**การตรวจจับผู้คนที่รอข้ามถนน ณ เวลานั้นเพื่อพิจารณาการให้สัญญาณไฟจราจร**

**Real time People Detector for Controlling traffic light**

*บทคัดย่อ* — ในปัจจุบันปัญหาเกี่ยวกับการใช้ถนนไม่ว่าจะเป็น ผู้ใช้ทางเท้า หรือ ผู้ใช้ยานพาหนะ ยังเป็นปัญหาที่ยังไม่ได้รับการแก้ไขอย่างจริงจังนัก มีอยู่บ่อยครั้งที่ผู้ใช้ทางเท้าไม่สามารถใช้ถนนได้อย่างปลอดภัย เช่น การข้ามถนน จึงเป็นที่มาของการทำโครงงานนี้เพื่อที่จะมาช่วยบรรเทาในส่วนนี้โดยการนำ Computer Vision ที่เป็น AI รูปแบบหนึ่งมาใช้เพื่อตรวจจับคนรอข้ามถนน ณ เวลานั้นเพื่อพิจารณาการเปลี่ยนสัญญาณไฟจารจร โดยเริ่มจากการเตรียมข้อมูล รูปภาพของคน และ ทำความสะอาดข้อมูลเพื่อให้ฝึกฝนโมเดลAIให้สามารถตรวจจับได้แม่นยำมากขึ้น ต่อมาคือ การฝึกฝนโมเดล โดยทดลองปรับค่าต่างๆไปเรื่อยๆจนได้โมเดลที่ดีที่สุดออกมา และนำไป deploy ใช้จริง ซึ่งโครงงานนี้จะนำโมเดลที่ดีที่สุดที่ได้ไป deploy บนบอร์ด Raspberry Pi ที่เป็นคอมพิวเตอร์ขนาดเล็กร่วมกับกล้อง เพื่อความสะดวกในการนำไปใช้งานเนื่องจากมีขนาดเล็ก

**I. บทนำ**

เนื่องจากในปัจจุบันปัญหาเกี่ยวกับการใช้ถนนไม่ว่าจะเป็น ผู้ใช้ทางเท้า หรือ ผู้ใช้ยานพาหนะ ยังเป็นปัญหาที่ยังไม่ได้รับการแก้ไขอย่างจริงจังนัก มีอยู่บ่อยครั้งที่ผู้ใช้ทางเท้าไม่สามารถใช้ถนนได้อย่างปลอดภัย เช่น การข้ามถนน ผู้ใช้งานทางเท้าไม่สามารถไว้วางใจในการข้ามถนนได้ และ ในกรณีของผู้ต้องการข้ามถนนที่เป็นผู้พิการ หรือ เป็นใครก็ตามที่ไม่สามารถกดสัญญาณเพื่อรอข้ามถนนได้ ทำให้เกิดความไม่สะดวกในการใช้งานทำให้บ่อยครั้งมักจะมีผู้ที่ข้ามถนนโดยไม่รอสัญญาณไฟ หรือ ข้ามถนนโดยไม่ใช้ทางม้าลาย ส่งผลให้เกิดอุบัติเหต เพื่อช่วยลดปัญหาเหล่านี้จึงได้มีแนวคิดของโครงงานนี้เกิดขึ้นมา



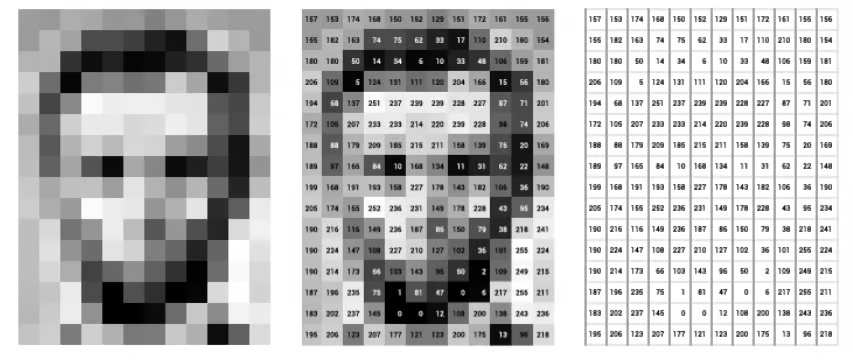
**2.ทฤษฎีที่เกี่ยวข้อง**

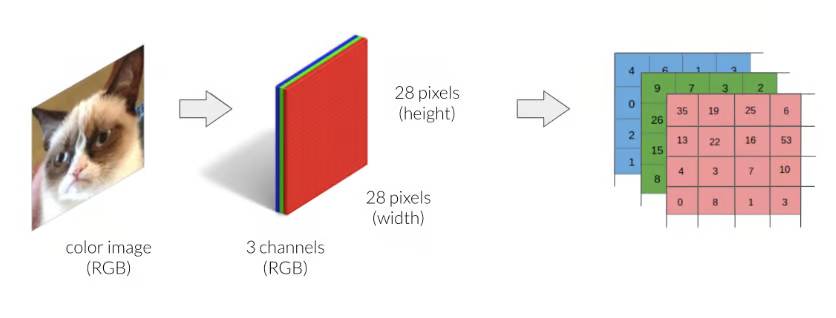
**2.1.Computer Vision**

Computer Vision เป็นสาขาหนึ่งของปัญญาประดิษฐ์ (AI) ที่ศึกษาวิธีทำให้คอมพิวเตอร์สามารถมองเห็นและเข้าใจภาพและวิดีโอดิจิทัล เป้าหมายหลักคือการเลียนแบบความสามารถในการมองเห็นของมนุษย์ แม้ว่ามนุษย์ใช้ดวงตาและสมองในการประมวลผลภาพ แต่คอมพิวเตอร์ต้องอาศัยองค์ประกอบทางเทคโนโลยี ได้แก่

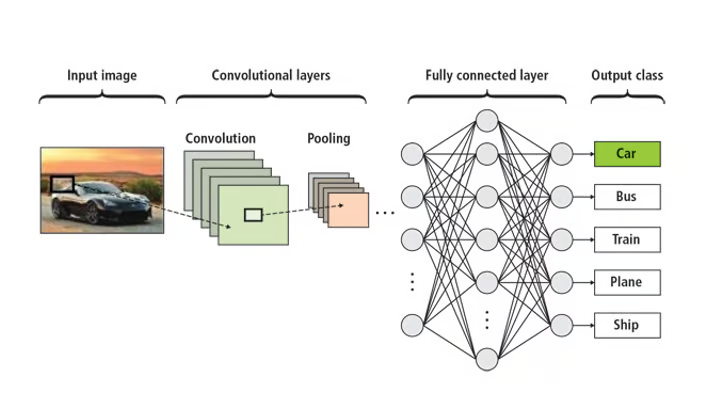
* เซ็นเซอร์ (Sensors): กล้องและอุปกรณ์ที่ติดตั้งเซ็นเซอร์พิเศษเพื่อเก็บข้อมูลภาพ
* ข้อมูล (Data): ครอบคลุมไฟล์รูปภาพ (.jpg, .png) และวิดีโอ (.mov, .avi) รวมถึงภาพจากกล้องหลายตัว, สแกนเนอร์ 3D, หรืออุปกรณ์ทางการแพทย์
* อัลกอริทึม (Algorithms): ใช้เทคนิคเตรียมข้อมูล เช่น การกรอง การปรับขนาด และการทำให้ภาพเป็นมาตรฐาน (Normalization) ก่อนนำไปใช้กับโมเดล Deep Learning เพื่อการวิเคราะห์และตรวจจับวัตถุที่มีประสิทธิภาพสูง

โดยการทำงานของมันจะทำการหาคุณลักษณะของรูปภาพโดยแปลงความเข้มของแต่ละ pixel ในรูปภาพให้เป็นตัวเลขเพื่อนำไปใช้คำนวณหาคุณลักษณะเล่น โดยค่าของแต่ละ pixel มีค่าได้ตั้งแต่ 0(เข้มมาก) จนถึง  255(จางจนเป็นสีขาว)

****

โดยสำหรับการทำเกี่ยวกับภาพสีจะทำให้ใช้การคำนวณมากขึ้น เนื่องจากต้องแบ่งแยกคุณลักษณะเป็น RGB จึงต้องหาคุณลักษณะเด่นของชั้น สีแดง สีเขียว และ สีน้ำเงิน ซึ่งมากกว่าภาพขาวดำที่ใช้เพียงชั้นเดียว****

หลังจากได้ค่าของคุณลักษณะเด่นแล้ว จะนำค่าเหล่านี้ไปคำนวณใน Neural Network เพื่อหาค่าความน่าจะเป็นว่ารูปภาพนี้มีค่าใกล้เคียงกับรูปภาพใดและทำนายออกมาเป็นรูปภาพนั้น โดยโมเดลที่นิยมใน  Computer Vision  คือ โมเดล CNN (Convolution Neuron Network)

****

2.2 **Pre-trained Model YOLO**

YOLO (You Only Look Once) เป็นหนึ่งในโมเดลที่นิยมใช้ในการตรวจจับวัตถุ (Object Detection) ในงานคอมพิวเตอร์วิชัน ซึ่ง YOLO ถูกพัฒนาเพื่อทำให้กระบวนการตรวจจับวัตถุเร็วขึ้นและมีประสิทธิภาพมากขึ้น โดยมันสามารถทำการตรวจจับวัตถุหลาย ๆ ชนิดในภาพเดียวกันได้ภายในเวลาเพียงแค่การคำนวณครั้งเดียว นั่นคือสิ่งที่ YOLO ทำให้มันมีชื่อเสียงว่า "You Only Look Once" นั่นเอง

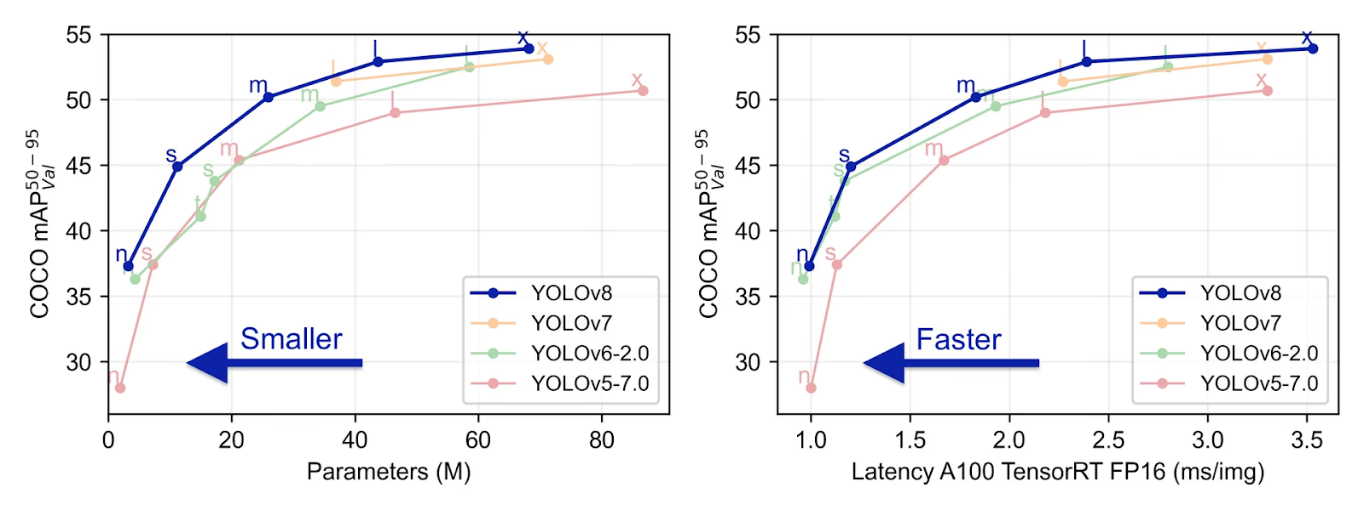
### **หลักการทำงานของ YOLO**

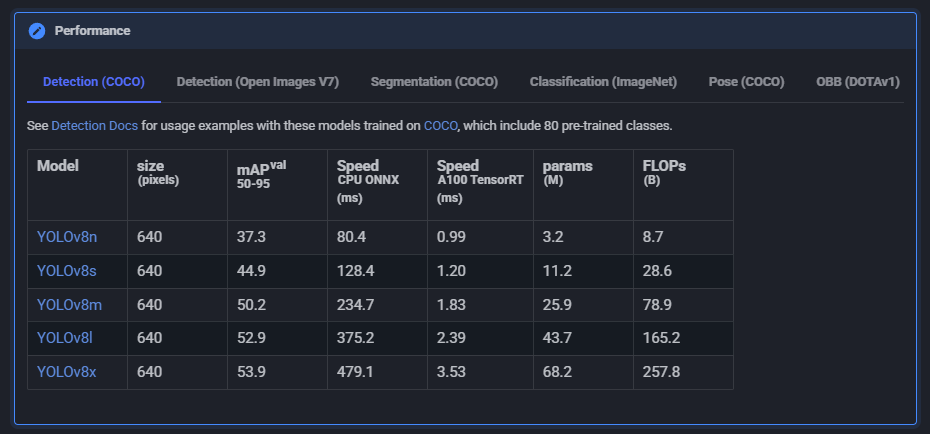
YOLO ใช้โครงข่ายประสาทเทียม (Neural Network) ในการทำการตรวจจับวัตถุ โดยมีหลักการทำงานหลัก ๆ ดังนี้:

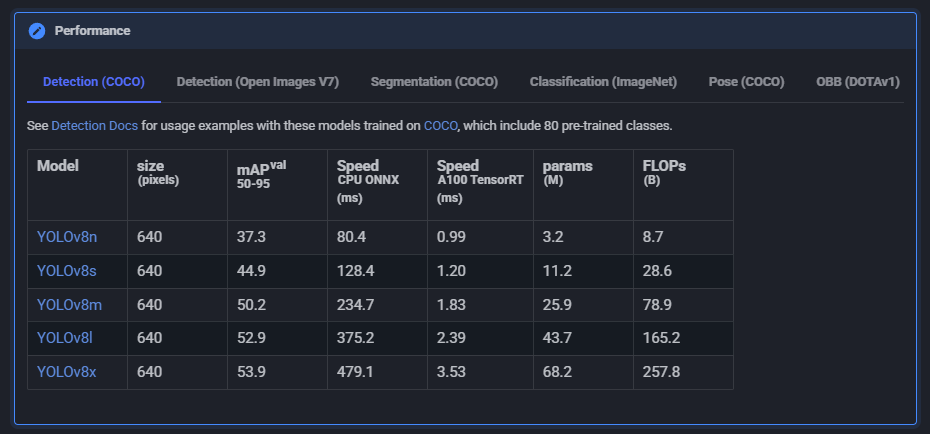
1. แบ่งภาพเป็น Grid: ภาพที่เข้ามาจะถูกแบ่งเป็นกริดเล็ก ๆ (เช่น 13x13, 19x19) โดยในแต่ละกริดจะทำการทำนายว่ามีวัตถุอะไรในแต่ละกริดบ้าง และอยู่ที่ตำแหน่งไหน
2. การทำนาย Output ของแต่ละ Grid: ในแต่ละกริดของภาพ YOLO จะทำนาย:
   * Bounding Box: สี่เหลี่ยมที่ล้อมรอบวัตถุ (ตำแหน่งและขนาด)
   * Class Probability: ความน่าจะเป็นของแต่ละประเภทวัตถุที่อาจจะอยู่ในกริดนั้น (เช่น แมว, สุนัข, รถยนต์ ฯลฯ)
   * Confidence Score: ค่าความมั่นใจว่าในกริดนั้นมีวัตถุอยู่หรือไม่
3. การคำนวณ Bounding Box: YOLO จะใช้ข้อมูลจากกริดที่แบ่งออกมาทำนายการตั้งค่าของ Bounding Box (ตำแหน่งและขนาด) รวมถึงการคำนวณคะแนนความมั่นใจที่วัตถุที่ตรวจจับได้จะตรงกับกริดนั้นจริง ๆ
4. การใช้ Anchor Boxes: YOLO ใช้การกำหนด Anchor Boxes เพื่อช่วยในการทำนายขนาดและอัตราส่วนของ Bounding Box ที่ดีที่สุดสำหรับแต่ละวัตถุ ซึ่ง Anchor Boxes จะช่วยให้ YOLO ตรวจจับวัตถุในอัตราส่วนต่าง ๆ ได้ดียิ่งขึ้น
5. การกรองผลลัพธ์: หลังจากได้ผลการทำนายจากกริดต่าง ๆ แล้ว YOLO จะใช้เทคนิค Non-Maximum Suppression (NMS) เพื่อลดจำนวน Bounding Box ที่ทับซ้อนกัน เพื่อให้ได้ผลลัพธ์ที่ดีที่สุดและไม่ซ้ำกัน

**2.1.1.YOLO 8**

YOLOv8 ถูกปล่อยโดย Ultralytics เมื่อวันที่ 10 มกราคม 2023 โดยมอบประสิทธิภาพที่ล้ำสมัยในแง่ของความแม่นยำและความเร็ว ซึ่งได้พัฒนาต่อยอดจากความก้าวหน้าของเวอร์ชัน YOLO ก่อนหน้า โดย YOLOv8 ได้นำเสนอคุณสมบัติและการ**ปรับแต่งใหม่**ๆ ที่ทำให้มันเป็นตัวเลือกที่เหมาะสมสำหรับงานตรวจจับวัตถุในหลากหลายแอปพลิเคชัน

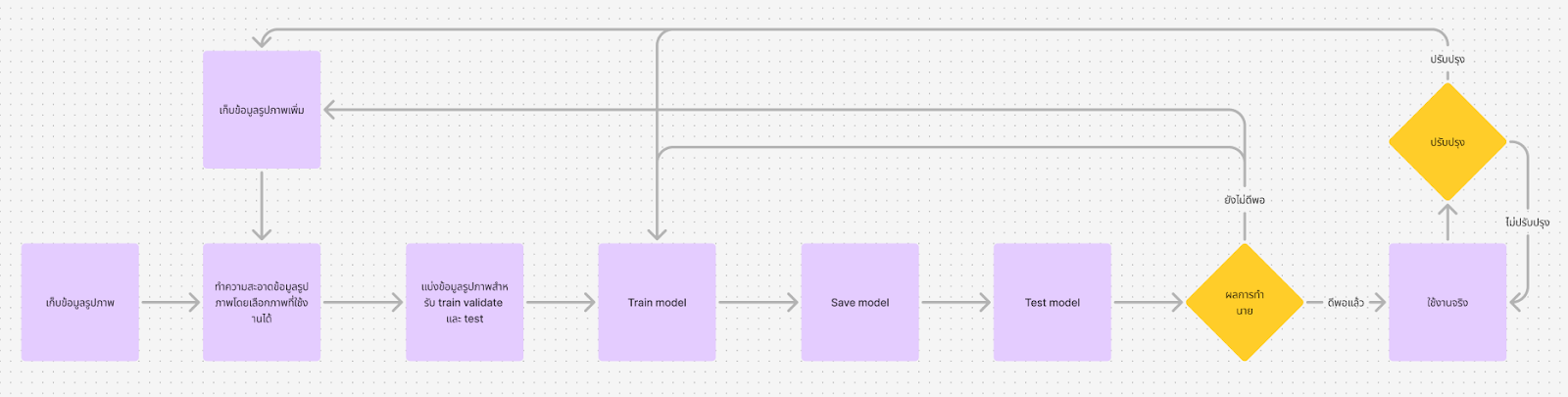
****



****

ตารางแสดงประสิทธิภาพคร่าวๆ

**3.ระบบการพัฒนาโมเดล**

****

3.1.1.เริ่มจากการเก็บรวมรวมรูปภาพ

3.1.2.นำความสะอาดข้อมูลรูปภาพให้สามารถใช้งานได้

3.1.3.แบ่งชุดข้อมูลรูปภาพ

3.1.4.ฝึกโมเดล

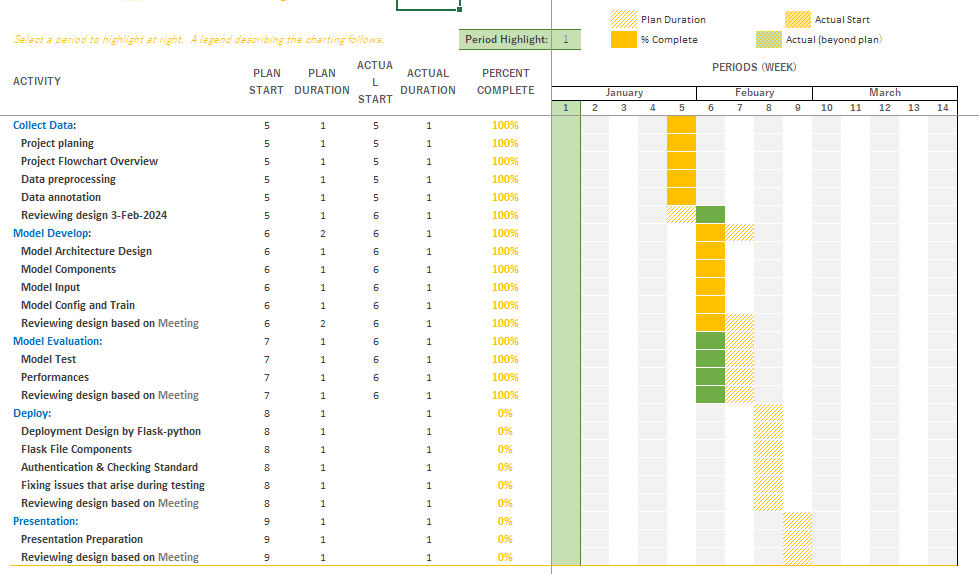
3.1.5.บันทึกโมเดลที่ฝึกได้

3.1.6.นำโมเดลที่บันทึกไปทดสอบ

3.1.7.สังเกตผลทำนายเพื่อพิจารณาว่าโมเดลที่ฝึกดีพอหรือไม่ ถ้าไม่ดีพอย้อนกลับไปทำ 3.1.4 หรือ เพิ่มจำนวนข้อมูลแล้วทำตั้งแต่ 3.1.2 เพิ่มเติม

3.1.8.เมื่อโมเดลที่ฝึกมีความแม่นยำดีพอแล้วจึงนำไปใช้งานจริง

3.1.9.ปรับปรุงโมเดลเพื่อความทันสมัยโดยการเพิ่มข้อมูลใหม่ๆแล้วทำตั้งแต่ 3.2 เพิ่มเติม

**3.2.แผนผังการพัฒนา**

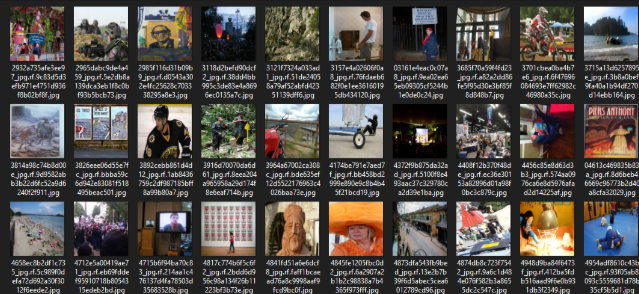
**หมายเหตุเนื่องจากผู้สั่งงานมีการเลื่อนกำหนดส่งให้ไวขึ้นโดยกระทันหัน กำหนดการจึงคลาดเคลี่อนไป 1 สัปดาห์**

**4.การพัฒนาโมเดล**

**4.1.การเตรียมข้อมูล**

ข้อมูลที่ใช้เป็นข้อมูลรูปภาพของคนจาก [Human Dataset](https://www.kaggle.com/datasets/fareselmenshawii/human-dataset)

สำหรับ version ที่ 2 มาจาก <https://universe.roboflow.com/vasu12360-gmail-com/people-detection-a5s5p/dataset/2>



4.1.1.แล้วนำมา annotate ใน  Roboflow เพื่อให้ได้ชุดข้อมูล เป็นอัตตราส่วน

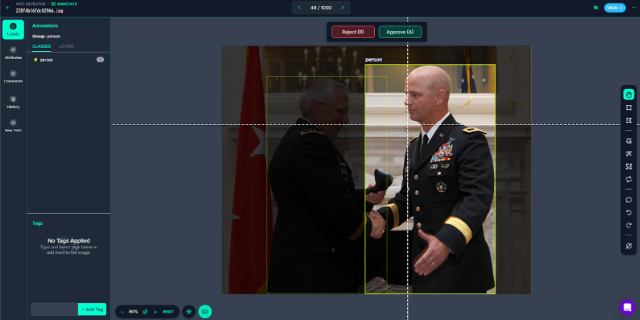
Data set v1: 701-200-99(70-20-10)

Data set v2: 929-265-133(70-20-10)

4.1.1.1 train สำหรับใช้ฝึกโมเดล

4.1.1.2 validate สำหรับใช้เทียบเพื่อหาค่าความแม่นยำ

4.1.1.3 test สำหรับใช้ทดสอบว่าสามารถทำนายได้จริงหรือไม่



4.1.2.เมื่อ annotate แล้วจะได้ชุดข้อมูลของ train validate และ test ซึ่งแต่ละชุดจะประกอบไปด้วย ชุดของรูปภาพ(image) และ ชุดของป้ายชื่อกำกับ(label) มาใส่ในFolderที่เตรียมไว้ดังรูปด้านล่าง แล้ว upload ลง Google Drive [datasets](https://drive.google.com/drive/folders/1rRpdoQ46n2ggeAFRr2LpFx0DSpeX2kQu?usp=drive_link) และ [datasetsv2](https://drive.google.com/drive/folders/1gRHUM1B6KLVOtaA3hGB_l-Z_2FboEUbh?usp=drive_link) เพื่อความสะดวกในการเปิดใช้งาน



**4.2.การฝึกโมเดล**

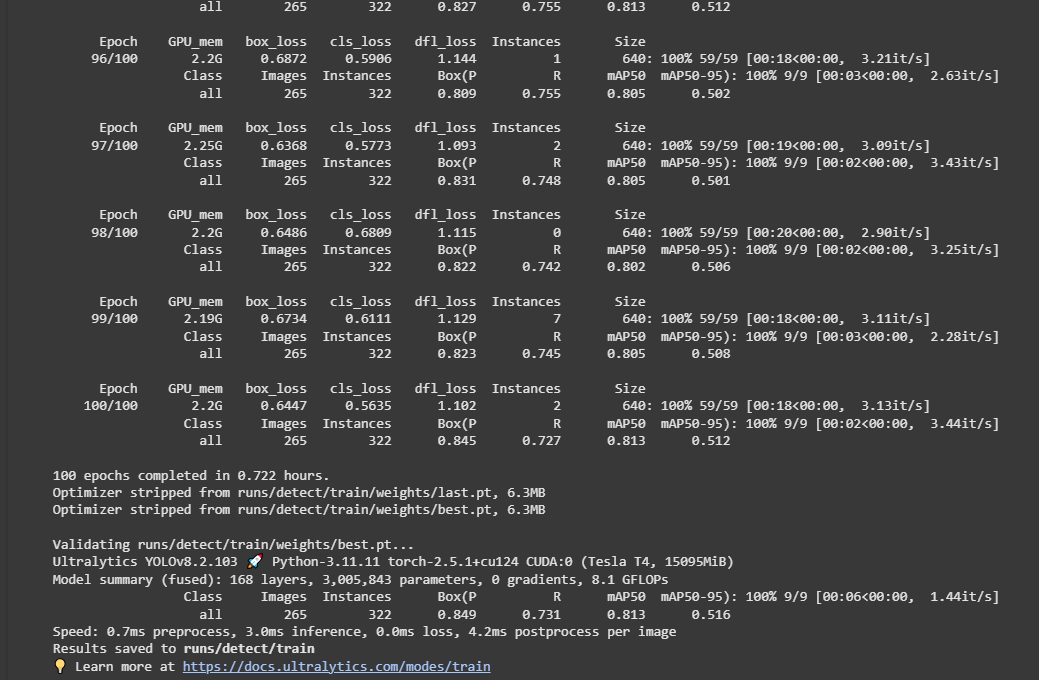
4.2.1.ทำการเตรียมสภาพแวดล้อมสำหรับการฝึก สำหรับโมเดลของโครงงานนี้ฝึกบน Google Colab ใช้ YOLOv8

|  |
| --- |
| !pip install ultralytics==8.2.103 -q  from IPython import display  display.clear\_output()  import ultralytics  ultralytics.checks()  from ultralytics import YOLO  from IPython.display import display, Image |

4.2.2.ทำการฝึกสอนโมเดล

|  |
| --- |
| !yolo task=detect mode=train model=yolov8s.pt data='/content/drive/MyDrive/towerlamp/datasets/data.yaml' epochs=100 imgsz=640 plots=True |

|  |  |  |
| --- | --- | --- |
| **พารามิเตอร์** | **คำนิยาม** | **ค่าเริ่มต้น** |
| **epochs** | จำนวนรอบการฝึกสอนโมเดล ยิ่งมาก โมเดลจะเรียนรู้มากขึ้น แต่ก็ใช้เวลานานขึ้น | 100 |
| **batch** | ขนาดของชุดข้อมูลที่ใช้ในการฝึกต่อรอบ ยิ่งมากช่วยให้โมเดลเรียนเร็วขึ้น แต่ใช้ RAM มากขึ้น | 16 |
| **imgsz** | ขนาดของภาพอินพุต (เช่น 640x640) ขนาดที่ใหญ่ขึ้นอาจช่วยให้ตรวจจับแม่นยำขึ้นแต่ช้าลง | 640 |
| **lr0** | ค่าเริ่มต้นของอัตราการเรียนรู้ (Learning Rate) | 0.01 |
| **optimizer** | ตัวเลือกอัลกอริธึมสำหรับการฝึกสอน (SGD, Adam, AdamW) | AdamW |
| **Model** | โมเดล เช่น Yolov8n, Yolov8s, Yolov8m, Yolov8l, Yolov8x | Yolov8n |



4.2.3 ผลลัพธ์ของการฝึกโมเดล

ตารางบันทึกผลการทดลองเพื่อหาโมเดล AI ที่ดีที่สุด

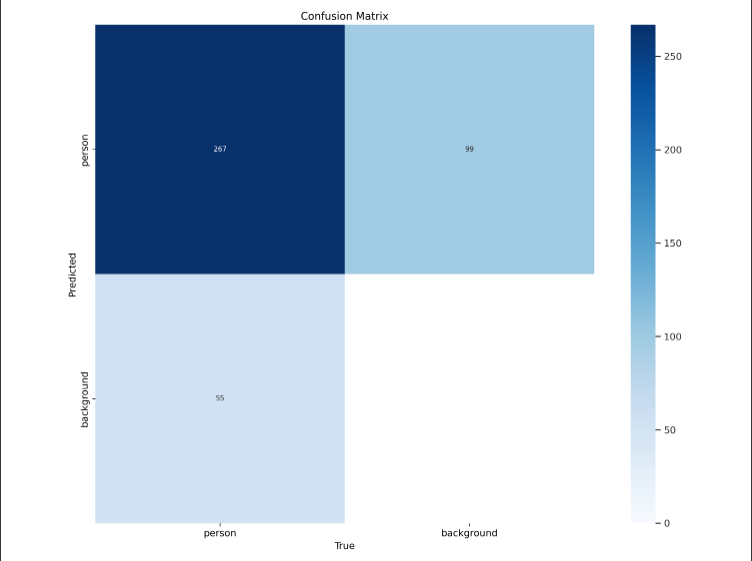
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Exp. No.** | **Model** | **Epoch** | **Batch Size** | **Precision 1** | **Recall** | **mAP50 1** | **mAP50-95** |
| **1** | Yolov8s | 10 | 16 | 0.638 | 0.540 | 0.592 | 0.338 |
| **2** | Yolov8n | 50 | 16 | 0.629 | 0.566 | 0.611 | 0.372 |
| **3** | Yolov8s | 100 | 32 | 0.604 | 0.572 | 0.604 | 0.35 |
| **4** | Yolov8s | 200 | 32 | 0.646 | 0.514 | 0.563 | 0.334 |
| **5** | Yolov8m | 10 | 32 | 0.309 | 0.336 | 0.247 | 0.0988 |
| **6** | Yolov8n | 100 | 16 | 0.691 | 0.522 | 0.615 | 0.368 |
| **7** | Yolov8n(Datasetsv2) | 100 | 16 | 0.849 | 0.731 | 0.813 | 0.516 |

โครงงานนี้ได้ทำการทดลองฝึกทั้งหมด 6รอบ ได้โมเดลที่ดีที่สุดคือ **Yolov8n** ที่ใช้การตั้งค่าตามค่าในตารางค่าเริ่มต้นด้านบน

ที่ได้ผลลัพธ์ Precision = 84.9% Recall = 73.1% mAP50 = 81.3% mAP50-95 = 51.6%

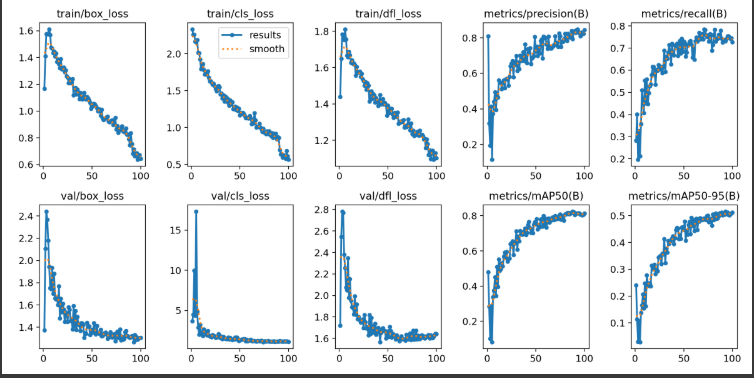
4.2.3.1.ผลลัพธ์ในรูปแบบ Confusion Matrices

|  |
| --- |
| Image(filename=f'/content/runs/detect/train/confusion\_matrix.png', width=600) |



4.2.3.2.ผลลัพธ์ในรูปแบบกราฟ

|  |
| --- |
| Image(filename=f'/content/runs/detect/train/results.png', width=600) |



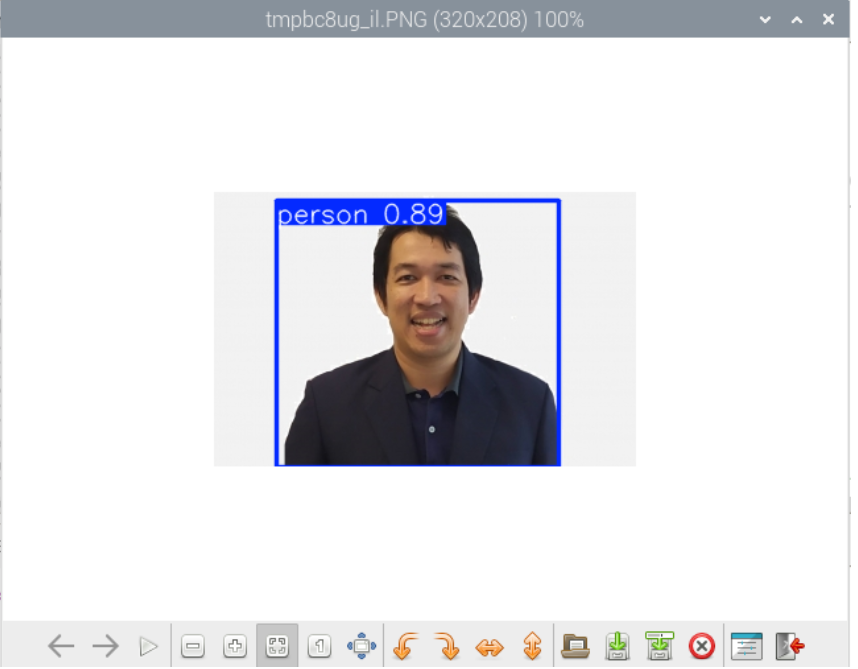
**4.3.ทดลองให้ทำนายรูปภาพ**

|  |
| --- |
| Image(filename=f'/content/runs/detect/train/val\_batch0\_pred.jpg', width=600) |



**4.4.นำโมเดลที่ฝึกได้ไปทดสอบใน Raspberry Pi โดยใช้โปรแกรม Thonny**

|  |
| --- |
| from ultralytics import YOLO  # Load a model  model = YOLO("/home/adn/Desktop/best.pt")  # pretrained YOLO11n model  results = model(["/home/adn/Desktop/adna.jpg"])  # return  for i, result in enumerate(results):      print(f"Results for Image {i+1}:")      boxes = result.boxes  # Bounding box outputs      if boxes is not None:          for j, box in enumerate(boxes.xyxy):              x\_min, y\_min, x\_max, y\_max = box.tolist()              print(f"Object {j+1}: x\_min={int(x\_min)}, y\_min={int(y\_min)}, x\_max={int(x\_max)}, y\_max={int(y\_max)}")        result.show()  # Display image with detections      result.save(filename=f"result\_{i+1}.jpg")  # Save to disk |



ผลลัพธ์ที่ได้

**5.สรุป**

โครงงานนี้สามารถดำเนินการไปได้ด้วยดี ถึงแม้ค่าความแม่นยำจะมีเพียง **Precision = 69.1% Recall = 52.2% mAP50 = 61.5% และ mAP50-95 = 36.8%** ก็สามารถทำนายออกมาได้ค่อนข้างแม่นยำในชุดข้อมูล version ที่ 1 ที่มีเพียง 1000รูป และ เวลาในการพัฒนาที่มีจำกัด

สำหรับชุดข้อมูล version ที่ 2 ที่มี 1327 รูป และ เป็นรูปที่มีคุณภาพกว่า จึงทำให้ได้ **Precision = 84.9% Recall = 73.1% mAP50 = 81.3% mAP50-95 = 51.6%** ซึ่งส่งผลต่างเป็นอย่างมากแม้จะฝึกสอนด้วยโมเดล Yolov8n ที่มีการตั้งค่าเหมือนกัน

สำหรับการพัฒนาในอนาคต ควรเตรียมชุดข้อมูลที่ดี และ มีจำนวนมากกว่านี้เพื่อเพิ่ม ค่าความแม่นยำ และ ประสิทธิภาพของโมเดล

**อ้างอิง**

**-**[**YOLOv8 - Ultralytics YOLO Docs**](https://docs.ultralytics.com/models/yolov8/#supported-tasks-and-modes)

**-**[**What is Computer Vision? A Beginner Guide to Image Analysis | DataCamp**](https://www.datacamp.com/blog/what-is-computer-vision)

**-**[RPI-for AIIoT\_Chp06.pdf](https://drive.google.com/file/d/1pgVv-2aJywXEm8BqH8kmV6ZMN7G1bfNX/view?usp=drive_link)

**-**[**RPI-for AIIoT\_Chp07.pdf**](https://drive.google.com/file/d/1HAev9DrLc27OwjIF5eebCLsW3iuqMuFY/view?usp=drive_link)

**-**<https://universe.roboflow.com/vasu12360-gmail-com/people-detection-a5s5p/dataset/2>

**จัดทำโดย**

**นายสิรธีร์ แทนจ้อย 2211311861**

**นายธีรชาติ คงอิ่ม 2211311630**

**ดูไฟล์งานทั้งหมดได้ที่**

**GitHub** <https://github.com/NONSRT/Real-time-People-Detector-for-Controlling-traffic-light>

**Google Drive** [**PersonDetector**](https://drive.google.com/drive/folders/11uwvVtE1pF5LC3bjAPSKm9diUsoRjhFo?usp=drive_link)