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

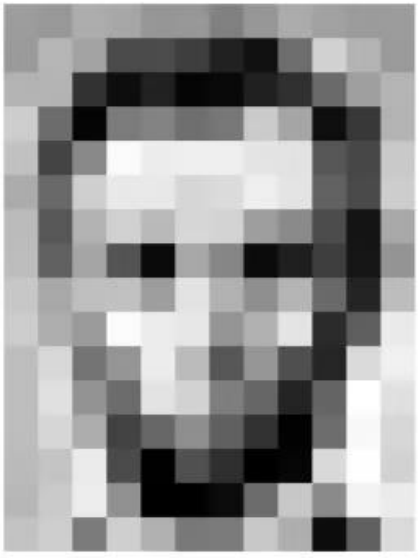
2.ทฤษฎีที่เกี่ยวข้อง

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

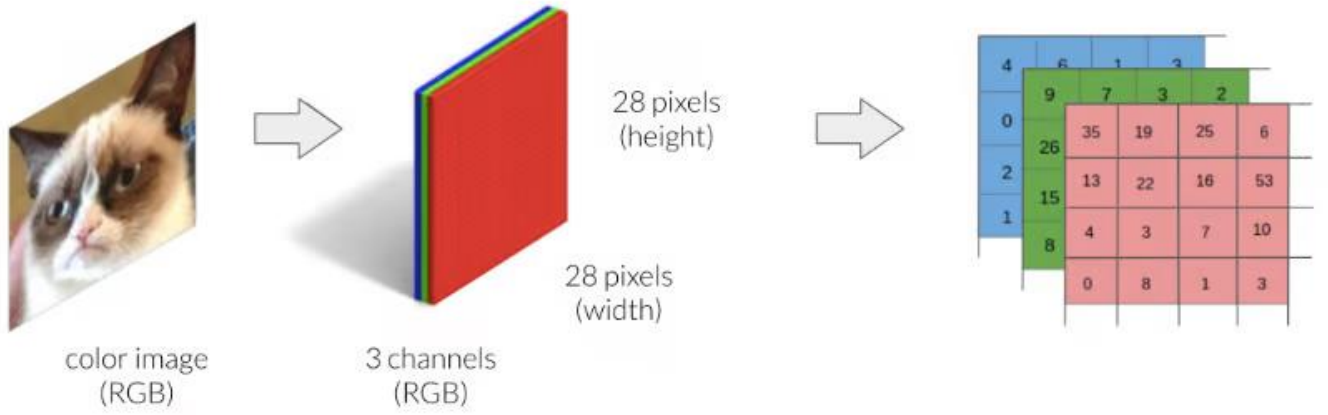
โดยการทำงานของมันจะทำการหาคุณลักษณะของรูปภาพโดยแปลงความเข้มของแต่ละ pixel ในรูปภาพให้เป็นตัวเลขเพื่อนำไปใช้คำนวณหาคุณลักษณะเล่น โดยค่าของแต่ละ pixel มีค่าได้ตั้งแต่ 0(เข้มมาก) จนถึง 255(จางจนเป็นสีขาว)



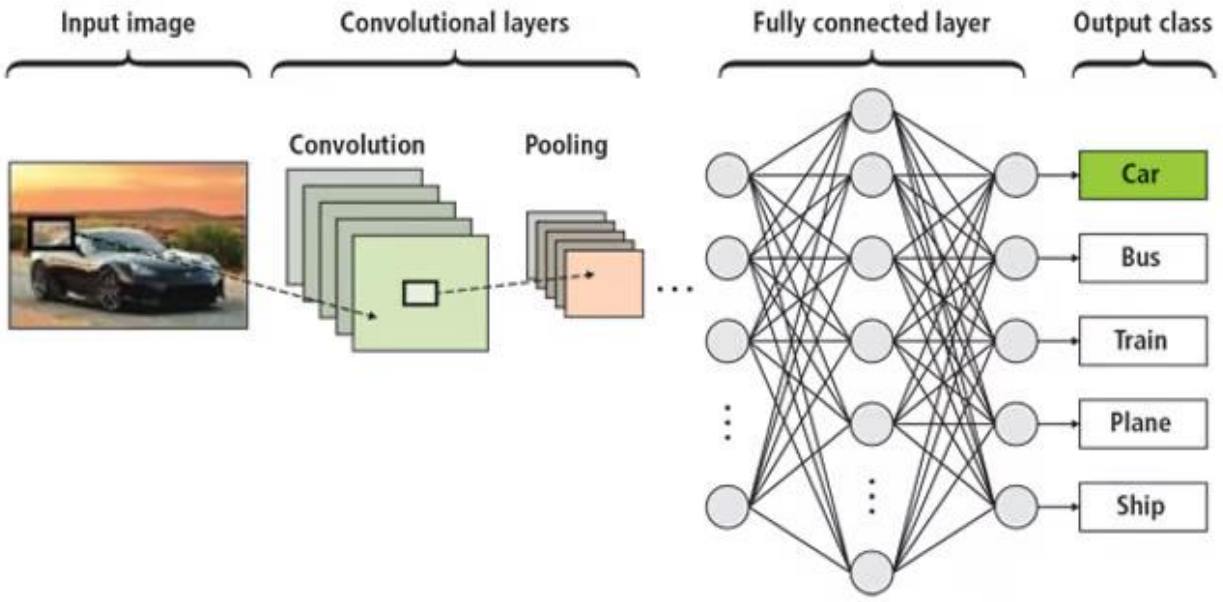
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

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

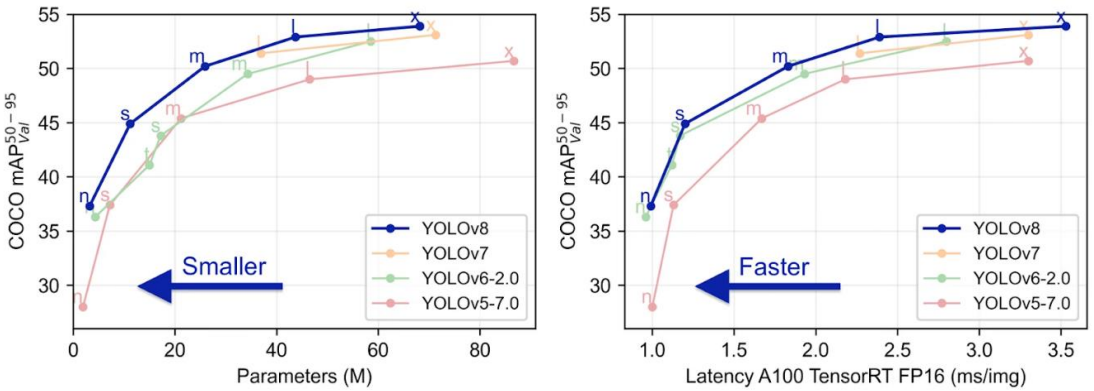
หลักการทำงานของ YOLO

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



Performance

Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) | Pose (COCO) | OBB (DOTAv1)

See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Performance

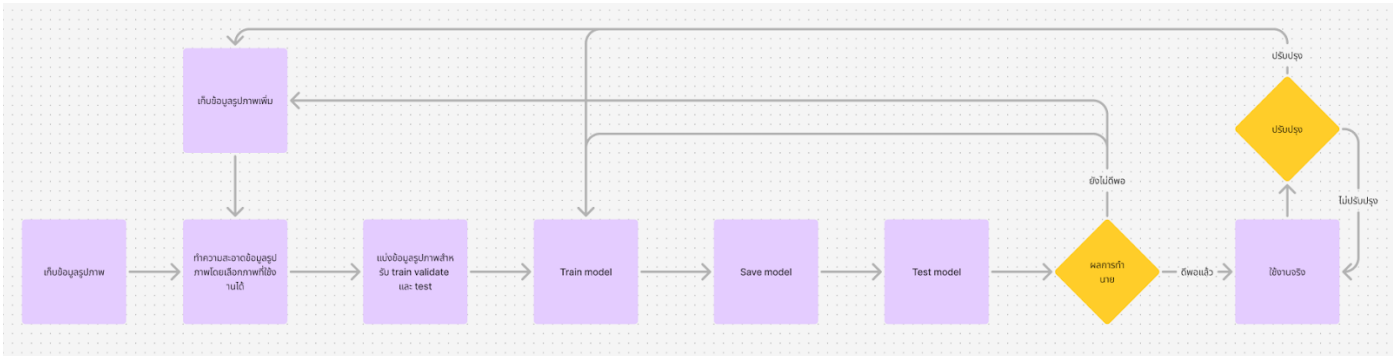
Detection (COCO) | Detection (Open Images V7) | Segmentation (COCO) | Classification (ImageNet) | Pose (COCO) | OBB (DOTAv1)

See Detection Docs for usage examples with these models trained on COCO, which include 80 pre-trained classes.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

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

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1

Plan Duration | Actual Start | % Complete | Actual (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Collect Data:	5	1	5	1	100%
Project planing	5	1	5	1	100%
Project Flowchart Overview	5	1	5	1	100%
Data preprocessing	5	1	5	1	100%
Data annotation	5	1	5	1	100%
Reviewing design 3-Feb-2024	5	1	6	1	100%
Model Develop:	6	2	6	1	100%
Model Architecture Design	6	1	6	1	100%
Model Components	6	1	6	1	100%
Model Input	6	1	6	1	100%
Model Config and Train	6	1	6	1	100%
Reviewing design based on Meeting	6	2	6	1	100%
Model Evaluation:	7	1	6	1	100%
Model Test	7	1	6	1	100%
Performances	7	1	6	1	100%
Reviewing design based on Meeting	7	1	6	1	100%
Deploy:	8	1		1	0%
Deployment Design by Flask-python	8	1		1	0%
Flask File Components	8	1		1	0%
Authentication & Checking Standard	8	1		1	0%
Fixing issues that arise during testing	8	1		1	0%
Reviewing design based on Meeting	8	1		1	0%
Presentation:	9	1		1	0%
Presentation Preparation	9	1		1	0%
Reviewing design based on Meeting	9	1		1	0%

PERIODS (WEEK)

January: 1 2 3 4 5 | Febuary: 6 7 8 9 | March: 10 11 12 13 14

หมายเหตุเนื่องจากผู้สั่งงานมีการเลื่อนกำหนดส่งให้ไวขึ้นโดยกระทันหัน กำหนดการจึงคลาดเคลื่อนไป 1 สัปดาห์

4.การพัฒนาโมเดล

4.1.การเตรียมข้อมูล

ข้อมูลที่ใช้เป็นข้อมูลรูปภาพของคนจาก 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

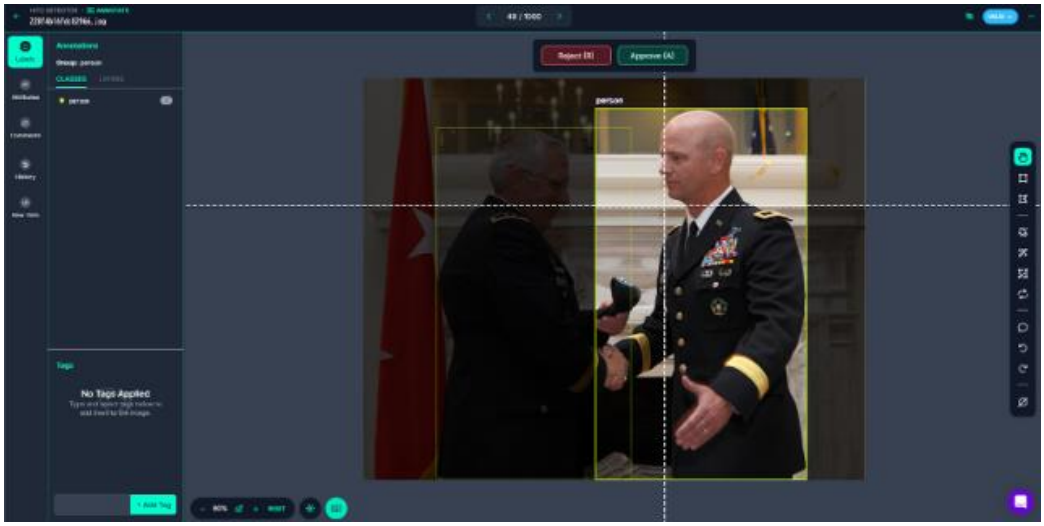
Data set v1: 701-200-99(70-20-10)

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) และชุดของป้ายชื่อกำกับ(label) มาใส่ในFolderที่เตรียมไว้ดังรูปด้านล่าง แล้ว upload ลง Google Drive datasets และ datasetsv2 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)	0.002
optimizer	ตัวเลือกอัลกอริธึมสำหรับการฝึกสอน (SGD, Adam, AdamW)	AdamW
Model	โมเดล เช่น Yolov8n, Yolov8s, Yolov8m, Yolov8l, Yolov8x	Yolov8n

```
                   all        265        322      0.827      0.755      0.813      0.512

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     96/100      2.2G     0.6872     0.5906      1.144          1        640: 100% 59/59 [00:18<00:00,  3.21it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:03<00:00,  2.63it/s]
                  all        265        322      0.809      0.755      0.805      0.502

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     97/100     2.25G     0.6368     0.5773      1.093          2        640: 100% 59/59 [00:19<00:00,  3.09it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:02<00:00,  3.43it/s]
                  all        265        322      0.831      0.748      0.805      0.501

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     98/100      2.2G     0.6486     0.6809      1.115          0        640: 100% 59/59 [00:20<00:00,  2.90it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:02<00:00,  3.25it/s]
                  all        265        322      0.822      0.742      0.802      0.506

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     99/100     2.19G     0.6734     0.6111      1.129          7        640: 100% 59/59 [00:18<00:00,  3.11it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:03<00:00,  2.28it/s]
                  all        265        322      0.823      0.745      0.805      0.508

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    100/100      2.2G     0.6447     0.5635      1.102          2        640: 100% 59/59 [00:18<00:00,  3.13it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:02<00:00,  3.44it/s]
                  all        265        322      0.845      0.727      0.813      0.512

100 epochs completed in 0.722 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.3MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.103 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 168 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 9/9 [00:06<00:00,  1.44it/s]
                  all        265        322      0.849      0.731      0.813      0.516
Speed: 0.7ms preprocess, 3.0ms inference, 0.0ms loss, 4.2ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

4.2.3 ผลลัพธ์ของการฝึกโมเดล

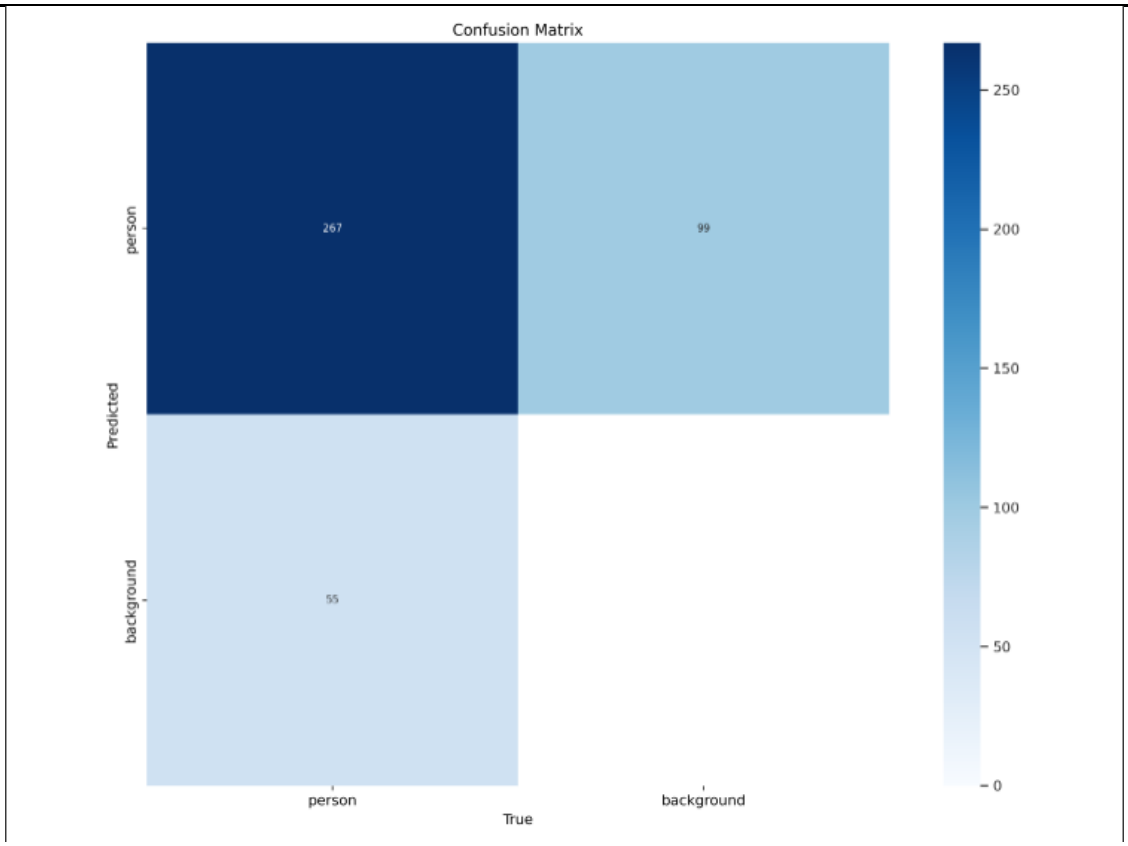
ตารางบันทึกผลการทดลองเพื่อหาโมเดล AI ที่ดีที่สุด

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%

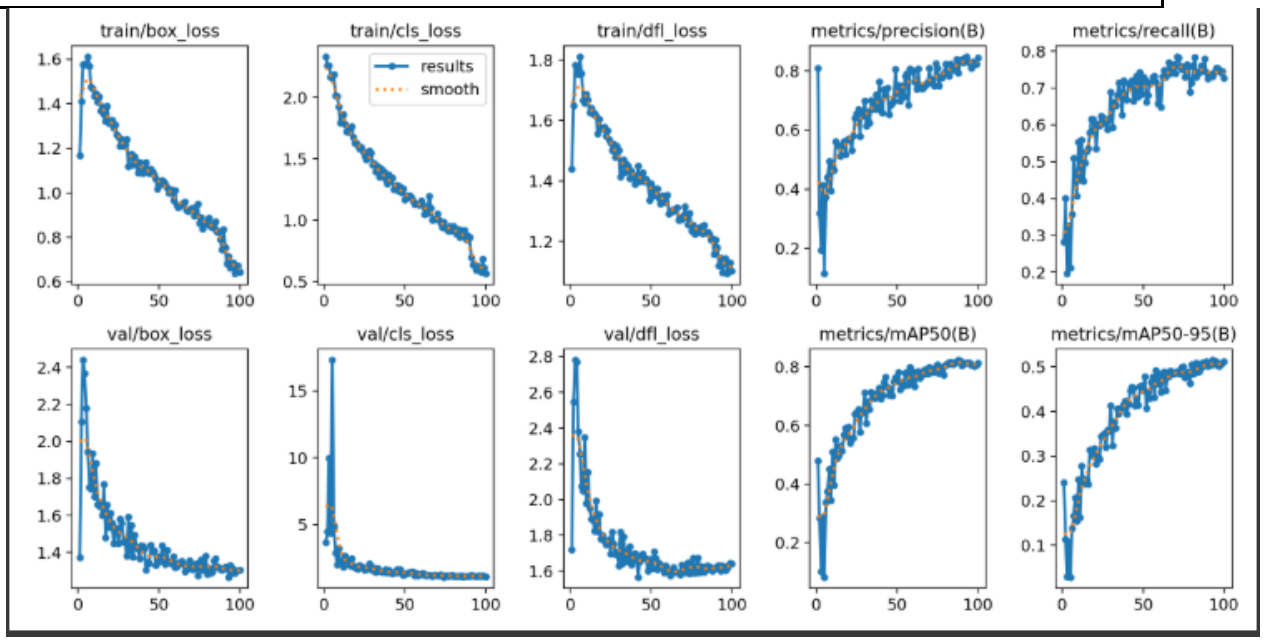
4.2.3.1.ผลลัพธ์ในรูปแบบ Confusion Matrices

```
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)
```

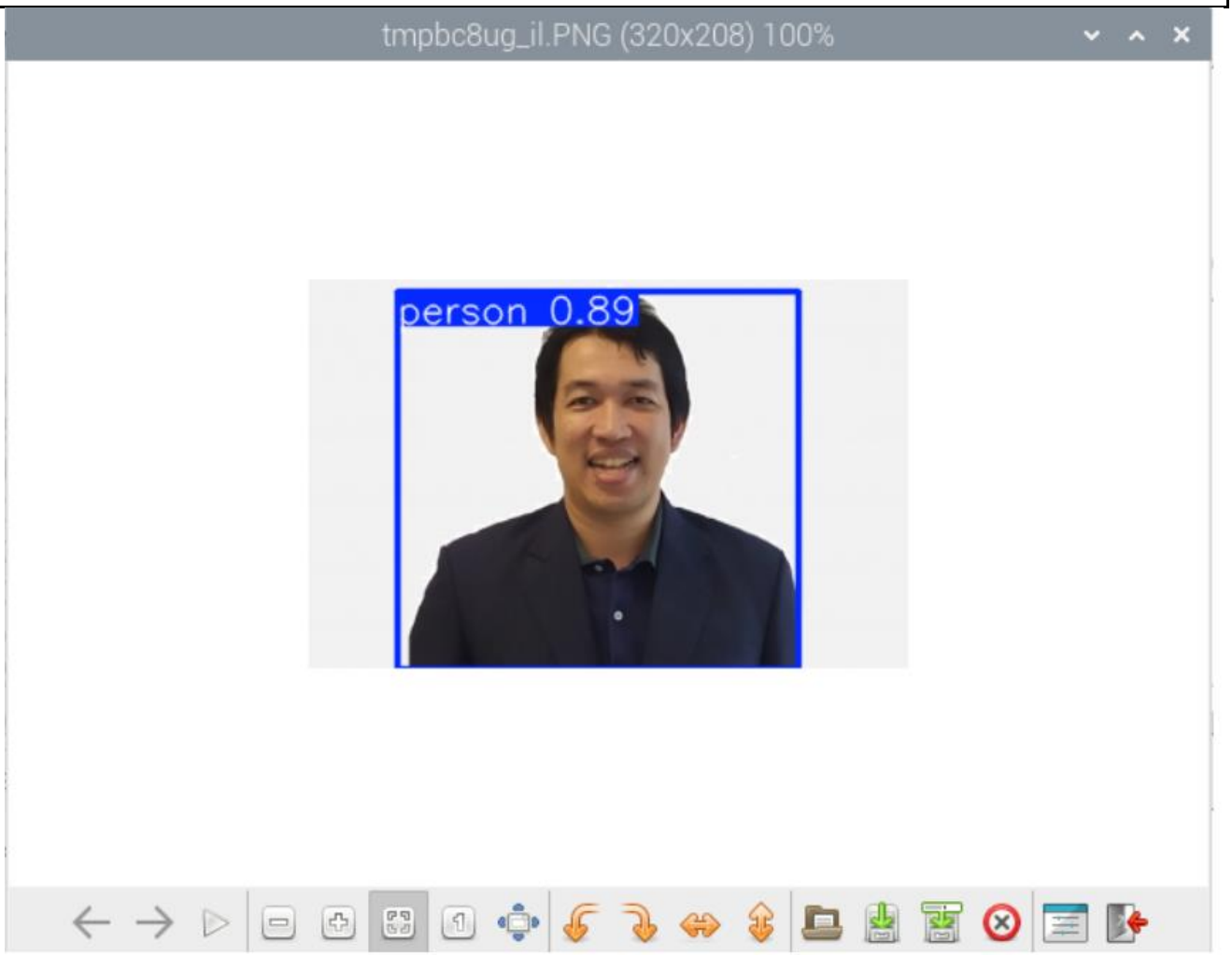
4.4.นำโมเดลที่ฝึกได้ไปทดสอบใน Raspberry Pi โดยใช้โปรแกรม Thonny

```
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**

-**What is Computer Vision? A Beginner Guide to Image Analysis | DataCamp**

-RPI-for AIIoT_Chp06.pdf

-**RPI-for AIIoT_Chp07.pdf**

-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**