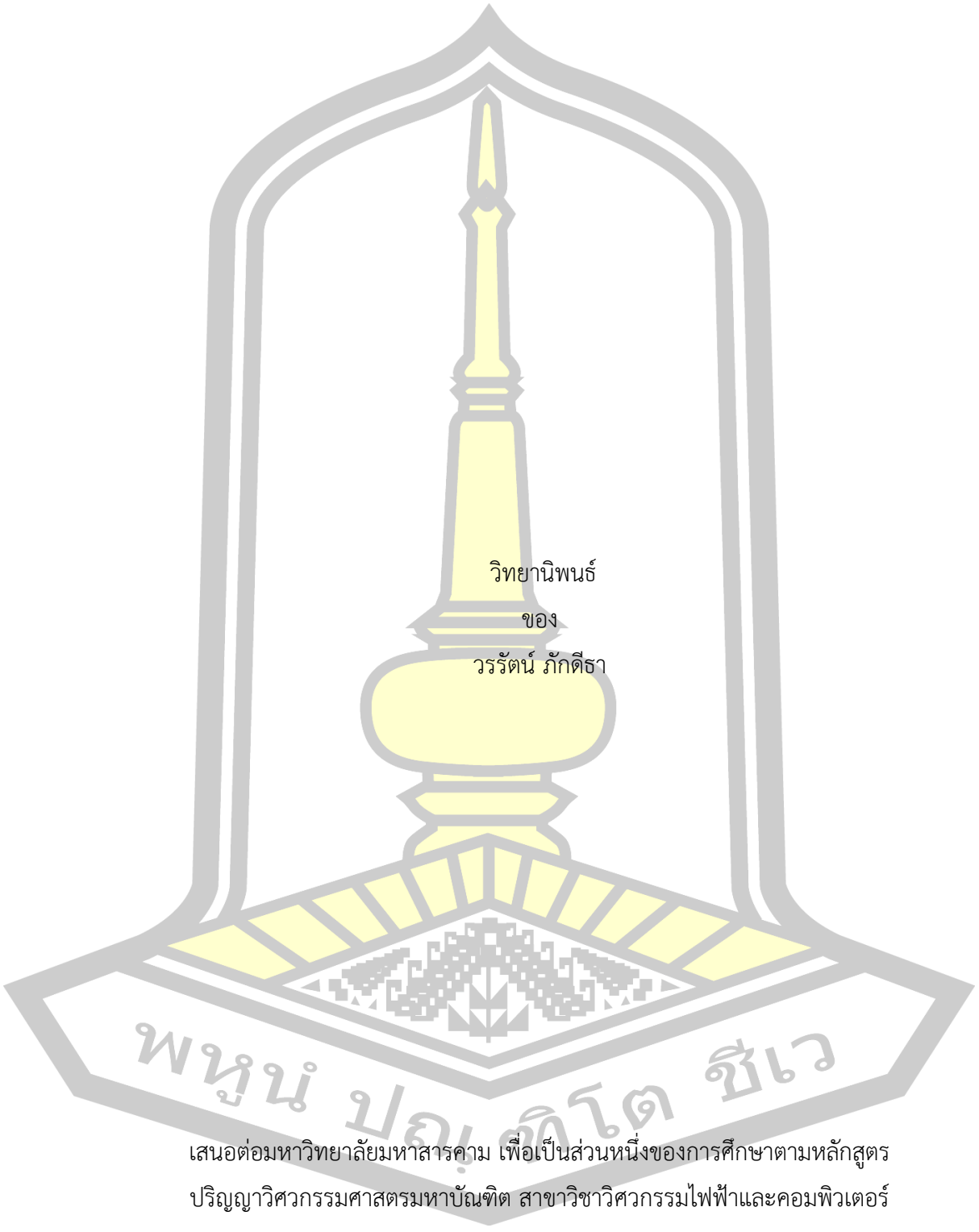
วรรัตน์ ภักดีธา

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

พฤษภาคม 2568

การตรวจจับเปลวไฟจากการเรียนรู้เชิงลึก



วิทยานิพนธ์

ของ

วรรัตน์ ภักดีธา

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

พฤษภาคม 2568

Flame Detection Based on Deep Learning

Worarat Phakditha

A Thesis Submitted in Partial Fulfillment of Requirements

for Master of Engineering (Electrical and Computer Engineering)

May 2025

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของนายวรรัตน์ ภักดีธา แล้ว เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รศ. ดร. อนันต์ เครือทรัพย์ถาวร)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผศ. ดร. ณัฐวุฒิ สุวรรณทา)

.....กรรมการ
(ผศ. ดร. ชัยยงค์ เสริมผล)

.....กรรมการ
(ศ. ดร. วรวัฒน์ เสงี่ยมวิบูล)

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....
(รศ. ดร. จักรมาส เลาหวณิช)
คณบดีคณะวิศวกรรมศาสตร์

.....
(ผศ. ดร. พลเดช เชาวรัตน์)
คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง การตรวจจับเปลวไฟจากการเรียนรู้เชิงลึก
ผู้วิจัย วรรัตน์ ภักดีธา
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ สุวรรณทา
ปริญญา วิศวกรรมศาสตรมหาบัณฑิต **สาขาวิชา** วิศวกรรมไฟฟ้าและคอมพิวเตอร์
มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2568

บทคัดย่อ

งานวิจัยนี้เสนอพัฒนาวิธีการตรวจจับเปลวไฟและควันแบบตามเวลาจริง โดยการนำภาพถ่ายของไฟไหม้สถานที่ต่างๆ นำมาเรียนรู้ด้วยโครงข่ายประสารทเทียมชนิดคอนโวลูชัน (CNN) โดยใช้โมเดล YOLOv8 จำนวน 5 ชนิดได้แก่ YOLOv8n YOLOv8s YOLOv8m YOLOv8l และ YOLOv8x โดยการเปรียบเทียบพารามิเตอร์ต่างๆได้แก่ Precision Recall F1-Score และ mAP จากผลการทดลองพบว่า YOLOv8m มีความสมดุลดีที่สุด โดยมี Precision เท่ากับ 87.3% Recall เท่ากับ 66.3% และ mAP50 เท่ากับ 75.8% ในขณะที่ YOLOv8x ให้ความแม่นยำสูงกว่าแต่ต้องใช้ทรัพยากรคำนวณมากขึ้น นอกจากนี้โมเดลที่ใช้ยังสามารถตรวจจับควันได้แม่นยำกว่าเปลวไฟ เนื่องจากเปลวไฟอาจก่อให้เกิด False Positives จากแสงสะท้อน ทำให้ความแม่ยำลดลง

คำสำคัญ : ตรวจจับเปลวไฟ, การเรียนรู้เชิงลึก, โครงข่ายประสาทเทียม, การตรวจจับแบบเรียลไทม์, YOLOv8

TITLE Flame Detection Based on Deep Learning

AUTHOR Worarat Phakditha

ADVISORS Assistant Professor Nattawoot Suwannata , Ph.D.

DEGREE Master of Engineering **MAJOR** Electrical and Computer Engineering

UNIVERSITY Mahasarakham University **YEAR** 2025

ABSTRACT

This study suggests creating a system that can detect flames and smoke in real-time using a convolutional neural network (CNN) trained on images of fire from different locations, using five versions of YOLOv8: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. The review of different factors like precision, recall, F1-score, and mAP shows that YOLOv8m gives the best balance, with a precision of 87.3%, a recall of 66.3%, and an mAP50 of 75.8%. Conversely, YOLOv8x offers enhanced accuracy but consumes more computational resources. The employed model can identify smoke with greater precision than flames, as flames may generate false positives due to reflected light, hence diminishing accuracy.

Keyword : Flame detection, Deep learning, YOLOv8, Artificial neural network, Real-time detection

กิตติกรรมประกาศ

ข้าพเจ้าขอแสดงความ ขอบพระคุณเป็นอย่างสูง ต่อ ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ สุวรรณทา อาจารย์ที่ปรึกษาหลัก ผู้ให้คำแนะนำ อย่างทุ่มเทและจริงใจ ตลอดระยะเวลาของการทำวิจัย ความรู้และประสบการณ์ของท่านเป็น แนวทางสำคัญ ที่ช่วยให้ข้าพเจ้าสามารถพัฒนาผลงานวิจัยจนสำเร็จลุล่วง

ข้าพเจ้าขอกราบขอบพระคุณ รองศาสตราจารย์ ดร.อนันต์ เครือทรัพย์ถาวร ประธานสอบวิทยานิพนธ์ รวมถึง ผู้ช่วยศาสตราจารย์ ดร. ชัยยงค์ เสริมผล และ ศาสตราจารย์ ดร.วรวัฒน์ เสงี่ยมวิบูล คณะกรรมการสอบ ที่ได้กรุณามอบข้อเสนอแนะ ที่เป็นประโยชน์และมีความสำคัญต่อการปรับปรุงวิทยานิพนธ์ ให้มีความสมบูรณ์มากยิ่งขึ้น

ข้าพเจ้าขอขอบคุณ คณาจารย์ในหลักสูตรปรัชญาดุษฎีบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม ตลอดจน เจ้าหน้าที่ทุกท่าน ที่ให้ความช่วยเหลือในด้านวิชาการและการบริหารจัดการงานวิจัย

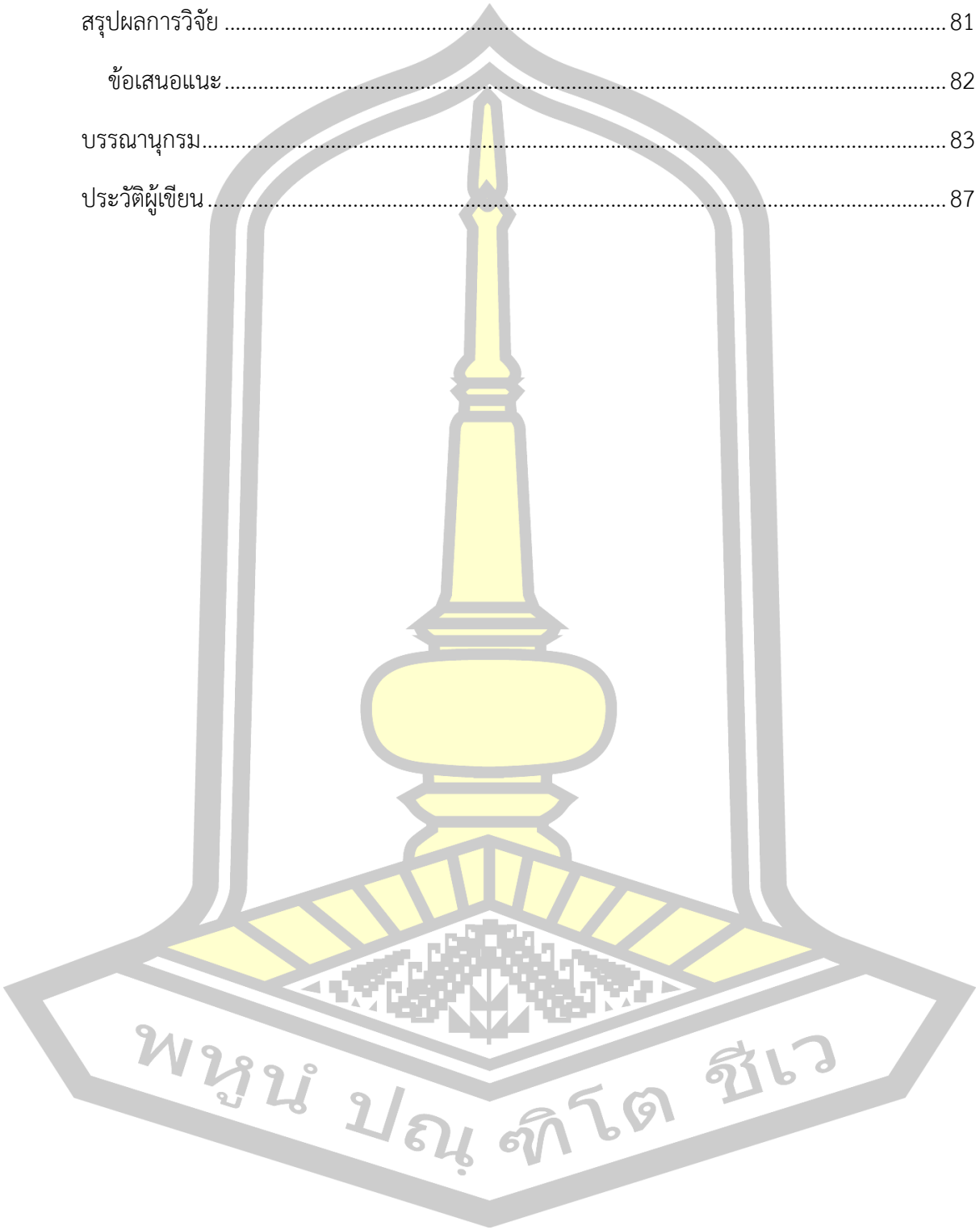
ข้าพเจ้าขอขอบพระคุณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหาสารคาม สำหรับการสนับสนุนทุนวิจัย IF ทุนวิจัยจาก งบประมาณเงินรายได้ ประจำปีงบประมาณ 2568 ซึ่งเป็นปัจจัยสำคัญที่เอื้อให้โครงการวิจัยนี้สามารถดำเนินไปได้จนสำเร็จ

สุดท้าย ข้าพเจ้าขอแสดง ความซาบซึ้งใจอย่างยิ่ง ต่อ บิดา มารดา และครอบครัว ที่ให้กำลังใจ ความรัก และการสนับสนุนอย่างไม่มีเงื่อนไข ตลอดระยะเวลาของการศึกษาวิจัย รวมถึงเพื่อนและเพื่อนร่วมงานทุกท่านที่ให้ แรงสนับสนุนและคำแนะนำที่เป็นประโยชน์ วิทยานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยความร่วมมือและแรงสนับสนุนจากหลายฝ่าย ข้าพเจ้าขอขอบพระคุณทุกท่านที่มีส่วนช่วยให้ผลงานนี้เกิดขึ้นและพัฒนาไปอย่างสมบูรณ์

วรรัตน์ ภักดีธา

สารบัญ

หน้า



สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1

บทนำ

1.1.ที่มาและความสำคัญ

การพัฒนาระบบตรวจจับเพลิงไหม้ในปัจจุบัน[1] เกิดขึ้นเป็นจำนวนมาก เนื่องจากการเกิดเหตุเพลิงไหม้แต่ละครั้งทำให้เกิดการสูญเสียชีวิตและทรัพย์สินจำนวนมหาศาล[2] การตรวจจับเพลิงไหม้ตั้งแต่เนิ่น ๆ ก่อนที่จะเกิดเพลิงไหม้ลุกลามเป็นวงกว้างจนเกินกว่าจะควบคุม มีความสำคัญเป็นอย่างมาก ด้วยปัจจุบันทางด้านปัญญาประดิษฐ์ (Artificial Intelligence) หรือ AI[3] กล่าวคือ วิทยาศาสตร์ที่เกี่ยวข้องกับการวิจัยและพัฒนาระบบคอมพิวเตอร์ที่สามารถมีความสามารถในการจดจำ แยกแยะ และประมวลผลข้อความ ภาพ เสียง เช่นการสร้างหุ่นยนต์ สร้างสมองกล สร้างระบบประสาทรับรู้เลียนแบบมนุษย์ สร้างระบบการมองเห็น การประมวลภาษาธรรมชาติของมนุษย์ การช่วยตัดสินใจ คาดการณ์ ทำนาย หรือการทำงานอื่น ๆ ที่เพียงแทนมนุษย์ในการตัดสินใจ Machine Learning[4] เป็นศาสตร์ย่อยของ AI ที่ทำให้คอมพิวเตอร์ หรือสมองกลเรียนรู้จากข้อมูล หรือกระบวนการเรียนรู้ แล้วนำความรู้นั้นมาใช้งาน วิเคราะห์ คาดการณ์ หรือขับเคลื่อนสิ่งต่าง ๆ ให้กับ AI เช่น การแสดงข้อมูล การแนะนำ การตัดสินใจ การควบคุมหุ่นยนต์ เครื่องจักร หรืออื่น ๆ ส่วน Deep Learning เป็นศาสตร์ย่อยของ Machine Learning ซึ่งมีวัตถุประสงค์เดียวกัน กล่าวได้ว่า ทำให้คอมพิวเตอร์เรียนรู้ และนำความรู้นั้นมาใช้งาน แต่ Deep Learning ใช้เทคนิคที่เรียกว่า เครือข่ายประสาทเทียม (Artificial Neural Network) ที่มีความลึกหลายชั้น (Deep Neural Network) หรือ DNN ที่จำลองการทำงานของเซลล์เครือข่ายสมอง หรือที่เรียกว่า "การเรียนรู้เชิงลึก" ในทางภาษาไทย

จากปัญหาการเกิดเพลิงไหม้ ที่กล่าวมาข้างต้น ในปัจจุบัน มีการสูญเสียทั้งชีวิตและทรัพย์สิน ซึ่งเครื่องตรวจจับอัคคีภัยในปัจจุบัน เช่น เครื่องตรวจจับอุณหภูมิแบบควัน เครื่องตรวจจับแบบอุณหภูมิ มีการใช้กันอย่างแพร่หลาย แต่จะช้ากว่าการตรวจจับ โดยใช้การประมวลผลภาพและการตรวจจับโดยใช้ปัญญาประดิษฐ์ ความแตกต่างระหว่างเทคนิคการตรวจจับไฟไหม้โดยใช้การประมวลผลภาพด้วย Deep Learning และ Machine Learning อยู่ที่ว่า Deep Learning จะใช้อัลกอริทึมที่เป็นปัญญาประดิษฐ์เชิงลึก เช่น Convolutional Neural Network (CNN) หรือ Recurrent Neural Network (RNN) ในขณะที่ Machine Learning จะใช้อัลกอริทึมที่ไม่ใช่ปัญญาประดิษฐ์เชิงลึก เช่น Support Vector Machine (SVM) หรือ Decision Tree โดยปกติแล้วเทคนิคการตรวจจับไฟไหม้โดยใช้การประมวลผลภาพด้วย Deep Learning มีความสามารถในการ

ตรวจจับไฟไหม้ได้ดีกว่าเทคนิคการตรวจจับไฟไหม้โดยใช้การประมวลผลภาพด้วย Machine Learning เนื่องจาก Deep Learning สามารถเรียนรู้ลักษณะของวัตถุในภาพได้ละเอียดมากยิ่งขึ้น

ดังนั้นสาระสำคัญของวิทยานิพนธ์นี้ ได้ถูกนำเสนอเพื่อวิเคราะห์ และเสนอแนวทางการป้องกันการเกิดเพลิงไหม้ ก่อนที่จะเกิดเพลิงไหม้ขนาดใหญ่อย่างเป็นระบบ ด้วยการใช้กล้องที่มีเทคโนโลยีการตรวจจับไฟไหม้ และการแจ้งเตือนทันทีผ่านระบบตรวจจับโดยใช้ปัญญาประดิษฐ์ โดยจะพบว่าวิทยานิพนธ์นี้ ได้ใช้เทคนิคการตรวจจับไฟไหม้ โดยใช้การประมวลผลภาพด้วยเทคโนโลยี Deep Learning ที่กำลังพัฒนาอย่างต่อเนื่อง การใช้อัลกอริทึม YOLOv8 และการเทรนโมเดลภาษา Python[5] เขียนใน PyCharm เป็นส่วนสำคัญของการพัฒนานี้ สุดท้ายนี้ คาดหวังว่าความมีประสิทธิภาพของระบบจะเพิ่มขึ้นอย่างมีนัยสำคัญในอนาคต

1.2.วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อสร้างระบบตรวจจับเปลวไฟ

1.2.2 เพื่อตรวจสอบความแม่นยำในการตรวจจับเปลวไฟ

1.2.3 โมเดลที่ฝึกสามารถนำไปประยุกต์ใช้งานต่อได้

1.3.ขอบเขตของงานวิจัย

1.3.1 ใช้ You Only Look Once Version 8 Algorithm ในการฝึกข้อมูล

1.3.2 ตรวจจับภาพด้วยกล้องแบบเรียลไทม์

1.3.3 นำเข้าข้อมูลภาพจำนวน 4,842 ภาพ โดยระบุตำแหน่งของไฟและควันด้วย Roboflow.com

1.4.ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้เข้าใจหลักการทำงานของระบบตรวจจับเปลวไฟ

1.4.2 ได้เข้าใจหลักการทำงานของการเรียนรู้เชิงลึก

1.4.3 สามารถนำความรู้ที่ได้ไปพัฒนาต่อยอดในการตรวจจับเปลวไฟ

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึง Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL), Convolutional Neural Network (CNN), การตรวจจับวัตถุ, การพัฒนาการของ YOLO และงานวิจัยที่เกี่ยวข้อง

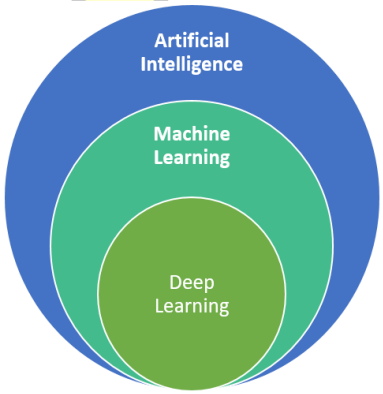
2.1 ปัญญาประดิษฐ์ (Artificial Intelligence)

ความหมายของปัญญาประดิษฐ์ AI [3] เป็นปัญญาประดิษฐ์ที่ใช้ประโยชน์จากคอมพิวเตอร์และเครื่องจักรเพื่อเลียนแบบความสามารถในการ แก้ปัญหาและการตัดสินใจเสมือนความรู้สึกนึกคิดของมนุษย์ โดยคำจำกัดความของปัญญาประดิษฐ์ (AI) ปรากฏขึ้นในช่วงสองสามทศวรรษที่ผ่านมา ในปี 1950 ได้มีการสนทนาเรื่องปัญญาประดิษฐ์ ผลงานของ Alan Turing เรื่อง “Computing Machinery and Intelligence” ถือเป็นจุดเริ่มต้น และในเวลาต่อมาถูกเรียกว่า “บิดาแห่งวิทยาการคอมพิวเตอร์” โดยมีการตั้งคำถามว่า “เครื่องจักร คิดได้หรือไม่” นอกจากนี้ ได้เสนอการทดสอบที่ในปัจจุบันรู้จักกันในชื่อ “Turing Test” โดยผู้ตรวจสอบที่ เป็นมนุษย์จะพยายามแยกแยะระหว่างการตอบกลับข้อความจากคอมพิวเตอร์และการตอบกลับ ข้อความของมนุษย์ การทดสอบนี้จึงเป็นส่วนสำคัญของประวัติศาสตร์ AI ที่นำแนวคิดเชิงปรัชญามา ใช้ในด้านภาษาศาสตร์ จากนั้นในปี พ.ศ. 2499 (ค.ศ. 1956) จอห์น แม็กคาร์ธีบัญญัติคำว่า 'ปัญญาประดิษฐ์' ในการประชุม AI ครั้งแรกที่วิทยาลัยดาร์ตมัธ (แม็กคาร์ธีจะคิดค้นภาษา Lisp ต่อไป) ต่อมาในปีนั้น Allen Newell, JC Shaw และ Herbert Simon ได้สร้าง Logic Theorist ซึ่งเป็นโปรแกรมซอฟต์แวร์ AI ตัวแรกที่รัน และได้ให้คำจำกัดความของ AI ว่าเป็น วิทยาศาสตร์ และวิศวกรรมของการสร้างเครื่องจักรอัจฉริยะ โดยเฉพาะโปรแกรมคอมพิวเตอร์ อัจฉริยะ มีความเกี่ยวข้องกับงานที่คล้ายกันของการใช้คอมพิวเตอร์เพื่อทำความเข้าใจความฉลาด ของมนุษย์ และไม่มีข้อจำกัดทางชีวภาพ

ในช่วงทศวรรษ 1950 และ 1960 ทางด้าน AI มุ่งเน้นไปที่การพัฒนาระบบที่สามารถคิดและแก้ปัญหาได้อย่างคล้ายคลึงกับมนุษย์ ตัวอย่างเช่น มีการวิจัยเกี่ยวกับระบบที่สามารถสร้างเหตุผล แปลภาษา และเล่นเกม ในช่วงทศวรรษ 1970 ทางด้าน AI ประสบกับความล้มเหลวครั้งใหญ่หลายครั้ง เช่น ระบบคอมพิวเตอร์ที่ไม่สามารถเอาชนะมนุษย์ในการเล่นหมากรุกได้ ซึ่งนำไปสู่การวิพากษ์วิจารณ์ว่า AI นั้นเป็นไปไม่ได้หรือไม่สามารถบรรลุได้ อย่างไรก็ตาม ในช่วงทศวรรษ 1980 และ 1990 AI ฟื้นตัวและเริ่มพัฒนาอย่างมีนัยสำคัญ ในช่วงนี้ มีการวิจัยเกี่ยวกับการเรียนรู้ของเครื่อง

(Machine Learning) ซึ่งเป็นสาขาย่อยของ AI ที่ช่วยให้เครื่องจักรสามารถเรียนรู้จากข้อมูลได้โดยไม่ต้องมีการเขียนโปรแกรมอย่างละเอียด การเรียนรู้ของเครื่องเป็นจุดเปลี่ยนครั้งสำคัญสำหรับ AI และนำไปสู่การพัฒนาระบบที่มีประสิทธิภาพมากขึ้น เช่น ระบบการจดจำใบหน้า ระบบแปลภาษา และระบบการวินิจฉัยทางการแพทย์

ในช่วงทศวรรษ 2000 และ 2010 ทางด้าน AI ยังคงพัฒนาอย่างต่อเนื่องและเริ่มมีบทบาทสำคัญในหลากหลายสาขา เช่น การแพทย์ การเงิน การผลิต และยานยนต์ ในช่วงนี้ มีการวิจัยเกี่ยวกับการเรียนรู้เชิงลึก (Deep Learning) ซึ่งเป็นสาขาย่อยของการเรียนรู้ของเครื่องที่มุ่งเน้นการใช้โครงข่ายประสาทเทียม (Neural Networks) เพื่อเรียนรู้จากข้อมูล การเรียนรู้เชิงลึกมีประสิทธิภาพเหนือกว่าเทคนิคการเรียนรู้ของเครื่องอื่น ๆ ในหลายด้าน เช่น ความสามารถในการเรียนรู้จากข้อมูลจำนวนมากและซับซ้อน และความสามารถในการเรียนรู้คุณสมบัติที่ซ่อนอยู่ในข้อมูล



ภาพประกอบ 1 AI, Machine Learning และ Deep Learning[6]

จากภาพประกอบ 1 จะเห็นว่า Artificial Intelligence: AI เป็นศาสตร์แขนงหนึ่ง ที่มีหลายสิบปีแล้ว เป็นเรื่องที่เกี่ยวกับการศึกษาวิจัยและพัฒนาระบบคอมพิวเตอร์รวมถึงจักรกลให้มีความสามารถอันชาญฉลาด สามารถจดจำ แยกแยะ และประมวลผลข้อความ ภาพ เสียงได้ เช่น การสร้างหุ่นยนต์ สร้างสมองกล สร้างระบบประสาทรับรู้เรียนแบบมนุษย์ สร้างระบบการมองเห็น การประมวลภาษาธรรมชาติของมนุษย์ การตัดสินใจแทนมนุษย์ การทำให้หุ่นยนต์สื่อสารโต้ตอบกับมนุษย์ได้และมีอื่นๆอีกมากมาย

Machine Learning หรือ ML [7] กล่าวคือ ศาสตร์ย่อยแขนงหนึ่งของ Artificial Intelligence (ดังภาพประกอบที่1) ที่ทำให้คอมพิวเตอร์หรือสมองกลเกิดการเรียนรู้จากข้อมูลหรือจากกระบวนการเรียนรู้ด้วยเครื่อง แล้วนำความรู้มาใช้งาน วิเคราะห์ คาดการณ์ หรือขับเคลื่อนสิ่ง

ต่างๆ ให้กับ AI เช่น แสดงข้อมูล แนะนำ ตัดสินใจ ควบคุมหุ่นยนต์ เครื่องจักร หรืออื่น ๆ อาจกล่าวได้ว่า Machine Learning เป็นส่วนสำคัญที่สนับสนุนการทำงานของ AI

Deep Learning [8] กล่าวคือ เป็นศาสตร์แขนงย่อยที่อยู่ใน Machine Learning อีกชั้นหนึ่งที่มีเป้าหมายเช่นเดียวกับ Machine Learning คือทำให้คอมพิวเตอร์เกิดการเรียนรู้ แล้วนำความรู้นั้นมาใช้งานแต่ Deep Learning ใช้วิธีการหรือเทคนิคลักษณะเครือข่ายประสาทเทียม (Artificial Neural Network) มีความลึกหลายชั้น (Deep Neural Network หรือ DNN) ที่เลียนแบบการทำงานของเซลล์เครือข่ายสมอง (ภาษาไทยใช้คำว่า การเรียนรู้เชิงลึก)

2.2 เครื่องจักรกลเรียนรู้ (Machine Learning)

การเรียนรู้ของเครื่อง (Machine Learning) หรือ ML กล่าวคือ ศาสตร์ย่อยของปัญญาประดิษฐ์ (Artificial Intelligence) หรือ AI ที่มุ่งเน้นการพัฒนาอัลกอริทึมและเทคนิคการเรียนรู้จากข้อมูล โดยไม่ต้องเขียนโปรแกรมแบบดั้งเดิม กล่าวอีกนัยหนึ่ง ML คือการสอนคอมพิวเตอร์ให้เรียนรู้และปรับตัวจากข้อมูล โดยสามารถทำนายผลลัพธ์ในอนาคตหรือตัดสินใจได้อย่างชาญฉลาด คล้ายกับกระบวนการเรียนรู้ของมนุษย์ กระบวนการทำงานของ Machine Learning ดังต่อไปนี้

- การรวบรวมข้อมูล คือ ขั้นตอนแรกคือการรวบรวมข้อมูลที่เกี่ยวข้องกับงานที่ต้องการ ตัวอย่างเช่น ข้อมูลภาพ เสียง ข้อความ หรือตัวเลข ข้อมูลเหล่านี้จะกลายเป็น "อาหาร" ให้โมเดล ML เรียนรู้
- การเตรียมข้อมูล คือ ข้อมูลที่รวบรวมมาอาจมีสภาพไม่พร้อมใช้งาน ต้องผ่านการทำความสะอาด จัดการค่าที่หายไป แปลงข้อมูลให้เป็นรูปแบบที่เหมาะสม
- การเลือกโมเดล คือ มีโมเดล ML หลายประเภท แต่ละแบบเหมาะกับงานที่ต่างกัน ตัวอย่างเช่น การคาดการณ์ตัวเลข เช่น Linear Regression, Decision Tree การจำแนกประเภท เช่น Logistic Regression, Support Vector Machine การหาความสัมพันธ์ เช่น K-Nearest Neighbors การเรียนรู้ภาพหรือเสียง เช่น Convolutional Neural Network (CNN)
- การฝึกโมเดล คือ ขั้นตอนสำคัญคือการฝึกโมเดล โดยให้อาหารข้อมูลให้โมเดลเรียนรู้ ปรับแต่งพารามิเตอร์ภายในตัวเองให้สามารถทำนายผลลัพธ์ได้แม่นยำ
- การประเมินโมเดล คือ เมื่อฝึกเสร็จ ต้องประเมินประสิทธิภาพโมเดลด้วยข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน ว่าโมเดลสามารถทำนายได้ถูกต้องแค่ไหน
- การใช้งานโมเดล คือ โมเดลที่มีประสิทธิภาพดีสามารถนำไปใช้งานจริง เช่น แนะนำสินค้า กรองอีเมล แปลภาษา วิเคราะห์การตลาด ช่วยแพทย์วินิจฉัยโรค ฯลฯ

เครื่องจักรกลเรียนรู้ (Machine Learning) แบ่งออกเป็นประเภทหลัก ๆ ทั้งหมด 3 ประเภท ตามลักษณะของข้อมูลที่ใช้เรียนรู้และเป้าหมายของการเรียนรู้ จะกล่าวดังต่อไปนี้

2.2.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning) [9]

เป็นประเภทของ Machine Learning ที่มีการจัดเตรียมข้อมูลที่เรียกว่า "Training Data" ซึ่งประกอบด้วยทั้งข้อมูลนำเข้า (Input) และผลลัพธ์ที่ต้องการ (Target) โดยอัลกอริทึมจะเรียนรู้จากข้อมูลเหล่านี้เพื่อค้นหาฟังก์ชันที่สามารถแปลงข้อมูล Input ให้กลายเป็น Target ได้อย่างแม่นยำที่สุด สามารถแบ่งย่อยออกเป็น 2 กลุ่ม ดังนี้

1. Classification การจำแนกแยกแยะ กล่าวคือ เป็นการจำแนก คัดแยก แยกแยะ เช่น แยกแยะว่าเป็น แมว หรือ สุนัข ระบบคัดแยก Spam Mail ก็จะแยกได้เพียงแค่ Mail ไหนเป็น Spam Mail ไหนไม่เป็น Spam ระบบคัดแยกภาพเนื้อเยื่อว่าป่วยหรือไม่ป่วย ฯลฯ

2. Regression กล่าวคือ คำนวณทำนายค่าเป็นตัวเลข เช่น ทำนายปริมาณการใช้กระแสไฟฟ้าในอนาคต ทำนายการขึ้นลงของหุ้น ทำนายการใช้วัตถุดิบในอนาคต ทำนายผลกำไร ฯลฯ

2.2.2 การเรียนรู้แบบไร้ผู้สอน (Unsupervised Learning) [10]

เป็นการเรียนรู้แบบไม่ต้องมีการสอน ไม่จำเป็นต้องใช้ข้อมูลที่มี Output หรือไม่ต้องมีข้อมูล Label/Target จึงเรียกว่า Unsupervised ลักษณะการทำงานคือป้อนข้อมูลที่ต้องการทำนาย จากนั้นระบบจะทำการประมวลผลข้อมูลให้เอง การเรียนรู้ประเภทนี้ แบ่งออกเป็น การจัดข้อมูล (Clustering) และการหาความสัมพันธ์ (Association)

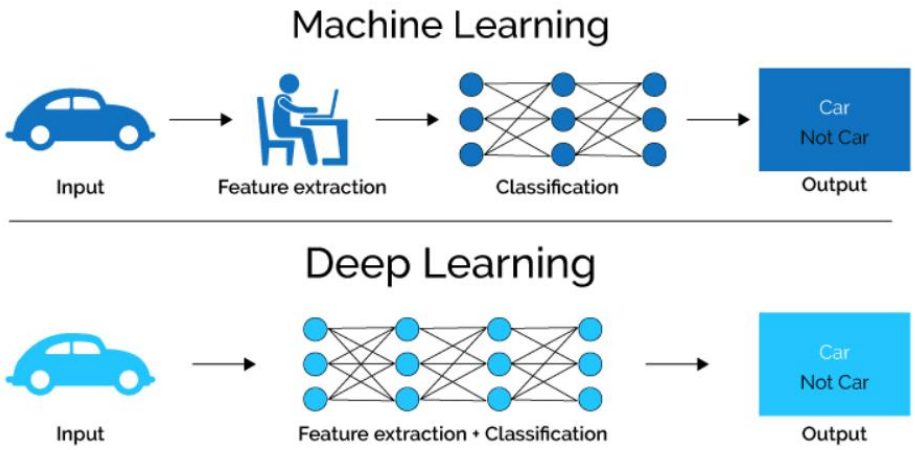
2.2.3 การเรียนรู้แบบเสริมแรง (Reinforcement Learning)[11]

เป็นระบบการเรียนรู้ที่อาศัยการป้อนกลับแล้วให้ระบบเรียนรู้แล้วปรับปรุงตัวเอง เช่น ระบบทรงตัวของหุ่นยนต์ เริ่มแรกตัวหุ่นอาจล้มในตอนแรก ๆ ระบบจะทำการป้อนกลับแล้วนำค่าข้อมูลต่าง ๆ ในการทรงตัวมาปรับปรุงตัวเอง จนในที่สุดสามารถทรงตัวได้อย่างเสถียรภาพ ฯลฯ

2.3 Deep Learning

การเรียนรู้เชิงลึก (Deep Learning) [12] กล่าวคือ เป็นศาสตร์ย่อยของการเรียนรู้ของเครื่อง (Machine Learning) ที่ใช้โครงข่ายประสาทเทียม (Artificial Neural Network) หลายชั้นในการประมวลผลข้อมูลและเรียนรู้รูปแบบที่ซับซ้อนโครงข่ายประสาทเทียมเป็นโมเดลทางคณิตศาสตร์ที่จำลองการทำงานของสมองมนุษย์ โดยประกอบด้วยหน่วยประมวลผล (Neuron) จำนวนมากเชื่อมต่อกันแบบเครือข่าย แต่ละหน่วยประมวลผลจะรับข้อมูลขาเข้าจากหน่วยประมวลผลอื่น ๆ และ

ประมวลผลข้อมูลเหล่านั้นเพื่อสร้างข้อมูลขาออกการเรียนรู้เชิงลึกใช้โครงข่ายประสาทเทียมหลายชั้นในการประมวลผลข้อมูล โดยแต่ละชั้นของโครงข่ายจะเรียนรู้ รูปแบบที่ซับซ้อนที่แตกต่างกันไป



ภาพประกอบ 2 Machine Learning และ Deep Learning[12]

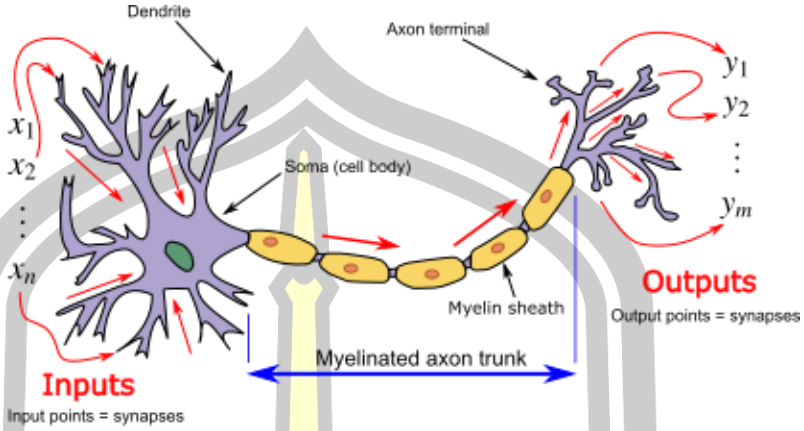
จากภาพประกอบ 2 ความแตกต่างระหว่าง Machine Learning และ Deep Learning โดยรูปด้านบนคือ Machine Learning หลังจาก Input ข้อมูลเข้ามา จะทำการค้นหาข้อมูลที่มีลักษณะเด่นโดยมนุษย์ ก่อนจะนำไป Classification แล้วค่อย Output ออกมา ส่วนรูปด้านล่างคือ Deep Learning หลังจาก Input ข้อมูลเข้ามา จะทำการเรียนรู้และค้นหา Feature Extraction เองด้วยเครือข่ายหลายชั้น เหมือนกับเรียนรู้จากภาพเองโดยตรง

2.3.1 Artificial Neural Network

สถาปัตยกรรมของ Deep Learning ประกอบด้วย Artificial Neural Network (ANN) โครงข่ายประสาทเทียม หรือที่เรียกสั้น ๆ ว่า Neural Networks (NNs) เป็นสาขาหนึ่งของโมเดล การเรียนรู้ของเครื่อง ที่สร้างขึ้นโดยใช้หลักการของการจัดระเบียบเซลล์ประสาทที่ค้น พบโดยการเชื่อมโยงในโครงข่ายประสาทเทียมทางชีววิทยา

2.3.2 Neuron

Artificial Neural Network เป็นของระบบคอมพิวเตอร์ ระบบดังกล่าวได้ออกแบบโดยอาศัยหลักการของเซลล์ประสาทจริง โดยในร่างกายเรามีเซลล์ประสาท (Neuron) จำนวนมากมายมหาศาล เชื่อมต่อกันเป็นโครงข่ายที่ซับซ้อน ทำหน้าที่รับรู้สิ่งเร้าภายนอกและภายในร่างกาย เช่น รับรู้ความรู้สึก ความร้อน เสียง แสง และทำหน้าที่ตอบสนองต่อสิ่งต่าง ๆ โครงสร้างของ Neuron ประกอบด้วยส่วนต่าง ๆ ดังนี้

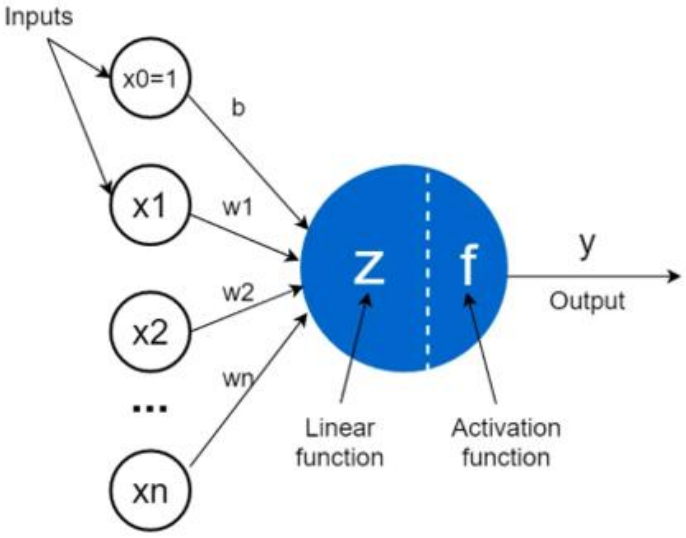


ภาพประกอบ 3 เซลล์ประสาท (Neuron)[13]

- หน่วยรับอินพุตหรือใยประสาทนำเข้า Dendrite ทำหน้าที่รับสัญญาณสิ่งเร้าภายนอกและภายในร่างกาย (สัญญาณจากเซลล์อื่น ๆ)
- ตัวเซลล์ (Cell Body) ภายในมีนิวเคลียส ซึ่งเป็นตัวประมวลผลหรือตัดสินว่าจะต้องกระตุ้นเซลล์อื่นๆต่อหรือไม่ ถ้ามีสัญญาณอินพุตแรงพอ ก็จะทำการกระตุ้นเซลล์อื่น ๆ ต่อไปโดยส่งเอาต์พุตผลลัพธ์ (Output) ซึ่งเป็นสัญญาณไฟฟ้าเคมีอย่างหนึ่ง ผ่านใยสมองหรือแกนประสาทนำออก (Axon) ไปยังหน่วยรับอินพุตลำดับถัดไป (Dendrite โครงข่ายของชุดเซลล์ประสาทอื่น)
- ใยประสาทนำออก (Axon) ทำหน้าที่ส่งสัญญาณออก (Output)

2.3.3 Artificial Neuron และ Perceptron

Artificial Neuron [14] เป็นเซลล์ประสาทที่มีการเลียนแบบหรืออาศัยหลักการทำงานของเซลล์ประสาทของมนุษย์ ในทางคอมพิวเตอร์เรียกว่า ประสาทเทียม หรือ Artificial Neuron (AN) เป็นโปรแกรมคอมพิวเตอร์ที่เขียนขึ้น จะเรียกว่าเป็นฟังก์ชันทางคณิตศาสตร์ก็ได้ ทำหน้าที่คล้ายเซลล์ประสาทจริง คือรับอินพุต (x_1 x_2 ... x_n ซึ่งใน Machine Learning และ Deep Learning ก็คือข้อมูล Feature) แล้วทำการประมวลผลหรือตัดสินใจ จากนั้นให้ผลลัพธ์ Output ออกมา ดังภาพประกอบที่ 4 โดยหน่วย Neuron นี้ เรียกว่า Perceptron หน้าที่รับข้อมูลอินพุต (Feature x_1 ... x_n) ได้แก่ข้อมูลต่างๆแล้วทำการประมวลผล ตัดสินหรือทำนาย (Predict) ว่าผลลัพธ์จะเป็น 1 หรือ 0 ซึ่งค่า Output อาจแทนผลว่าใช่หรือไม่ใช่



ภาพประกอบ 4 Perceptron และ ฟังก์ชัน[15]

การทำงาน

เริ่มต้นหน่วยอินพุตจะรับข้อมูลหรือค่า Input เข้ามา เช่น รับค่าน้ำหนัก ความดัน อายุ ฯลฯ เมื่อรับค่ามาแล้วก็จะเข้าสู่ขั้นตอนการประมวลผล ประกอบด้วย 2 ขั้นตอน ดังต่อไปนี้

รวมผล (Sum)

การรวมผลของ Perceptron ทำโดยนำค่า Input คูณด้วย weight ของแต่ละตัว ($W_n \times X_n$) จากนั้นนำผลลัพธ์มารวมกัน โดยรวมค่า w_0 (bias) ด้วย เมื่อได้ผลรวมแล้วก็ส่งไปตัดสินในขั้นต่อไป โดยสมการทางคณิตศาสตร์ที่ใช้รวมผล มีฐานมาจากสมการเชิงเส้นทั่วไปคือ y = mx+b ดังสมการที่ 1

S คือ ผลรวมหรือ Sum = bias (w_0) + [ค่า Input แต่ละตัว (x_n) คูณ weight (w_n)...]

$$S = W_0 + (W_1 \times X_1) + (W_2 \times X_2) + \ldots (W_n \times X_n) \quad (1)$$

ในทางคณิตศาสตร์เขียนเป็นผลรวมดังสมการที่ 2

$$s = \sum_{i=0}^{n} w_i \times x_i \quad (2)$$

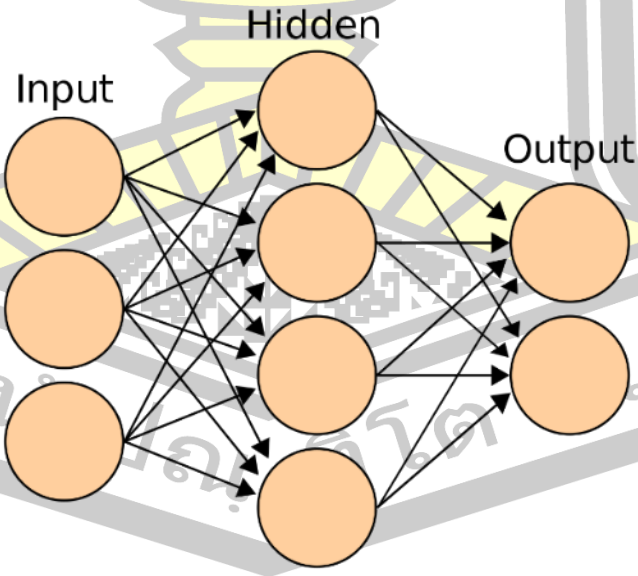
ตัดสิน (Activation function)

ค่าผลรวมหรือ Sum นำมาประมวลผลตัดสินต่อ โดยจะมีค่าตัวเลขที่ใช้เป็นค่าเกณฑ์ตัดสิน ที่เรียกว่า Threshold ค่านี้จะกำหนดให้ใช้ค่าเท่าไรก็ได้ ขึ้นอยู่กับการออกแบบ การทำงานมีขั้นตอนดังนี้คือ นำ Sum มาเปรียบเทียบกับค่า Threshold(t) ถ้าผลรวม S ในสมการ มีค่าเท่ากับหรือสูงกว่าค่า Threshold(t) การตัดสินจะให้ผลลัพธ์ Output เป็น 1 แต่ถ้า S น้อยกว่า Threshold(t) ผลการตัดสินจะให้ Output เป็น 0 ดังสมการที่ 3

$$y = \begin{cases} 1 \text{ if } W_i X_i \geq \text{Threshold} \\ 0 \text{ if } W_i X_i \leq \text{Threshold} \end{cases} \quad (3)$$

2.3.4 Multi-layer Perceptron

จากที่ผ่านมาคือ 1 หน่วย Perceptron หรือ 1 หน่วย Neuron ภาพประกอบที่ 4 โดย Input รับข้อมูล ส่งเข้า Neuron แล้ว Sum และ Activation ให้ผลเอาต์พุตเลย ซึ่งจะทำงานในลักษณะแบ่งกลุ่มข้อมูลออกเป็น 2 ส่วน โดยเส้นแบ่งมีลักษณะเป็นเส้นตรงเท่านั้น แต่ในงานที่จำเป็นต้องใช้เส้นแนวแบ่งแบบเส้นโค้งเช่น XOR ตัว Neuron จะมีปัญหา ทำงานผิดพลาด จำเป็นต้องใช้ Neuron แบบหลายชั้นที่เรียกว่า Multi-layer Perceptron (MLP) โครงสร้าง MLP ประกอบไปด้วยชั้น Layer ต่างๆ โดยแต่ละชั้นจะประกอบด้วยโนด (Node) ดังภาพประกอบที่ 5 โดยทั้งหมดเรียกว่า Model



ภาพประกอบ 5 MLP มี Hidden Layer [15]

การทำงาน

Input Layar คือ เป็นส่วนที่รับ Input ซึ่งก็คือรับค่า Feature โดยแต่ละ Node ในชั้นนี้จะไม่มีตัวตัดสิน (Activation Function) จะทำหน้าที่รับข้อมูลส่งเข้าไปประมวลผลต่อไปในชั้นถัดไปนั้นเอง

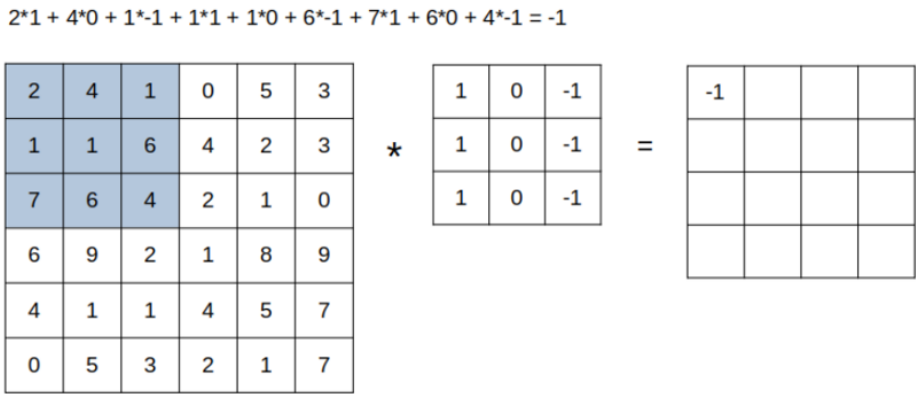
Hidden Layar คือ เป็น Perceptron หรือ Neuron จะมีกี่ Node ก็ได้ รับค่ามาจาก Input Layer ทำการประมวลผลแล้วตัดสิน คล้ายกับ Perceptron แต่ผลการตัดสินหรือ Output ที่ได้จาก Hidden Layer นี้ยังไม่ใช่ Output สุดท้าย แต่จะส่งให้ Perceptron อีกชั้นเป็นการประมวลผลต่อไป

Output Layer คือ เป็นส่วนที่รับข้อมูลต่อจาก Hidden Layer แล้วประมวลผลรวมและตัดสินใจ จากนั้นให้ค่าผลลัพธ์เอาต์พุตสุดท้ายออกมา

2.4 Convolutional Neural Network (CNN)

โครงข่ายประสาทเทียมแบบหมุนวน (Convolutional Neural Network) หรือ CNN [15] เป็น Neural Network (NN) ที่มีแนวคิดมาจากการรับรู้ของระบบประสาทการมองเห็นของมนุษย์ ที่มองวัตถุหรือภาพเป็นพื้นที่ย่อย ๆ ซึ่งการมองเห็นพื้นที่ย่อยของมนุษย์จะทำการแยกคุณลักษณะเด่นเอาไว้เพื่อประกอบการพิจารณา แล้วนำลักษณะเด่นที่แยกไว้มารวมกัน เพื่อให้ทราบรายละเอียดว่าวัตถุหรือภาพนั้นคืออะไร การ Convolution ในงาน Image Processing เป็นกระบวนการที่นำภาพ 2 ภาพมา ประมวลผลด้านคณิตศาสตร์เข้าด้วยกันทำให้เกิดข้อมูลภาพขึ้นมาใหม่ โดยมีขนาดเล็กลง เรียกว่า Mask หรือ Filter หรือ Kernel วัตถุประสงค์ของการทำ Convolution ก็เพื่อนำภาพต้นฉบับมาประมวลผลให้เกิด Effect ในลักษณะต่าง ๆ เช่น การตรวจจับหาเส้นขอบของวัตถุที่อยู่ในภาพ (Edge Detection) การทำให้ภาพคมชัดขึ้น การทำให้ภาพเบลอ การลด Noise ในภาพ ฯลฯ ซึ่งในงาน Deep Learning จะใช้ Convolution ในการหาคุณลักษณะเด่น Feature ที่มีอยู่ในภาพ

2.4.1 ลักษณะการทำงานของ Filter



ภาพประกอบ 6 แสดงการทำ Convolution [16]

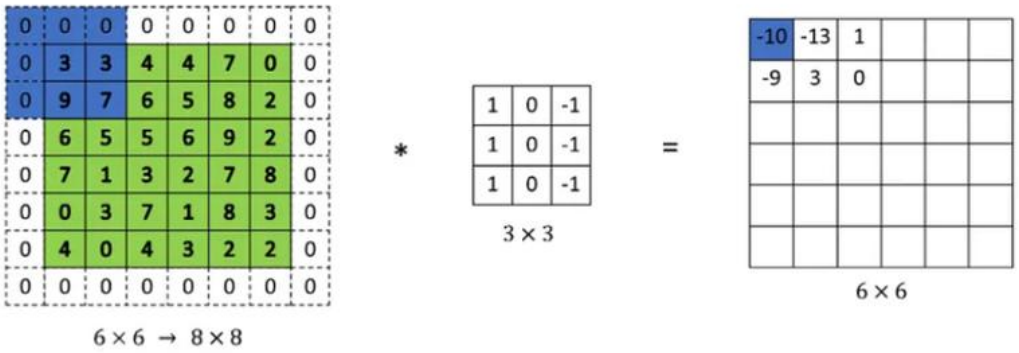
1) นำ Kernel (เรียกว่า Mask หรือ Filter ก็ได้) ไปทาบกับ Input Image ลักษณะ Pixel ต่อ Pixel ช่องต่อช่องในภาพประกอบ 6
2) นำข้อมูลใน Input Image และ Kernel มาคูณกัน แล้วรวมผลลัพธ์ จากนั้นนำไปสร้างเป็น Pixel ใหม่เรียกว่า Feature Map
3) ทำการเลื่อน Kernel ไปอีกช่อง ในแนวระนาบแล้วทำการคำนวณในลักษณะเดิม นำค่าที่ได้ไปสร้างเป็น Pixel ลำดับถัดไปของ Feature Map
4) เลื่อน Kernel และทำซ้ำลักษณะเดียวกันจนสุดท้ายจะได้ภาพ Feature Map สำหรับ 1 Kernel

2.4.2 Stride

จากตัวออย่างที่กล่าวมา เป็นการเลื่อนขยับในแนวนอนและแนวดิ่งแบบทีละ 1 ช่อง หรือ 1 Pixel ในทางเทคนิคเรียกว่า Stride = (1, 1) นอกจากนี้ยังกำหนดให้ขยับทีละเท่าใดก็ได้ เช่น (2, 2) คือขยับแนวนอนทีละ 2 เมื่อสุดแล้วขยับลงทีละ 2 ฯลฯ

2.4.3 Padding

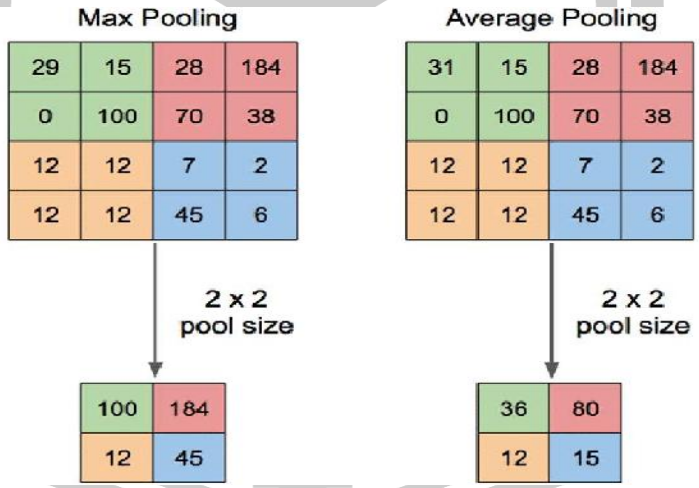
ลักษณะการทำ Convolution จะทำให้ภาพผลลัพธ์ Feature Map ที่ได้ มีขนาด (WxH) เล็กลง เนื่องจากการทาบ Kernel จะอยู่ในบริเวณภาพ Input Image ทำให้ขอบต้นฉบับหายไป คล้ายกับถูกลบขอบ ยิ่งถ้าใช้ Kernel ขนาดใหญ่ ขนาดของภาพ Feature Map ที่ได้จะเล็กลงมากตามไปด้วย อย่างไรก็ตามหากต้องการให้มิติ Feature Map มีขนาดเท่ากับภาพต้นฉบับ จะต้องเติมขอบเข้าไปก่อนการทำ Convolution เรียกว่าการ Padding ในภาพประกอบ 7 ทำการเติมขอบจาก 6x6 เป็น 8x8 ซึ่งหลังจากการทำ Convolution แล้วจะได้ผล 6x6 เท่าเดิม



ภาพประกอบ 7 แสดงการทำ Padding เติมขอบรูป[17]

2.4.4 Pooling

Pooling เป็นการย่อหรือลดขนาดของรูปภาพลง ใน Image Processing การลดขนาดมีหลายวิธี แต่ในงาน CNN Deep Learning หลักๆ มีวิธี Max Pooling และ Average Pooling โดยลักษณะการทำงานของ Pooling คล้ายกับการทำ Convolution คือการใช้ภาพขนาดเล็กหรือ Pooling Layer ไปทาบแล้วเลื่อนขยับไปเรื่อย ๆ ดังภาพประกอบ 8



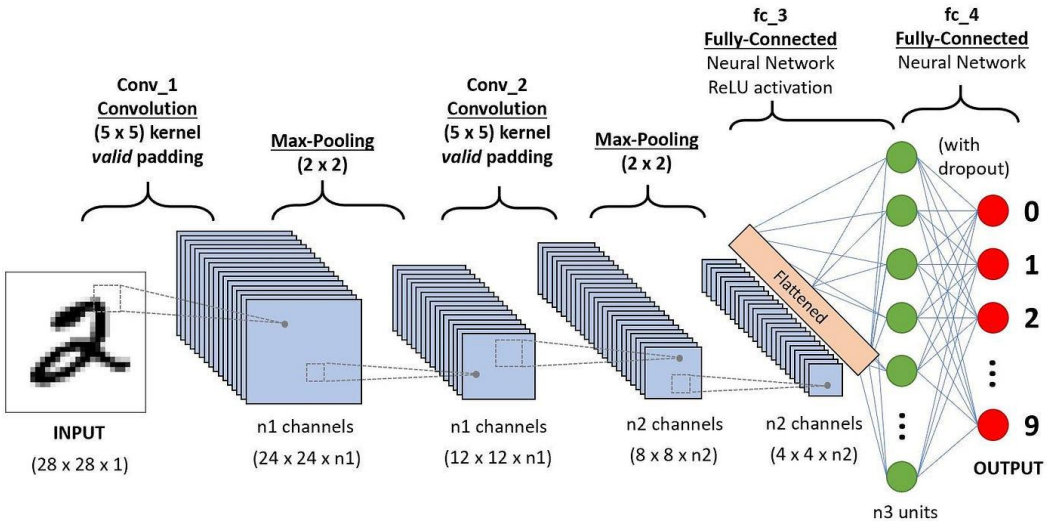
ภาพประกอบ 8 แสดงวิธีการทำ Pooling วิธี Max Pooling และ Average Pooling[17]

Max Pooling ใช้วิธีการหาค่า Max ในการทาบแต่ละครั้ง ส่วน Average Pooling ใช้วิธีการหาค่าเฉลี่ย โดยนิยมใช้ Max Pooling มากกว่า Pool Size ขึ้นอยู่ที่การเลือกใช้ โดยทั่วไปจะพบ Pooling 2x2 มากกว่า การใช้ Size 2x2 เท่ากับว่า ลดขนาดภาพลงไปด้านละครึ่ง(W/2,H/2) ถ้าพิจารณาในลักษณะพื้นที่หรือมิติ Feature จะลดลงไปถึง 75% เลย เช่น ภาพต้นฉบับ WxH =

10x10 = 100 เท่ากับว่ามีมิติ Feature เท่ากับ 100 แต่ถ้าทำ Pooling ด้วย 2x2 ขนาดจะลดลงเหลือเพียง 5x5 = 25 ซึ่งเท่ากับ Feature มิติเพียง 25 เท่านั้น เหลือเพียง 3/4 ลดลงไป 75% เลยทีเดียว ทำให้ลดภาระการประมวลผลของคอมพิวเตอร์ลงไปได้มาก (การลดมากเกินไป ก็จะทำให้ประสิทธิภาพลดลงตามไปด้วย)

โดยรวมแล้ว Pooling ถือว่าลดขนาดมิติของภาพแต่ยังคงค่าคุณลักษณะเด่น (Feature) ของเค้าโครงสิ่งที่อยู่ในภาพไว้ ในงาน Deep Learning ไม่ต้องการความชัดของภาพแต่ต้องการข้อมูลคุณลักษณะเด่นที่อยู่มนภาพ เพื่อจำแนกแยกแยะ

CNN เป็นการรวมของสองส่วนคือ Feature Extraction และ ส่วนของ Neural Network หรือ ส่วน Classification



ภาพประกอบ 9 แสดงสถาปัตยกรรมและกระบวนการ CNN[8]

สำหรับการแสดงสถาปัตยกรรมและกระบวนการของ CNN จะมีลักษณะคล้ายภาพประกอบ 9 ซึ่งมีความลึก Deep ของชั้น Layer หลายชั้น โดยการทำงานมีดังนี้

Input เป็นข้อมูลที่ป้อนเข้าระบบเพื่อทำการ Train, Validate, Test และทำการ Predict ซึ่งจะเป็นภาพโทนเทา (Grayscale) 1 Channel หรือ ภาพสีก็ได้ โดยถ้าเป็นภาพสี ก็จะมีขนาดมิติด้านลึกเป็น 3 Channel จากรูปตัวอย่าง Input Image เป็นภาพขนาด WxH 28x28x1 (1 Channel)

Convolution เป็นชั้นที่นำข้อมูล Input มาผ่านกระบวนการ Convolution ด้วย Kernel แล้วแต่ขนาดที่กำหนด เช่น 3x3 5x5 ฯลฯ จะได้ผลลัพธ์เป็น Feature Map 1 ภาพ ต่อ 1 Kernel ดังนั้น ถ้ากำหนดให้มี Kernel มี 8 ตัว ก็จะได้ Feature Map จำนวน 8 ภาพ ปกติในชั้น

Convolution เดียวกัน จะใช้ Kernel ที่มีขนาดเท่ากัน แต่ค่าภายในของ Kernel จะแตกต่างกัน เช่น อันแรกตรวจจับขอบตรง อันต่อไปตรวจจับขอบโค้ง เป็นต้น.

Max Pooling เป็นชั้นที่นำ Feature Map มาลดขนาดลง เช่น ถ้าใช้ Pool Size 2x2 ก็จะลดขนาดความกว้างความสูง (WxH) ของ Feature Map ลงไปครึ่งนึ่ง หลังการทำ Convolution เสร็จ จะต่อด้วย Max Pooling เลย (Convolution -> Max Pooling) หรือ Convolution 2 ครั้ง ต่อด้วย Max Pooling 1 ครั้งก็ได้ (Convolution2 -> Max Pooling) หรือจะทำการ Convolution + Max Pooling ต่อกัน ซึ่งรวมๆ แล้วจะทำ 1 ครั้ง หรือมากกว่าก็ได้ จากตัวอย่างรูป ทำ Convolution + Max Pooling 2 ครั้ง ก็จะได้ภาพ Feature Map ที่ขนาดเล็กลง เช่น 24x24 ผ่าน Convolution + Max Pooling ครั้งแรก จะเหลือ 12x12 พอผ่าน Convolution + Max Pooling อีกรอบ จะเหลือ 4x4 ฯลฯ (ใช้ Pool Size เท่ากันในขั้นเดียวกัน) ส่วนใหญ่แล้ว Convolution + Max Pooling ชั้นถัดๆไป จะกำหนดให้มีความลึกจำนวนมากขึ้น (จำนวน Kernel มากขึ้น) เป็นค่า 2 ยกกำลัง เช่น 16 32 64 ....

Fully Connected Layer (FCL) กล่าวคือ ภาพ Feature Map ที่ได้จากชั้นก่อนหน้าจะมีข้อมูลเป็นลักษณะหลายมิติ ดังนั้นระบบจะทำการจัดข้อมูลให้เป็นแบบ เวเตอร์ (Vector) หรือ เมทริกซ์ 1 มิติ ก่อน เรียกว่า Flatten แล้วป้อนเข้าสู่ FCL ซึ่งเป็นชั้นที่เป็น Neural Network หรือ MLP โดยชั้นนี้จะมีอินพุตและ Weight เหมือนกับ Neural Network ปกติ ทำหน้าที่ Classification ให้ผลลัพธ์ Output เช่น จากรูปตัวอย่าง จะได้ผลลัพธ์ Predict ว่ารูปภาพที่ป้อนเข้า Input Image เป็นเลขอะไร

2.5 การตรวจจับวัตถุ (Object Detection)

การตรวจจับวัตถุเป็นเทคโนโลยี Computer Vision ที่ทำการประมวลผลภาพเพื่อตรวจจับ (Detect) หรือค้นหาว่าในภาพ หรือเฟรมในวิดีโอนั้น มีวัตถุ (Object) อะไรบ้างและอยู่ในตำแหน่งใด ซึ่งวัตถุนั้นเป็นอะไรก็ได้ที่เราสนใจหรือกำลังพัฒนาเพื่อไปตรวจจับสิ่งใด โดยเมื่อพบแล้วจะนำไปตีกรอบ (Bounding box) รอบวัตถุนั้น วิธีการตรวจจับวัตถุ แบ่งออกเป็น 2 กลุ่มหลักๆ ดังนี้

2.5.1 Machine Learning

การตรวจจับวัตถุด้วย Machine Learning หรือ Non-neural Network approach เป็นวิธีที่ไม่ใช้ Neural Network แต่จะอาศัย Feature ของภาพเป็นหลัก เช่น Histogram ขอบ (Edges) หรือจุดเด่นต่างๆ วิธีการจำแนก (Classify) ใช้อัลกอริทึมต่างๆ ของ Machine Learning เช่น SVM k-NN ฯลฯ

2.5.2 Deep Learning

การตรวจจับวัตถุด้วย Deep Learning หรือวิธี Neural approach เป็นวิธี Neural Network ที่ไม่ต้องใช้ Feature ตรงๆ แต่จะใช้ CNN เป็นหลัก ซึ่งวิธีนี้ได้รับความนิยมมากในปัจจุบัน วิธีนี้ยังแบ่งออกเป็น 2 กลุ่ม คือ

1) วิธีการประมวลผล 2 ครั้ง (Two-stage detector) ประกอบด้วยตัวแกนหลักหรือ Backbone จะใช้ CNN ประมวลผล ทำหน้าที่คัดแยก Feature และคัดขอบเขตของส่วนที่สนใจ Region Proposal ออกมาก่อน ซึ่งก็คือส่วนที่คาดว่าจะเป็นวัตถุที่กำลังสนใจ สิ่งที่ได้คือขอบเขตที่สนใจ Proposal จากนั้นทำการจำแนกว่าเป็นวัตถุอะไร เช่นภาพประกอบ 10 (a) ภาพหรือวิดีโอถูกป้อนเข้าสู่ CNN จะได้ Proposal ที่มีขอบเขตตำแหน่ง Loc จากนั้นป้อนเข้าสู่ CNN เพื่อจำแนกว่าเป็นคลาสใดต่อไป เนื่องจากต้องทำการประมวลผล 2 ครั้ง ดังนั้น โดยรวมจะช้ากว่าวิธีประมวลผล 1 ครั้ง แต่จะมีข้อดีคือได้ความแม่นยำสูงกว่า ตัวอย่างวิธีนี้ได้แก่ Region Proposals R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN ฯลฯ

2) วิธีประมวลผล 1 ครั้ง จะใช้ CNN ประมวลผลเพียงครั้งเดียว ภาพประกอบ 10 (b) ผลลัพธ์ทั้งคลาส Cls และขอบเขต Loc ของวัตถุเลย โดยส่วนใหญ่จะตีกรอบรอบวัตถุนั้นๆ ไว้ด้วย วิธีนี้จะดีกว่าวิธีแรกคือ มีความเร็วสูงกว่า นิยมใช้กับการตรวจจับภาพเคลื่อนไหว เช่น กล้องจับภาพบนถนน ระบบตรวจจับหรือนับจำนวนคนในพื้นที่ ฯลฯ วิธีนี้ ได้แก่ Single Shot Detector (SSD), You Only Look Once (YOLO), Deconvolutional Single Shot Detector (DSSD), Retina Net, M2Det ฯลฯ

ภาพประกอบ 10 (a) การตรวจจับแบบประมวลผล 2 ครั้ง (b) การตรวจจับแบบประมวลผล 1 ครั้ง [18]

การตรวจจับด้วย Deep Learning เป็นวิธีสมัยใหม่ที่ได้รับความนิยมสูง แต่อย่างไรก็ตามการที่จะเลือกใช้วิธี Machine Learning หรือ Deep Learning นั้นมีปัจจัยที่เกี่ยวข้องคือ ลักษณะของงาน กับอุปกรณ์ที่ใช้ ดังที่ทราบแล้วว่า Deep Learning ใช้กำลังการประมวลผลสูงมากหากมีอุปกรณ์มีทรัพยากรการประมวลผลสูงพอ เช่น CPU หรือ GPU แรง ก็จะสามารถใช้วิธี Deep Learning ได้ แต่หากไม่แรงพอ เช่น Device ขนาดเล็ก IoT Mobile ฯลฯ ก็ต้องหันมาใช้ Machine Learning แทน

2.6 YOLO

You Only Look Once (YOLO) [18] เป็นอัลกอริทึมแบบประมวลผล 1 ครั้ง มีจุดเด่นตามชื่อ คือประมวลผลครั้งเดียวได้ผลลัพธ์เลย มีความเร็วและความแม่นยำสูง ทำให้เป็นที่นิยมมากในปัจจุบันสำหรับการตรวจจับวัตถุ โดยเฉพาะแบบ Real time

2.6.1 หลักการของ YOLO

หลักการทำงานของ YOLO คือแบ่งภาพออกเป็นตารางกริด (Grid) เช่น 3x3 4x4 จากนั้นแต่ละกริด จะถูกส่งเข้าประมวลผลเพื่อจำแนกว่าคลาส (Classification) และหาตำแหน่ง (Localization) ซึ่งค่า Label ที่ใช้ใน YOLO บางส่วนจะแตกต่างกับ CNN ของ Image

Classification ทั่วไป คือ Image Classification ทั่วไปจะมี Label คือคลาสของภาพ เช่น สุนัข แมว รถ.... ส่วน YOLO จะมี Label เป็นเวกเตอร์หรือเมทริกซ์ 1D เช่นสมมุติว่างานมี 3 คลาส (c) คือ รถ สุนัข แมว (c_1, c_2, c_3ตามลำดับ) เมทริกซ์ที่ได้จะเป็นดังสมการที่ 4

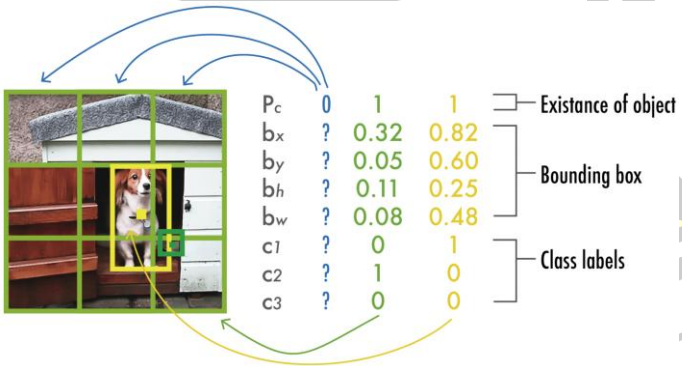
$$y_{r,c} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad (4)$$

เมื่อ r,c = row , column

p_c คือ ค่าที่บอกว่าวัตถุอยู่ในกริด (หน้าต่าง) ที่กำลังพิจารณาหรือไม่

b_x, b_y, b_h, b_w คือค่าที่บอกขอบเขต (Bounding box) ของวัตถุ (ถ้ามีวัตถุอยู่ในกริดที่กำลังพิจารณา) คือ x,y ความสูงและความกว้าง

c_1, c_2, c_3 คือค่าที่บอกว่าวัตถุอยู่ในคลาสใด เช่น ถ้าเป็น รถ ค่า c_1 จะเท่ากับ 1 ส่วน c_2, c_3 จะมีค่าเท่ากับ 0 ดังภาพประกอบ 11

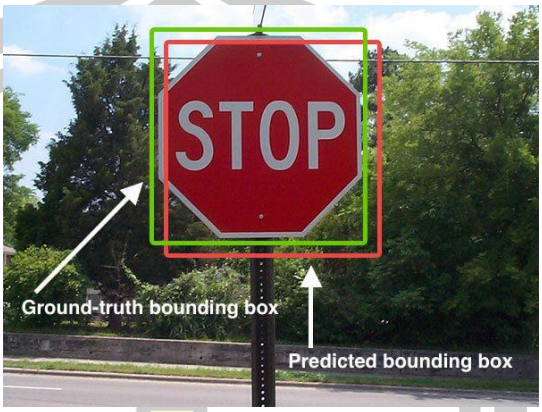


ภาพประกอบ 11 กริดแบ่งภาพและข้อมูลเมทริกซ์ของ Label[19]

2.6.2 Intersect Over Union

ผลลัพธ์ที่ได้จากการทำนายเบื้องต้นอาจได้กรอบขอบเขต (Boundary) จำนวนมาก เช่น ภาพประกอบที่ 12 ขอบเขตจริงของวัตถุคือเส้นสีเขียว ส่วนที่ระบบทำนายจะเป็นเส้นสีแดง ซึ่งจริง ๆ

แล้วขอบเขตที่ระบบทำนายให้อยู่รอบ ๆ วัตถุจำนวนมาก ดังนั้นการจะเลือกขอบเขตที่เหมาะสมที่สุดจะต้องมีการคำนวณคัดกรอง ดังภาพประกอบ 12



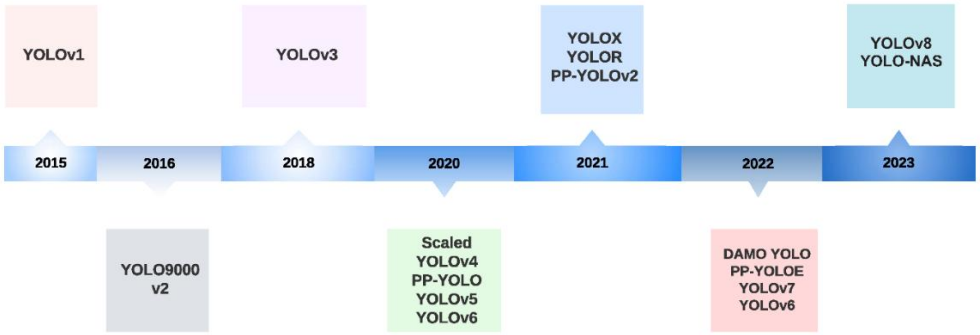
ภาพประกอบ 12 ขอบเขตที่ทำนายได้และขอบเขตจริงของวัตถุ[19]

Intersect Over Union (IoU) เป็นการคำนวนหาว่าขอบเขตที่ระบบทำนายหรือเลือกมานั้นมีความเหมาะสมแค่ไหน โดยวิธีคำนวณคือ นำค่าพื้นที่ที่เกยทับกัน (Intersection) หารด้วย พื้นที่ผลรวมแบบ Union ภาพประกอบ 13 โดยค่า IoU จะอยู่ที่ระหว่าง 0 – 1 หากผลการคำนวณได้ค่า IoU สูง แสดงว่าขอบเขตหรือพื้นที่ค่อนข้างทันกันมาก โดยเฉพาะถ้าได้ค่า IoU = 1 แสดงว่า 2 ขอบเขตนั้นมีพื้นที่เกยทับกัน 100% แต่ถ้าได้ค่าน้อย เช่น 0.1 แสดงว่ามีส่วนที่เกยทับกันน้อยมาก โดยปกติแล้วนิยมยึดค่า 0.5 ขึ้นไปเป็น Threshold ซึ่งถือว่าเป็นค่าที่เหมาะสมที่จะนำไปใช้

ภาพประกอบ 13 ตัวอย่าการคำนวณและการคำนวณ IoU[19]

2.6.3 You Only Look Once v1

YOLO โดย Joseph Redmon และคณะ ได้รับการเผยแพร่ในงาน CVPR 2016 เป็นครั้งแรกที่นำเสนอแนวทางการตรวจจับวัตถุแบบ end-to-end แบบเรียลไทม์ ชื่อ YOLO ย่อมาจาก “You Only Look Once” ซึ่งหมายถึงข้อเท็จจริง ที่ว่ามันสามารถบรรลุภารกิจการตรวจจับ ให้สำเร็จได้ ด้วยการส่งผ่านเครือข่ายเพียงครั้งเดียว ดังภาพประกอบ 14 ไทม์ไลน์ของเวอร์ชัน YOLO



ภาพประกอบ 14 ไทม์ไลน์ของเวอร์ชัน You Only Look Once [19]

ข้อดีและข้อเสียของ You Only Look Once v1

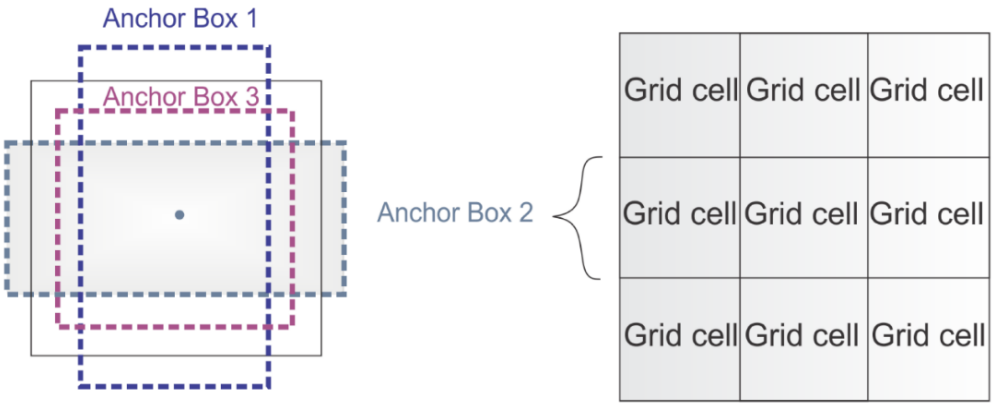
สถาปัตยกรรมที่เรียบง่ายของ You Only Look Once ควบคู่ไปกับการถดถอยด้วยการถ่ายภาพเดียวแบบเต็มภาพ ทำให้เร็วกว่าเครื่องตรวจจับวัตถุที่มีอยู่มาก ทำให้สามารถทำงานได้แบบเรียลไทม์แม้ว่า You Only Look Once จะดำเนินการได้เร็วกว่าเครื่องตรวจจับวัตถุใดๆ ก็ตาม แต่ข้อผิดพลาดในการระบุตำแหน่งนั้นมีมากกว่าเมื่อเทียบกับวิธีการที่ทันสมัย เช่น Fast R-CNN สาเหตุหลักคือ สามารถตรวจจับวัตถุประเภทเดียวกันได้สูงสุดสองชิ้นในเซลล์กริด ซึ่งจำกัดความสามารถในการทำนายวัตถุใกล้เคียง, มีปัญหาในการทำนายวัตถุที่มีอัตราส่วนภาพที่ไม่เห็นในข้อมูลการฝึกและเรียนรู้จากคุณสมบัติของวัตถุหยาบเนื่องจากการสุ่มตัวอย่าง เลเยอร์

2.6.4 You Only Look Once v2

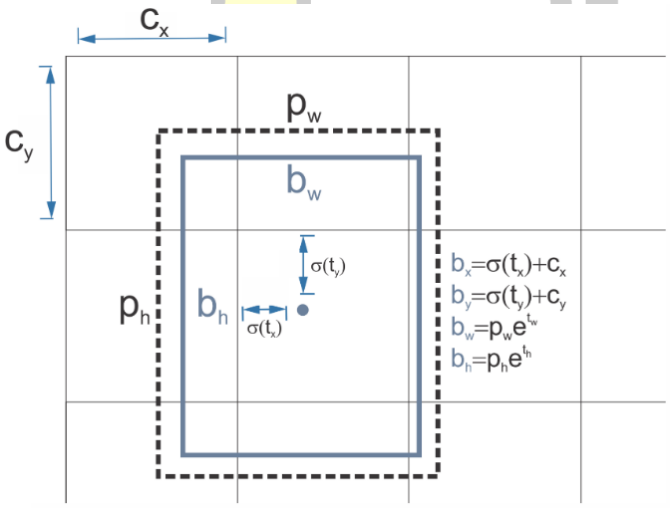
You Only Look Once 2 ได้รับการเผยแพร่ที่ CVPR 2017 โดย Joseph Redmon และ Ali Farhadi มันรวมการปรับปรุงหลายประการจาก You Only Look Once ดั้งเดิม เพื่อให้ดีขึ้น โดยรักษาความเร็วเท่าเดิม และยังแข็งแกร่งขึ้นอีกด้วย การปรับปรุงมีดังนี้

1. การทำให้เป็นมาตรฐานแบบแบทช์บนเลเยอร์การบิดทั้งหมดปรับปรุงการลู่เข้าและทำหน้าที่เป็นตัวทำให้สม่ำเสมอเพื่อลดการโอเวอร์ฟิต

2. ลักษณนามความละเอียดสูง เช่นเดียวกับ You Only Look Once v1 พวกเขาฝึกอบรมโมเดลล่วงหน้าด้วย ImageNet ที่ 224 × 224 อย่างไรก็ตาม ในครั้งนี้ พวกเขาได้ปรับแต่งโมเดลสำหรับสิบยุคบน ImageNet ด้วยความละเอียดที่ 448 × 448 ปรับปรุงประสิทธิภาพของเครือข่ายด้วยอินพุตความละเอียดสูงขึ้น
3. Fully convolutional เอาชั้นที่หนาแน่นออกและใช้สถาปัตยกรรมแบบบิดเต็มรูปแบบ
4. ใช้กล่องสมอเพื่อทำนายกล่องที่มีขอบเขต พวกเขาใช้ชุดกล่องก่อนหน้าหรือกล่องยึดซึ่งเป็นกล่องที่มีรูปร่างที่กำหนดไว้ล่วงหน้าซึ่งใช้จับคู่รูปร่างต้นแบบของวัตถุดังแสดงในภาพประกอบ 15 กล่องจุดยึดหลายกล่องถูกกำหนดไว้สำหรับแต่ละเซลล์ตาราง และระบบจะคาดการณ์พิกัดและคลาสสำหรับกล่องจุดยึดทุกกล่อง ขนาดของเอาต์พุตเครือข่ายเป็นสัดส่วนกับจำนวนกล่องจุดยึดต่อเซลล์กริด
5. คลัสเตอร์มิติ การเลือกกล่องก่อนหน้าที่ดีจะช่วยให้เครือข่ายเรียนรู้ที่จะทำนายกรอบขอบเขตที่แม่นยำยิ่งขึ้น ผู้เขียนจัดกลุ่มเคมีนบนกล่องขอบเขตการฝึกอบรมเพื่อค้นหานักบวชที่ดี พวกเขาเลือกกล่องก่อนหน้าห้ากล่อง ซึ่งให้การแลกเปลี่ยนที่ดีระหว่างการเรียกคืนและความซับซ้อนของโมเดล
6. การทำนายตำแหน่งโดยตรง ไม่เหมือนวิธีอื่นๆ ที่ทำนายออฟเซ็ต You Only Look Once v2 ปฏิบัติตามหลักปรัชญาเดียวกันและคาดการณ์พิกัดตำแหน่งที่สัมพันธ์กับเซลล์กริด เครือข่ายคาดการณ์กรอบขอบเขตห้ากล่องสำหรับแต่ละเซลล์ โดยแต่ละกล่องจะมีค่า 5 t_x , t_y , t_w , t_h และ t_0 ที่ t_0 เทียบเท่ากับ p_c จาก YOLOv1 และขอบเขตจะได้ดังแสดงในภาพประกอบ 16
7. คุณสมบัติที่ละเอียดยิ่งขึ้น You Only Look Once v2 เมื่อเปรียบเทียบกับ You Only Look Once v1 ได้ลบหนึ่งเลเยอร์ของการรวมกลุ่มออกเพื่อรับแผนผังคุณลักษณะเอาต์พุตหรือกริดของ 13 × 13 สำหรับใส่รูปภาพของ 416 × 416 You Only Look Once v2 ยังใช้เลเยอร์การส่งผ่านที่รับ26 × 26 × 512 แผนที่คุณลักษณะและจัดระเบียบใหม่โดยการซ้อนคุณลักษณะที่อยู่ติดกันลงในช่องต่างๆ แทนที่จะสูญเสียไปผ่านการสุ่มตัวอย่างเชิงพื้นที่ สิ่งนี้ทำให้เกิด13 × 13 × 2048 แผนที่คุณลักษณะที่ต่อกันในมิติช่องสัญญาณที่มีความละเอียดต่ำกว่า13 × 13 × 1024 แผนที่ที่จะได้รับ13 × 13 × 3072
8. การฝึกอบรมหลายระดับ เนื่องจาก YOLOv2 ไม่ได้ใช้เลเยอร์ที่เชื่อมต่อกันอย่างสมบูรณ์ อินพุตจึงสามารถมีขนาดแตกต่างกันได้ เพื่อให้ YOLOv2 ทนทานต่อขนาดอินพุตที่แตกต่างกัน ผู้เขียนได้ฝึกฝนโมเดลแบบสุ่ม โดยเปลี่ยนขนาดอินพุตจาก 320 × 320 จนถึง 608 × 608 ทุก ๆ สิบชุด



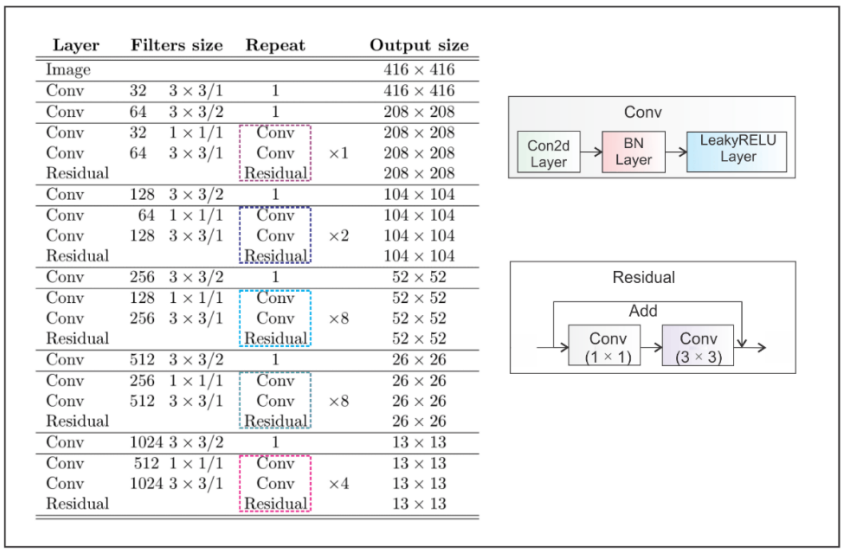
ภาพประกอบ 15 Anchor Box You Only Look Once v2 กำหนดกล่องจุดยึดหลายช่องสำหรับแต่ละเซลล์ตาราง [19]



ภาพประกอบ 16 การทำนายกรอบขอบเขตของ You Only Look Once v2 [19]

2.6.5 You Only Look Once v3

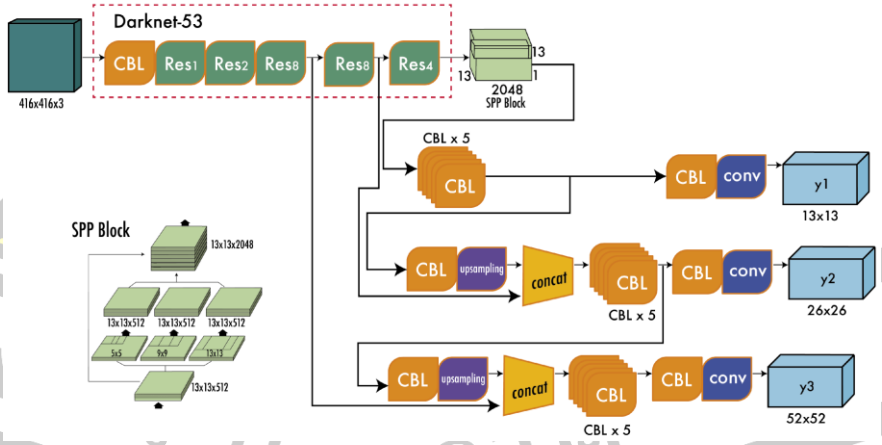
You Only Look Once v3ได้รับการเผยแพร่ใน ArXiv ในปี 2018 โดย Joseph Redmon และ Ali Farhadi รวมถึงการเปลี่ยนแปลงที่สำคัญและสถาปัตยกรรมที่ใหญ่ขึ้นเพื่อให้ทัดเทียมกับความทันสมัยในขณะที่ยังคงประสิทธิภาพแบบเรียลไทม์ สถาปัตยกรรม You Only Look Once v3 โครงสร้างหลักสถาปัตยกรรมที่นำเสนอใน You Only Look Once v3 เรียกว่า Darknet-53 มันแทนที่เลเยอร์การรวมกลุ่มสูงสุดทั้งหมดด้วยการบิดแบบเป็นเส้นและเพิ่มการเชื่อมต่อที่เหลือ โดยรวมแล้วมีชั้นบิดจำนวน 53 ชั้น ภาพประกอบ 17 แสดงรายละเอียดสถาปัตยกรรม



Layer	Filters	size	Repeat		Output size
Image					416 × 416
Conv	32	3 × 3/1	1		416 × 416
Conv	64	3 × 3/2	1		208 × 208
Conv	32	1 × 1/1	Conv		208 × 208
Conv	64	3 × 3/1	Conv	×1	208 × 208
Residual			Residual		208 × 208
Conv	128	3 × 3/2	1		104 × 104
Conv	64	1 × 1/1	Conv		104 × 104
Conv	128	3 × 3/1	Conv	×2	104 × 104
Residual			Residual		104 × 104
Conv	256	3 × 3/2	1		52 × 52
Conv	128	1 × 1/1	Conv		52 × 52
Conv	256	3 × 3/1	Conv	×8	52 × 52
Residual			Residual		52 × 52
Conv	512	3 × 3/2	1		26 × 26
Conv	256	1 × 1/1	Conv		26 × 26
Conv	512	3 × 3/1	Conv	×8	26 × 26
Residual			Residual		26 × 26
Conv	1024	3 × 3/2	1		13 × 13
Conv	512	1 × 1/1	Conv		13 × 13
Conv	1024	3 × 3/1	Conv	×4	13 × 13
Residual			Residual		13 × 13

ภาพประกอบ 17 You Only Look Once 3 Darknet-53 backbone [19]

นอกเหนือจากสถาปัตยกรรมที่ใหญ่ขึ้นแล้ว คุณสมบัติที่สำคัญของ You Only Look Once v3 ก็คือการคาดการณ์แบบหลายสเกล เช่น การคาดการณ์ที่ขนาดกริดหลายขนาด สิ่งนี้ช่วยให้ได้กล่องที่มีรายละเอียดปลีกย่อยและปรับปรุงการทำนายวัตถุขนาดเล็กได้อย่างมาก ซึ่งเป็นหนึ่งในจุดอ่อนหลักของ You Only Look Once เวอร์ชันก่อนหน้า

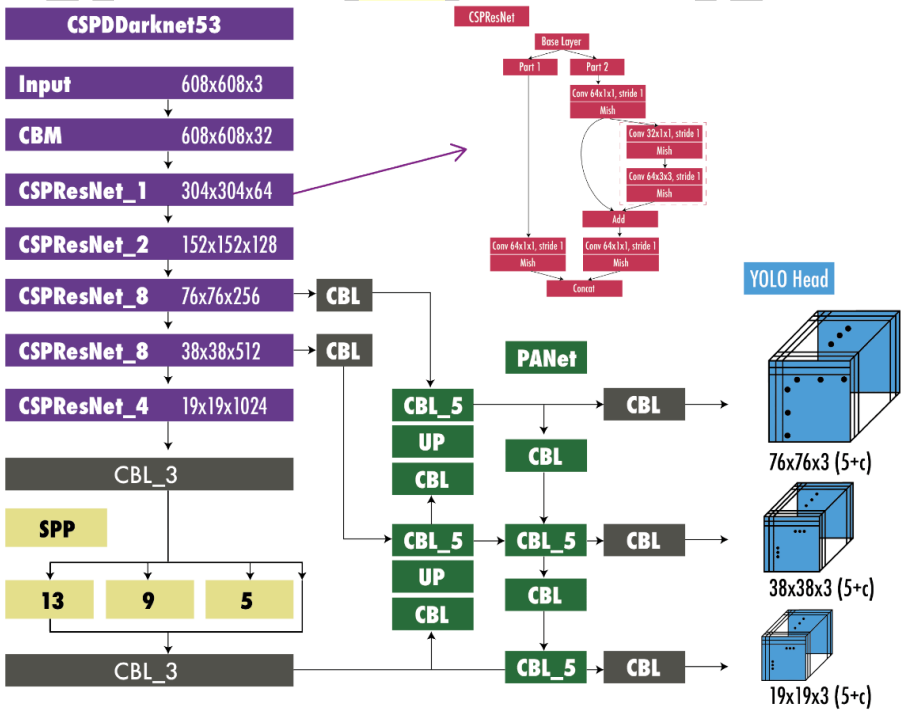


ภาพประกอบ 18 สถาปัตยกรรมการตรวจจับหลายสเกล You Only Look Once v3[19]

2.6.6 You Only Look Once v4

เดือนเมษายน 2020 Alexey Bochkovskiy, Chien-Yao Wang และ Hong-Yuan Mark Liao ได้เผยแพร่บทความสำหรับ You Only Look Once v4 ใน ArXiv ในตอนแรก รู้สึกแปลกที่ผู้เขียนหลายคนนำเสนอ You Only Look Once เวอร์ชัน "อย่างเป็นทางการ" ใหม่ อย่างไรก็ตาม You Only Look Once v4 ยังคงปรัชญา You Only Look Once เดิมไว้ ไม่ว่าจะเป็นแบบเรียลไทม์ โอเพ่นซอร์ส Single Shot และเฟรมเวิร์ก Darknet และการปรับปรุงก็เป็นที่น่าพอใจมาก จนชุมชนยอมรับเวอร์ชันนี้อย่างรวดเร็วในฐานะ You Only Look Once v4 อย่างเป็นทางการ

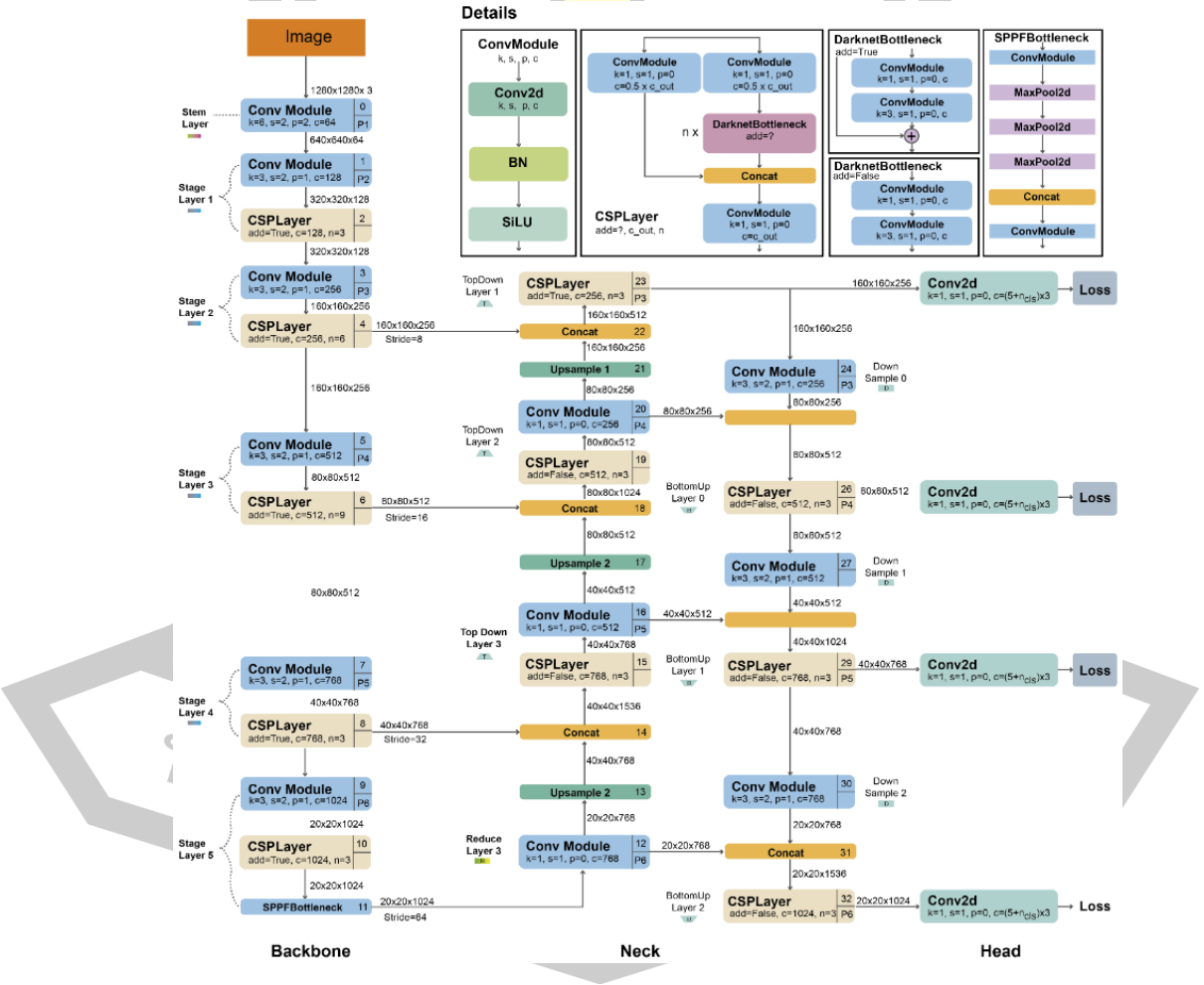
You Only Look Once v4 พยายามค้นหาจุดสมดุลที่เหมาะสมโดยการทดลองกับการเปลี่ยนแปลงหลายอย่างซึ่งจัดเป็นถุงของแจกฟรีและถุงของพิเศษ วิธีการแจกฟรีเพียงเปลี่ยนกลยุทธ์การฝึกอบรมและเพิ่มต้นทุนการฝึกอบรม แต่ไม่เพิ่มเวลาในการอนุมาน โดยทั่วไปการเพิ่มข้อมูลที่พบบ่อยที่สุด ในทางกลับกัน วิธีการแบบ Bag-of-Special จะเพิ่มต้นทุนการอนุมานเล็กน้อย แต่ปรับปรุงความแม่นยำได้อย่างมาก ตัวอย่างของวิธีการเหล่านี้ ได้แก่ การขยายฟิลด์รับสัญญาณ รวมคุณสมบัติ และขั้นตอนหลังการประมวลผล และอื่น ๆ อีกมากมาก



ภาพประกอบ 19 สถาปัตยกรรม You Only Look Once v4 สำหรับการตรวจจับวัตถุ[19]

2.6.7 You Only Look Once v5

You Only Look Once v5 เปิดตัวสองสามเดือนหลังจาก You Only Look Once v4 ในปี 2020 โดย Glen Jocher ผู้ก่อตั้งและ CEO ของ Ultralights ใช้การปรับปรุงหลายอย่างที่อธิบายไว้ในส่วน You Only Look Once v4 แต่พัฒนาใน Pytorch แทนที่จะเป็น Darknet You Only Look Once v5 รวมอัลกอริทึม Ultralights ที่เรียกว่า Auto Anchor เครื่องมือก่อนการฝึกนี้จะตรวจสอบและปรับ Anchor Box หากไม่เหมาะกับชุดข้อมูลและการตั้งค่าการฝึก เช่น ขนาดรูปภาพ ขั้นแรกจะใช้ฟังก์ชัน K-Means กับป้ายกำกับชุดข้อมูลเพื่อสร้างเงื่อนไขเริ่มต้นสำหรับอัลกอริทึมวิวัฒนาการทางพันธุกรรม (GE) จากนั้นอัลกอริทึม GE จะพัฒนาจุดยึดเหล่านี้มากกว่า 1,000 รุ่นโดยค่าเริ่มต้น โดยใช้การสูญเสีย CIOU และการเรียกคืนที่ดีที่สุดที่เป็นไปได้เป็นฟังก์ชันด้านฟิตเนส ดังภาพประกอบที่ 20 แสดงสถาปัตยกรรมโดยละเอียดของ You Only Look Once v5



ภาพประกอบ 20 สถาปัตยกรรม You Only Look Once v5[19]

สถาปัตยกรรม You Only Look Once v5

แกนหลักคือ CSPDarknet53 ที่ได้รับการดัดแปลงซึ่งเริ่มต้นด้วย Stem ซึ่งออกเป็นเลเยอร์ Convolution แบบลายเส้นที่มีขนาดหน้าต่างขนาดใหญ่ เพื่อลดต้นทุนหน่วยความจำและการคำนวณ ตามด้วยเลเยอร์ Convolutional ที่แยกคุณสมบัติที่เกี่ยวข้องจากรูปภาพอินพุต เลเยอร์ SPPF (การรวมกลุ่มพีระมิดเชิงพื้นที่อย่างรวดเร็ว) และเลเยอร์ Convolution ต่อไปนี้จะประมวลผลคุณลักษณะต่างๆ ในขนาดต่างๆ ในขณะที่ เลเยอร์ตัวอย่างจะเพิ่มความละเอียดของแผนผังคุณลักษณะ เลเยอร์ SPPF มีเป้าหมายเพื่อเพิ่มความเร็วในการคำนวณของเครือข่ายโดยการรวมคุณลักษณะของขนาดต่างๆ ไว้ในแผนผังคุณลักษณะที่มีขนาดคงที่ การบิดแต่ละครั้งจะตามด้วยการทำให้เป็นมาตรฐานแบบแบทช์ (BN) และการเปิดใช้งาน SiLU ส่วนคอใช้ SPPF และ CSP-PAN ที่ได้รับการแก้ไข ในขณะที่ส่วนหัวมีลักษณะคล้ายกับ You Only Look Once v3 ดังภาพประกอบ 20 สถาปัตยกรรม You Only Look Once v5

You Only Look Once v5 ใช้การเสริมหลายอย่าง เช่น โมเสก การคัดลอกวางสุ่ม affine MixUp การเสริม HSV การพลิกแนวนอนแบบสุ่ม เช่นเดียวกับการเสริมอื่น ๆ จากแพ็คเกจอัลบั้ม นอกจากนี้ยังปรับปรุงความไวของตารางเพื่อให้มีเสถียรภาพมากขึ้นในการไล่ระดับสีแบบ Runaway

You Only Look Once v5 มีเวอร์ชันปรับขนาดได้ห้าเวอร์ชัน Y You Only Look Once v5n (นาโน), You Only Look Once v5s (เล็ก), You Only Look Once v5m (กลาง), You Only Look Once v5l (ใหญ่) และ You Only Look Once v5x (ใหญ่พิเศษ) โดยที่ความกว้างและความลึกของโมดูล Convolution จะแตกต่างกันไปเพื่อให้เหมาะกับแอปพลิเคชันและฮาร์ดแวร์เฉพาะ ความต้องการ. ตัวอย่างเช่น You Only Look Once v5n และ You Only Look Once v5s เป็นรุ่นน้ำหนักเบาที่กำหนดเป้าหมายสำหรับอุปกรณ์ที่ใช้ทรัพยากรต่ำ ในขณะที่ You Only Look Once v5x ได้รับการปรับปรุงให้มีประสิทธิภาพสูง แม้ว่าจะแลกกับความเร็วก็ตาม You Only Look Once v5 เวอร์ชันที่เผยแพร่ในขณะที่เขียนบทความนี้คือเวอร์ชัน 7.0 รวมถึงเวอร์ชัน You Only Look Once v5 ที่สามารถจำแนกประเภทและแบ่งส่วนอินสแตนซ์ได้

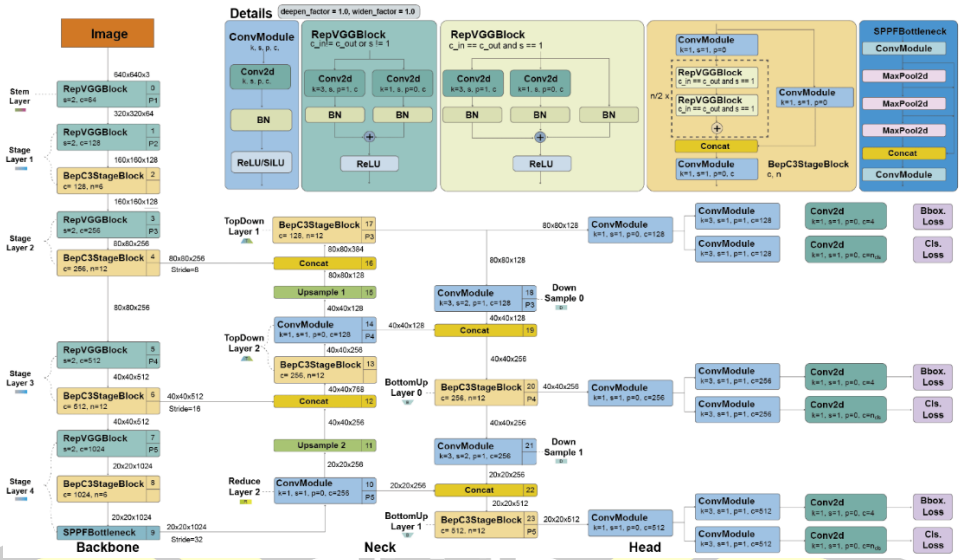
You Only Look Once v5 เป็นโอเพ่นซอร์สและได้รับการดูแลอย่างแข็งขันโดย Ultralytics โดยมีผู้มีส่วนร่วมมากกว่า 250 รายและมีการปรับปรุงใหม่อยู่บ่อยครั้ง You Only Look Once v5 ใช้งานง่าย ฝึกฝน และปรับใช้ Ultralytics นำเสนอเวอร์ชันมือถือสำหรับ iOS และ Android และการผสานรวมมากมายสำหรับการติดฉลาก การฝึกอบรม และการปรับใช้

จากการประเมินบนชุดข้อมูล MS COCO test-dev 2017 You Only Look Once v5x ได้รับ AP ที่ 50.7% ด้วยขนาดภาพ 640 พิกเซล เมื่อใช้ขนาดแบตช์ 32 จะสามารถบรรลุ

ความเร็ว 200 FPS บน NVIDIA V100 ด้วยการใช้ขนาดอินพุตที่ใหญ่กว่า 1536 พิกเซลและการทดสอบการเพิ่มเวลาทดสอบ (TTA) You Only Look Once v5 จึงมี AP อยู่ที่ 55.8%

2.6.8 You Only Look Once v6

You Only Look Once v6 ได้รับการเผยแพร่ใน ArXiv ในเดือนกันยายน 2022 โดยแผนก Meituan Vision AI การออกแบบเครือข่ายประกอบด้วยแบ็คโบนที่มีประสิทธิภาพด้วยบล็อก RepVGG หรือ CPSStackRep คอโทโพโลยี PAN และส่วนหัวแยกส่วนที่มีประสิทธิภาพพร้อมกลยุทธ์ช่องสัญญาณแบบไฮบริด นอกจากนี้ บทความนี้ยังแนะนำเทคนิคการหาปริมาณที่ได้รับการปรับปรุงโดยใช้การหาปริมาณหลังการฝึกและการกลั่นแบบ Channel-Wise ส่งผลให้เครื่องตรวจจับเร็วขึ้นและแม่นยำยิ่งขึ้น โดยรวมแล้ว YOLOv6 มีประสิทธิภาพเหนือกว่าโมเดลล้ำสมัยก่อนหน้านี้ เช่น You Only Look Once v5, YOLX และ PP-YOLOE ในเรื่องการวัดความแม่นยำและความเร็ว ดังภาพประกอบที่ 21 สถาปัตยกรรม You Only Look Once v6

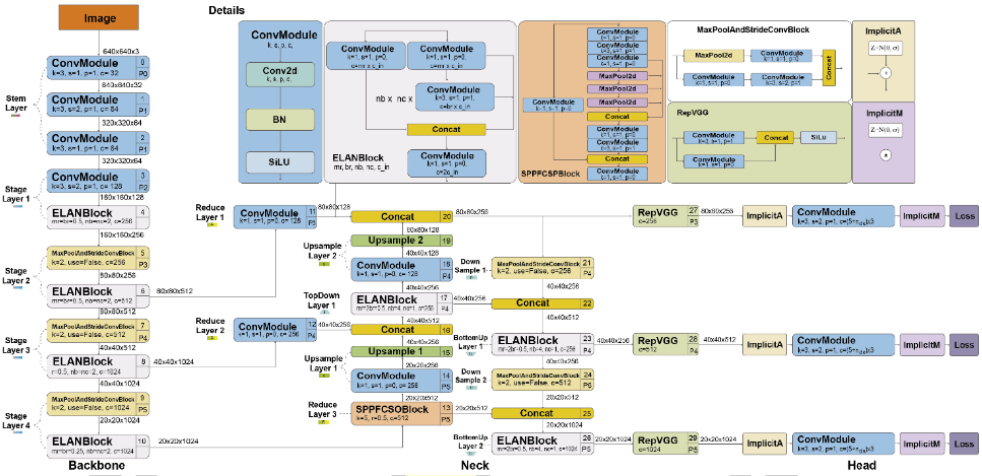


ภาพประกอบ 21 สถาปัตยกรรม You Only Look Once v6[19]

2.6.9 You Only Look Once v7

You Only Look Once v7 ได้รับการตีพิมพ์ในArXivในเดือนกรกฎาคม 2022 โดยผู้เขียน You Only Look Once v4 และ YOLOR คนเดียวกัน ในเวลานั้น มันแซงหน้าเครื่องตรวจจับวัตถุที่รู้จักทั้งหมดในด้านความเร็วและความแม่นยำในช่วง 5 FPS ถึง 160 FPS เช่นเดียวกับ You Only Look Once v4 มันถูกฝึกฝนโดยใช้เฉพาะชุดข้อมูล MS COCO โดยไม่มีแบ็คโบนที่ได้รับการฝึกล่วงหน้า You Only Look Once v7 เสนอการเปลี่ยนแปลงสถาปัตยกรรมสองสามรายการและ

ชุดวิธีการแจกฟรี ซึ่งเพิ่มความแม่นยำโดยไม่ส่งผลกระทบต่อความเร็วในการอนุมาน โดยส่งผลต่อเวลาการฝึกอบรมเท่านั้น



ภาพประกอบ 22 สถาปัตยกรรม You Only Look Once v7[19]

การเปลี่ยนแปลงสถาปัตยกรรมของ You Only Look Once v7 คือ ขยายเครือข่ายการรวมเลเยอร์ที่มีประสิทธิภาพ (E-ELAN) ELAN เป็นกลยุทธ์ที่ช่วยให้โมเดลเชิงลึกเรียนรู้และมาบรรจบกันได้อย่างมีประสิทธิภาพมากขึ้นโดยการควบคุมเส้นทางไล่ระดับที่สั้นที่สุดที่ยาวที่สุด You Only Look Once v7 เสนอ E-ELAN ที่เหมาะกับโมเดลที่มีบล็อกการคำนวณแบบสแต็กไม่จำกัด E-ELAN รวมคุณสมบัติของกลุ่มต่างๆ โดยการสับเปลี่ยนและผสานคาร์ดินาลิตี้เพื่อเพิ่มประสิทธิภาพการเรียนรู้ของเครือข่ายโดยไม่ทำลายเส้นทางการไล่ระดับสีดั้งเดิม การปรับขนาดโมเดลสำหรับโมเดลที่อิงการต่อข้อมูล ภาพประกอบ 22 สถาปัตยกรรม You Only Look Once v7

การปรับขนาดจะสร้างโมเดลที่มีขนาดแตกต่างกันโดยการปรับแอตทริบิวต์โมเดลบางอย่าง สถาปัตยกรรมของ You Only Look Once v7 เป็นสถาปัตยกรรมที่อิงการต่อข้อมูลซึ่งเทคนิคการปรับขนาดมาตรฐาน เช่น การปรับขนาดความลึก ทำให้เกิดการเปลี่ยนแปลงอัตราส่วนระหว่างช่องสัญญาณเข้าและช่องสัญญาณออกของเลเยอร์การเปลี่ยนแปลง ซึ่งในทางกลับกัน ส่งผลให้ฮาร์ดแวร์ลดลง การใช้งานโมเดล You Only Look Once v7 เสนอกลยุทธ์ใหม่สำหรับการปรับขนาดแบบจำลองตามการต่อข้อมูล โดยความลึกและความกว้างของบล็อกจะถูกปรับขนาดด้วยปัจจัยเดียวกันเพื่อรักษาโครงสร้างที่เหมาะสมที่สุดของแบบจำลอง

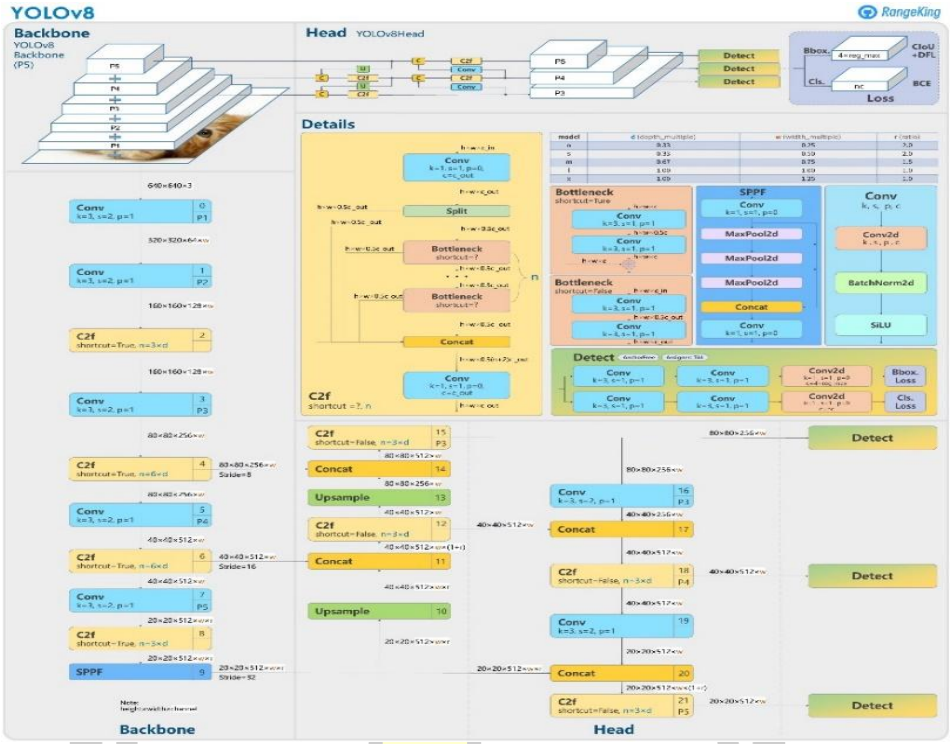
2.6.10 You Only Look Once v8

You Only Look Once v8 เปิดตัวในเดือนมกราคม พ.ศ. 2566 โดย Ultralytics บริษัทที่พัฒนา You Only Look Once v5 และ You Only Look Once v8 มีเวอร์ชันปรับขนาดได้ 5 เวอร์ชัน YOLOv8n (นาโน), YOLOv8s (เล็ก), YOLOv8m (กลาง), YOLOv8l (ใหญ่) และ YOLOv8x (ใหญ่พิเศษ) YOLOv8 รองรับงานการมองเห็นหลายอย่าง เช่น การตรวจจับวัตถุ การแบ่งส่วน การประมาณท่าทาง การติดตาม และการจัดหมวดหมู่

You Only Look Once v8 ใช้แกนหลักที่คล้ายกันกับ You Only Look Once v5 โดยมีการเปลี่ยนแปลงบางอย่างใน CSPLayer ซึ่งปัจจุบันเรียกว่าโมดูล C2f โมดูล C2f (ปัญหาคอขวดบางส่วนข้ามขั้นตอนที่มีการบิดงอสองครั้ง) ผสมผสานคุณสมบัติระดับสูงเข้ากับข้อมูลเชิงบริบทเพื่อปรับปรุงความแม่นยำในการตรวจจับ ดังภาพประกอบ 23 สถาปัตยกรรม You Only Look Once v8

You Only Look Once v8 ใช้โมเดลที่ไม่มีจุดยึดซึ่งมีส่วนหัวที่แยกออกจากกันเพื่อประมวลผลงานที่เป็นวัตถุ การจัดหมวดหมู่ และการถดถอยอย่างเป็นอิสระ การออกแบบนี้ช่วยให้แต่ละสาขามุ่งเน้นไปที่งานของตนและปรับปรุงความแม่นยำโดยรวมของแบบจำลอง ในเลเยอร์เอาต์พุตของ You Only Look Once v8 พวกเขาใช้ฟังก์ชัน sigmoid เป็นฟังก์ชันการเปิดใช้งานสำหรับคะแนนความเป็นวัตถุ ซึ่งแสดงถึงความน่าจะเป็นที่กล่องขอบเขตจะมีวัตถุอยู่ ใช้ฟังก์ชัน softmax สำหรับความน่าจะเป็นของคลาส ซึ่งแสดงถึงความน่าจะเป็นของอ็อบเจ็กต์ที่เป็นของแต่ละคลาสที่เป็นไปได้

You Only Look Once v8 ใช้ฟังก์ชันการสูญเสีย CIOU และ DFL สำหรับการสูญเสียขอบเขตและเอนโทรปีแบบไบนารีสำหรับการสูญเสียการจำแนกประเภท การสูญเสียเหล่านี้ทำให้ประสิทธิภาพการตรวจจับวัตถุดีขึ้น โดยหลักแล้วเมื่อต้องจัดการกับวัตถุขนาดเล็ก



ภาพประกอบ 23 สถาปัตยกรรม You Only Look Once v8[19]

You Only Look Once v8 ยังมีโมเดลการแบ่งส่วนเชิงความหมายที่เรียกว่าโมเดล You Only Look Once v8-Seg แกนหลักเป็นตัวแยกคุณสมบัติ CSPDarknet53 ตามด้วยโมดูล C2f แทนที่จะเป็นสถาปัตยกรรมคอ YOLO แบบดั้งเดิม โมดูล C2f ตามด้วยส่วนหัวของการแบ่งส่วนสองส่วน ซึ่งเรียนรู้ที่จะทำนายมาการแบ่งส่วนความหมายสำหรับรูปภาพอินพุต โมเดลนี้มีหัวตรวจจับที่คล้ายกันกับ You Only Look Once v8 ซึ่งประกอบด้วยโมดูลการตรวจจับห้าโมดูลและเลเยอร์การคาดการณ์ โมเดล You Only Look Once v8-Seg ได้รับผลลัพธ์ที่ล้ำสมัยในการตรวจจับวัตถุและเกณฑ์มาตรฐานการแบ่งส่วนความหมายต่าง ๆ ขณะเดียวกันก็รักษาความเร็วและประสิทธิภาพสูงเอาไว้

You Only Look Once v8 สามารถเรียกใช้จากอินเทอร์เฟซบรรทัดคำสั่ง (CLI) หรือสามารถติดตั้งเป็นแพ็คเกจ PIP ก็ได้ นอกจากนี้ยังมาพร้อมกับการผสานรวมที่หลากหลายสำหรับการติดป้ายกำกับ การฝึกอบรมและการปรับใช้ จากการประเมินบนชุดข้อมูล MS COCO test-dev 2017 You Only Look Once v8x ได้รับ AP 53.9% ด้วยขนาดภาพ 640 พิกเซล (เทียบกับ 50.7% ของ YOLOv5 ในขนาดอินพุตเดียวกัน) ด้วยความเร็ว 280 FPS บน NVIDIA A100 และ TensorRT

ตาราง 1 สรุปสถาปัตยกรรม You Only Look Once [16]

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	78.6
YOLOv3	2018	Yes	Darknet	Darknet53	33.0
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	YOLOv5CSPDarknet	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.2
Scaled-YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101 -vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	YOLOXCSPDarknet	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	YOLOv7Backbone	56.8
DAMO-YOLO	2022	NO	Pytorch	MAE-NAS	50.0
YOLOv8	2023	No	Pytorch	YOLOv8CSPDarknet	53.9
YOLO-NAS	2023	No	Pytorch	NAS	52.2

Anchors คือ โมเดล You Only Look Once ดั้งเดิมนั้นค่อนข้างเรียบง่ายและไม่ได้ใช้จุดยึด ในขณะที่ความทันสมัยอาศัยเครื่องตรวจจับแบบสองขั้นตอนพร้อมจุดยึด You Only Look Once v2 รวมจุดยึด ซึ่งนำไปสู่การปรับปรุงความแม่นยำในการทำนายขอบเขต แนวโน้มนี้ยังคงมีอยู่เป็นเวลาห้าปีจนกระทั่ง YOLX นำเสนอแนวทางที่ไร้จุดยึดซึ่งบรรลุผลลัพธ์ที่ล้ำสมัย ตั้งแต่นั้นมา You Only Look Once เวอร์ชันต่อ ๆ ไปก็ได้ยกเลิกการใช้จุดยึด

Framework คือ ในขั้นต้น You Only Look Once ได้รับการพัฒนาโดยใช้กรอบงาน Darknet โดยมีเวอร์ชันต่อมาตามมา อย่างไรก็ตาม เมื่อ Ultralytics ย้าย You Only Look Once v3 ไปยัง PyTorch แล้ว You Only Look Once เวอร์ชันที่เหลือก็ได้รับการพัฒนาโดยใช้ PyTorch ซึ่งนำไปสู่การปรับปรุงที่เพิ่มขึ้นอย่างรวดเร็ว ภาษาการเรียนรู้เชิงลึกอีกภาษาหนึ่งที่ใช้คือ PaddlePaddle ซึ่งเป็นเฟรมเวิร์กโอเพ่นซอร์สที่เริ่มพัฒนาโดย Baidu

Backbone คือ สถาปัตยกรรมแบ็คโบนของโมเดล You Only Look Once มีการเปลี่ยนแปลงที่สำคัญเมื่อเวลาผ่านไป เริ่มต้นด้วยสถาปัตยกรรม Darknet ซึ่งประกอบด้วยเลเยอร์ Convolutional แบบธรรมดาและการรวมเลเยอร์สูงสุด รุ่นต่อมาได้รวมการเชื่อมต่อบางส่วนแบบข้ามขั้นตอน (CSP) ใน You Only Look Once v4 การปรับพารามิเตอร์ใหม่ใน You Only Look Once v6 และ You Only Look Once v7 และการค้นหาสถาปัตยกรรมประสาทใน DAMO- You Only Look Once และ You Only Look Once -NAS

Performance คือ แม้ว่าประสิทธิภาพของโมเดล You Only Look Once จะได้รับการปรับปรุงเมื่อเวลาผ่านไป แต่ก็น่าสังเกตว่าโมเดลเหล่านี้มักจะให้ความสำคัญกับความเร็วและความแม่นยำที่สมดุล มากกว่ามุ่งเน้นไปที่ความแม่นยำเพียงอย่างเดียว การแลกเปลี่ยนนี้มีความสำคัญต่อกรอบงาน You Only Look Once ช่วยให้สามารถตรวจจับวัตถุแบบเรียลไทม์ในแอปพลิเคชันต่างๆ

2.7 การเปรียบเทียบความเร็วและความแม่นยำ You Only Look Once

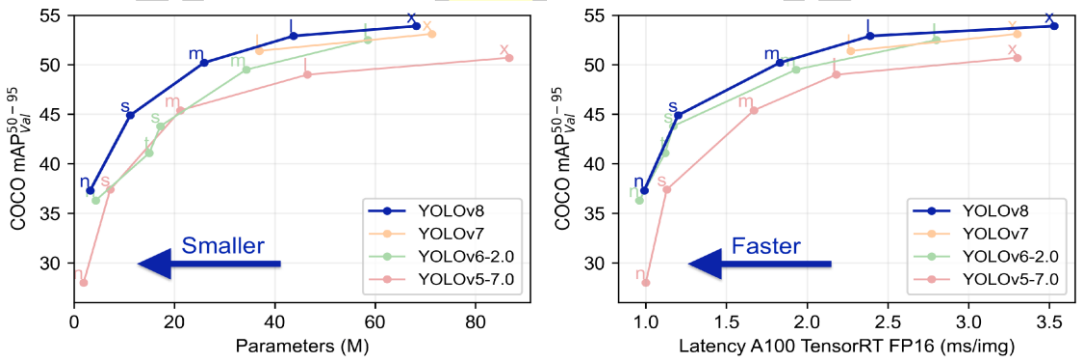
โมเดลการตรวจจับวัตถุในตระกูล You Only Look Once มุ่งเน้นไปที่การรักษาสมดุลความเร็วและความแม่นยำอย่างต่อเนื่อง โดยมีเป้าหมายเพื่อมอบประสิทธิภาพแบบเรียลไทม์โดยไม่กระทบต่อคุณภาพของผลลัพธ์การตรวจจับ เนื่องจากกรอบงาน You Only Look Once มีการพัฒนาผ่านการทำซ้ำต่าง ๆ การแลกเปลี่ยนนี้จึงเป็นธีมที่เกิดซ้ำ โดยแต่ละเวอร์ชัน พยายามปรับวัตถุประสงค์การแข่งขันให้เหมาะสมแตกต่างกัน ในโมเดล You Only Look Once ดั้งเดิม จุดสนใจหลักอยู่ที่การตรวจจับวัตถุความเร็วสูง แบบจำลองนี้ใช้โครงข่ายประสาทเทียมแบบหมุนเดี่ยว (CNN) เพื่อคาดการณ์ตำแหน่งของวัตถุและคลาสได้โดยตรงจากรูปภาพอินพุต ทำให้สามารถประมวลผลแบบเรียลไทม์ได้ อย่างไรก็ตาม การเน้นที่ความเร็วนี้นำไปสู่การประนีประนอมความแม่นยำ โดยส่วนใหญ่เมื่อต้องจัดการกับวัตถุขนาดเล็กหรือวัตถุที่มีกรอบขอบเขตที่ทับซ้อนกัน

You Only Look Once เวอร์ชันต่อมาได้นำเสนอการปรับปรุงและการปรับปรุงเพื่อแก้ไขข้อจำกัดเหล่านี้ ในขณะที่ยังคงรักษาความสามารถแบบเรียลไทม์ของเฟรมเวิร์กไว้ ตัวอย่างเช่น You Only Look Once v2 (YOLO9000) ได้แนะนำกล่องพุกและเลเยอร์พาสทรูเพื่อปรับปรุงการแปลวัตถุ ส่งผลให้มีความแม่นยำสูงขึ้น นอกจากนี้ You Only Look Once 3 ยังปรับปรุงประสิทธิภาพของโมเดลด้วยการใช้สถาปัตยกรรมการแยกคุณสมบัติหลายสเกล ซึ่งช่วยให้การตรวจจับวัตถุดีขึ้นในสเกลต่าง ๆ

การแลกเปลี่ยนระหว่างความเร็วและความแม่นยำนั้นเหมาะสมยิ่งขึ้นเมื่อกรอบงาน YOLO พัฒนาขึ้น โมเดลอย่าง You Only Look Once v4 และ You Only Look Once v5 นำเสนอนวัตกรรม เช่น แบ็คโบนเครือข่ายใหม่ เทคนิคการเพิ่มข้อมูลที่ได้รับการปรับปรุง และกลยุทธ์การ

ฝึกอบรมที่ปรับให้เหมาะสม การพัฒนาเหล่านี้นำไปสู่การเพิ่มความแม่นยำอย่างมาก โดยไม่ส่งผลกระทบอย่างมากต่อประสิทธิภาพแบบเรียลไทม์ของโมเดล

จาก You Only Look Once v5 รุ่น YOLO อย่างเป็นทางการทั้งหมดได้ปรับแต่งข้อดีระหว่างความเร็วและความแม่นยำ โดยเสนอขนาดรุ่นที่แตกต่างกันเพื่อให้เหมาะกับการใช้งานเฉพาะและข้อกำหนดด้านฮาร์ดแวร์ ตัวอย่างเช่น เวอร์ชันเหล่านี้มักมีโมเดลน้ำหนักเบาที่ได้รับการปรับให้เหมาะกับอุปกรณ์ Edge มีความแม่นยำในการแลกเปลี่ยนเพื่อลดความซับซ้อนในการคำนวณและเวลาประมวลผลที่เร็วขึ้น ภาพประกอบที่ 24 แสดงการเปรียบเทียบขนาดโมเดลที่แตกต่างกันตั้งแต่ You Only Look Once v5 ถึง You Only Look Once v8 รูปภาพนี้นำเสนอการวิเคราะห์เปรียบเทียบโมเดล YOLO เวอร์ชันต่างๆ ในแง่ของความซับซ้อนและประสิทธิภาพ กราฟด้านซ้ายแสดงจำนวนพารามิเตอร์ (เป็นล้าน) เทียบกับความแม่นยำเฉลี่ยเฉลี่ย (mAP) ในชุดการตรวจสอบความถูกต้องของ COCO ตั้งแต่เกณฑ์ IOU ที่ 50 ถึง 95 ซึ่งแสดงให้เห็นแนวโน้มที่ชัดเจนซึ่งการเพิ่มขึ้นของจำนวนพารามิเตอร์จะช่วยเพิ่ม ความแม่นยำของโมเดล แต่ละรุ่นมีสเกลต่างๆ ที่ระบุด้วย n (นาโน), s (เล็ก), m (กลาง), l (ใหญ่) และx (ใหญ่พิเศษ)



ภาพประกอบ 24 การเปรียบเทียบประสิทธิภาพของโมเดลการตรวจจับวัตถุ You Only Look Once [20]

กราฟด้านขวาจะเปรียบเทียบความหน่วงของการอนุมานบน NVIDIA A100 GPU โดยใช้ TensorRT FP16 โดยมีเมตริกประสิทธิภาพ mAP เดียวกัน จะเห็นความแตกต่างระหว่างความเร็วในการอนุมานและความแม่นยำในการตรวจจับ ค่าเวลาแฝงที่ต่ำกว่า ซึ่งบ่งชี้ถึงการอนุมานโมเดลที่เร็วขึ้น โดยทั่วไปส่งผลให้ความแม่นยำลดลง ในทางกลับกัน โมเดลที่มีความหน่วงสูงกว่ามักจะได้รับประสิทธิภาพที่ดีขึ้นบนตัววัด COCO mAP ความสัมพันธ์นี้ถือเป็นส่วนสำคัญสำหรับแอปพลิเคชันที่การประมวลผลแบบเรียลไทม์เป็นสิ่งสำคัญ และการเลือกรุ่นจะได้รับอิทธิพลจากข้อกำหนดในการสร้างสมดุลความเร็วและความแม่นยำ

2.8 PyThorch

Pytorch เป็น AI & Machine Learning Framework[21] ที่ถูกพัฒนาโดยทีม Facebook's AI Research Lab ในปี 2016 ซึ่งเป็น open-source library เปิดให้ใช้งานฟรี โดยภาษาหลักที่นิยมนำไปใช้กันคือ Python ปัจจุบัน Pytorch ได้รับความนิยมอย่างมากในการนำไปสร้างโมเดล Neural Network (Deep Learning) เนื่องจากมีลักษณะที่ค่อนข้างพร้อมต่อการใช้งานทันที จึงใช้งานได้ง่ายกว่า Tensorflow ที่เป็น Framework ประเภทเดียวกัน แต่ก็ยังสามารถปรับแต่งอะไรเพิ่มเติมได้มากกว่า Keras และ Pytorch สามารถเขียนเพื่อให้ใช้ GPU ในการคำนวณได้ ซึ่ง GPU มีจำนวน cores มากกว่า CPU (บางรุ่นอาจมีหลายพัน cores) และถูกออกแบบมาเพื่อจัดการกับงานที่มีการคำนวณในลักษณะเดียวกันซ้ำๆ (parallel processing) เหมาะกับการประมวลผลข้อมูลที่ต้องทำซ้ำจำนวนมาก เช่น การคำนวณใน neural networks สามารถประมวลผลข้อมูลในปริมาณมากได้พร้อมกัน ทำให้สามารถเทรนโมเดลได้เร็วกว่า CPU อย่างมาก โดยเฉพาะในงานที่เกี่ยวข้องกับ deep learning และ large-scale models

2.9 การเชื่อมต่อข้อมูล

การเชื่อมต่อข้อมูลที่ใช้ร่วมกับ โมเดล YOLO บน NVIDIA Jetson Nano สามารถทำได้โดยการดึงวิดีโอจากกล้องผ่านโปรโตคอล RTSP(Real-Time Streaming Protocol) หรือ HTTP URL แล้วนำมาใช้ประมวลผลด้วย OpenCV และ YOLO โดยมีความแตกต่างในการออกแบบและวัตถุประสงค์ดังนี้

1) RTSP (Real-Time Streaming Protocol), เล่นต่อ (Play) และหยุด (Stop) การส่งข้อมูล นิยมทำงานร่วมกับโปรโตคอล RTP (Real-Time Transport Protocol) สำหรับส่งข้อมูลที่มีความหน่วงต่ำและเสถียร
2) HTTP (Hyper Text Transfer Protocol) เหมาะสำหรับการถ่ายโอนข้อมูลบนเว็บ เช่น ไฟล์ HTML, ภาพ, วิดีโอ หรือเอกสาร โดยถูกออกแบบมาให้เป็นโปรโตคอลที่ใช้บนอินเตอร์เน็ต เพื่อการเรียกดูผ่านเว็บและการดาวโหลดข้อมูล การส่งข้อมูลเป็นแบบขอแล้วตอบ (Request-Response) โดยผู้ใช้ต้องร้องขอข้อมูลจากเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะส่งข้อมูลกลับมา

โดยทั่วไปแล้ว RTSP จะถูกใช้งานในระบบกล้องวงจรปิด (CCTV) และระบบเฝ้าระวัง ในขณะที่ HTTP ถูกใช้งานในแอปพลิเคชันบนเว็บและการส่งข้อมูลผ่านอินเทอร์เน็ตเป็นหลัก

2.10 งานวิจัยที่เกี่ยวข้อง

ในหัวข้อนี้จะเป็นการสรุปข้อมูลที่ได้ศึกษามาจากงานวิจัยต่างๆ ที่เกี่ยวข้อง และเป็นประโยชน์ต่อการดำเนินงานวิจัย

F. M. Talaat and H. ZainEldin (2023) [22] ได้กล่าวถึง วิธีการตรวจจับไฟในเมืองฉลองที่พัฒนาจากอัลกอริทึม YOLOv8 เรียกว่าระบบตรวจจับไฟสมาร์ท (SFDS) เพื่อลดความเสียหายจากไฟโดยการตรวจจับลักษณะที่เกี่ยวข้องกับไฟในเวลาที่เป็นจริง มีศักยภาพที่จะเพิ่มความแม่นยำ, ลดการเตือนเทียบเท่า, และเป็นทางเลือกที่มีความคุ้มค่าทางตรงเมื่อเปรียบเทียบกับวิธีการ ระบบ SFDS นี้ใช้ Fog และ Cloud computing, ร่วมกับ IoT layer, เพื่อรวบรวมและประมวลผลข้อมูลในเวลาที่เป็นจริง, เพื่อให้มีการตอบสนองที่รวดเร็วและลดความเสี่ยงของความเสียหายต่อทรัพย์สินและชีวิตมนุษย์ ระบบ SFDS ได้รับประสิทธิภาพสูงทั้งในเรื่องของความแม่นยำและ recall, ด้วยอัตราความแม่นยำสูงถึง 97.1% สำหรับทุกคลาส นอกจากนี้, มีความเป็นไปได้ที่จะขยายไปยังการตรวจจับวัตถุที่น่าสนใจอื่น ๆ ในเมืองฉลอง รวมถึงการจัดการความปลอดภัยจากไฟในพื้นที่สาธารณะ, การตรวจสอบไฟป่า, และระบบรักษาความปลอดภัยอัจฉริยะ.

Terven, Juan, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González.(2023) "การทบทวนสถาปัตยกรรม YOLO อย่างครอบคลุมในคอมพิวเตอร์วิทัศน์: จาก YOLOv1 ถึง YOLOv8 และ YOLO-NAS"[19] ได้กล่าวถึง YOLO ที่ได้รับการพัฒนาตั้งแต่เริ่มก่อตั้งโดยพัฒนาไปสู่ระบบตรวจจับวัตถุแบบเรียลไทม์ที่ซับซ้อนและมีประสิทธิภาพ ความก้าวหน้าตั้งแต่ YOLO ไปจนถึง YOLOv8 รวมถึง YOLO-NAS และ YOLO with transformers ได้แสดงให้เห็นถึงขอบเขตใหม่ในการตรวจจับวัตถุ และแสดงให้เห็นว่า YOLO ยังคงเป็นสาขาการวิจัยที่สำคัญ การผสมผสานระหว่างการปรับปรุงสถาปัตยกรรม เทคนิคการฝึกอบรม และการเพิ่มชุดข้อมูล ได้ขับเคลื่อนการปรับปรุงประสิทธิภาพของตระกูล YOLO นอกจากนี้ แนวทางการเรียนรู้แบบถ่ายโอนยังเป็นปัจจัยสำคัญต่อความสำเร็จของ YOLO ทำให้สามารถปรับกรอบงานให้เข้ากับงานตรวจจับวัตถุต่างๆ ได้

Licheng, Jiao., Fan, Zhang., Fang, Liu., Shuyuan, Yang., Lingling, Li., Zhixi, Feng., Rong, Qu. (2019) [23] ได้กล่าวถึง การตรวจจับวัตถุเป็นหนึ่งในสาขาที่สำคัญและท้าทายที่สุดของคอมพิวเตอร์วิทัศน์ ซึ่งถูกนำไปใช้อย่างกว้างขวางในชีวิตของผู้คน เช่น การตรวจสอบความปลอดภัย การขับขี่อัตโนมัติ เป็นต้น โดยมีวัตถุประสงค์ในการค้นหาอินสแตนซ์ของวัตถุเชิงความหมายของคลาสหนึ่ง ด้วยการพัฒนาอย่างรวดเร็วของอัลกอริทึมการเรียนรู้เชิงลึกสำหรับงานการตรวจจับประสิทธิภาพของเครื่องตรวจจับวัตถุได้รับการปรับปรุงอย่างมาก เพื่อให้เข้าใจสถานะการพัฒนาหลักของไปป์ไลน์การตรวจจับวัตถุอย่างละเอียดและลึกซึ้ง ในแบบสำรวจนี้ เราจะวิเคราะห์วิธีการของ

โมเดลการตรวจจับทั่วไปที่มีอยู่ และอธิบายชุดข้อมูลเบนช์มาร์กในตอนแรก หลังจากนั้นและในเบื้องต้น นำเสนอภาพรวมที่ครอบคลุมเกี่ยวกับวิธีการตรวจจับวัตถุต่างๆ อย่างเป็นระบบ ครอบคลุมเครื่องตรวจจับแบบขั้นตอนเดียวและสองขั้นตอน นอกจากนี้เรายังแสดงรายการแอปพลิเคชันแบบดั้งเดิมและแอปพลิเคชันใหม่ มีการวิเคราะห์การตรวจจับวัตถุบางสาขาที่เป็นตัวแทนด้วยเช่นกัน ในที่สุด หารือเกี่ยวกับสถาปัตยกรรมของการใช้ประโยชน์จากวิธีการตรวจจับวัตถุเหล่านี้เพื่อสร้างระบบที่มีประสิทธิภาพและชี้ให้เห็นชุดแนวโน้มการพัฒนาเพื่อให้เป็นไปตามอัลกอริทึมที่ล้ำสมัยและการวิจัยเพิ่มเติมได้ดียิ่งขึ้น

S. Kanmani. (2023) [24]ได้กล่าวถึง อัลกอริทึมการตรวจจับวัตถุที่มี YOLO เวอร์ชันต่างๆ จะถูกนำมาเปรียบเทียบกับพารามิเตอร์ เช่น วิธีการ ชุดข้อมูลที่ใช้ ขนาดภาพ ความแม่นยำ การเรียกคืน เทคโนโลยีที่ใช้ ฯลฯ เพื่อให้ได้ข้อสรุปว่าอัลกอริทึมใดจะดีที่สุดและมีประสิทธิภาพสำหรับการตรวจจับวัตถุ ในปัจจุบัน เนื่องจากราคาที่ต่ำและใช้งานง่าย โดรนจึงอาจเป็นภัยคุกคามที่เป็นอันตรายได้ ในด้านความมั่นคงสาธารณะและความเป็นส่วนตัว การติดตั้งระบบตรวจจับโดรนในพื้นที่หวงห้ามถือเป็นสิ่งสำคัญ โมเดลการวิเคราะห์เชิงเปรียบเทียบนี้ให้ภาพรวมกว้างๆ ว่าอัลกอริทึมการตรวจจับวัตถุต่างๆ ทำงานอย่างไร และช่วยในการทำความเข้าใจอัลกอริทึมที่ดีที่สุดที่จะใช้สำหรับการตรวจจับโดรนที่มีความแม่นยำและแม่นยำสูงสุด

Sarfaraz, Masood., Saima, Majid., Fayadh, Alenezi., Fatma, Ulusal., Sarfaraz, Masood., Musheer, Ahmad., Emine, Selda, Gündüz., Kemal, Polat. (2022) [25] ได้กล่าวถึง อัคคีภัยเป็นภัยธรรมชาติที่รุนแรงซึ่งก่อให้เกิดอันตรายร้ายแรงต่อชีวิตมนุษย์และสิ่งแวดล้อม ผลงานล่าสุดได้เสนอการใช้คอมพิวเตอร์วิทัศน์เพื่อพัฒนาระบบตรวจจับอัคคีภัยอัตโนมัติที่คุ้มค่า บทความนี้นำเสนอกรอบการทำงานที่กำหนดเองสำหรับการตรวจจับไฟโดยใช้การเรียนรู้แบบถ่ายโอนกับ CNN ที่ล้ำสมัยซึ่งได้รับการฝึกฝนเกี่ยวกับภาพการลุกลามของไฟในโลกแห่งความเป็นจริง เฟรมเวิร์กยังใช้วิธีการ Grad-CAM สำหรับการแสดงภาพและการแปลตำแหน่งของไฟในภาพ โมเดลยังใช้กลไกความสนใจที่ช่วยให้เครือข่ายได้รับประสิทธิภาพที่ดีขึ้นอย่างมาก สังเกตได้จากผลลัพธ์ของ Grad-CAM ว่าการใช้ความสนใจที่เสนอนั้นทำให้แบบจำลองไปสู่การแปลตำแหน่งของไฟในภาพได้ดีขึ้น ในบรรดาโมเดลที่มีการสำรวจมากมาย Efficient-NetB0 กลายเป็นตัวเลือกเครือข่ายที่เหมาะสมที่สุดสำหรับปัญหานี้ สำหรับชุดข้อมูลภาพไฟในโลกแห่งความเป็นจริงที่เลือก ความแม่นยำในการทดสอบที่ 95.40% สนับสนุนประสิทธิภาพของแบบจำลองในการตรวจจับไฟจากตัวอย่างภาพที่นำเสนออย่างมาก นอกจากนี้ การเรียกคืน 97.61 ที่สูงมากยังเน้นย้ำว่าแบบจำลองนี้มีผลลบลวงเล็กน้อย ซึ่งบ่งบอกว่าเครือข่ายมีความน่าเชื่อถือสำหรับการตรวจจับอัคคีภัย

บทที่ 3

วิธีดำเนินการวิจัย

บทนี้จะกล่าวถึง เครื่องมือที่ใช้ในการวิจัย ขั้นตอนการดำเนินงานวิจัย การรวบรวม การเตรียมข้อมูล การเลือกโมเดลและการประเมินโมเดล โดยมีรายละเอียดดังนี้

3.1 เครื่องมือที่ใช้ในการวิจัย

เครื่องมือที่ใช้ในงานวิจัยนี้ประกอบด้วย ฮาร์ดแวร์ (Hardware) และ ซอฟต์แวร์ (Software) ต่างๆดังนี้

3.1.1 ฮาร์ดแวร์ (Hardware)

แล็ปท็อป

- หน่วยประมวลผลข้อมูล Central Processing Unit (CPU): Intel Core i9-13900HX (2.2GHz-5.4GHz)
- หน่วยประมวลผลภาพ Graphics Processing Unit (GPU): NVIDIA GeForces RTX 4060 (8GB) Laptop
- หน่วยความจำระยะสั้น Random-Access Memory (RAM): ขนาด 16 GB DDR5-5600

3.1.2 ซอฟต์แวร์ (Software)

Python เวอร์ชัน 3.12 เป็นภาษาการเขียนโปรแกรมที่ใช้อย่างแพร่หลายในเว็บแอปพลิเคชัน การพัฒนาซอฟต์แวร์ วิทยาศาสตร์ข้อมูล และ Machine Learning (ML) งานวิจัยนี้ใช้ Python เนื่องจากมีประสิทธิภาพ สามารถเขียนเพื่อให้ใช้ GPU ในการคำนวณได้ ซึ่ง GPU มีจำนวน cores มากกว่า CPU (บางรุ่นอาจมีหลายพัน cores) และถูกออกแบบมาเพื่อจัดการกับงานที่มีการคำนวณในลักษณะเดียวกันซ้ำๆ (parallel processing) เหมาะกับการประมวลผลข้อมูลที่ต้องทำซ้ำจำนวนมาก เช่น การคำนวณใน neural networks สามารถประมวลผลข้อมูลในปริมาณมากได้พร้อมกัน ทำให้สามารถเทรนโมเดลได้เร็วกว่า CPU อย่างมาก โดยเฉพาะในงานที่เกี่ยวข้องกับ deep learning และ large-scale models

PyCharm เป็นโปรแกรมสำหรับเขียนโค้ดสำหรับภาษา Python มีรุ่นฟรี (EDU) สำหรับการฝึกเรียนรู้ (Learners)

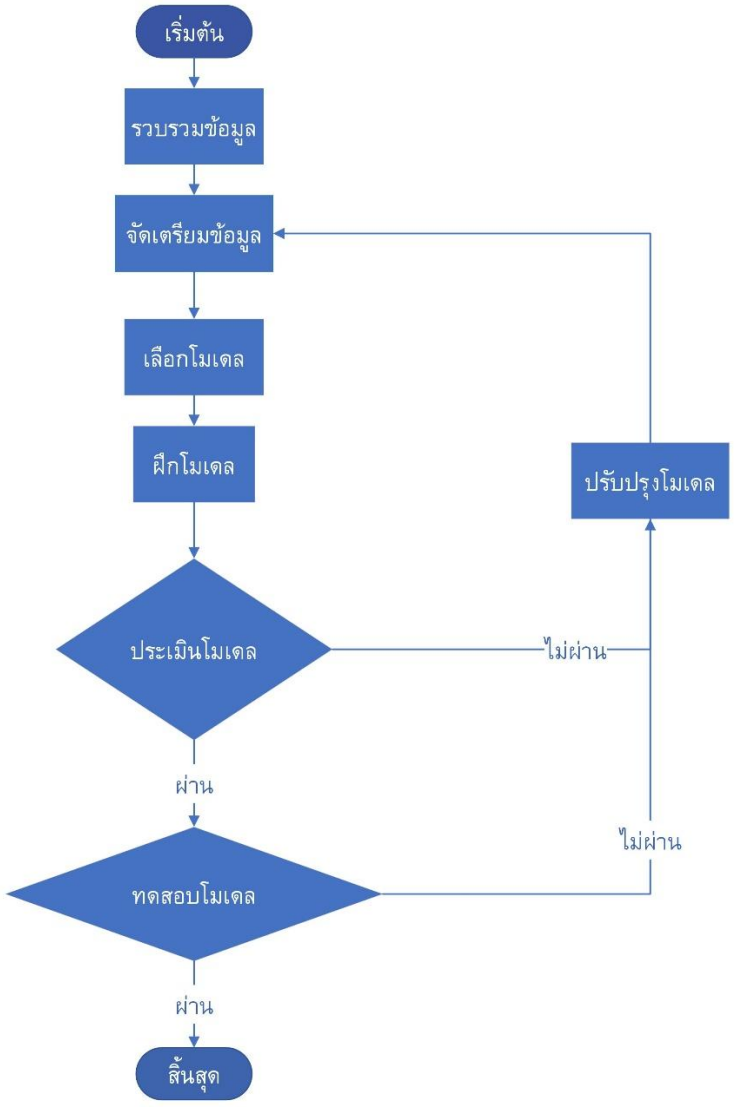
YOLO เวอร์ชัน 8 เป็นอัลกอริทึมที่ใช้ตรวจจับวัตถุที่มีความแม่นยำและความเร็วสูงที่เหนือชั้นของ YOLOv8 ทำให้โมเดลคอมพิวเตอร์วิทัศน์โดดเด่นจากเวอร์ชันก่อนหน้า เนื่องจากสามารถตรวจจับวัตถุได้แบบเรียลไทม์ ได้อย่างรวดเร็วยิ่งขึ้น ไม่เหมือนในเวอร์ชันก่อนๆ

Git Hub YOLOv8 เป็นเว็บไซต์ Git (Version Control Repository) ที่อยู่บนอินเตอร์เน็ต มีการทำงานแบบเดียวกับ Git เลย แต่สามารถเข้าถึงข้อมูลและจัดการไปผ่าน Web โดยไม่ต้องเสียเงินหรือลงทุกตั้ง Server เพื่อติดตั้ง Git แต่ Code Project ทั้งหมดจะถูกแจกจ่ายให้คนอื่นๆสามารถเห็นได้ด้วย

Visio เป็นโปรแกรมที่ใช้ในการออกแบบแผนผังต่างๆ

Roboflow เป็นเว็บไซต์ที่ช่วยในการกำหนดตำแหน่งบนรูปภาพ โดยการตีกรอบ และสร้างชุดข้อมูลสำหรับเทรนข้อมูล

3.2 วิธีการดำเนินงานวิจัย



ภาพประกอบ 25 แผนภาพวิธีดำเนินงานวิจัย

ขั้นตอนที่ 1 การรวบรวมข้อมูล ขั้นตอนนี้ เป็นการรวบรวมข้อมูลรูปภาพและวิดีโอ ที่จะใช้ในการดำเนินงานวิจัย

ขั้นตอนที่ 2 การเตรียมข้อมูล ขั้นตอนนี้เป็นการจัดสรรข้อมูลที่ได้ทำการรวบรวมข้อมูลมา โดยแบ่งออกเป็นชุดข้อมูลการฝึกการเรียนรู้ให้ระบบ (Training) ตรวจสอบความถูกต้อง (Validation) และทำสอบระบบ (Testing)

ขั้นตอนที่ 3 การเลือกแบบโมเดล ขั้นตอนนี้เกี่ยวข้องกับการเลือกอัลกอริทึมการตรวจจับวัตถุที่เหมาะสมสำหรับการฝึกแบบจำลองการตรวจจับอัคคีภัย มีอัลกอริทึมหลายตัวให้เลือก เช่น YOLOv8, Faster R-CNN และ SSD ซึ่งแต่ละอัลกอริทึม ก็มีข้อดีและข้อเสียของตัวเอง อัลกอริทึมที่เลือกควรมีประสิทธิภาพที่ดีกับชุดข้อมูลที่รวบรวม และสามารถจัดการสถานการณ์อัคคีภัยที่แตกต่างกัน ขึ้นอยู่กับข้อกำหนดของระบบตรวจจับอัคคีภัยอัจฉริยะ YOLOv8 เป็นตัวเลือกยอดนิยมเนื่องจากความเร็วและความแม่นยำ แต่อัลกอริทึมอื่นๆ ก็สามารถใช้ได้ตามความต้องการเฉพาะเช่นกัน

ขั้นตอนที่ 4 การฝึกโมเดล ขั้นตอนนี้ โมเดล YOLOv8 ได้รับการฝึกฝนโดยใช้เฟรมเวิร์กการเรียนรู้เชิงลึก คือ Pytorch

ขั้นตอนที่ 5 การประเมินโมเดล ขั้นตอนนี้จะประเมินประสิทธิภาพของแบบจำลองที่ได้รับการฝึก จะได้รับการประเมินโดยใช้ตัวชี้วัดต่างๆ เช่น ความแม่นยำ ความแม่นยำ การเรียกคืน และคะแนน F1 ตัวชี้วัดเหล่านี้เป็นตัววัดว่าแบบจำลองทำงานได้ดีเพียงใด หากประสิทธิภาพของแบบจำลองไม่เป็นที่น่าพอใจ สามารถปรับแบบละเอียดได้โดยการปรับไฮเปอร์พารามิเตอร์หรือเพิ่มข้อมูลการฝึกเพิ่มเติม สิ่งสำคัญคือต้องหาสมดุลระหว่างการโอเวอร์ฟิตและการลดขนาดลง เพื่อให้แน่ใจว่าโมเดลจะสรุปข้อมูลใหม่ได้ดี

ขั้นตอนที่ 6 การปรับปรุงโมเดล ขั้นตอนนี้เกี่ยวข้องกับการปรับใช้โมเดลที่ผ่านการฝึกอบรมในระบบเรียลไทม์ที่สามารถประมวลผลสตรีมวิดีโอสดจากกล้องได้ ต้องใช้คอมพิวเตอร์หรือเซิร์ฟเวอร์ที่มีพลังประมวลผลสูงและ GPU เพื่อประมวลผลสตรีมวิดีโอแบบเรียลไทม์ ระบบควรจะสามารถอ่านเฟรมวิดีโอจากกล้องหรือสตรีมวิดีโอ ประมวลผลผ่านโมเดลที่ผ่านการฝึกอบรม และสร้างการแจ้งเตือนเมื่อตรวจพบเพลิงไหม้ ผลบวกลวงสามารถจัดการได้โดยใช้ค่าเกณฑ์ ซึ่งจะกำหนดระดับความเชื่อมั่นขั้นต่ำที่จำเป็นสำหรับแบบจำลองในการตรวจจับเพลิงไหม้ และการตรวจจับที่ต่ำกว่าค่าเกณฑ์นี้จะละทิ้งเป็นผลบวกลวง

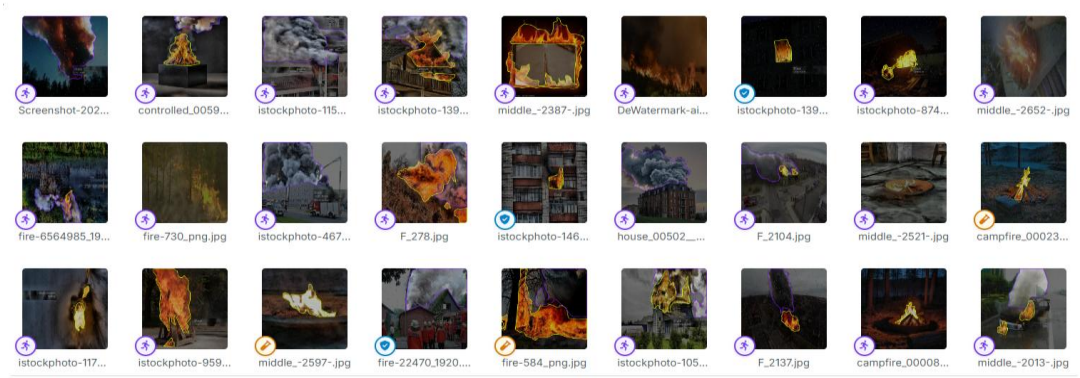
ขั้นตอนที่ 7 การทดสอบ ขั้นตอนนี้เป็นการทดสอบประสิทธิภาพผลลัพธ์ของโมเดลและบันทึกผลลัพธ์ที่ได้

3.3 การรวบรวมข้อมูลและการเตรียมข้อมูล

ในงานวิจัยนี้ได้นำคลิปและภาพที่มีการเกิดเหตุเพลิงไหม้ซึ่งสามารถหาได้จากเว็บไซต์ต่างๆ เช่น เว็บไซต์ข่าวหรือโซเชียลมีเดียต่างๆ จำนวน 10 วิดีโอ ที่ใช้ในการทดสอบ และภาพจำนวน 4,842

ภาพ และเมื่อทำการตีกรอบภาพใน Roboflow จะได้ภาพจำนวน 4,842x3 เท่ากับ 14,526 ภาพและทำการแบ่งเป็นชุดข้อมูลภาพเพื่อใช้ในการ ฝึกการเรียนรู้ให้ระบบ (Training) ตรวจสอบความถูกต้อง (Validation) และทำสอบระบบ (Testing) โดยแบ่งเป็น 87.5%,8.3%,4.2% ตามลำดับ ดังตาราง 2

ภาพประกอบ 26 ตัวอย่างรูปภาพก่อนนำมาแบ่งชุดข้อมูล



ภาพประกอบ 27 ตัวอย่างรูปภาพหลังนำมาแบ่งชุดข้อมูลและตีกรอบโดย Roboflow

ในส่วนของวิดีโอจะนำมาทดสอบในการประเมินประสิทธิภาพในการตรวจจับ ซึ่งมีจำนวน 10 คลิป โดยจะใช้ในการทดสอบความแม่นยำและความสามารถการทำงานแบบเรียลไทม์ได้ในเวลาเฉลี่ยว่ากี่ ms ต่อภาพ

ตาราง 2 การแบ่งชุดข้อมูลที่ใช้ในงานวิจัย

ชุดข้อมูลที่	การใช้งาน	เปอร์เซ็นต์	จำนวนภาพ (รูป)	จำนวนวิดีโอ
1	ฝึกการเรียนรู้ให้ระบบ (Training)	87.5	12,711	-
2	ตรวจสอบความถูกต้อง (Validation)	8.3	1,211	-
3	ทำสอบระบบ (Testing)	4.2	604	10
	รวม	100	14,526	10

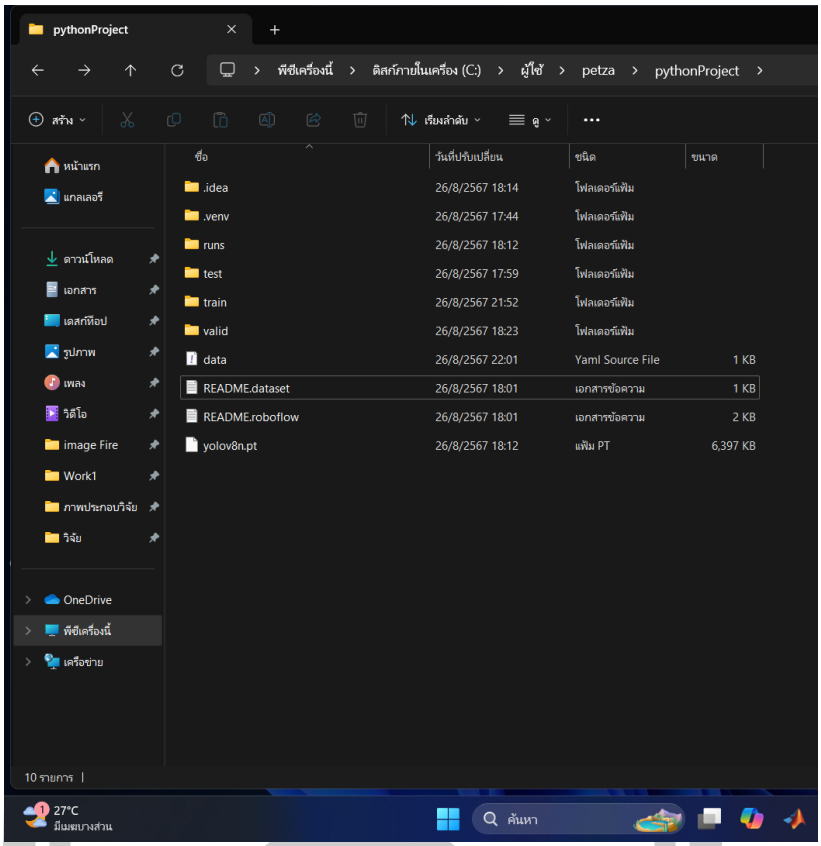
การจัดกลุ่มชุดข้อมูลเป็นกระบวนการนำรูปภาพที่เกี่ยวกับเหตุการณ์ไฟไหม้ รวมถึงควันไฟทั้งในและนอกอาคาร มาตีกรอบตำแหน่งและประเภทของวัตถุในแต่ละภาพด้วยโปรแกรม Roboflow เพื่อระบุลักษณะและตำแหน่งที่ต้องการตรวจจับ ได้อย่างมีประสิทธิภาพ นอกจากนี้ยังเพิ่มกระบวนการ Data Augmentation เพื่อขยายจำนวนข้อมูลสำหรับการเรียนรู้ของ AI โดยการปรับมุมมองของภาพ เช่น การหมุน การปรับแสง หรือการย่อ-ขยาย ซึ่งช่วยสร้างภาพใหม่ที่หลากหลายแต่ยังคงลักษณะเดิมของภาพต้นฉบับ กระบวนการนี้เพิ่มจำนวนข้อมูลให้มากขึ้นและช่วยให้ระบบมีความแม่นยำยิ่งขึ้น หลังจากจัดกลุ่มและปรับปรุงข้อมูลเสร็จสิ้นแล้ว ข้อมูลจะถูกแบ่งออกเป็น 3 ชุด คือ

Training set สำหรับการฝึกโมเดล

Validation set สำหรับตรวจสอบประสิทธิภาพของโมเดลระหว่างการฝึก

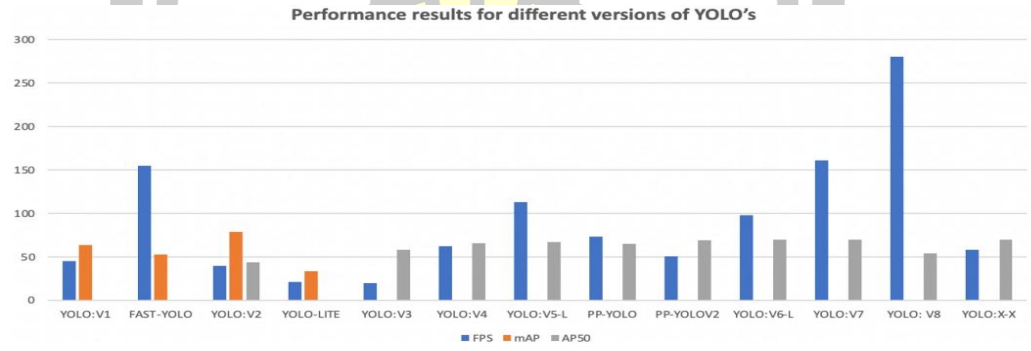
Test set สำหรับประเมินผลสุดท้ายของโมเดล

กระบวนการนี้ช่วยให้ AI สามารถเรียนรู้และตรวจจับไฟไหม้หรือควันไฟในภาพได้อย่างแม่นยำและครอบคลุมมุมมองที่หลากหลายมากขึ้น ดังภาพประกอบ 28



ภาพประกอบ 28 หลังนำออกจาก Roboflow

3.4 การเลือกโมเดล



ภาพประกอบ 29 เปรียบเทียบประสิทธิภาพของ YOLO แต่ละเวอร์ชัน

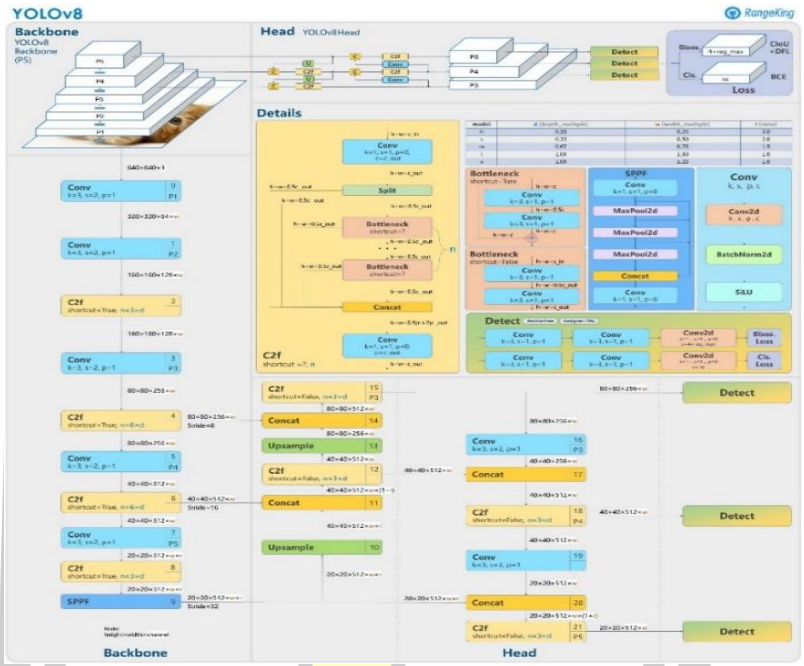
เนื่องจากระบบตรวจจับเปลวเพลิงเป็นระบบแบบ Real Time จึงควรเลือกอัลกอริทึมที่เหมาะสมกับการตรวจจับวัตถุแบบ Real-Time และเมื่อคำนึงถึงความแม่นยำ (AP) และความรวดเร็วในการประมวลผลภาพ (FPS) เป็นสำคัญ ดังนั้นการศึกษาในครั้งนี้ผู้วิจัยจึงเลือกใช้อัลกอริทึม YOLO ซึ่งเป็นอัลกอริทึมที่เหมาะกับการใช้งานแบบ real-time โดยเลือกใช้เป็น YOLOv8 ซึ่งมีค่า FPS ที่สูงสุด

การทำงาน

เครือข่ายแกนหลัก Darknet-53 ใน YOLOv8 ซึ่งได้รับแรงบันดาลใจจาก VGG จะเพิ่มจำนวนช่องสัญญาณเป็นสองเท่าหลังจากรวมการดำเนินการ นอกจากนี้ ยังวางเคอร์เนลแบบบิดขนาด 1 × 1 ไว้ระหว่างเคอร์เนลแบบบิดขนาด 3 × 3 เพื่อบีบอัดคุณสมบัติ และใช้การรวมค่าเฉลี่ยทั่วโลกเลเยอร์ทำให้เป็นมาตรฐานแบบแบตช์ใช้เพื่อทำให้การฝึกโมเดลมีความเสถียร เร่งการลู่เข้า และจัดให้มีการทำให้เป็นมาตรฐาน โมดูล CSP เป็นโมดูลการแยกคุณลักษณะที่มีจุดมุ่งหมายเพื่อเพิ่มประสิทธิภาพในการแยกคุณลักษณะโดยใช้การเชื่อมต่อข้ามขั้นตอน ช่วยให้สามารถแบ่งปันคุณลักษณะต่างๆ ในหลายขั้นตอน และปรับปรุงพารามิเตอร์และประสิทธิภาพการคำนวณโดยการเชื่อมต่อบางส่วนจากขั้นตอนต่างๆ ส่วน Neck ของรุ่น YOLOv8 ใช้แนวคิด CSP เพื่อหลอมรวมคุณสมบัติดั้งเดิมเข้ากับคุณสมบัติที่ประมวลผลผ่านการดำเนินการแบบ Convolution หลายครั้ง จึงช่วยเพิ่มความสามารถในการแยกคุณสมบัติ เครือข่ายแกนหลักและส่วนคอของ YOLOv8 ได้รับแรงบันดาลใจจากหลักการออกแบบของ YOLOv7 ELAN และแนะนำโมดูล C2F โมดูล C2F มีเป้าหมายเพื่อลดจำนวนพารามิเตอร์ ในขณะที่ยังคงรักษาข้อมูลโฟลว์เกรเดียนต์ที่สมบูรณ์ ในช่วง 10 ยุคสุดท้ายของการฝึกอบรม การเพิ่มข้อมูลของโมเสค ถูกปิดใช้งานเพื่อปรับปรุงความแม่นยำ

ส่วนหัวของ YOLOv8 จะแยกส่วนหัวการจำแนกประเภทและการตรวจจับออกจากกัน สาขาการจำแนกประเภทยังคงใช้การสูญเสียแบบไบนารีข้ามเอนโทรปี (BCE) ในขณะที่สาขาการถดถอยรวมการสูญเสียโฟกัสการกระจายฟังก์ชันการสูญเสียนี้จัดการปัญหาต่างๆ เช่น ความไม่สมดุลของคลาสและความยากลำบากในการจำแนกตัวอย่างที่ท้าทายได้อย่างมีประสิทธิภาพ

ในแง่ของกลยุทธ์การกำหนดตัวอย่างเชิงบวกและเชิงลบ YOLOv8 แยกออกจากกลยุทธ์การจัดสรรแบบคงที่ที่ใช้ใน YOLOv5 แต่จะใช้ Task-Aligned Assigner จากอัลกอริทึม TOOD แทน ผู้มอบหมายรายนี้เลือกตัวอย่างที่เป็นบวกโดยพิจารณาจากคะแนนถ่วงน้ำหนักของงานจำแนกประเภทและงานการถดถอย



ภาพประกอบ 30 การทำงานของ YOLOv8

YOLOv8 มีหลายขนาด ซึ่งแต่ละขนาดถูกออกแบบมาให้มีสมดุลระหว่างความเร็วและความแม่นยำที่แตกต่างกัน เพื่อให้เหมาะสมกับความสามารถของฮาร์ดแวร์และความต้องการของแอปพลิเคชันต่าง ๆ ดังนี้

1. YOLOv8n (Nano)

ขนาดเล็กที่สุด ความเร็วสูงสุดในการประมวลผล จำนวนพารามิเตอร์ประมาณ 3-5 ล้าน เหมาะสำหรับอุปกรณ์ขอบเขต เช่น โทรศัพท์มือถือ, Raspberry Pi, และ NVIDIA Jetson เหมาะกับแอปพลิเคชันแบบเรียลไทม์ที่ต้องการความหน่วงต่ำ แต่ไม่จำเป็นต้องมีความแม่นยำสูงสุด

2. YOLOv8s (Small)

ใหญ่กว่า nano แต่ยังคงมีขนาดกะทัดรัด ช้ากว่า nano เล็กน้อย แต่ความแม่นยำเพิ่มขึ้น จำนวนพารามิเตอร์ประมาณ 7-10 ล้าน เหมาะสำหรับแอปพลิเคชันที่ต้องการความสมดุลระหว่างความเร็วและความแม่นยำ ยังเหมาะสมสำหรับอุปกรณ์ขอบเขต แต่มีประสิทธิภาพในการตรวจจับที่ดีกว่า

3. YOLOv8m (Medium)

ขนาดกลาง ความเร็วปานกลาง มีสมดุลที่ดีระหว่างความแม่นยำและเวลาประมวลผลจำนวนพารามิเตอร์ประมาณ 20-30 ล้าน เหมาะสำหรับแอปพลิเคชันที่ต้องการทั้งความเร็วและความแม่นยำ เหมาะสำหรับงานที่มีความซับซ้อนมากขึ้น แต่ยังสามารถทำงานได้บนอุปกรณ์ที่มีทรัพยากรเพียงพอ

4. YOLOv8l (Large)

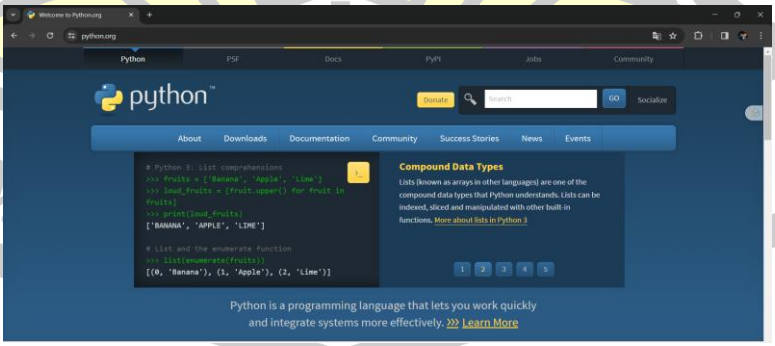
ขนาดใหญ่ขึ้น ช้ากว่ารุ่นเล็ก แต่มีความแม่นยำสูงกว่าจำนวนพารามิเตอร์ประมาณ 40-60 ล้าน เหมาะสำหรับกรณีที่ต้องการความแม่นยำสูงสุด เหมาะสมสำหรับการใช้งานบน GPU ที่มีประสิทธิภาพหรือระบบบนคลาวด์

5. YOLOv8x (Extra Large)

ขนาดใหญ่ที่สุด ความเร็วต่ำสุดเนื่องจากจำนวนพารามิเตอร์ที่มาก จำนวนพารามิเตอร์ประมาณ 60-90 ล้านเหมาะกับงานที่ต้องการความแม่นยำสูง เหมาะสำหรับระบบที่มีความสามารถคอมพิวเตอร์สูง เช่น GPU ระดับสูงหรือระบบคลาวด์

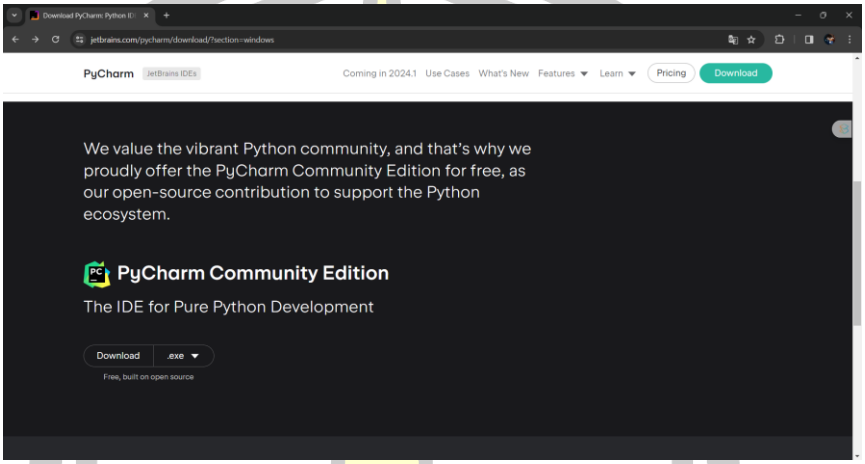
3.4.1 ซอฟต์แวร์ที่จำเป็น

ซอฟต์แวร์ที่จำเป็นสำหรับ YOLOv8 คือ Python ซึ่งเป็นซอฟต์แวร์ สำหรับภาษา Python โดยเลือกติดตั้งเวอร์ชัน 3.12.2 สามารถดาวน์โหลดได้จากเว็บไซต์ python.org ดังแสดงในภาพประกอบ 31



ภาพประกอบ 31 เว็บไซต์ดาวโหลด Python

ซอฟต์แวร์ที่จำเป็นสำหรับเขียนภาษา Python คือ PyCharm เป็นโปรแกรมที่ใช้เขียนภาษา Python โดยเลือกใช้เวอร์ชันล่าสุดคือ 2023.3.4 มีรุ่นฟรี (EDU) สำหรับการฝึกเรียนรู้ (Learners) ที่เปิดให้ดาวน์โหลดได้ฟรีที่ Jetbrains.com ดังภาพประกอบ 32

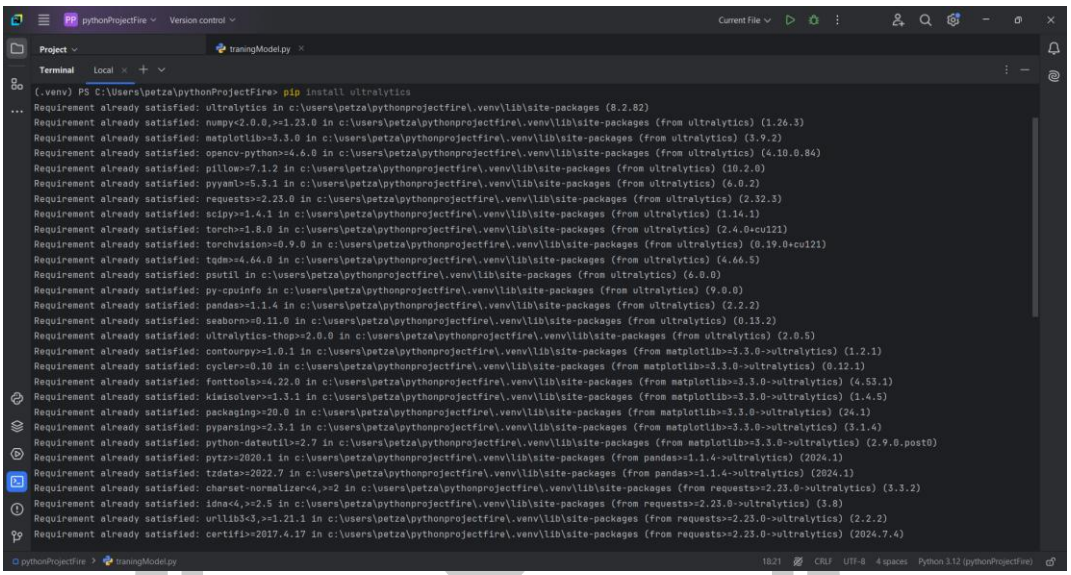


ภาพประกอบ 32 เว็บไซต์ดาวโหลด PyCharm

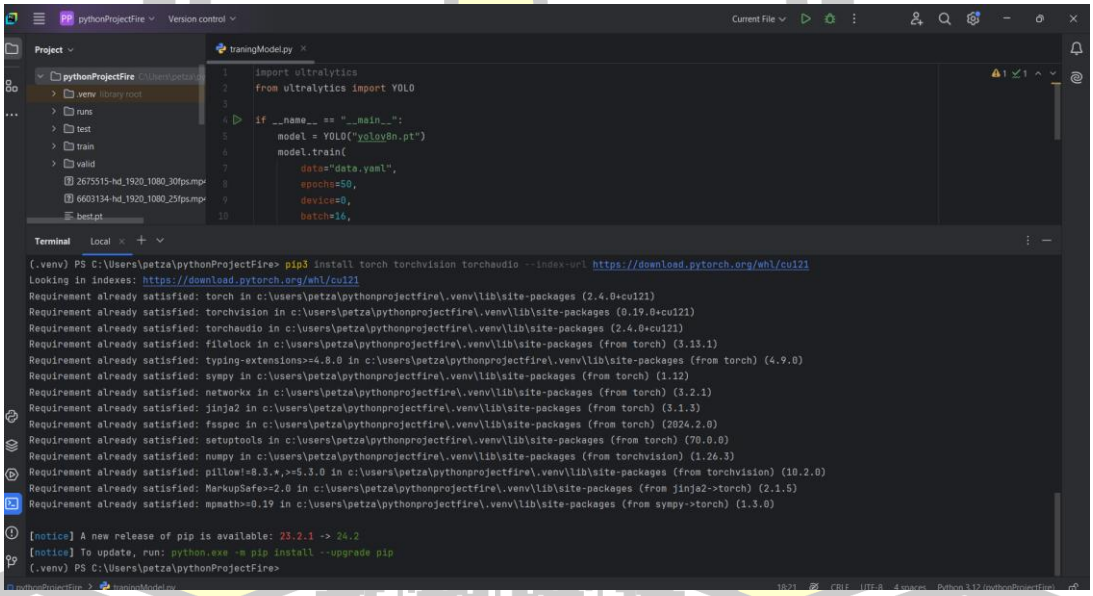
3.4.2 Library ที่จำเป็น

โดย Library หลักๆ เช่น Ultralytics, Torch เป็นต้น ซึ่งเมื่อติดตั้ง Ultralytics จะมีการดาวน์โหลด Library เพิ่มเติมโดยอัตโนมัติ ดังภาพประกอบ 33 และ ภาพประกอบ 34 ตามลำดับ

Ultralytics YOLOv8 โมเดลการตรวจจับวัตถุและการแบ่งส่วนรูปภาพแบบเรียลไทม์ที่ได้รับการยกย่อง YOLOv8 สร้างขึ้นจากความก้าวหน้าล้ำสมัยในการเรียนรู้เชิงลึกและการมองเห็นด้วยคอมพิวเตอร์ โดยนำเสนอประสิทธิภาพที่เหนือชั้นในแง่ของความเร็วและความแม่นยำ การออกแบบที่คล่องตัวทำให้เหมาะสำหรับแอปพลิเคชันต่างๆ และปรับให้เข้ากับแพลตฟอร์มฮาร์ดแวร์ต่างๆ ได้อย่างง่ายดาย ตั้งแต่อุปกรณ์ Edge ไปจนถึง Cloud API



ภาพประกอบ 33 ติดตั้ง Library ultralytics



ภาพประกอบ 34 ติดตั้ง Library torch

3.5 ประเมินโมเดล

ในการประเมินความสามารถของโมเดลปัญญาประดิษฐ์ (AI) ในการจำแนกประเภทหรือทำนายผลลัพธ์ มีสมการคำนวณหลายแบบเพื่อวัดผลต่าง ๆ ซึ่งสามารถแบ่งออกเป็น 4 กลุ่มหลัก คือ Accuracy, Precision, RecallและF1 Score โดยรายละเอียดสมการมีดังนี้

3.5.1 Accuracy (ความแม่นยำ)

ความแม่นยำคือการวัดจำนวนการทำนายที่ถูกต้องทั้งหมด หารด้วยจำนวนการทำนายทั้งหมด ดังสมการที่ 4

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

TP คือ True Positives (ทำนายถูกต้องว่าเป็นคลาสที่สนใจ)

TN คือ True Negatives (ทำนายถูกต้องว่าไม่ใช่คลาสที่สนใจ)

FP คือ False Positives (ทำนายผิดพลาดว่าเป็นคลาสที่สนใจ)

FN คือ False Negatives (ทำนายผิดพลาดว่าไม่ใช่คลาสที่สนใจ)

3.5.2 Precision (ความเที่ยงตรง)

ความเที่ยงตรงคือการวัดความถูกต้องของการทำนายเป็นคลาสที่สนใจ โดยคิดจากจำนวนการทำนายที่ถูกต้องเป็นคลาสที่สนใจ หารด้วยจำนวนการทำนายทั้งหมดที่เป็นคลาสที่สนใจ ดังสมการที่ 5

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5)$$

3.5.3 Recall (การเรียกคืน)

การเรียกคืนคือการวัดความสามารถของโมเดลในการทำนายว่าตัวอย่างเป็นคลาสที่สนใจจากจำนวนตัวอย่างทั้งหมดที่เป็นคลาสนั้นจริง ดังสมการที่ 6

$$\text{Recall} = \frac{TP}{TP+FN} \quad (6)$$

3.5.4 Mean average precision (mAP)

mAP เป็นการวัดประสิทธิภาพของโมเดลที่ใช้ในการตรวจจับวัตถุและจำแนกประเภทในภาพ คำนวณโดยการหาพื้นที่ใต้กราฟ Precision-Recall (PR Curve) ของแต่ละคลาส ยิ่งกราฟ Precision-Recall ใกล้กับมุมขวาบนมากเท่าไร ค่าของ AP จะยิ่งสูง โดยจะคำนวณจากค่าเฉลี่ยของค่า AP (Average Precision) สำหรับทุกคลาสของวัตถุที่ต้องการตรวจจับ ตัวอย่างเช่น mAP@50

หมายถึงการคำนวณ mAP ที่ใช้ค่า IoU (Intersection over Union) ที่เกณฑ์ 0.5และmAP@50-95 คำนวณค่า mAP โดยเฉลี่ยของ IoU ในช่วง 0.5 ถึง 0.95 โดยเพิ่มขึ้นทีละ 0.05 ซึ่งเป็นมาตรฐานการประเมินในชุดข้อมูล COCO โดยมีการคำนวณ ดังสมการที่ 7

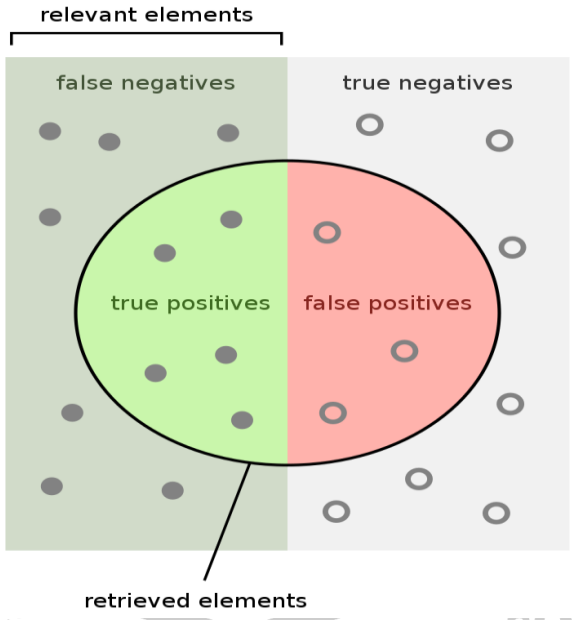
$$\text{mAP} = \frac{\sum \text{AP}}{\text{N(Class)}} \tag{7}$$

3.5.5 F1 Score

F1 Score เป็นค่าเฉลี่ยเชิงฮาร์โมนิกระหว่าง Precision และ Recall ซึ่งเหมาะสำหรับใช้วัดประสิทธิภาพของโมเดลในกรณีที่ข้อมูลไม่สมดุล ดังสมการที่ 8

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

สรุปจากสมการข้างต้น เพื่อที่จะเข้าใจง่ายขึ้นสามารถดูภาพประกอบ 35



ภาพประกอบ 35 ความแตกต่างของ (TP),(FN),(FP)และ(TN)

True Positive (TP) = ทายว่าใช่ ผลคือใช่ (แม่นยำ)

False Negative (FN) = ทายว่าไม่ ผลคือใช่ (ไม่แม่นยำ)

False Positive (FP) = ทายว่าใช่ ผลคือไม่ (ไม่แม่นยำ)

True Negative (TN) = ทายว่าไม่ ผลคือไม่ (แม่นยำ)

3.6 ทดสอบโมเดล

หลังจากการฝึกโมเดลเสร็จสิ้น จะมีการประเมินประสิทธิภาพของโมเดลด้วยชุดข้อมูลที่ไม่มีคำตอบล่วงหน้าเพื่อคาดการณ์ผลลัพธ์ ซึ่งเรียกว่ากระบวนการ ทดสอบโมเดล (Testing) โดยในขั้นตอนนี้ โมเดลจะนำชุดข้อมูลดังกล่าวมาใช้ในการประมาณผลลัพธ์ จากนั้นเปรียบเทียบผลลัพธ์ที่โมเดลคาดการณ์กับคำตอบจริงเพื่อวัดความแม่นยำของการทำงาน

การทดสอบโมเดลมีจุดมุ่งหมายเพื่อประเมิน ประสิทธิภาพ (Performance) ของโมเดลที่พัฒนาขึ้นจากข้อมูลฝึก (Training Data) โดยทั่วไป ชุดข้อมูลที่ใช้สำหรับการทดสอบ (Test Data) จะไม่ถูกใช้ในกระบวนการฝึกโมเดล เพื่อให้มั่นใจว่าโมเดลสามารถทำงานได้ดีเมื่อเจอกับข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน การทดสอบมักทำเพียงครั้งเดียวเพื่อวิเคราะห์ประสิทธิภาพของโมเดลหรือเพื่อเปรียบเทียบกับโมเดลอื่น ๆ เพื่อปรับปรุงให้ดียิ่งขึ้น

ขั้นตอนการทดสอบโมเดล มักประกอบด้วย

1. โหลดชุดข้อมูลสำหรับการทดสอบ (Test Data)

2. ใช้โมเดลที่ผ่านการฝึกด้วยข้อมูลฝึก (Training Data) ในการคาดการณ์ผลลัพธ์ของข้อมูลทดสอบ

3. ประมวลผลและวิเคราะห์ผลลัพธ์โดยใช้ฟังก์ชัน Predict

เมื่อดำเนินการทดสอบใน PyCharm โปรแกรมจะสร้างโฟลเดอร์ชื่อ Predict ซึ่งประกอบด้วยชุดข้อมูลที่ใช้ในการทดสอบ พร้อมผลลัพธ์ที่โมเดลคาดการณ์ไว้ในรูปแบบป้ายกำกับ (Labels) การจัดเก็บนี้ช่วยให้ผู้วิจัยสามารถตรวจสอบผลลัพธ์ได้อย่างสะดวก และนำผลการทดสอบไปปรับปรุงหรือเปรียบเทียบโมเดลให้มีประสิทธิภาพสูงขึ้น

บทที่ 4

ผลการวิจัยและการอภิปรายผล

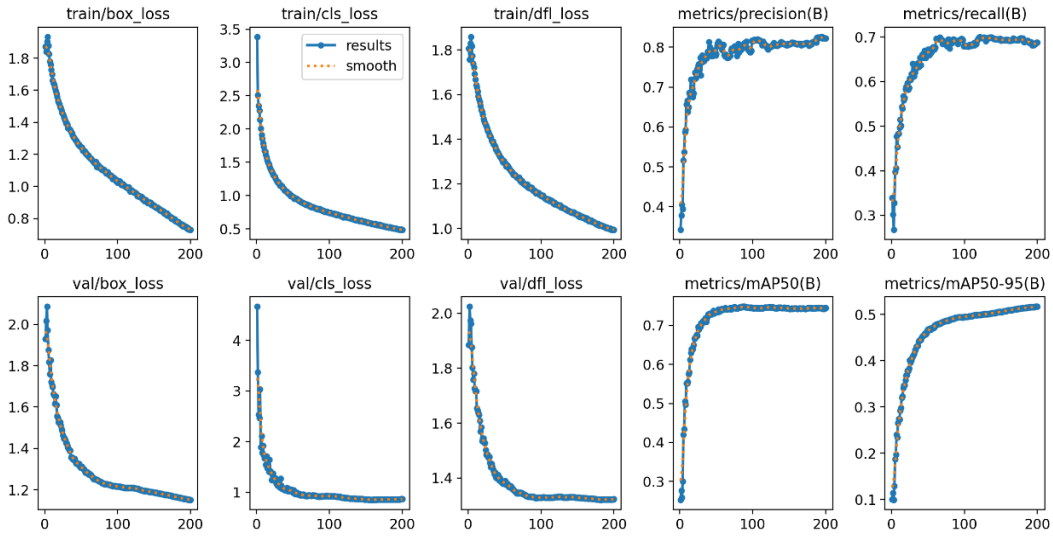
จากการศึกษาแนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง เพื่อนำความรู้มาใช้ในงานวิจัยในครั้งนี้ การวิจัยครั้งนี้ได้ทำการตรวจจับเปลวไฟด้วยการเรียนรู้เชิงลึก ด้วยการนำโมเดลที่ผู้วิจัยได้ทำการฝึกโมเดล มาทำการทดสอบความแม่นยำของการตรวจจับเปลวไฟ เมื่อใช้งานกล้องในสถานการณ์จริง

4.1 การประเมินผลการฝึก

การทดสอบประสิทธิภาพของระบบตรวจจับไฟไหม้สามารถพิจารณาจากตัวชี้วัดหลายประการ เช่น Precision, Recall และ F1-Score ซึ่งช่วยในการประเมินว่าโมเดลใดมีประสิทธิภาพดีที่สุด นอกจากนี้ยังสามารถใช้เป็นเกณฑ์ในการวิเคราะห์ว่าระบบสามารถตรวจจับเหตุการณ์ไฟไหม้ที่เกิดขึ้นจริงได้มากน้อยเพียงใด ทั้งนี้สามารถจัดลำดับโมเดลตามขนาดของโมเดลการตรวจจับจากเล็กไปใหญ่สุดได้ดังนี้

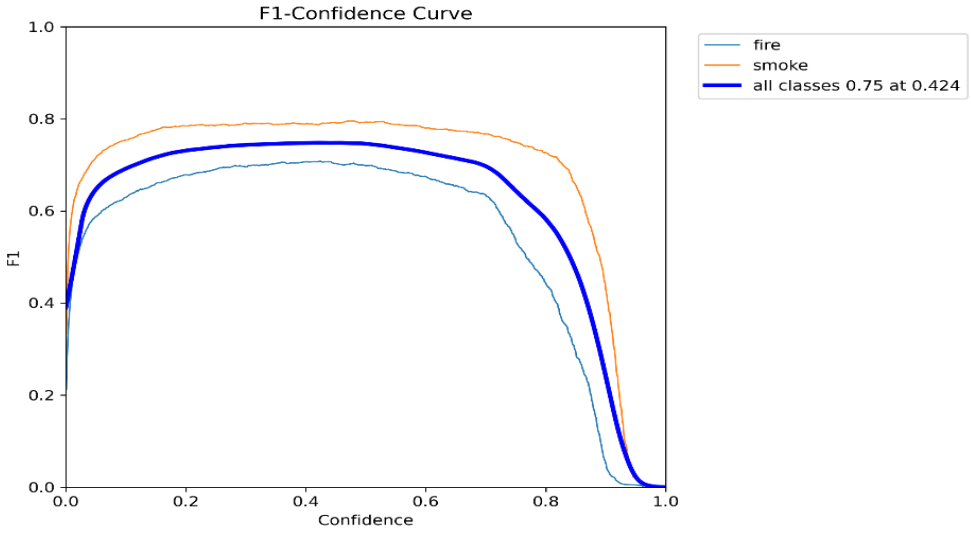
4.1.1 โมเดล n

YOLOv8n เป็นเวอร์ชันที่เล็กที่สุดของ YOLOv8 ออกแบบมาเพื่อให้ ความเร็วสูงและใช้ทรัพยากรต่ำ เหมาะสำหรับ อุปกรณ์ Edge AI เช่น Jetson Nano, Raspberry Pi และระบบที่มีข้อจำกัดด้านพลังงาน โมเดลนี้มี Inference Time ต่ำที่สุด ทำให้สามารถทำงานได้แบบเรียลไทม์ โดยไม่ต้องใช้ฮาร์ดแวร์ที่ทรงพลังมาก อย่างไรก็ตาม ความแม่นยำของ YOLOv8n ต่ำกว่ารุ่นที่ใหญ่กว่า ทำให้เกิด False Positives หรือ False Negatives ได้มากขึ้น โดยสรุป YOLOv8n เหมาะสำหรับงานที่ต้องการความเร็วสูง ใช้พลังงานต่ำ และสามารถทำงานบนอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร แต่หากต้องการความแม่นยำสูงขึ้น อาจต้องพิจารณาใช้โมเดลที่ใหญ่ขึ้นในตระกูลเดียวกัน



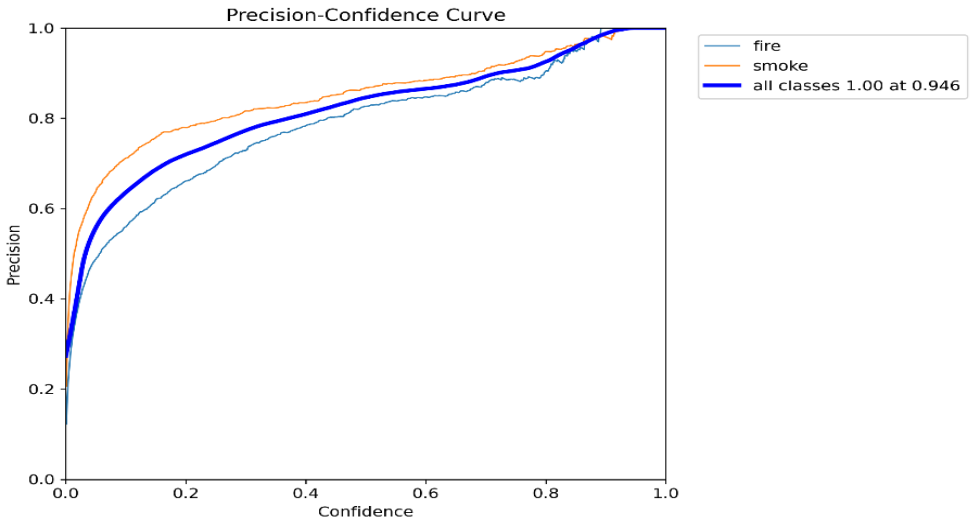
ภาพประกอบ 36 การฝึกโมเดล YOLOv8n

จากภาพประกอบ 36 การฝึกโมเดล YOLOv8n แสดงให้เห็นถึงการปรับปรุงที่ดีตลอด 200 Epochs โดยค่า Loss ในการกำหนดขอบเขตวัตถุ (box_loss), การจำแนกประเภท (cls_loss) และการคาดการณ์ตำแหน่ง (dfl_loss) ลดลงอย่างต่อเนื่อง แสดงว่าโมเดลเรียนรู้และปรับปรุงความแม่นยำได้ดีขึ้น ในขณะเดียวกัน ค่า Precision และ Recall เพิ่มขึ้น โดย Precision คงที่ที่ประมาณ 0.8 แสดงว่าโมเดลสามารถแยกแยะวัตถุได้ดี ขณะที่ Recall อยู่ที่ 0.7 บ่งบอกว่าโมเดลยังอาจพลาดการตรวจจับบางวัตถุ นอกจากนี้ ค่า mAP50 และ mAP50-95 เพิ่มขึ้นอย่างต่อเนื่องและมีความเสถียรที่ระดับสูง ซึ่งสะท้อนถึงประสิทธิภาพโดยรวมของโมเดลที่ดีขึ้น โดยสรุป YOLOv8n สามารถเรียนรู้และพัฒนาได้อย่างมีประสิทธิภาพ แต่ยังสามารถปรับปรุงเพิ่มเติมได้โดย เพิ่มชุดข้อมูลที่ครอบคลุมขึ้น หรือ ปรับแต่งค่า Hyperparameters เพื่อเพิ่มความสามารถในการตรวจจับวัตถุที่ซับซ้อนขึ้น



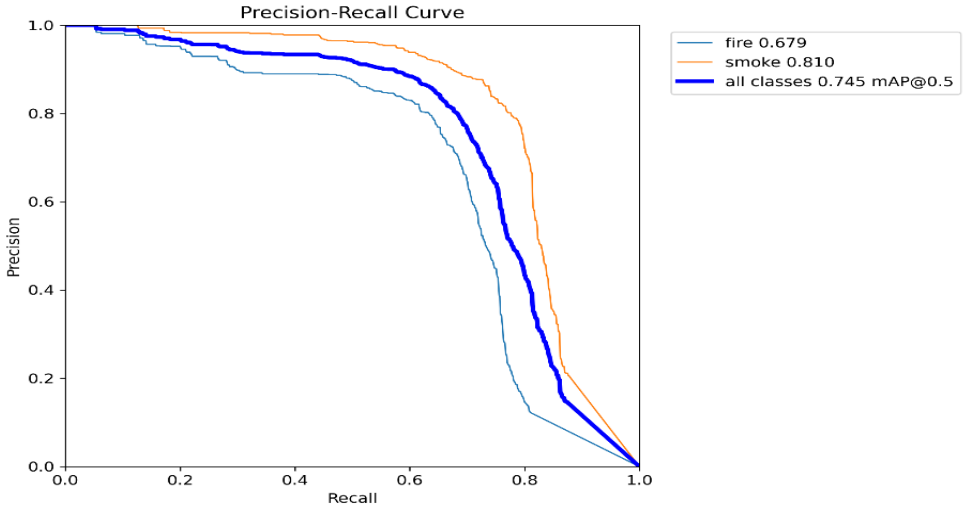
ภาพประกอบ 37 กราฟ F1-Confidence Curve (Model n)

จากภาพประกอบ 37 กราฟแสดงความสัมพันธ์ระหว่างค่า F1-score กับ Confidence YOLOv8n เป็นโมเดลขนาดเล็กที่ทำงานรวดเร็วและเหมาะสำหรับ Jetson Nano โดยมี F1-Score สูงสุดที่ 0.75 เมื่อ Confidence = 0.424 ซึ่งเป็นจุดสมดุลระหว่าง Precision และ Recall โมเดลตรวจจับ ควัน (smoke) ได้แม่นยำกว่าไฟ (fire) และเมื่อ Confidence สูงขึ้น F1-Score จะลดลง อาจเกิด False Positives หรือ False Negatives หากตั้งค่าไม่เหมาะสม เพื่อปรับปรุงประสิทธิภาพควรผสานรวมกับเซ็นเซอร์ เช่น อินฟราเรดหรือเซ็นเซอร์ตรวจจับควันในการใช้งานจริง



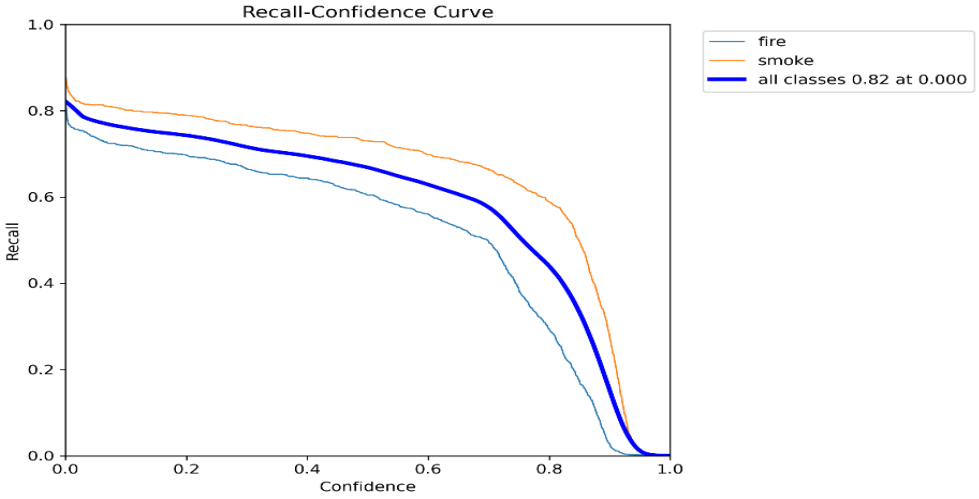
ภาพประกอบ 38 กราฟ Precision-Confidence Curve (Model n)

จากภาพประกอบ 38 Precision-Confidence Curve แสดงความสัมพันธ์ระหว่าง Precision และ Confidence ของโมเดล YOLOv8n ในการตรวจจับ ไฟ (fire) และควัน (smoke) โดยพบว่า Precision สูงสุดที่ 1.00 เมื่อ Confidence = 0.946 ซึ่งหมายความว่า หากตั้งค่า Confidence สูงสุด โมเดลจะไม่มีข้อผิดพลาด แต่ Recall อาจลดลง ทำให้พลาดการตรวจจับบางส่วน ทั้งนี้ โมเดลตรวจจับ ควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่อยู่สูงกว่าเส้นสีฟ้า การเลือกค่า Confidence จึงต้องพิจารณาร่วมกับ Recall เพื่อให้เกิดสมดุลที่เหมาะสมในการใช้งานจริง



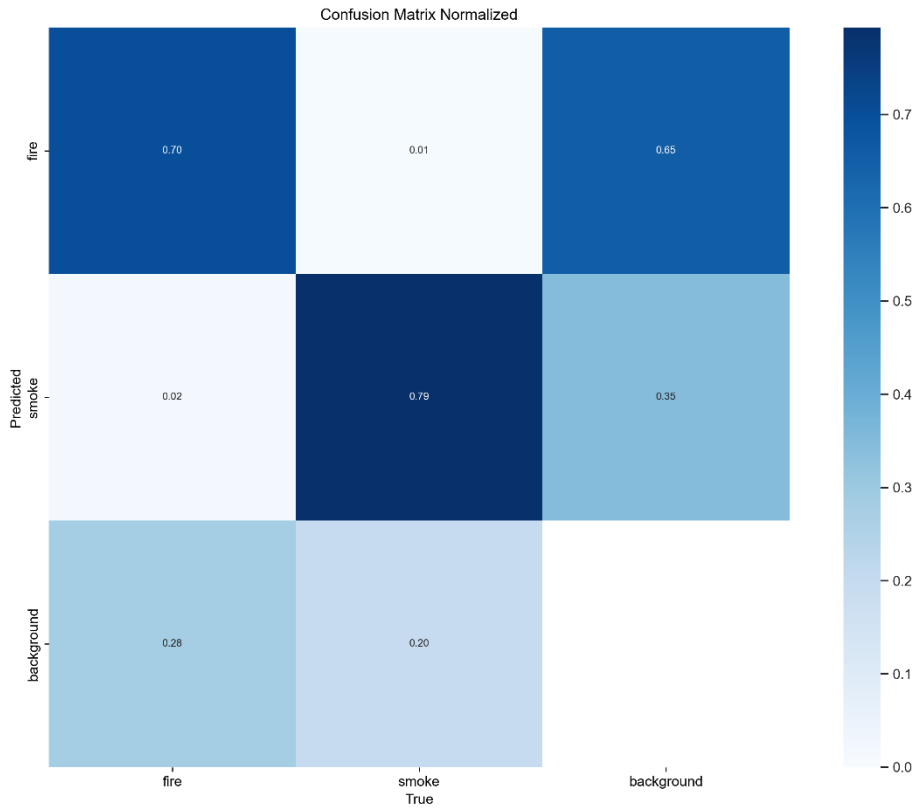
ภาพประกอบ 39 กราฟ Precision-Recall Curve (Model n)

จากภาพประกอบ 39 Precision-Recall Curve แสดงว่าโมเดล YOLOv8n สามารถตรวจจับ ควัน (Smoke) ได้แม่นยำกว่าการตรวจจับ ไฟ (Fire) โดยมีค่า mAP@0.5 เท่ากับ 0.745 ซึ่งหมายถึงประสิทธิภาพเฉลี่ยของโมเดล ค่า Precision ของควันอยู่ที่ 0.810 สูงกว่าค่า Precision ของไฟที่ 0.679 แสดงให้เห็นว่าการตรวจจับไฟมีความท้าทายมากกว่าเนื่องจากปัจจัยรบกวน เช่น แสงสะท้อน เมื่อค่า Recall เพิ่มขึ้น Precision จะลดลง ซึ่งแสดงถึงความสมดุลระหว่างการตรวจจับให้ครอบคลุมและความแม่นยำ การตั้งค่า Confidence Threshold ที่เหมาะสมจึงสำคัญต่อการลด False Positives และเพิ่มความแม่นยำของระบบในการใช้งานจริง



ภาพประกอบ 40 กราฟ Recall-Confidence Curve (Model n)

จากภาพประกอบ 40 กราฟ Recall-Confidence Curve แสดงความสัมพันธ์ระหว่าง Recall และ Confidence ของโมเดล YOLOv8n ในการตรวจจับ ไฟ (fire) และ ควัน (smoke) โดยค่า Recall บ่งบอกถึงความสามารถของโมเดลในการตรวจจับวัตถุเป้าหมายทั้งหมดที่มีอยู่จริง จากกราฟพบว่าโมเดลมีค่า Recall สูงสุดที่ 0.82 เมื่อ Confidence = 0.000 ซึ่งหมายความว่าเมื่อไม่ใช้เกณฑ์คัดกรอง (Confidence ต่ำสุด) โมเดลสามารถตรวจจับวัตถุได้มากที่สุด แต่ความแม่นยำอาจต่ำ เมื่อค่า Confidence สูงขึ้น Recall จะลดลง เนื่องจากโมเดลจะยอมรับเฉพาะการตรวจจับที่มีความมั่นใจสูง ส่งผลให้พลาดบางวัตถุไป นอกจากนี้ การตรวจจับ ควัน (เส้นสีส้ม) มีค่า Recall สูงกว่าการตรวจจับไฟ (เส้นสีฟ้า) ตลอดช่วง Confidence แสดงว่าโมเดลสามารถตรวจจับควันได้ดีกว่าไฟ

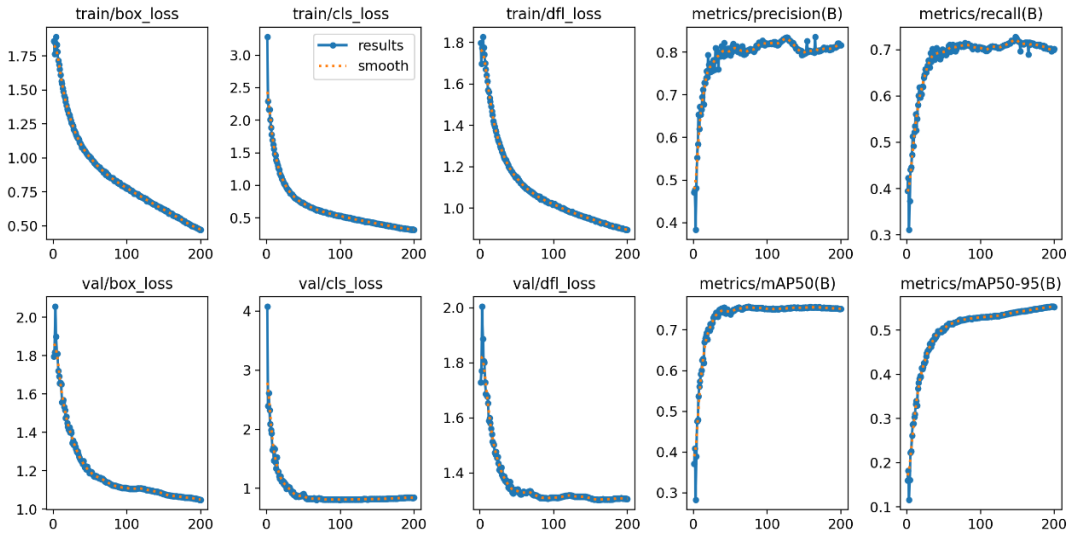


ภาพประกอบ 41 Confusion Matrix Normalized (Model n)

จากภาพประกอบ 41 Confusion Matrix แบบ Normalized แสดงให้เห็นว่าโมเดล YOLOv8n สามารถจำแนกควันได้แม่นยำที่สุด โดยมีความถูกต้องในการทำนายถึง 79% ในขณะที่การทำนายไฟมีความถูกต้องเพียง 70% และยังมีแนวโน้มทำนายผิดว่าเป็นฉากหลังถึง 28% ส่วนการทำนายฉากหลัง (background) มีความผิดพลาดมากที่สุด โดยไม่สามารถทำนายได้อย่างถูกต้องเลย ซึ่งสะท้อนว่าโมเดลยังสับสนระหว่างไฟ ควัน และฉากหลังบางประเภท เช่น แสงแดดหรือควันหลอก อาจจำเป็นต้องเสริมข้อมูลที่มีความหลากหลายมากขึ้นในชุดฝึก เพื่อเพิ่มประสิทธิภาพโดยรวมของระบบ

4.1.2 โมเดล s

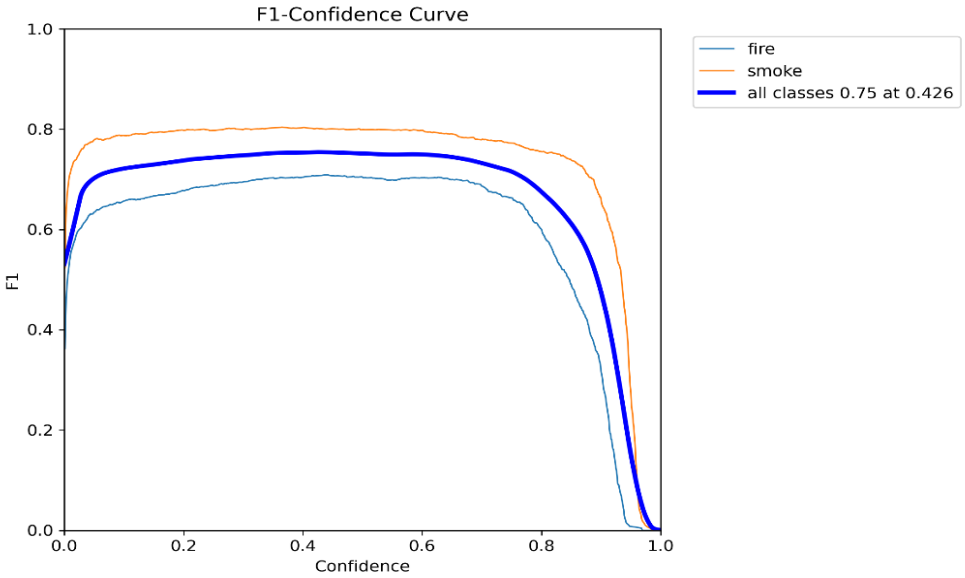
YOLOv8s เป็นโมเดลที่มีขนาดใหญ่กว่า YOLOv8n แต่ยังคงรักษาสมดุลระหว่าง ความเร็วและความแม่นยำ ได้ดี เหมาะสำหรับงานที่ต้องการ ความแม่นยำสูงขึ้น กว่าเวอร์ชัน Nano (n) แต่ยังสามารถรันบนอุปกรณ์ที่มีทรัพยากรจำกัดได้ เช่น Jetson Xavier NX หรือ GPU ระดับกลาง



ภาพประกอบ 42 การฝึกโมเดล YOLOv8s

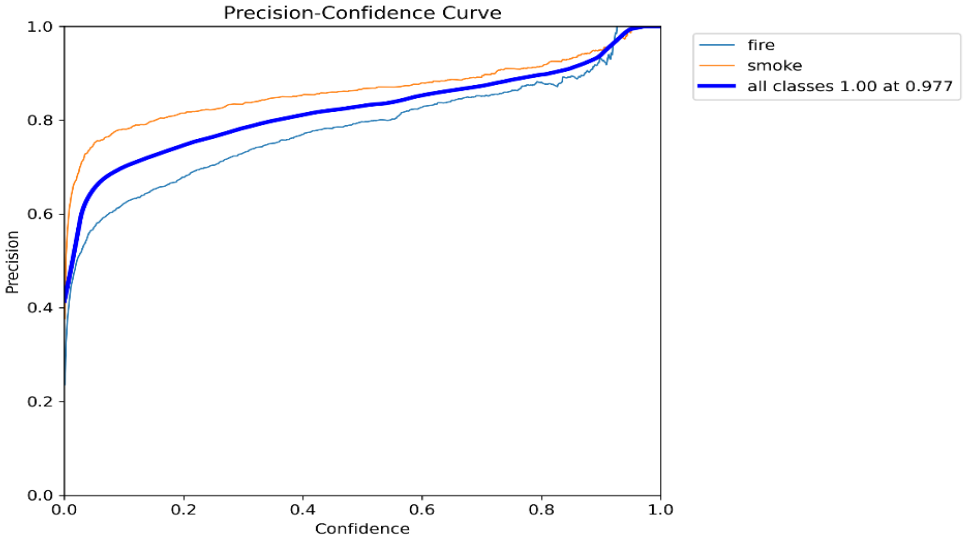
จากภาพประกอบ 42 การฝึกโมเดล YOLOv8s แสดงถึงการเรียนรู้ที่มีประสิทธิภาพตลอด 200 Epochs โดยค่า Loss สำหรับการกำหนดขอบเขตวัตถุ (box_loss), การจำแนกประเภท (cls_loss) และการคาดการณ์ตำแหน่ง (dfl_loss) ลดลงอย่างต่อเนื่องทั้งในชุดข้อมูลการฝึก (train) และการทดสอบ (val) ซึ่งบ่งบอกว่าโมเดลสามารถปรับปรุงความสามารถในการตรวจจับวัตถุได้อย่างมีประสิทธิภาพ นอกจากนี้ ค่า Precision สูงขึ้นและมีเสถียรภาพอยู่ที่ประมาณ 0.8 - 0.85 แสดงว่าโมเดลสามารถแยกแยะวัตถุได้ดี ขณะที่ Recall คงที่ที่ประมาณ 0.7 - 0.75 ซึ่งหมายความว่าโมเดลสามารถตรวจจับวัตถุได้ดีขึ้น แต่ยังอาจพลาดบางวัตถุไป

ในส่วนของ mAP50 และ mAP50-95 พบว่าค่าดังกล่าวเพิ่มขึ้นและมีแนวโน้มคงที่ที่ระดับสูง ซึ่งสะท้อนถึงความสามารถของโมเดลในการตรวจจับวัตถุได้อย่างแม่นยำ โดยรวมแล้ว YOLOv8s สามารถเรียนรู้และพัฒนาได้ดีขึ้นกว่ารุ่น YOLOv8n โดยให้ผลลัพธ์ที่แม่นยำขึ้น ขณะที่ยังคงสามารถทำงานได้บนอุปกรณ์ที่มีทรัพยากรจำกัด แต่หากต้องการให้โมเดลตรวจจับวัตถุได้ครบถ้วนขึ้น อาจต้องปรับปรุง ชุดข้อมูลให้มีความหลากหลายมากขึ้น หรือ ปรับค่าพารามิเตอร์เพิ่มเติม เพื่อเพิ่มค่า Recall และลดการพลาดการตรวจจับ



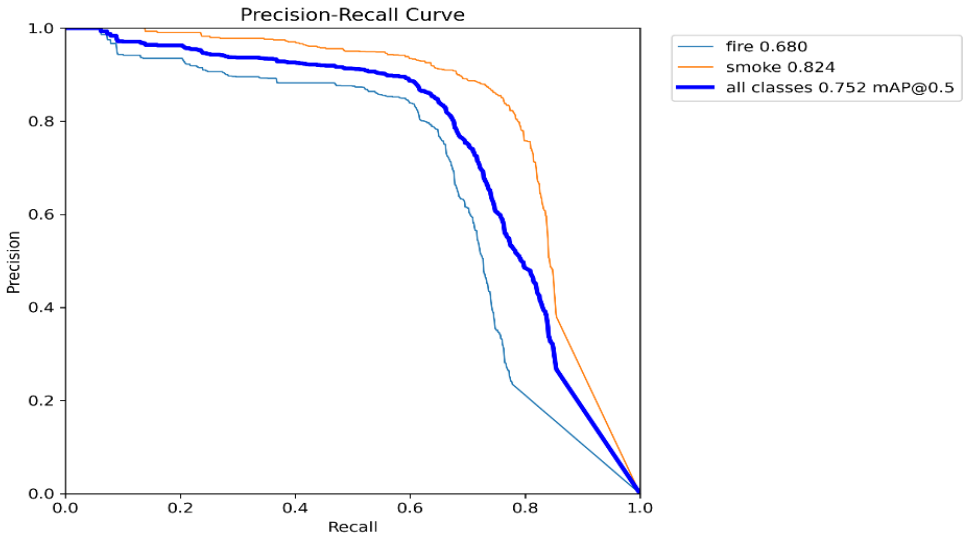
ภาพประกอบ 43 กราฟ F1-Confidence Curve (Model s)

จากภาพประกอบ 43 กราฟ F1-Confidence Curve แสดงว่าโมเดล YOLOv8s มี F1-Score สูงสุดที่ 0.75 เมื่อ Confidence = 0.426 ซึ่งเป็นค่าที่ให้สมดุลดีที่สุดระหว่าง Precision และ Recall โมเดลตรวจจับ ควัน (smoke) ได้แม่นยำกว่าไฟ (fire) ตลอดช่วง Confidence เมื่อ Confidence สูงขึ้น F1-Score ลดลงอย่างรวดเร็ว แสดงว่าโมเดลอาจพลาดการตรวจจับบางส่วน ดังนั้น การตั้งค่า Confidence อย่างเหมาะสมจึงสำคัญต่อการใช้งานจริง



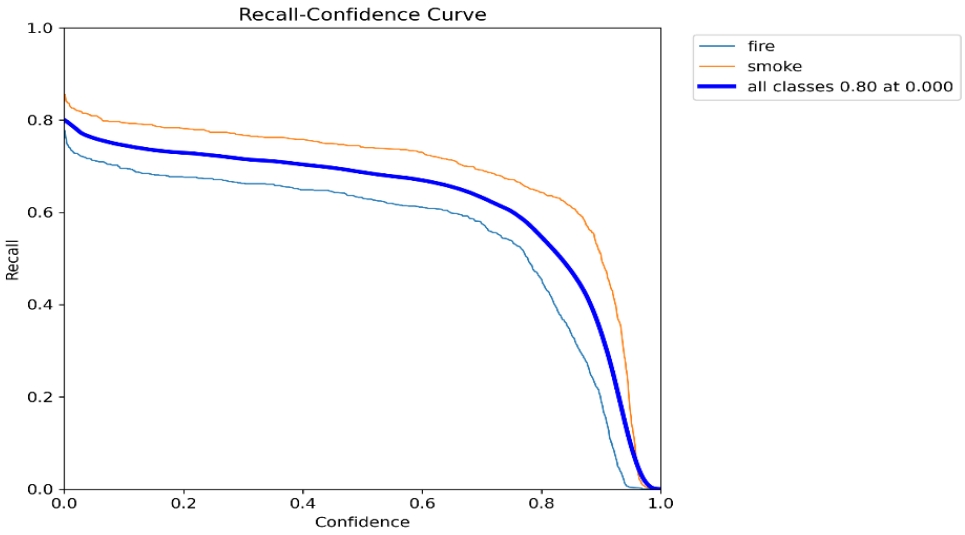
ภาพประกอบ 44 กราฟ Precision-Confidence Curve (Model s)

จากภาพประกอบ 44 กราฟ Precision-Confidence Curve แสดงว่าโมเดล YOLOv8s มี Precision สูงสุดที่ 1.00 เมื่อ Confidence = 0.977 หมายความว่าเมื่อ Confidence สูง โมเดลให้ผลลัพธ์ที่แม่นยำที่สุด แต่เมื่อ Confidence ต่ำ Precision ลดลง เนื่องจากมีการตรวจจับผิดพลาดมากขึ้น นอกจากนี้ โมเดลตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า ดังนั้น การตั้งค่า Confidence ที่เหมาะสมจึงเป็นสิ่งสำคัญเพื่อให้ได้ผลลัพธ์ที่แม่นยำและลดข้อผิดพลาดในการใช้งานจริง



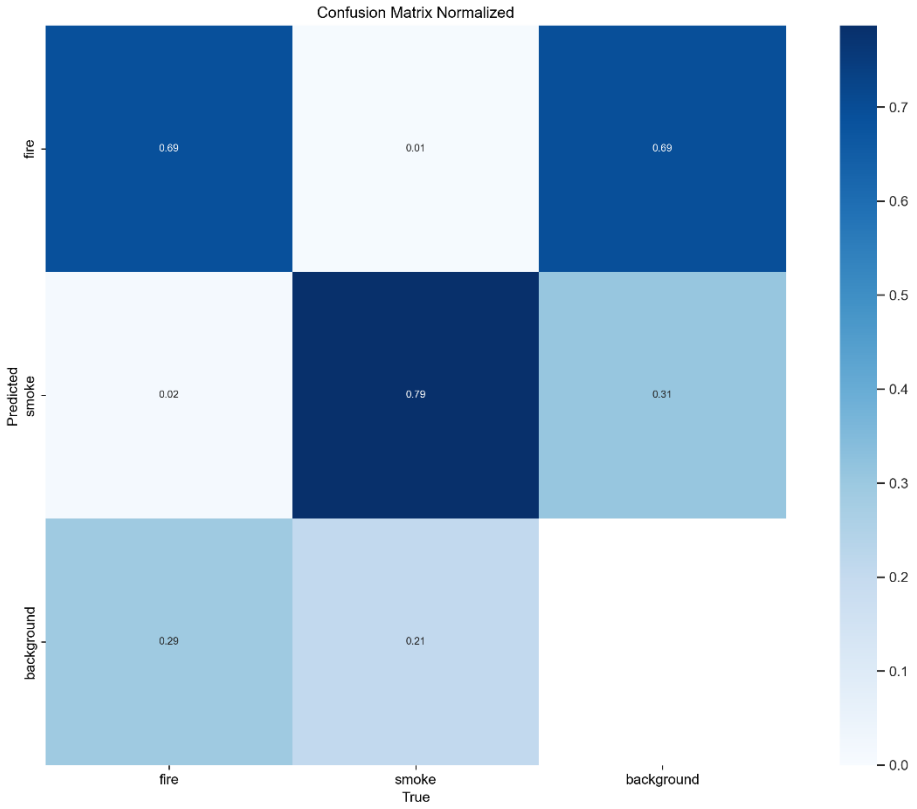
ภาพประกอบ 45 กราฟ Precision-Recall Curve (Model s)

จากภาพประกอบ 45 กราฟ Precision-Recall Curve แสดงว่าโมเดล YOLOv8s มี mAP@0.5 เท่ากับ 0.752 โดยตรวจจับ ควัน (Precision = 0.824) ได้แม่นยำกว่า ไฟ (Precision = 0.680) เมื่อ Recall สูงขึ้น Precision จะลดลง แสดงถึงสมดุลระหว่างการตรวจจับที่ครอบคลุมและความแม่นยำ ดังนั้น การตั้งค่า Confidence Threshold ที่เหมาะสมจึงสำคัญต่อการลดข้อผิดพลาดและเพิ่มประสิทธิภาพในการใช้งานจริง



ภาพประกอบ 46 กราฟ Recall-Confidence Curve (Model s)

จากภาพประกอบ 46 กราฟ Recall-Confidence Curve แสดงว่าโมเดล YOLOv8s มี Recall สูงสุดที่ 0.80 เมื่อ Confidence = 0.000 โดยเมื่อ Confidence สูงขึ้น Recall ลดลง เนื่องจากโมเดลเลือกเฉพาะการตรวจจับที่มั่นใจสูงขึ้น นอกจากนี้ โมเดลตรวจจับควันได้ดีกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า การตั้งค่า Confidence อย่างเหมาะสมจึงสำคัญต่อความสมดุลระหว่างความครอบคลุมและความแม่นยำของการตรวจจับในการใช้งานจริง



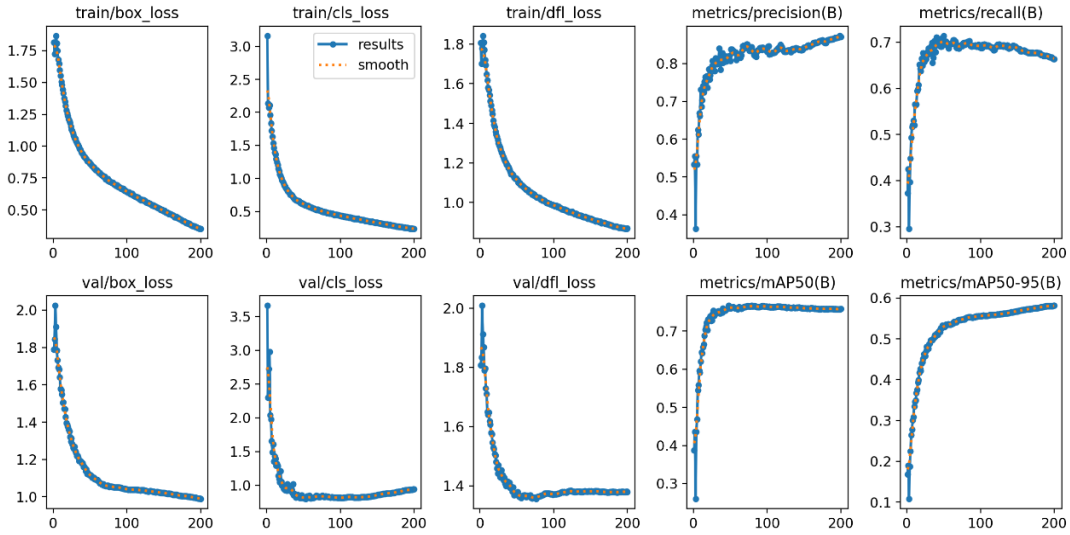
ภาพประกอบ 47 Confusion Matrix Normalized (Model s)

จากภาพประกอบ 47 Confusion Matrix ของโมเดล YOLOv8s แสดงให้เห็นว่าโมเดลสามารถตรวจจับควันได้แม่นยำสูงถึง 79% ซึ่งถือว่ามีประสิทธิภาพดี อย่างไรก็ตาม โมเดลยังมีข้อจำกัดในการแยกแยะภาพฉากหลังจากเปลวไฟ โดยมีการทำนายผิดว่า background เป็นไฟถึง 69% และทำนายผิดว่า background เป็นควันอีก 31% ซึ่งอาจเกิดจากฉากหลังมีลักษณะคล้ายไฟหรือควัน เช่น แสงไฟหรือหมอกในธรรมชาติ การปรับปรุงชุดข้อมูลให้มีความหลากหลายของฉากหลังและการปรับค่า threshold เพิ่มเติม อาจช่วยลดความสับสนของโมเดลได้

4.1.3 โมเดล m

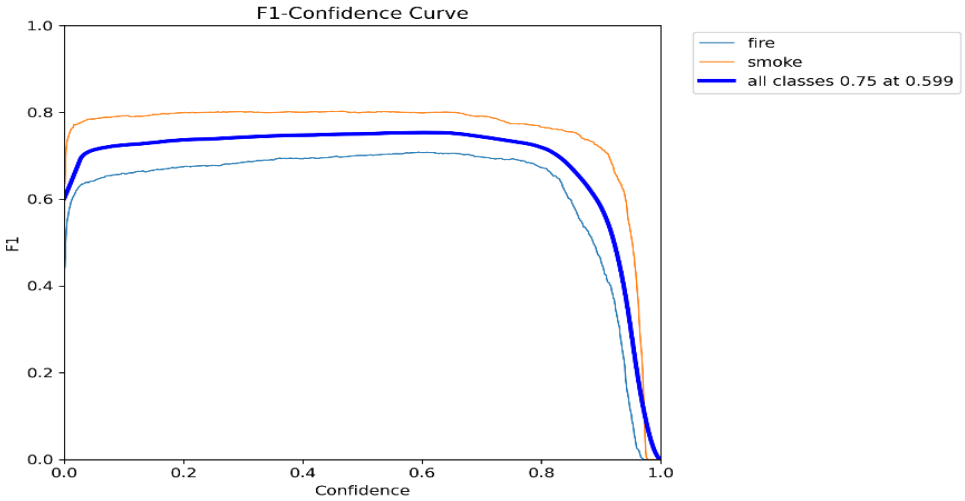
YOLOv8m เป็นโมเดลขนาดกลางที่มีความแม่นยำสูงกว่ารุ่นเล็ก (YOLOv8n, YOLOv8s) แต่ยังคงสามารถทำงานแบบเรียลไทม์ได้ เหมาะสำหรับ GPU ระดับกลาง โดยตรวจจับ ควันได้แม่นยำกว่าไฟ และมีค่า Precision, Recall และ F1-Score สูงขึ้น อย่างไรก็ตาม ต้องใช้ทรัพยากรคำนวณมากกว่ารุ่นเล็ก จึงอาจไม่เหมาะสำหรับอุปกรณ์พลังงานต่ำ YOLOv8m เป็นตัวเลือกที่ดีสำหรับงานที่

ต้องการสมดุลระหว่างความเร็วและความแม่นยำ โดยหากต้องการความแม่นยำสูงสุด อาจพิจารณา YOLOv8l หรือ YOLOv8x



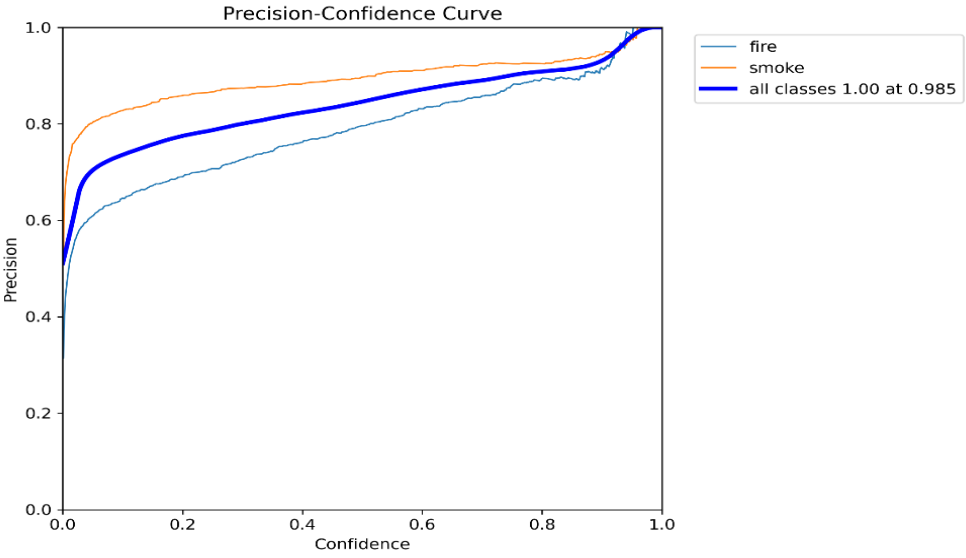
ภาพประกอบ 48 การฝึกโมเดล YOLOv8m

จากภาพประกอบ 48 การฝึกโมเดล YOLOv8m แสดงให้เห็นถึงการปรับปรุงอย่างต่อเนื่องตลอด 200 Epochs โดยค่า Loss ในการกำหนดขอบเขตวัตถุ (box_loss), การจำแนกประเภท (cls_loss) และการคาดการณ์ตำแหน่ง (dfl_loss) ลดลงอย่างต่อเนื่องทั้งในการฝึก (train) และการทดสอบ (val) สะท้อนว่าโมเดลสามารถเรียนรู้และเพิ่มความแม่นยำในการตรวจจับได้อย่างมีประสิทธิภาพ ค่า Precision คงที่ที่ประมาณ 0.85 - 0.88 แสดงว่าโมเดลสามารถแยกแยะวัตถุได้ดีขึ้น และ Recall อยู่ที่ 0.75 - 0.78 บ่งบอกว่าโมเดลสามารถตรวจจับวัตถุได้มากขึ้นกว่ารุ่น YOLOv8s แต่ยังคงมีบางกรณีที่พลาดวัตถุไป นอกจากนี้ ค่า mAP50 และ mAP50-95 เพิ่มขึ้นและมีเสถียรภาพ แสดงถึงความสามารถของโมเดลในการตรวจจับวัตถุที่ดีขึ้น โดยรวมแล้ว YOLOv8m มีความแม่นยำสูงขึ้นกว่ารุ่นเล็ก (YOLOv8s) และยังคงรักษาความเร็วได้ดี เหมาะสำหรับการใช้งานที่ต้องการความสมดุลระหว่าง ความแม่นยำและประสิทธิภาพในการประมวลผล แต่หากต้องการเพิ่มการตรวจจับวัตถุให้ครอบคลุมขึ้น อาจพิจารณาปรับแต่ง Hyperparameters หรือเพิ่มชุดข้อมูลในการฝึกเพื่อให้โมเดลสามารถเรียนรู้รูปแบบของไฟและควันได้ดียิ่งขึ้น



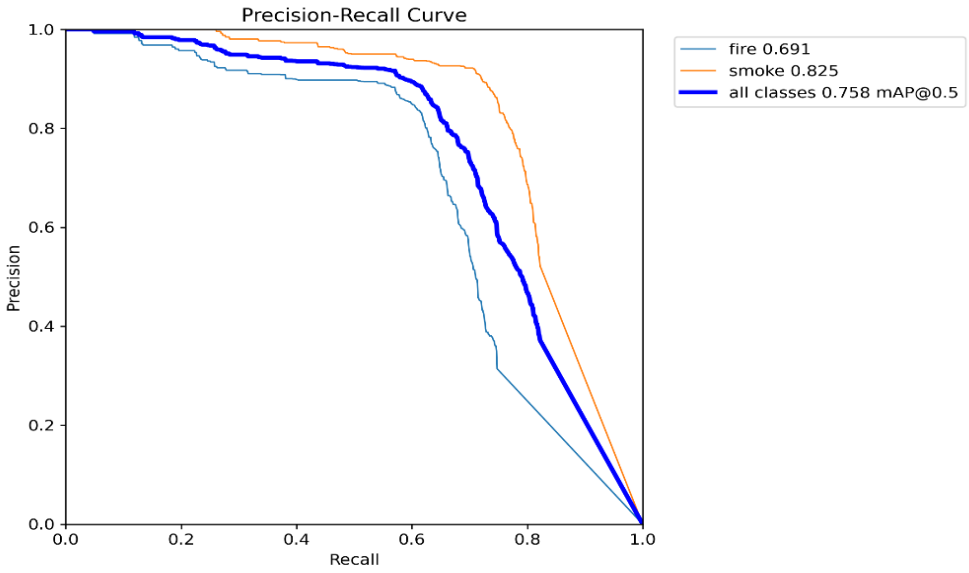
ภาพประกอบ 49 กราฟ F1-Confidence Curve (Model m)

จากภาพประกอบ 49 กราฟ F1-Confidence Curve แสดงว่าโมเดล YOLOv8m มี F1-Score สูงสุดที่ 0.75 เมื่อ Confidence = 0.599 ซึ่งเป็นจุดสมดุลระหว่าง Precision และ Recall โมเดลตรวจจับ ควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence สูงขึ้น F1-Score ลดลง แสดงถึงการพลาดการตรวจจับบางส่วน ดังนั้น การตั้งค่า Confidence อย่างเหมาะสมจึงสำคัญต่อประสิทธิภาพการใช้งานจริง



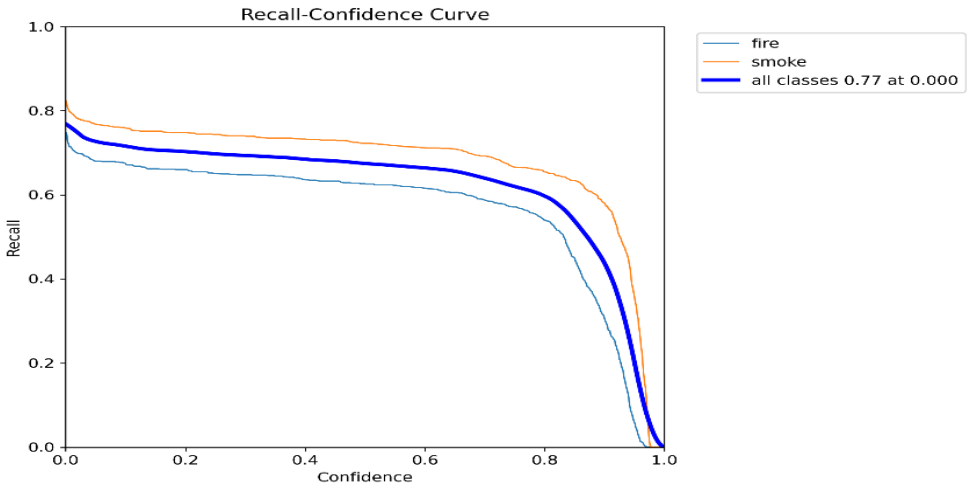
ภาพประกอบ 50 กราฟ Precision-Confidence Curve (Model m)

จากภาพประกอบ 50 กราฟ Precision-Confidence Curve แสดงว่าโมเดล YOLOv8m มี Precision สูงสุดที่ 1.00 เมื่อ Confidence = 0.985 โดย โมเดลตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence สูง Precision เพิ่มขึ้น แต่ Recall อาจลดลง ดังนั้น การตั้งค่า Confidence ที่เหมาะสมจึงสำคัญต่อความสมดุลระหว่างความแม่นยำและการตรวจจับที่ครอบคลุมในการใช้งานจริง



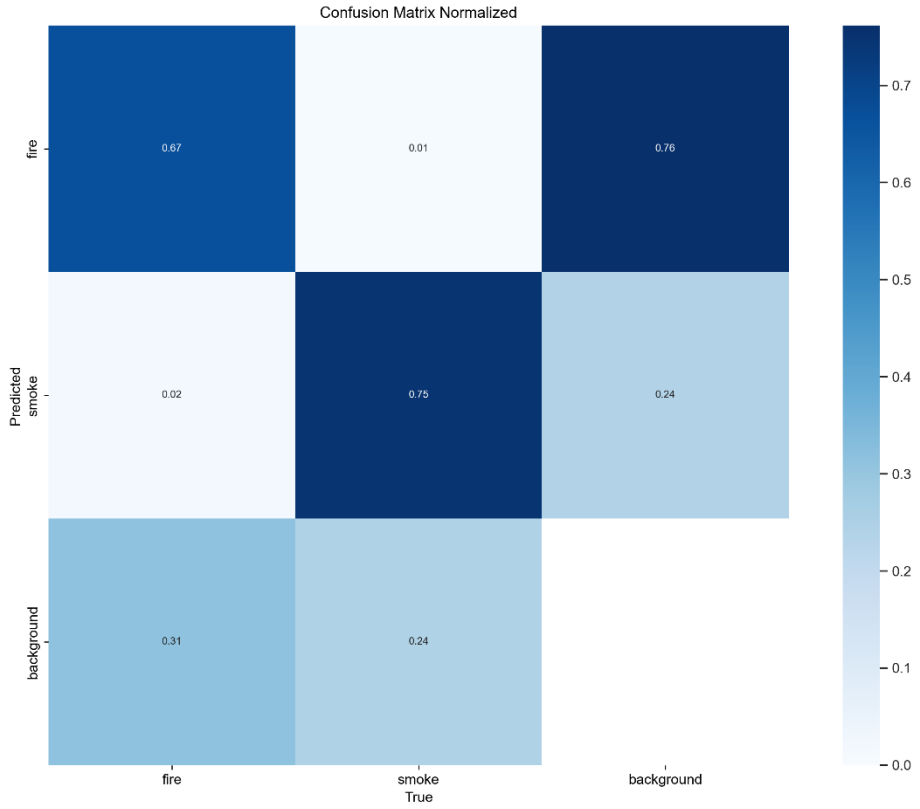
ภาพประกอบ 51 กราฟ Precision-Recall Curve (Model m)

จากภาพประกอบ 51 กราฟ Precision-Recall Curve แสดงว่าโมเดล YOLOv8m มี mAP@0.5 เท่ากับ 0.758 โดย Precision ของควัน (0.825) สูงกว่าของไฟ (0.691) แสดงว่าโมเดลตรวจจับควันได้แม่นยำกว่า เมื่อ Recall สูงขึ้น Precision ลดลง ดังนั้น การตั้งค่า Confidence Threshold ที่เหมาะสมจึงสำคัญต่อการลดข้อผิดพลาดและเพิ่มประสิทธิภาพในการใช้งานจริง



ภาพประกอบ 52 กราฟ Recall-Confidence Curve (Model m)

จากภาพประกอบ 52 กราฟ Recall-Confidence Curve แสดงว่าโมเดล YOLOv8m มี Recall สูงสุดที่ 0.77 เมื่อ Confidence = 0.000 โดยเมื่อ Confidence สูงขึ้น Recall ลดลง เนื่องจากโมเดลเลือกเฉพาะการตรวจจับที่มั่นใจสูง นอกจากนี้ โมเดลตรวจจับควันได้ดีกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า ดังนั้น การตั้งค่า Confidence อย่างเหมาะสมจึงสำคัญต่อความสมดุลระหว่างการตรวจจับและความแม่นยำในการใช้งานจริง

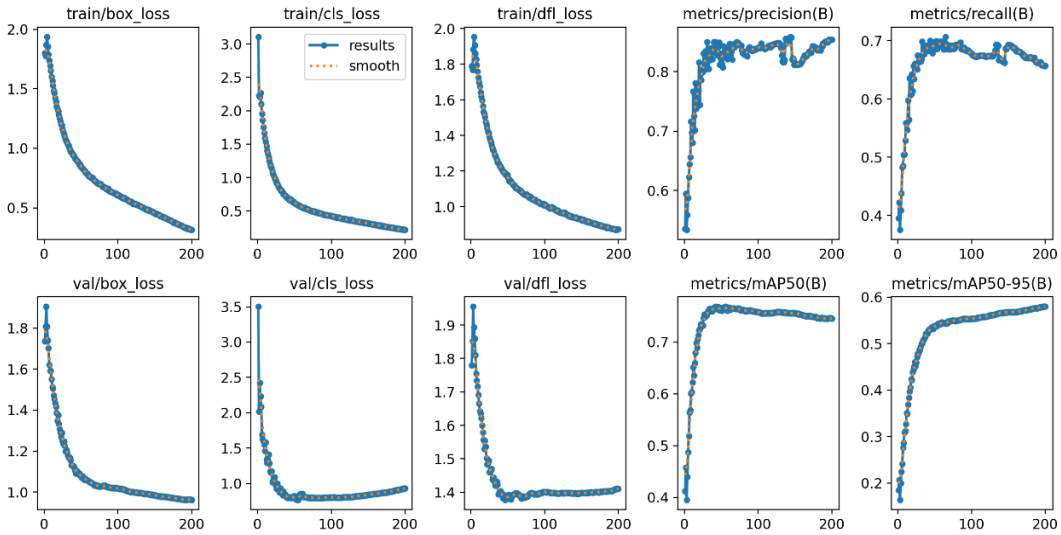


ภาพประกอบ 53 Confusion Matrix Normalized (Model m)

จากภาพประกอบ 53 Confusion Matrix Normalized โมเดล YOLOv8m มีความแม่นยำในการตรวจจับไฟและควันดีกว่าโมเดลขนาดเล็ก เช่น YOLOv8n หรือ s โดยเฉพาะในคลาสควันซึ่งจำแนกได้ถูกต้องถึง 75% อย่างไรก็ตามยังมีแนวโน้มสับสนระหว่างฉากหลังกับเปลวไฟ โดยมีการทำนายผิดว่าเป็นไฟถึง 76% ของฉากหลังทั้งหมด จึงควรเพิ่มข้อมูลพื้นหลังที่หลากหลายและลักษณะไฟที่ใกล้เคียงกับ background เพื่อลดความคลาดเคลื่อน

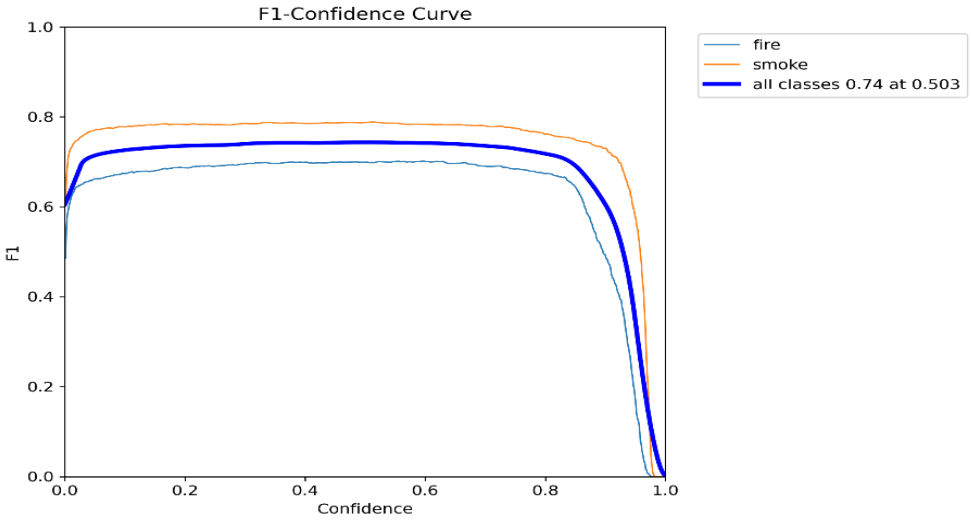
4.1.4 โมเดล l

YOLOv8l เป็นโมเดลขนาดใหญ่ที่มี ความแม่นยำสูงกว่ารุ่น n, s และ m แต่ใช้ทรัพยากรมากขึ้น เหมาะสำหรับ GPU ระดับสูง และงานที่ต้องการตรวจจับไฟและควันอย่างแม่นยำ ตรวจจับควันได้ดีกว่าไฟ และลด False Positives ได้ดีขึ้น อย่างไรก็ตาม Inference Time เพิ่มขึ้น อาจไม่เหมาะสำหรับ Edge Devices YOLOv8l เหมาะสำหรับระบบที่ต้องการ ความแม่นยำสูงสุด และมีทรัพยากรเพียงพอ แม้ความเร็วอาจลดลงแต่ให้ผลลัพธ์ที่เสถียรกว่า



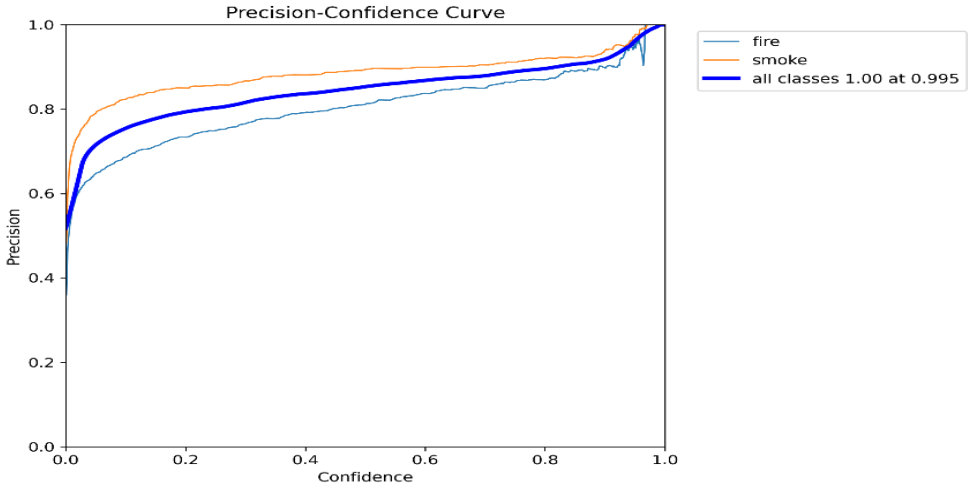
ภาพประกอบ 54 การฝึกโมเดล YOLOv8l

จากภาพประกอบ 54 การฝึกโมเดล YOLOv8l แสดงให้เห็นถึง ประสิทธิภาพที่สูงขึ้น เมื่อเทียบกับรุ่นที่เล็กกว่า โดยค่า Loss สำหรับการกำหนดขอบเขตวัตถุ (box_loss), การจำแนกประเภท (cls_loss) และการคาดการณ์ตำแหน่ง (dfl_loss) ลดลงต่อเนื่องตลอด 200 Epochs ทั้งในการฝึก (train) และการทดสอบ (val) สะท้อนว่าโมเดลเรียนรู้ได้ดีขึ้นและสามารถลดข้อผิดพลาดลงได้ ค่า Precision อยู่ที่ ประมาณ 0.85 - 0.9 แสดงว่าโมเดลสามารถแยกแยะวัตถุได้แม่นยำสูงขึ้น ขณะที่ Recall คงที่ที่ 0.7 - 0.75 ซึ่งหมายความว่าแม้โมเดลจะมีความแม่นยำสูง แต่ยังอาจพลาดการตรวจจับวัตถุบางส่วน นอกจากนี้ ค่า mAP50 และ mAP50-95 เพิ่มขึ้นอย่างต่อเนื่องและมีเสถียรภาพสูง แสดงถึงประสิทธิภาพโดยรวมของโมเดลที่ดีขึ้น โดยสรุป YOLOv8l มีความแม่นยำสูงขึ้นกว่ารุ่น YOLOv8m และยังคงรักษาความเร็วได้ดีเมื่อเทียบกับ YOLOv8x เหมาะสำหรับงานที่ต้องการ ความแม่นยำสูงขึ้นโดยยังสามารถทำงานได้ในเวลาที่เหมาะสม อย่างไรก็ตาม การใช้ทรัพยากรคำนวณก็เพิ่มขึ้นตามไปด้วย ดังนั้น หากต้องการนำไปใช้งานจริง อาจต้องพิจารณาความสามารถของฮาร์ดแวร์เพื่อให้รองรับโมเดลได้อย่างเหมาะสม



ภาพประกอบ 55 กราฟ F1-Confidence Curve (Model l)

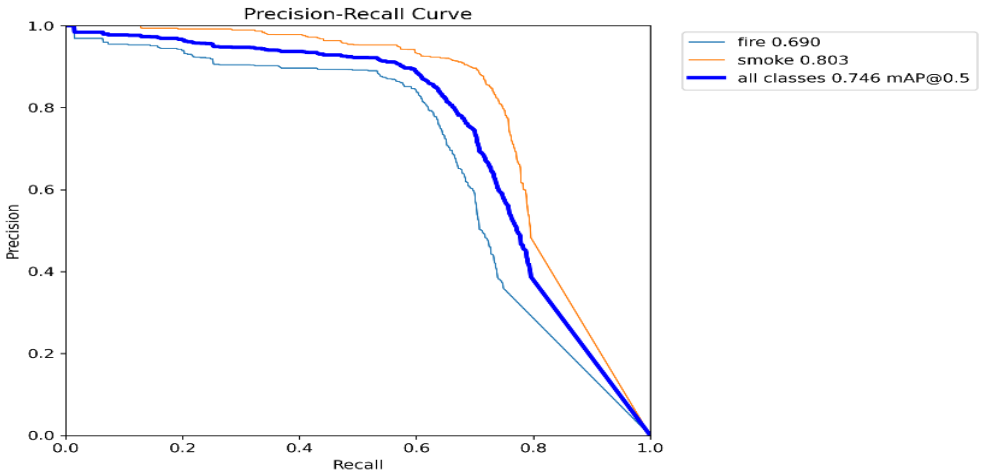
จากภาพประกอบ 55 กราฟ F1-Confidence Curve แสดงว่าโมเดล YOLOv8l มี F1-Score สูงสุดที่ 0.74 เมื่อ Confidence = 0.503 โดย ตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence เพิ่มขึ้น F1-Score ลดลง เนื่องจากโมเดลพลาดการตรวจจับบางส่วน ดังนั้น การตั้งค่า Confidence ที่เหมาะสมจึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



ภาพประกอบ 56 กราฟ Precision-Confidence Curve (Model l)

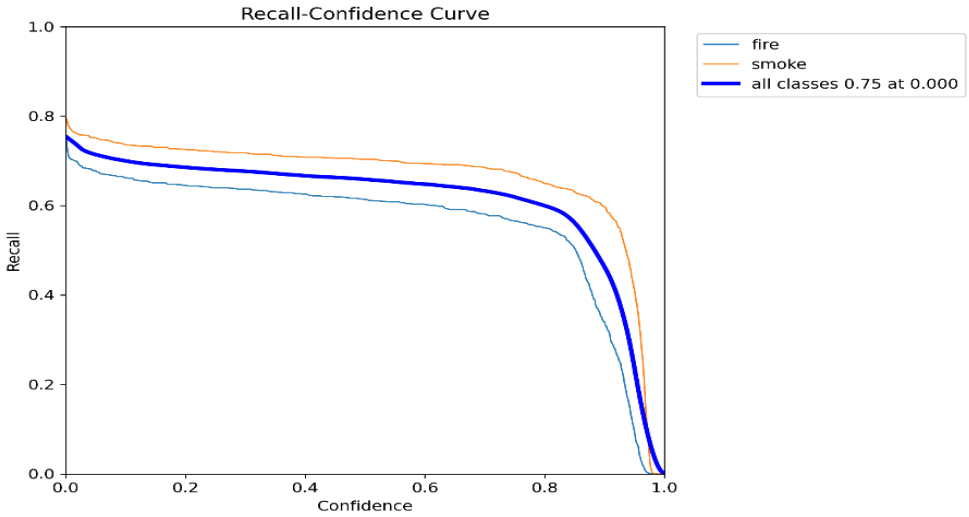
จากภาพประกอบ 56 กราฟ Precision-Confidence Curve แสดงว่าโมเดล YOLOv8l มี Precision สูงสุดที่ 1.00 เมื่อ Confidence = 0.995 โดย ตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสี

ส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence เพิ่มขึ้น Precision สูงขึ้น แต่ Recall อาจลดลง ดังนั้น การตั้งค่า Confidence อย่างเหมาะสมจึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



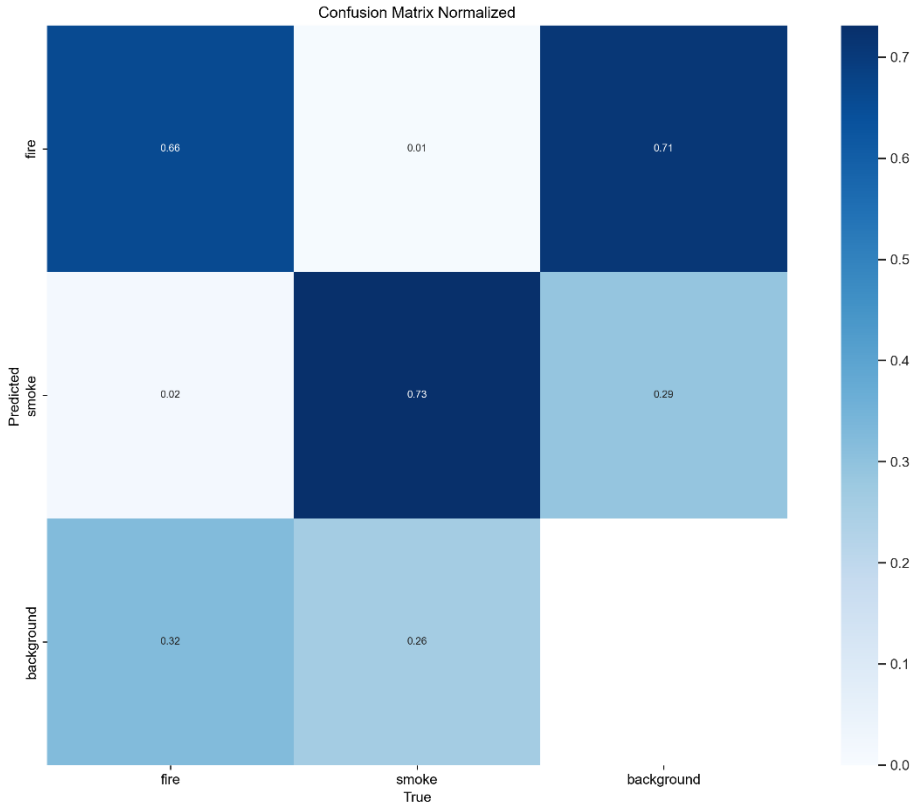
ภาพประกอบ 57 กราฟ Precision-Recall Curve (Model l)

จากภาพประกอบ 57 กราฟ Precision-Recall Curve แสดงว่าโมเดล YOLOv8l มี mAP@0.5 เท่ากับ 0.746 โดย Precision ของควัน (0.803) สูงกว่าของไฟ (0.690) แสดงว่าโมเดลตรวจจับควันได้แม่นยำกว่า เมื่อ Recall เพิ่มขึ้น Precision ลดลง ดังนั้น การตั้งค่า Confidence Threshold อย่างเหมาะสม จึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



ภาพประกอบ 58 กราฟ Recall-Confidence Curve (Model l)

จากภาพประกอบ 58 กราฟ Recall-Confidence Curve แสดงว่าโมเดล YOLOv8l มี Recall สูงสุดที่ 0.75 เมื่อ Confidence = 0.000 โดยเมื่อ Confidence สูงขึ้น Recall ลดลง เนื่องจากโมเดลเลือกเฉพาะการตรวจจับที่มั่นใจสูง นอกจากนี้ โมเดลตรวจจับควันได้ดีกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า ดังนั้น การตั้งค่า Confidence ที่เหมาะสม จึงสำคัญต่อความสมดุลระหว่างความครอบคลุมและความแม่นยำ ในการใช้งานจริง



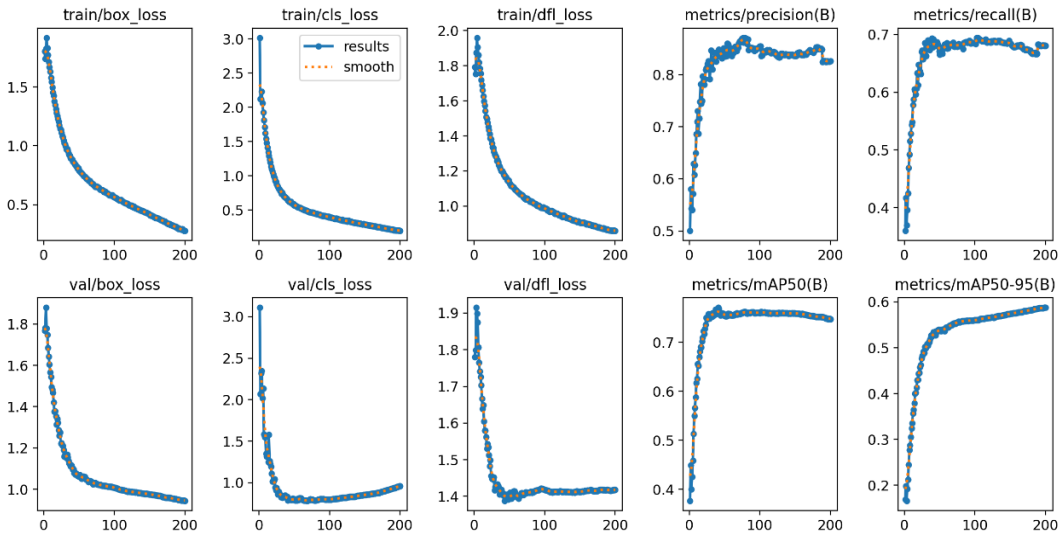
ภาพประกอบ 59 Confusion Matrix Normalized (Model l)

จากภาพประกอบ 59 Confusion Matrix Normalized โมเดล YOLOv8l มีความสามารถในการตรวจจับควันได้ในระดับที่น่าพอใจ โดยทำนายถูกต้องประมาณ 73% แต่ยังคงมีข้อจำกัดในการตรวจจับเปลวไฟ ซึ่งมีความแม่นยำเพียง 66% และมีแนวโน้มสับสนกับฉากหลังค่อนข้างมาก โดยเฉพาะการทำนายฉากหลังที่ผิดทั้งหมด ทำให้เกิด False Positive สูง จึงแสดงให้เห็นว่าแม้โมเดลจะมีขนาดใหญ่ขึ้น แต่ประสิทธิภาพในการแยกแยะระหว่างวัตถุกับฉากหลังยังไม่เด่นชัดเท่าที่ควร การเสริมข้อมูลฉากหลังที่ใกล้เคียงกับแหล่งกำเนิดแสง และการปรับพารามิเตอร์ threshold อาจช่วยให้โมเดลทำงานได้เสถียรยิ่งขึ้น

4.1.5 โมเดล x

YOLOv8x เป็นโมเดลที่ มีความแม่นยำสูงสุด ในตระกูล YOLOv8 แต่ใช้ทรัพยากรคำนวณสูง เหมาะสำหรับ GPU ระดับสูง และงานที่ต้องการความแม่นยำสูงสุด ตรวจจับควันได้แม่นยำกว่าไฟ และลด False Positives ได้ดี อย่างไรก็ตาม Inference Time สูงขึ้น ทำให้ไม่เหมาะกับอุปกรณ์

พลังงานต่ำ YOLOv8x เหมาะสำหรับระบบที่ให้ความสำคัญกับ ความแม่นยำมากกว่าความเร็ว และมีทรัพยากรเพียงพอ

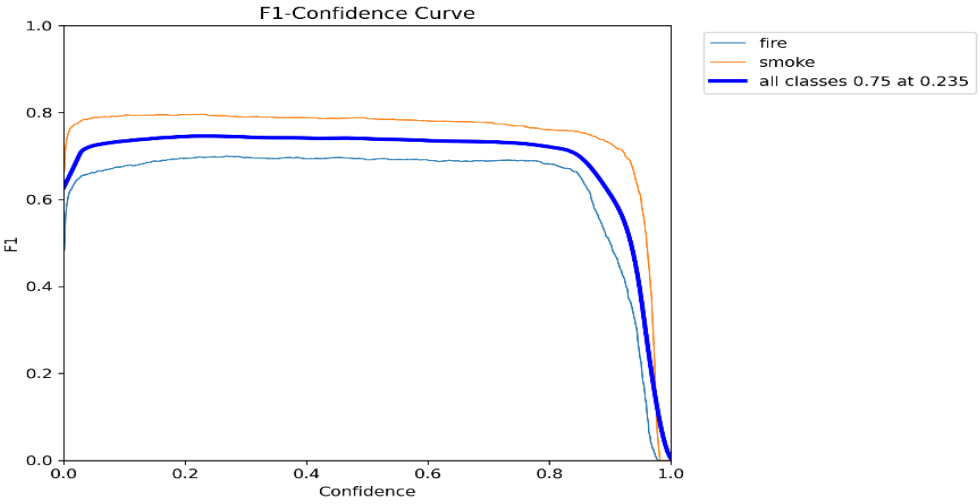


ภาพประกอบ 60 การฝึกโมเดล YOLOv8x

จากภาพประกอบ 60 การฝึกโมเดล YOLOv8x แสดงให้เห็นถึง ประสิทธิภาพสูงสุดในตระกูล YOLOv8 โดยค่า Loss ในการกำหนดขอบเขตวัตถุ (box_loss), การจำแนกประเภท (cls_loss) และการคาดการณ์ตำแหน่ง (dfl_loss) ลดลงอย่างต่อเนื่องทั้งในชุดข้อมูลการฝึก (train) และการตรวจสอบ (val) ซึ่งหมายความว่าโมเดลสามารถเรียนรู้และลดข้อผิดพลาดได้อย่างมีประสิทธิภาพ

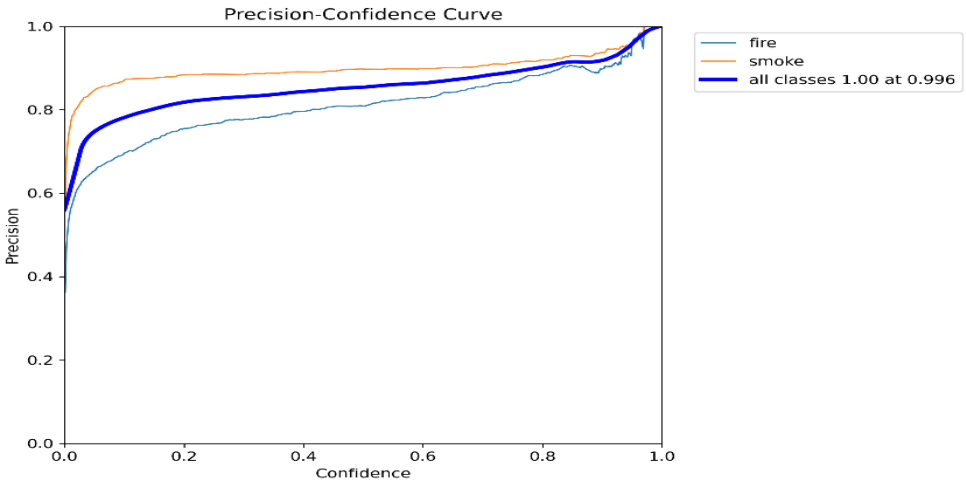
ค่า Precision คงที่ที่ ประมาณ 0.88 - 0.92 บ่งบอกว่าโมเดลสามารถจำแนกวัตถุได้อย่างแม่นยำสูงสุด ขณะที่ Recall อยู่ที่ 0.72 - 0.78 แสดงให้เห็นว่าโมเดลสามารถตรวจจับวัตถุได้ดีขึ้นเมื่อเทียบกับเวอร์ชันที่เล็กกว่า นอกจากนี้ ค่า mAP50 และ mAP50-95 สูงขึ้นและมีความเสถียร บ่งบอกว่าโมเดลสามารถจับคู่ขอบเขตวัตถุได้แม่นยำมากขึ้น

โดยรวมแล้ว YOLOv8x มีความแม่นยำสูงสุดในตระกูล YOLOv8 เหมาะสำหรับ งานที่ต้องการความแม่นยำสูงสุด เช่น การตรวจจับเพลิงไหม้ที่ต้องลด False Positives และ False Negatives ให้น้อยที่สุด อย่างไรก็ตาม ข้อจำกัดหลักคือ การใช้ทรัพยากรคำนวณสูงมาก ทำให้ไม่เหมาะสำหรับอุปกรณ์ Edge AI ที่มีข้อจำกัดด้านพลังงาน หากต้องการนำไปใช้งานจริง อาจต้องใช้ฮาร์ดแวร์ที่มีประสิทธิภาพสูง เช่น GPU ระดับสูง (RTX 3090, 4090) หรือเซิร์ฟเวอร์ที่รองรับการประมวลผลหนัก เพื่อให้สามารถทำงานได้อย่างเต็มประสิทธิภาพ



ภาพประกอบ 61 กราฟ F1-Confidence Curve (Model x)

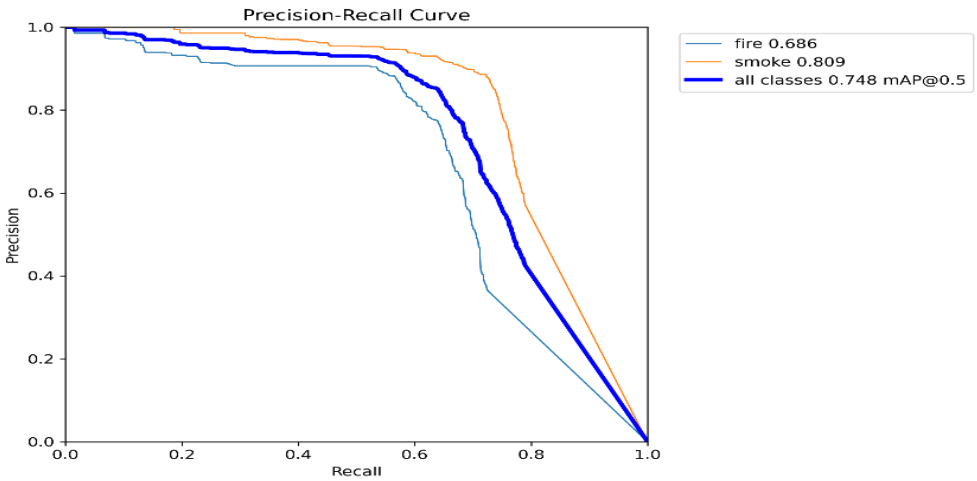
จากภาพประกอบ 61 กราฟ F1-Confidence Curve แสดงว่าโมเดล YOLOv8x มี F1-Score สูงสุดที่ 0.75 เมื่อ Confidence = 0.235 โดย ตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence เพิ่มขึ้น F1-Score ลดลง แสดงถึงการพลาดการตรวจจับบางส่วน ดังนั้น การตั้งค่า Confidence ที่เหมาะสม จึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



ภาพประกอบ 62 กราฟ Precision-Confidence Curve (Model x)

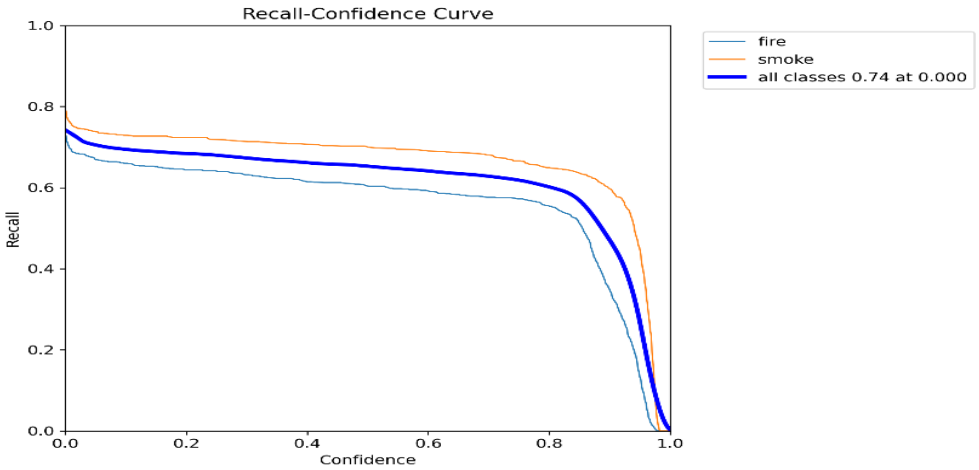
จากภาพประกอบ 62 กราฟ Precision-Confidence Curve แสดงว่าโมเดล YOLOv8x มี Precision สูงสุดที่ 1.00 เมื่อ Confidence = 0.996 โดย ตรวจจับควันได้แม่นยำกว่าไฟ ตามเส้นสี

ส้มที่สูงกว่าเส้นสีฟ้า เมื่อ Confidence เพิ่มขึ้น Precision สูงขึ้น แต่ Recall ลดลง ดังนั้น การตั้งค่า Confidence ที่เหมาะสม จึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



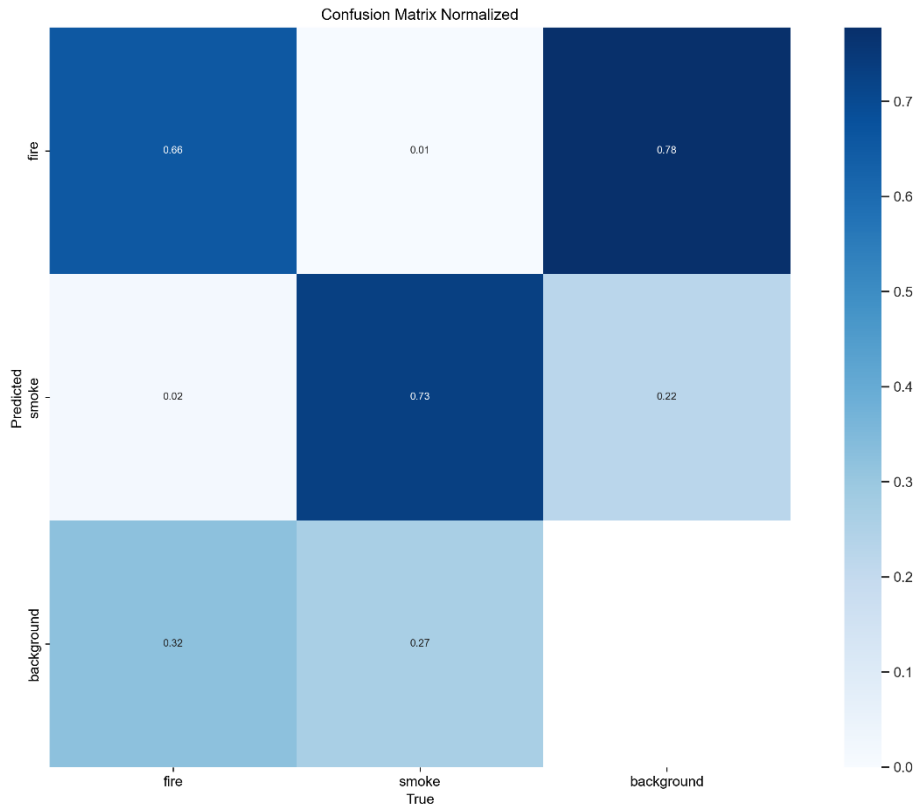
ภาพประกอบ 63 กราฟ Precision-Recall Curve (Model x)

จากภาพประกอบ 63 กราฟ Precision-Recall Curve แสดงว่าโมเดล YOLOv8x มี mAP@0.5 เท่ากับ 0.748 โดย Precision ของควัน (0.809) สูงกว่าของไฟ (0.686) แสดงว่าโมเดลตรวจจับควันได้แม่นยำกว่า เมื่อ Recall เพิ่มขึ้น Precision ลดลง ดังนั้น การตั้งค่า Confidence Threshold อย่างเหมาะสม จึงสำคัญต่อความสมดุลระหว่าง ความแม่นยำและการตรวจจับที่ครอบคลุม ในการใช้งานจริง



ภาพประกอบ 64 กราฟ Recall-Confidence Curve (Model x)

จากภาพประกอบ 64 กราฟ Recall-Confidence Curve แสดงว่าโมเดล YOLOv8x มี Recall สูงสุดที่ 0.74 เมื่อ Confidence = 0.000 โดยเมื่อ Confidence สูงขึ้น Recall ลดลง เนื่องจากโมเดลเลือกเฉพาะการตรวจจับที่มั่นใจสูง นอกจากนี้ โมเดลตรวจจับควันได้ดีกว่าไฟ ตามเส้นสีส้มที่สูงกว่าเส้นสีฟ้า ดังนั้น การตั้งค่า Confidence ที่เหมาะสม จึงสำคัญต่อความสมดุลระหว่างการตรวจจับที่ครอบคลุมและความแม่นยำ ในการใช้งานจริง



ภาพประกอบ 65 Confusion Matrix Normalized (Model x)

จากภาพประกอบ 65 Confusion Matrix ของโมเดล YOLOv8x แสดงให้เห็นว่าโมเดลมีความสามารถในการจำแนกควันได้ในระดับที่ดี โดยทำนายถูกต้องถึง 73% ในขณะที่การจำแนกไฟอยู่ที่ 66% อย่างไรก็ตาม โมเดลยังคงมีปัญหาในการแยกฉากหลังออกจากเปลวไฟ โดยทำนายผิดว่าเป็นไฟถึง 78% ของกรณีที่เป็น background จริง ซึ่งบ่งชี้ว่าแม้จะเป็นโมเดลขนาดใหญ่ที่สุด แต่ยังไม่สามารถแก้ปัญหา False Positive ได้ดีนัก การเพิ่มข้อมูลฉากหลังที่หลากหลายและลักษณะไฟปลอมอาจช่วยให้โมเดลมีความแม่นยำมากขึ้นในสถานการณ์จริง

4.2 ผลการเปรียบเทียบการตรวจจับในแต่ละโมเดล

ตาราง 3 ผลการเปรียบเทียบการตรวจจับในแต่ละโมเดล

โมเดล	ระยะเวลา (ชั่วโมง)	Precision	Recall	mAP50	mAP50-95	F1-Score
YOLOv8n	4.835	82.3%	68.7%	74.5%	51.7%	75% @ 42.4%
YOLOv8s	8.309	81.4%	70.2%	75.2%	55.4%	75% @ 42.6%
YOLOv8m	16.680	87.3%	66.3%	75.8%	58.2%	75% @ 59.9%
YOLOv8l	28.379	85.4%	65.8%	74.6%	57.9%	74% @ 50.3%
YOLOv8x	47.330	85.6%	68.0%	74.8%	58.8%	75% @ 23.5%

จากตาราง 3 ผลการเปรียบเทียบโมเดล YOLOv8 ในแต่ละเวอร์ชัน พบว่า YOLOv8m เป็นตัวเลือกที่สมดุลที่สุด ระหว่างความแม่นยำและเวลาในการฝึก โดยมี Precision 87.3% และ F1-Score สูงสุดที่ 75% @ 59.9% ใช้เวลาฝึก 16.680 ชั่วโมง ในขณะที่ YOLOv8x และ YOLOv8l แม้จะให้ความแม่นยำสูง (Precision 85.6% และ 85.4%) แต่ใช้ทรัพยากรมากขึ้นอย่างมาก โดย YOLOv8x ใช้เวลาฝึกสูงสุดถึง 47.330 ชั่วโมง ด้าน YOLOv8n และ YOLOv8s มี Inference Time ต่ำที่สุด เหมาะสำหรับอุปกรณ์ที่มีข้อจำกัดด้านพลังงาน แม้ว่า Precision และ Recall จะต่ำกว่ารุ่นที่ใหญ่กว่า โดยรวมแล้ว YOLOv8m เป็นตัวเลือกที่ดีที่สุดสำหรับการใช้งานจริง เนื่องจากให้ผลลัพธ์ที่แม่นยำสูงและยังคงมีประสิทธิภาพในการประมวลผลที่เหมาะสม ขณะที่ YOLOv8x เหมาะสำหรับงานที่ต้องการความแม่นยำสูงสุด และสามารถรองรับการใช้ทรัพยากรคำนวณที่มากขึ้น

4.3 ผลการทดสอบวิดีโอในโมเดลต่างๆ

ตาราง 4 ผลการทดสอบวิดีโอในโมเดลต่างๆ

วิดีโอ	model n		model s		model m		model l		model x	
	FPS	ms/frame	FPS	ms/frame	FPS	ms/frame	FPS	ms/frame	FPS	ms/frame
1	114.79	11.09	114.63	11.18	98.45	12.7	68.07	17.29	46.57	24.19
2	121.78	12.19	132.05	11.61	103.23	13.95	74.61	17.85	49.37	24.83
3	106.4	16.12	119.47	10.51	102.85	11	70.79	15.31	46.01	22.92
4	119.09	12.23	122.79	12.12	100.87	14.2	70.07	18.4	45.89	26.15
5	119.61	10.99	116.42	11.52	103.43	12.57	71.12	16.97	47.67	24.01
6	114.61	10.57	113.88	10.61	99.75	11.87	66.77	16.77	45.5	23.87
7	83.08	14.81	98.2	12.2	78.27	14.72	66.38	16.96	44.62	24.47
8	116.27	10.35	115.87	10.26	98.1	12.03	68.45	16.27	47.16	23.03
9	121.05	12.11	116.98	12.45	101.67	13.84	69.34	18.7	47.64	25.43
10	119.08	12.67	119.46	12.67	104.01	14.23	72.94	18.42	48.57	25.53

จากตาราง 4 แสดงผลการทดสอบวิดีโอใน YOLOv8 แต่ละเวอร์ชัน (n, s, m, l, x) โดยวัดค่า FPS (Frames Per Second) และ ms/frame (เวลาเฉลี่ยต่อเฟรม) สามารถสรุปผลได้ดังนี้

1. โมเดล YOLOv8n และ YOLOv8s มี FPS สูงสุด
   - YOLOv8n ให้ค่า FPS เฉลี่ยประมาณ 115-122 FPS และ ms/frame อยู่ที่ 10-12 ms
   - YOLOv8s มีค่าที่ใกล้เคียงกัน ซึ่งหมายถึงทั้งสองโมเดลนี้สามารถรันได้เร็วและมี latency ต่ำ
2. YOLOv8m และ YOLOv8l ให้ความสมดุลระหว่าง FPS และความแม่นยำ
   - FPS ของ YOLOv8m อยู่ระหว่าง 98-104 FPS
   - FPS ของ YOLOv8l ต่ำลงไปอีก อยู่ระหว่าง 66-74 FPS
   - เวลาเฉลี่ยต่อเฟรมของ YOLOv8m อยู่ที่ 13-14 ms และของ YOLOv8l อยู่ที่ 16-18 ms
3. YOLOv8x ช้าที่สุด แต่มีความแม่นยำสูง
   - FPS ต่ำสุด อยู่ที่ประมาณ 44-49 FPS
   - ใช้เวลาต่อเฟรมมากที่สุด ประมาณ 23-26 ms/frame
   - บ่งบอกว่าโมเดลนี้ต้องการทรัพยากรสูง และอาจไม่เหมาะกับงานที่ต้องการประมวลผลเรียลไทม์บนอุปกรณ์ที่มีข้อจำกัด

บทที่ 5

สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาและเปรียบเทียบประสิทธิภาพของ YOLOv8 ในการตรวจจับไฟและควัน โดยทำการทดสอบโมเดล YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l และ YOLOv8x เพื่อวิเคราะห์ว่าโมเดลใดให้ผลลัพธ์ที่เหมาะสมที่สุดสำหรับการนำไปใช้งานจริง ผลการทดลองพบว่า YOLOv8x มีความแม่นยำสูงสุด โดยมีค่า mAP@0.5 เท่ากับ 0.748 ซึ่งหมายความว่าโมเดลสามารถตรวจจับไฟและควันได้อย่างแม่นยำที่สุด อย่างไรก็ตาม ขนาดโมเดลที่ใหญ่ขึ้นทำให้ต้องใช้ทรัพยากรคำนวณสูง และอาจไม่เหมาะสำหรับการใช้งานบนอุปกรณ์ที่มีข้อจำกัดด้านพลังงาน เช่น Edge AI หรือระบบตรวจจับแบบเรียลไทม์ ในขณะที่ YOLOv8m ให้ความสมดุลที่ดีที่สุดระหว่างความแม่นยำและความเร็ว โดยมีค่า F1-Score สูงสุดที่ 0.75 เมื่อ Confidence = 0.599 ส่งผลให้เป็นตัวเลือกที่เหมาะสมสำหรับการใช้งานจริงที่ต้องการความแม่นยำในระดับสูงโดยไม่สูญเสียประสิทธิภาพด้านความเร็วมากเกินไป

นอกจากนี้ ผลการทดลองยังแสดงให้เห็นว่า โมเดลสามารถตรวจจับควันได้แม่นยำกว่าไฟ โดยค่า Precision และ Recall ของควันสูงกว่าของไฟตลอดช่วง Confidence ซึ่งอาจเกิดจากลักษณะของข้อมูลที่ใช้ฝึกโมเดล เนื่องจากควันมักมีรูปร่างกระจายตัวและมีลักษณะที่โมเดลสามารถเรียนรู้ได้ชัดเจนกว่า ในขณะที่เปลวไฟมีลักษณะคล้ายกับแหล่งกำเนิดแสงอื่น ๆ เช่น หลอดไฟหรือแสงสะท้อน ทำให้เกิด False Positives ได้ง่ายกว่า นอกจากนี้ ยังพบว่าเมื่อ Confidence สูงขึ้น ค่า Recall จะลดลงอย่างชัดเจน เนื่องจากโมเดลจะเลือกแสดงผลเฉพาะการตรวจจับที่มีความมั่นใจสูง ส่งผลให้มีโอกาสพลาดวัตถุที่อยู่ในขอบเขตของไฟหรือควันที่อาจมีลักษณะไม่ชัดเจน

โดยสรุป งานวิจัยนี้แสดงให้เห็นว่า YOLOv8 เป็นโมเดลที่มีประสิทธิภาพในการตรวจจับไฟและควัน แต่การเลือกเวอร์ชันที่เหมาะสมขึ้นอยู่กับข้อจำกัดด้านทรัพยากรและวัตถุประสงค์ของการใช้งานจริง YOLOv8m เป็นตัวเลือกที่ดีที่สุดสำหรับการใช้งานจริง เนื่องจากให้ความสมดุลระหว่างความแม่นยำและความเร็ว ส่วน YOLOv8x เหมาะสำหรับงานที่ต้องการความแม่นยำสูงสุด และสามารถใช้ทรัพยากรคำนวณสูงได้ หากต้องการนำโมเดลไปใช้งานจริง ควรพิจารณาการปรับแต่งโมเดลเพิ่มเติม เช่น การใช้ Transfer Learning หรือ Data Augmentation เพื่อให้โมเดลสามารถเรียนรู้จากข้อมูลที่มีความหลากหลายมากขึ้น และเพิ่มประสิทธิภาพในการตรวจจับในสภาพแวดล้อมที่แตกต่างกัน

ข้อเสนอแนะ

1. การพัฒนาเพิ่มเติม ควรเพิ่มชุดข้อมูลที่ครอบคลุมสภาพแวดล้อมที่หลากหลาย เช่น แสงน้อย, หมอกควันหรือพื้นที่ที่มีสิ่งกีดขวาง ใช้ Transfer Learning กับชุดข้อมูลเฉพาะทางเพื่อปรับปรุงประสิทธิภาพของโมเดล ทดลองใช้ Data Augmentation เช่น การเพิ่ม Noise หรือการเปลี่ยนแปลงสีของภาพเพื่อทำให้โมเดลมีความยืดหยุ่นมากขึ้น

2. การปรับใช้ในงานจริง ควรทดสอบโมเดลใน สถานการณ์จริง เช่น โรงงาน, อาคารสูง, หรือสถานที่ที่มีความเสี่ยงต่อการเกิดไฟไหม้ พัฒนา Edge AI Solution โดยใช้ YOLOv8m หรือ YOLOv8s บน Jetson Nano หรือ Raspberry Pi ผสานรวมโมเดลกับระบบ Internet of Things (IoT) เพื่อให้สามารถแจ้งเตือนอัตโนมัติเมื่อเกิดเหตุเพลิงไหม้

3. การปรับปรุงโมเดลให้ทำงานได้ดีขึ้น ทดลองใช้ Ensemble Model หรือการรวมโมเดลหลายตัวเพื่อเพิ่มความแม่นยำ ผสาน YOLOv8 กับ เซ็นเซอร์ตรวจจับความร้อน หรือกล้องอินฟราเรดเพื่อลด False Positives ใช้ Optical Flow หรือ CNN-LSTM เพื่อเพิ่มประสิทธิภาพในการตรวจจับไฟที่มีการเคลื่อนไหว

4. แนวทางสำหรับงานวิจัยในอนาคต ทดลองใช้ โมเดลที่ล้ำสมัยขึ้น เช่น Vision Transformer (ViT), DETR, หรือ YOLO-NAS เพื่อตรวจสอบว่าให้ผลลัพธ์ที่ดีกว่าหรือไม่ ศึกษาวิธีลดพลังงานที่ใช้ในการประมวลผลโมเดล เพื่อให้สามารถใช้งานบน อุปกรณ์พลังงานต่ำได้ดีขึ้น วิเคราะห์การนำ Deep Learning มาผสานกับ AI-Based Fire Suppression Systems เพื่อให้เกิดระบบอัตโนมัติที่สามารถตรวจจับและระงับไฟไหม้ได้อย่างมีประสิทธิภาพ

บรรณานุกรม

[1] R. Ghali, M. Jmal, W. Souidene Mseddi, and R. Attia, "Recent Advances in Fire Detection and Monitoring Systems: A Review," in *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18), Vol.1*, Cham, M. S. Bouhlel and S. Rovetta, Eds., 2020// 2020: Springer International Publishing, pp. 332-340.

[2] A. Koltunov, S. L. Ustin, B. Quayle, B. Schwind, V. G. Ambrosia, and W. Li, "The development and first validation of the GOES Early Fire Detection (GOES-EFD) algorithm," *Remote Sensing of Environment,* vol. 184, pp. 436-453, 2016/10/01/ 2016, doi: https://doi.org/10.1016/j.rse.2016.07.021.

[3] V. Kaul, S. Enslin, and S. A. Gross, "History of artificial intelligence in medicine," *Gastrointestinal Endoscopy,* vol. 92, no. 4, pp. 807-812, 2020/10/01/ 2020, doi: https://doi.org/10.1016/j.gie.2020.06.040.

[4] R. Gupta, D. Srivastava, M. Sahu, S. Tiwari, R. K. Ambasta, and P. Kumar, "Artificial intelligence to deep learning: machine intelligence approach for drug discovery," *Molecular Diversity,* vol. 25, no. 3, pp. 1315-1360, 2021/08/01 2021, doi: 10.1007/s11030-021-10217-3.

[5] S. Gollapudi, "OpenCV with Python," in *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*, S. Gollapudi Ed. Berkeley, CA: Apress, 2019, pp. 31-50.

[6] A. Hanafi, L. Elaachak, and M. Bouhorma, "Safe laboratory practices & Procedures introduced to the students through an augmented reality application," in *ACM International Conference Proceeding Series*, 2019, doi: 10.1145/3368756.3369042. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85077743915&doi=10.1145%2f3368756.3369042&partnerID=40&md5=5ea3fbbb33a88a5fc308e4ff1f5c7122

[7] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet],* vol. 9, no. 1, pp. 381-386, 2020.

[8] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data,* vol. 8, no. 1, p. 53, 2021/03/31 2021, doi: 10.1186/s40537-021-00444-8.

[9] P. Cunningham, M. Cord, and S. J. Delany, "Supervised Learning," in *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, M. Cord and P. Cunningham Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21-49.

[10] Z. Ghahramani, "Unsupervised Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, and G. Rätsch Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72-112.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research,* vol. 4, pp. 237-285, 1996.

[12] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Archives of Computational Methods in Engineering,* vol. 27, no. 4, pp. 1071-1092, 2020/09/01 2020, doi: 10.1007/s11831-019-09344-w.

[13] P. G. Asteris and V. G. Mokos, "Concrete compressive strength using artificial neural networks," *Neural Computing and Applications,* vol. 32, no. 15, pp. 11807-11826, 2020/08/01 2020, doi: 10.1007/s00521-019-04663-2.

[14] Y.-c. Wu and J.-w. Feng, "Development and application of artificial neural network," *Wireless Personal Communications,* vol. 102, pp. 1645-1656, 2018.

[15] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition,* vol. 77, pp. 354-377, 2018/05/01/ 2018, doi: https://doi.org/10.1016/j.patcog.2017.10.013.

[16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 33, no. 12, pp. 6999-7019, 2022, doi: 10.1109/TNNLS.2021.3084827.

[17] D. Bhatt *et al.*, "CNN variants for computer vision: History, architecture,

application, challenges and future scope," *Electronics,* vol. 10, no. 20, p. 2470, 2021.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[19] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine learning and knowledge extraction,* vol. 5, no. 4, pp. 1680-1716, 2023.

[20] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast YOLO: A fast you only look once system for real-time embedded object detection in video," *arXiv preprint arXiv:*1709.05943*,* 2017.

[21] G. Nguyen *et al.*, "Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey," *Artificial Intelligence Review,* vol. 52, pp. 77-124, 2019.

[22] F. M. Talaat and H. ZainEldin, "An improved fire detection approach based on YOLO-v8 for smart cities," *Neural Computing and Applications,* vol. 35, no. 28, pp. 20939-20954, 2023/10/01 2023, doi: 10.1007/s00521-023-08809-1.

[23] L. Jiao *et al.*, "A survey of deep learning-based object detection," *IEEE access,* vol. 7, pp. 128837-128868, 2019.

[24] D. S. Kanmani, "A Comparative study of various versions of YOLO algorithm to detect drones," *rrrj,* vol. 2, no. 1, pp. 54-61, 2023.

[25] S. Majid, F. Alenezi, S. Masood, M. Ahmad, E. S. Gündüz, and K. Polat, "Attention based CNN model for fire detection and localization in real-world images," *Expert Systems with Applications,* vol. 189, p. 116114, 2022/03/01/ 2022, doi: https://doi.org/10.1016/j.eswa.2021.116114.

พหูน ปณ ฑิโต ชีเว

ประวัติผู้เขียน

ชื่อ	นาย วรรัตน์ ภักดีธา
วันเกิด	27 มิถุนายน พ.ศ. 2543
สถานที่เกิด	แขวงคลองสาน คลองสาน กรุงเทพมหานคร
สถานที่อยู่ปัจจุบัน	บ้านเลขที่ 41 หมู่ 6 บ้านขอนสัก ตำบลหนองแซง อำเภอบ้านแฮด จังหวัดขอนแก่น 40110
ประวัติการศึกษา	พ.ศ.2562 มัธยมศึกษาตอนปลาย โรงเรียนบ้านแฮดศึกษา จังหวัดขอนแก่น พ.ศ.2565 วิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหาสารคาม จังหวัดมหาสารคาม พ.ศ.2567 วิศวกรรมศาสตรมหาบัณฑิต (วศ.ม.) สาขาวิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหาสารคาม จังหวัดมหาสารคาม