โครงงาน Project-Based Learning 3 (PBL3)

Object Detection

รายวิชา 968-352 Machine Learning

วิทยาลัยการคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต

อาจารย์ประจำวิชา : ผศ.ดร. ขวัญกมล ดิฐกัญจน์

ผู้รับผิดชอบโครงการ

นาย รัฐภูมิ รอดนิล **รหัสนักศึกษา** 6630611025

หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาการคอมพิวเตอร์

ภาคเรียนที่ 1 **ปีการศึกษา** 2568

การเตรียมความพร้อมของข้อมูล

การเตรียมความพร้อมข้อมูลของโครงงานนี้เตรียมความพร้อมข้อมูลด้วย roboflow โดยใน roboflow ก็จะมีการเตรียมพร้อมด้วยการทำ Label และการแบ่งข้อมูลเพื่อใช้ทำ Model

1. การนำเข้าข้อมูล (Import Data)

ในขั้นแรก ผู้ทำทำการรวบรวมข้อมูลวิดีโอจากลิงก์ที่ผู้สอนให้ จากนั้นทำการแยกเฟรม (Extract Frames) ออกมาเป็นภาพนิ่ง โดยใช้การ 2 FPS ได้มา ~ 978 รูปภาพ เพื่อให้จำนวนภาพเพียงพอสำหรับการฝึกโมเดล
และกำหนด Label ตามโจทย์ ได้แก่

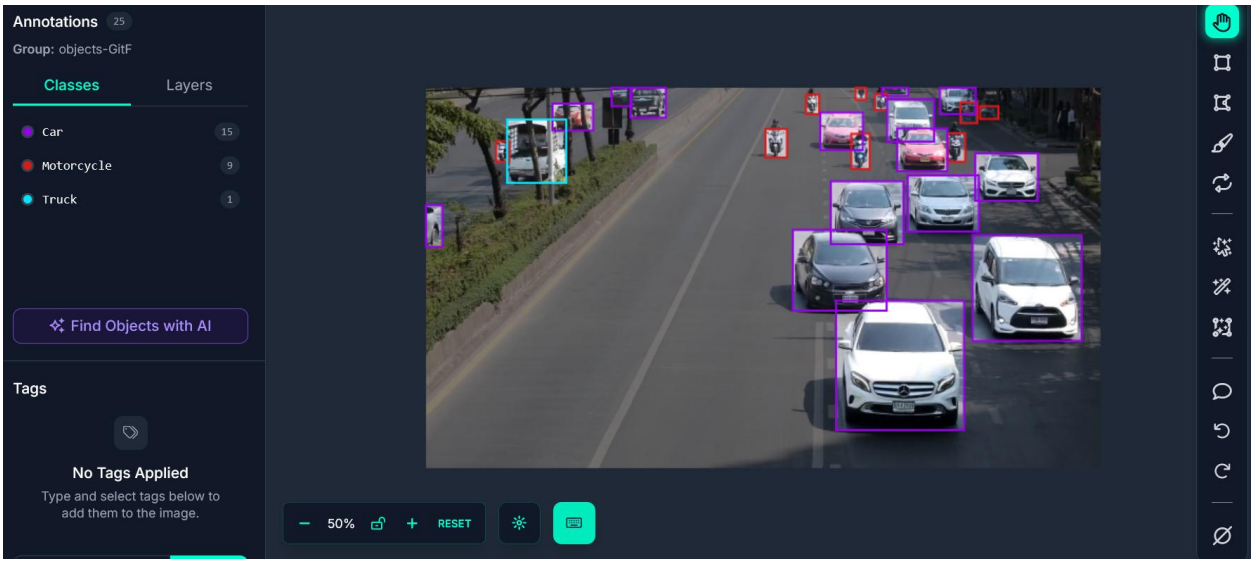
- รถยนต์ (Car)
- รถจักรยานยนต์ (Motorcycle)
- รถบรรทุก (Truck)

หลังจากนั้น ก็คัดกรองภาพที่ไม่มีประสิทธิภาพออกไป ก่อนเข้าสู่ขั้นถัดไป

2. การทำ Label ด้วย Roboflow Annotate

หลังจากอัปโหลดภาพทั้งหมดแล้ว ผู้ทำใช้เครื่องมือ **Roboflow Annotate** ในการทำ Bounding Box ให้กับยานพาหนะแต่ละประเภท
ขั้นตอนประกอบด้วย

1. เปิดภาพทีละเฟรม
2. ลากกรอบ Bounding Box ครอบวัตถุ
3. เลือก Label ให้ถูกต้อง
4. ตรวจสอบความถูกต้องก่อนกด Save



เพื่อให้ข้อมูลมีคุณภาพสูง มีหลักการกำกับดังนี้

- กรอบต้อง ครอบชิดวัตถุ (tight bounding box)
- กรอบไม่ควรเกินเลยออกนอกภาพ
- ภาพที่เบลอหรือเห็นไม่ชัดมากจะถูกลบออกเพื่อลด Noise
- วัตถุที่มีหลายคันในภาพต้องแยกกรอบออกจากกันทุกครั้ง

3. การแบ่งชุดข้อมูล

Roboflow ช่วยแบ่งข้อมูลอัตโนมัติ โดยกำหนดสัดส่วนเป็น

- Train 80%
- Validate 20%

หลังจากแบ่งแล้ว Roboflow จะแสดงสถิติเช่น

- จำนวนภาพแต่ละชุด
- จำนวนวัตถุในแต่ละ Class
- Visualization ของ distribution เพื่อให้ตรวจสอบได้ว่าไม่มี class ไหนถูกแบ่งน้อยเกินไป

ผลลัพธ์ที่ได้:

- Train: 608 images
- Validate: 152 images

Data Modeling

ในการทำ Object Detection ของโครงงานนี้ จะใช้ 3 โมเดล ได้แก่ Yolo11, Yolo8, Yolo5

กำหนดตัวแปลพื้นฐาน

```
5
4  DATA_YAML = ["yolov11/data.yaml","yolov8/data.yaml","yolov5/data.yaml"]
3  MODEL_WEIGHTS = ["yolo11n.pt","yolov8n.pt","yolov5s.pt"]
2
1  PROJECT = ["runs0/train","runs1/train","runs2/train"]
6  EXPERIMENT_NAME = ["exp_yolo11_custom","exp_yolo8_custom","exp_yolo5_custom"]
1
2  EPOCHS = 50
3  BATCH_SIZE = 16
4  IMAGE_SIZE = 640
5
6
```

List Variable

- Data_YAML
    1. คือ location ของข้อมูลที่ Locate รูปภาพที่แบ่งมาแล้ว
- MODEL_WEIGHTS
    1. เลือก model ที่จะใช้
    2. เลือก weight ที่จะใช้
- PROJECT, EXPERIMENT_NAME
    1. Path ที่ผลลัพธ์จะเข้าไปอยู่หลังจบการเทรน
       ชื่อไฟล์

Integer Variable

- EPOCHS = 50

    1. โมเดลจะวนอ่าน dataset ทั้งหมด 50 รอบ
    2. มากขึ้น = แม่นขึ้น แต่เสี่ยง overfit
3. BATCH_SIZE = 16
    1. จำนวนภาพที่ส่งเข้าโมเดลต่อ 1 รอบ
    2. ถ้าน้อยเกินไป ⟶ ช้า
    3. ถ้ามากเกินไป ⟶ GPU RAM ไม่พอ
4. IMAGE_SIZE = 640
    1. ขนาดรูปหลังจาก resize ก่อนเข้า YOLO
    2. 640 × 640 เป็นค่ามาตรฐาน

```python
def train(data_yaml_idx, model_weight_idx, project_idx, experiment_name_idx,modelName,device=0 ):
    data_yaml = DATA_YAML[data_yaml_idx]
    model_weight = MODEL_WEIGHTS[model_weight_idx]

    model = YOLO(model_weight)

results = model.train(
    data=data_yaml,
    epochs=EPOCHS,
    imgsz=IMAGE_SIZE,
    batch=BATCH_SIZE,
    project=PROJECT[project_idx],
    name=EXPERIMENT_NAME[experiment_name_idx],
    pretrained=True,
    patience=20,
    lr0=0.01,
    device=device,
    weight_decay=0.0005,
    optimizer="SGD",
)
```

ฟังก์ชัน train() ถูกออกแบบมาเพื่อจัดการกระบวนการฝึกโมเดล YOLO หลายรุ่นภายในโครงสร้างที่ยืดหยุ่น โดยใช้การอ้างอิงผ่าน index เพื่อลดการเขียนโค้ดซ้ำ และเพิ่มความเป็นระบบในการเปรียบเทียบโมเดลหลายเวอร์ชัน ฟังก์ชันนี้มีบทบาทสำคัญใน data configuration, เลือกน้ำหนักเริ่มต้นของโมเดล pretrained weights, ตั้งค่าพารามิเตอร์สำหรับการฝึก และเรียกใช้กระบวนการฝึกด้วยไลบรารี Ultralytics

1. การเลือกไฟล์ข้อมูลและ weight โมเดล

data_yaml = DATA_YAML[data_yaml_idx]

model_weight = MODEL_WEIGHTS[model_weight_idx]

คำสั่งสองบรรทัดนี้ทำหน้าที่เลือกไฟล์ data.yaml และไฟล์น้ำหนักเริ่มต้น .pt โดยอ้างอิงจากลิสต์ที่กำหนดไว้ล่วงหน้า การเลือกผ่าน index ช่วยให้ฟังก์ชันสามารถรองรับโมเดลหลายเวอร์ชันได้โดยไม่ต้องเขียนโค้ดซ้ำ

- data_yaml อธิบายโครงสร้างชุดข้อมูล เช่นเส้นทางของชุด train/validation และรายการ class
- model_weight คือ weight ที่ใช้สำหรับ transfer learning

2. การโหลดโมเดล YOLO

model = YOLO(model_weight)

คำสั่งนี้โหลดโมเดล YOLO พร้อม weight เริ่มต้นเพื่อเตรียมพร้อมสำหรับการฝึก การใช้ pretrained weights ทำให้โมเดลสามารถเรียนรู้ได้เร็วขึ้นและลดความเสี่ยงของการเรียนรู้ที่ไม่เสถียร โดยเฉพาะในกรณีที่จำนวนข้อมูลฝึกมีจำกัด

3. การตั้งค่าการฝึกโมเดล

ส่วนของโค้ดด้านล่างคือการกำหนดพารามิเตอร์สำหรับการฝึก

```
results = model.train(
    data=data_yaml,
    epochs=EPOCHS,
    imgsz=IMAGE_SIZE,
    batch=BATCH_SIZE,
    project=PROJECT[project_idx],
    name=EXPERIMENT_NAME[experiment_name_idx],
    pretrained=True,
    patience=20,
    lr0=0.01,
    device=device,
    weight_decay=0.0005,
    optimizer="SGD",
)
```

รายละเอียดของแต่ละพารามิเตอร์มีดังนี้

3.1 พารามิเตอร์ข้อมูลและโครงสร้างการจัดเก็บผลลัพธ์

- data: ไฟล์ data.yaml ที่ระบุชุดข้อมูลที่ใช้ในการฝึกและตรวจสอบ
- project: โฟลเดอร์หลักที่ใช้เก็บผลลัพธ์ของการฝึก

- name: ชื่อย่อยของ experiment เพื่อแยกผลลัพธ์ของแต่ละโมเดล

การกำหนดชื่อและโครงสร้างโฟลเดอร์ช่วยให้สามารถจัดการและเปรียบเทียบผลลัพธ์ของโมเดลต่าง ๆ ได้อย่างเป็นระบบ

3.2 **พารามิเตอร์สำหรับการฝึก**

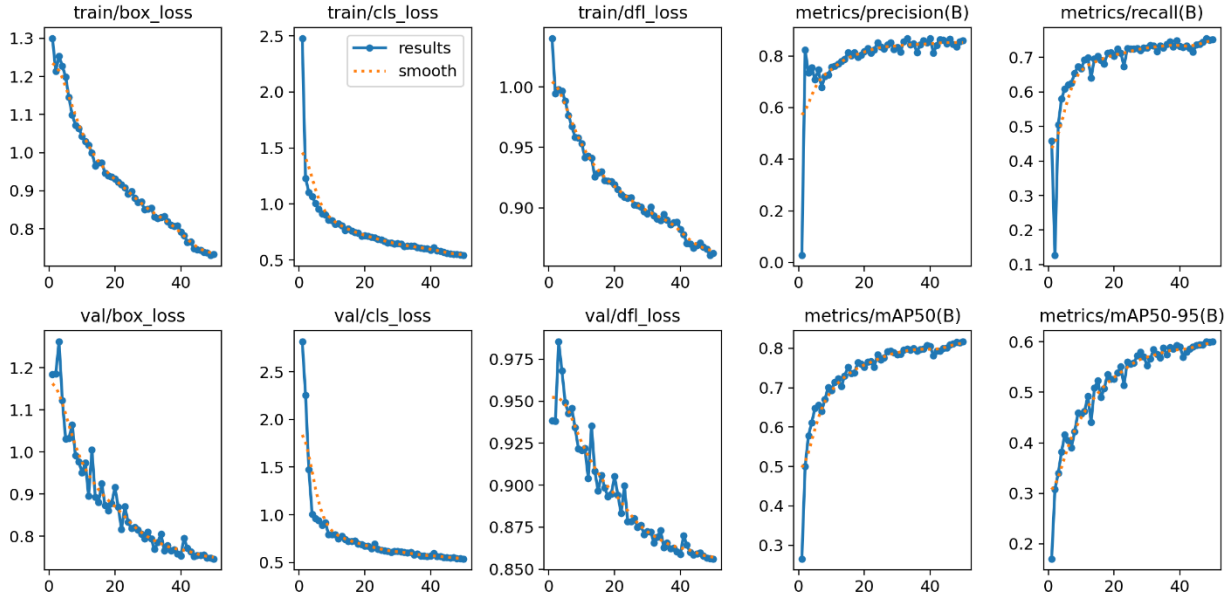
- epochs: จำนวนรอบการฝึกของโมเดล
- imgsz: ขนาดของภาพที่ป้อนเข้าสู่โมเดล
- batch: จำนวนตัวอย่างข้อมูลที่ใช้ในแต่ละขั้นตอนการอัปเดตน้ำหนัก
- pretrained=True: ใช้น้ำหนักที่ผ่านการฝึกเบื้องต้นเพื่อเพิ่มประสิทธิภาพในการเรียนรู้
- lr=0.01: อัตราการเรียนรู้ ซึ่งกำหนดความเร็วในการอัปเดตน้ำหนัก
- weight_decay=0.0005: ค่าการลดน้ำหนักเพื่อควบคุมความซับซ้อนของโมเดลและลดการเกิด overfitting
- optimizer="SGD": ระบุว่าใช้ตัวปรับปรุงน้ำหนักแบบ Stochastic Gradient Descent
- patience=20: ตัวควบคุม early stopping เพื่อหยุดการฝึกเมื่อผลลัพธ์ไม่ดีขึ้นภายในจำนวนรอบที่กำหนด

3.3 **พารามิเตอร์ด้านฮาร์ดแวร์**

- device: ระบุว่าโมเดลจะฝึกบน GPU หรือ CPU การเลือก GPU (device=0) จะทำให้การฝึกดำเนินไปได้เร็วกว่า

ผลลัพธ์การทดลองโครงงาน

Yolo 11:

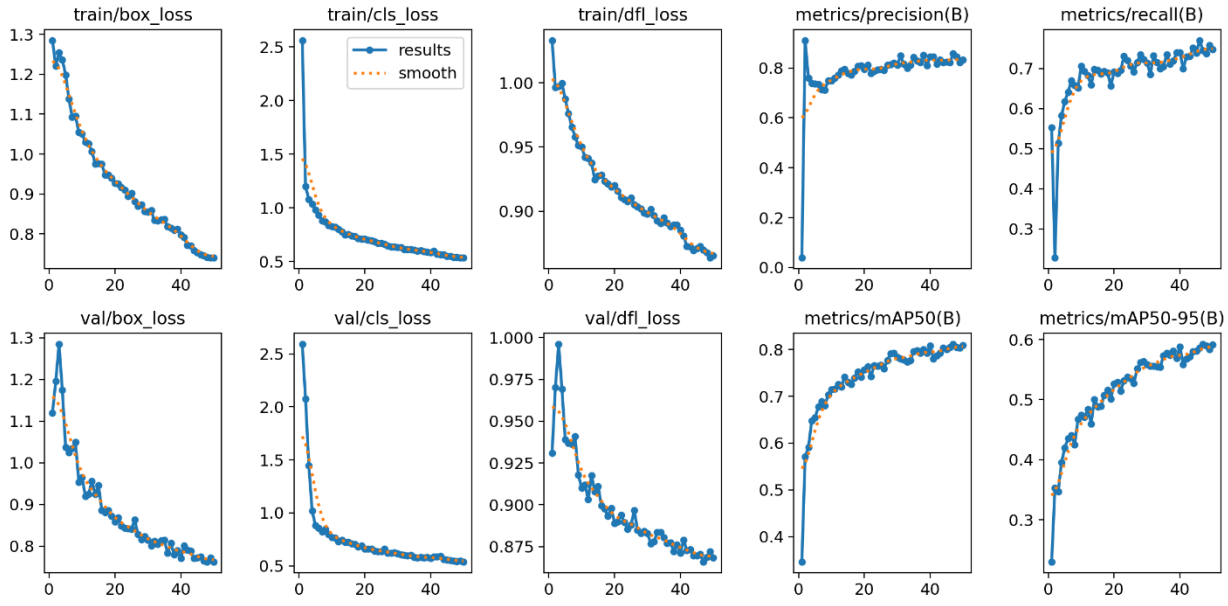


รูปภาพแสดงถึงค่าต่างๆ หลังเทรนโมเดล Yolo 11

รูปภาพแสดงถึงตัวอย่างการ label ในช่วง validation

รูปภาพแสดงถึงตัวอย่างการ predict ในช่วง validation

YOLO 8:

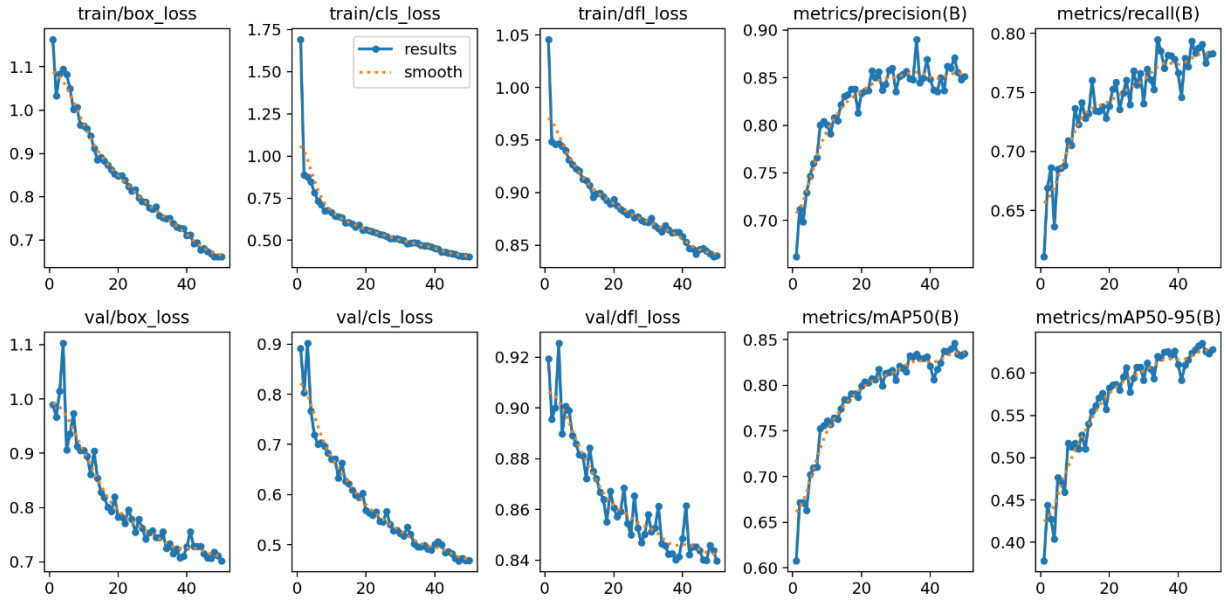


รูปภาพแสดงถึงค่าต่างๆ หลังเทรนโมเดล Yolo 8

รูปภาพแสดงถึงตัวอย่างการ label ในช่วง validation

รูปภาพแสดงถึงตัวอย่างการ predict ในช่วง validation

YOLO 5:

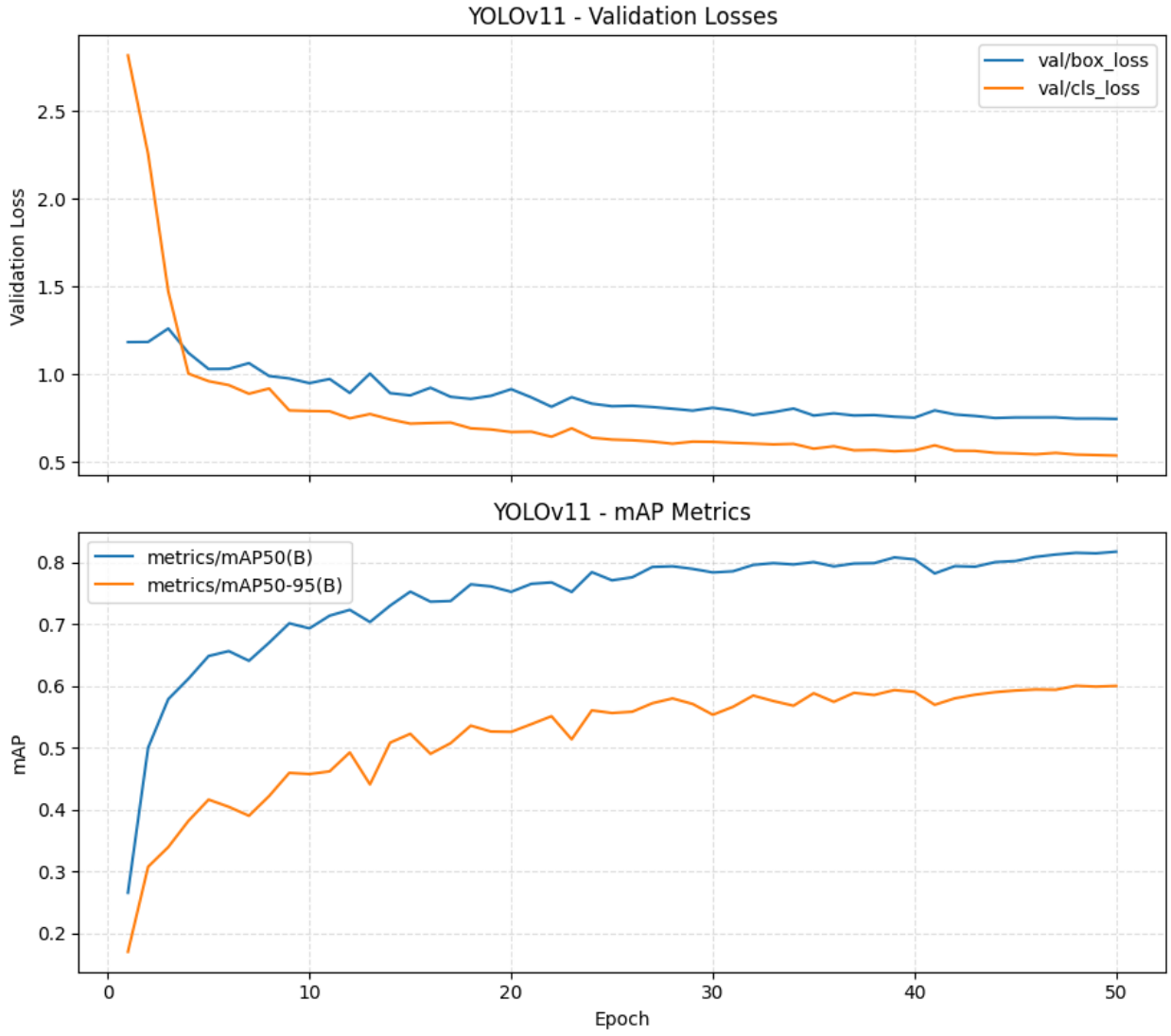


รูปภาพแสดงถึงค่าต่างๆ หลังเทรนโมเดล Yolo 5

รูปภาพแสดงถึงตัวอย่างการ label ในช่วง validation

รูปภาพแสดงถึงตัวอย่างการ predict ในช่วง validation

การประเมินประสิทธิภาพโมเดล



1. ผลการประเมินโมเดล YOLO11

1.1 Validation Loss

- box_loss ลดลงอย่างต่อเนื่องจากช่วงเริ่มต้นประมาณ 1.1 ลงสู่ค่าเฉลี่ยราว 0.70–0.75
- cls_loss ลดลงชัดเจนในช่วง 10 epochs แรก จาก 2.8 เหลือประมาณ 0.50–0.55

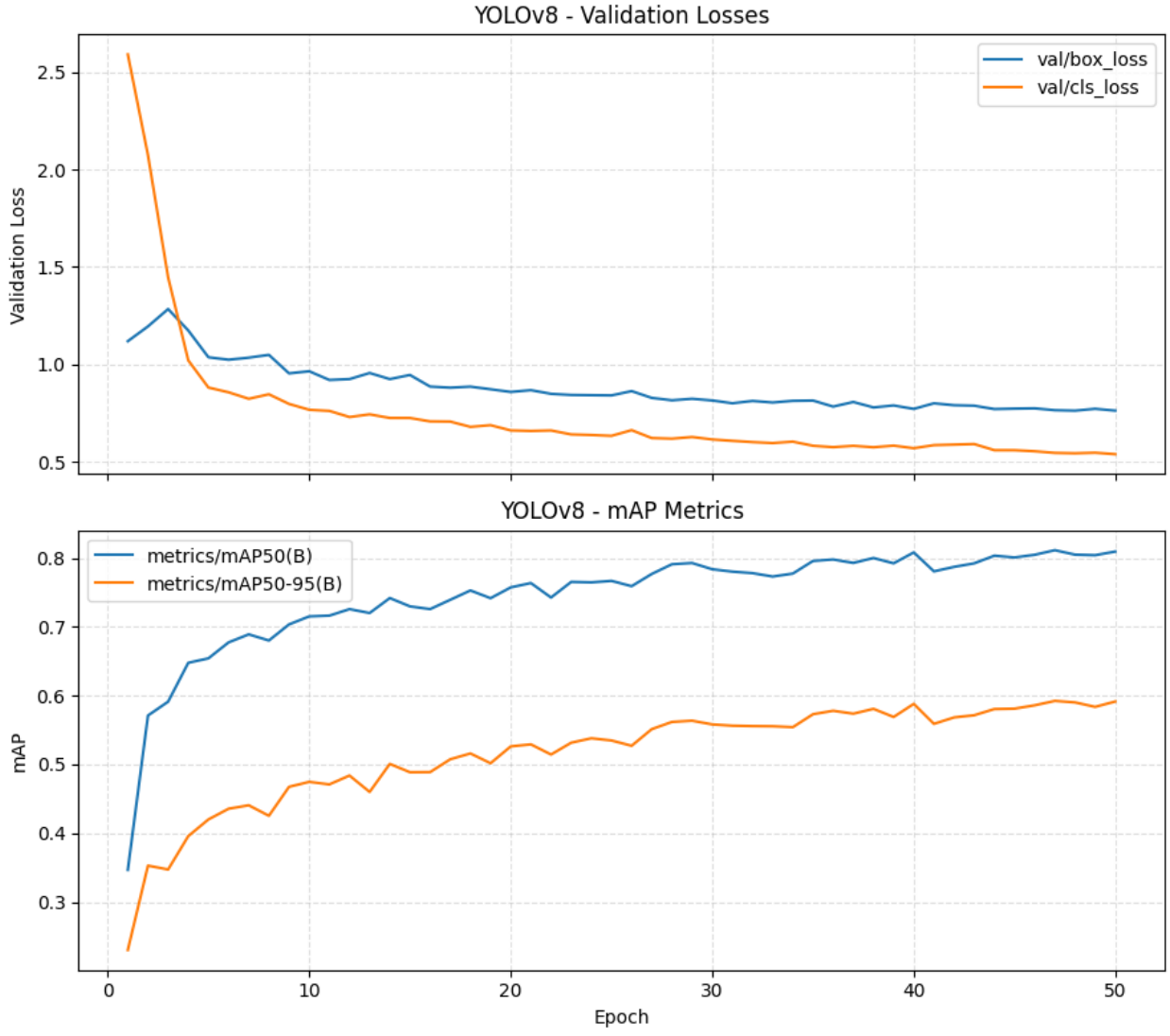
ภาพรวมแสดงให้เห็นว่าโมเดลมีการเรียนรู้ที่มั่นคงและไม่เกิด overfitting อย่างมีนัยสำคัญ

1.2 ค่า mAP

- mAP50 เพิ่มขึ้นต่อเนื่องจากประมาณ 0.28 ใน epoch แรก สู่ค่าคงที่บริเวณ 0.80–0.83
- mAP50–95 เพิ่มขึ้นจาก ~0.17 ไปอยู่ในช่วง 0.58–0.60

1.3 สรุปผลของ YOLO11

โมเดล YOLO11 ให้ค่า mAP สูงที่สุดในสามรุ่น และมี Loss ที่ลดลงเร็วที่สุด แสดงผลการเรียนรู้ที่มีเสถียรภาพ มีความแม่นยำสูง และสามารถตรวจจับวัตถุได้ดีในหลายระดับ IoU



2. ผลการประเมินโมเดล YOLO8

2.1 Validation Loss

- box_loss ลดจากราว 1.1 ลงสู่ระดับ 0.70–0.75 ใกล้เคียง YOLO11

- cls_loss จาก 2.6 ลดสู่ 0.50–0.55

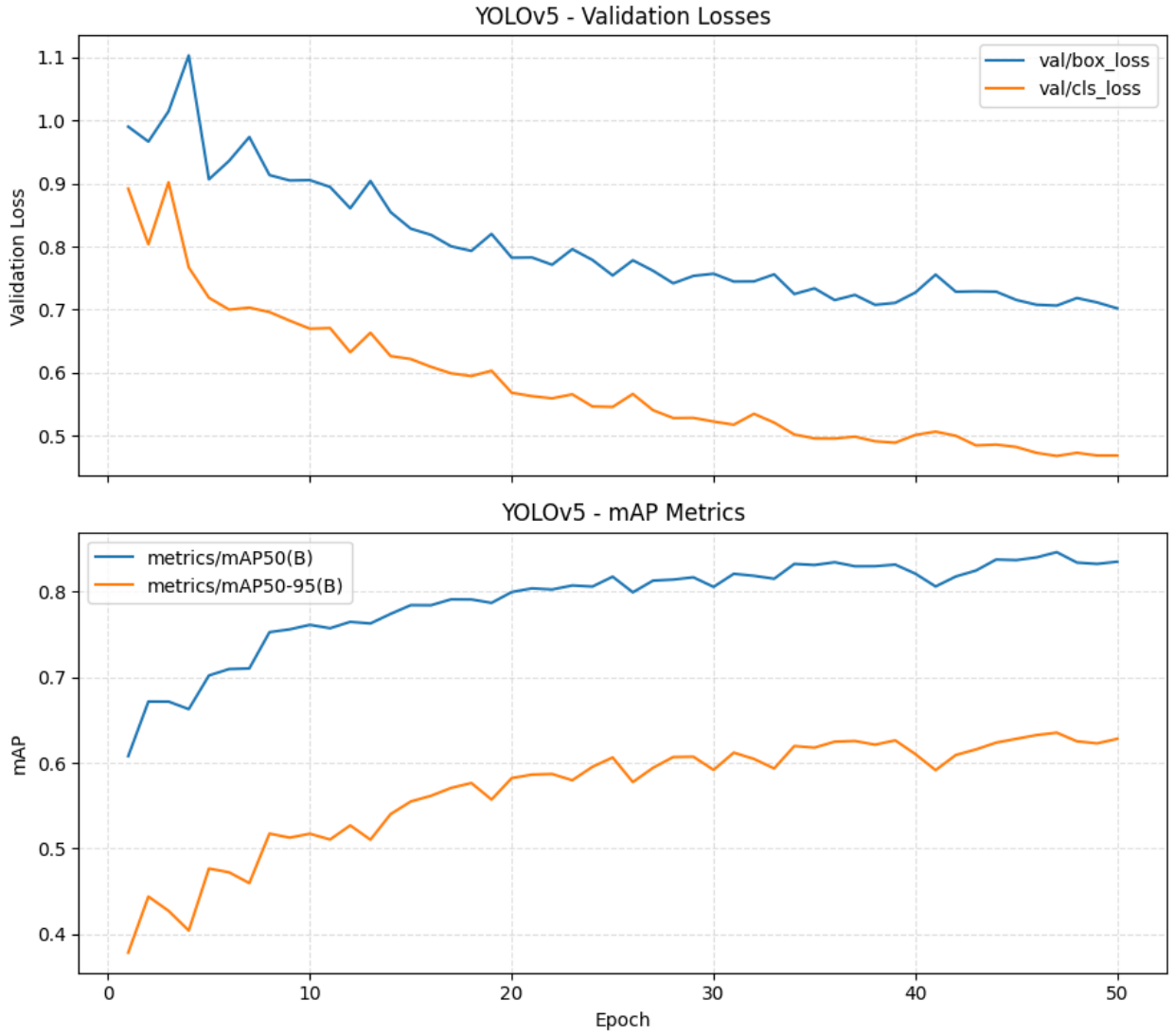
แนวโน้มการลดลงคล้ายกับ YOLO11 แต่มีค่า Loss โดยรวมสูงกว่าเล็กน้อยในบางช่วง

2.2 ค่า mAP

- mAP50 เริ่มจาก ~0.30 และเพิ่มขึ้นถึง 0.80–0.82
- mAP50–95 อยู่ในช่วง 0.55–0.58

2.3 สรุปผลของ YOLO8

ให้ประสิทธิภาพสูงใกล้เคียง YOLO11 แต่มีความผันผวนใน validation loss มากกว่าเล็กน้อย อย่างไรก็ตาม mAP50 ค่อนข้างสูงและมีความสม่ำเสมอ



3. ผลการประเมินโมเดล YOLO5

3.1 Validation Loss

- box_loss ลดลงจาก ~1.0 ลงสู่ค่าประมาณ 0.75–0.80 ซึ่งสูงกว่า YOLO11 และ YOLO8 เล็กน้อย
- cls_loss ลดลงจาก ~0.9 เหลือ ~0.45–0.48 ซึ่งใกล้เคียงกับสองรุ่นแรก

โดยรวม Loss ลดลงช้ากว่า YOLO11 และ YOLO8

3.2 ค่า mAP

- mAP50 ปรับตัวจากราว 0.60 ขึ้นสู่ประมาณ 0.82–0.84
- mAP50–95 อยู่ในช่วง 0.57–0.62 แม้จะใกล้เคียง YOLO11 แต่มีความผันผวนมากกว่า

3.3 สรุปผลของ YOLO5

แม้ YOLO5 จะมีค่า mAP50 ใกล้ YOLO11 แต่ค่า Loss สูงกว่าและเสถียรน้อยกว่า แสดงว่า YOLO5 จับกรอบได้ไม่แม่นเท่า YOLO11 ในด้าน localization

โมเดล	**box_loss ต่ำสุด**	**cls_loss ต่ำสุด**	**mAP50 สูงสุด**	**mAP50–95 สูงสุด**	**ภาพรวม**
YOLO11	~0.70	~0.50	0.82–0.83	0.58–0.60	ดีที่สุด
YOLO8	~0.70	~0.50	0.80–0.82	0.55–0.58	ดี
YOLO5	~0.75–0.80	~0.45	0.82–0.84	0.57–0.62	ดี แต่ Loss สูงกว่า

5. สรุปผลรวม (Conclusion)

จากผลการประเมินเชิงปริมาณ โมเดล YOLO11 แสดงประสิทธิภาพดีที่สุดในภาพรวม ทั้งในด้านความแม่นยำ (mAP) และการลดลงของ Loss แม้ว่า YOLO5 จะมีค่า mAP50 ใกล้เคียงหรือบางครั้งสูงกว่าเล็กน้อย แต่ YOLO11 มีความสม่ำเสมอกว่าทั้งคะแนนเฉลี่ยและความเสถียรของกราฟ นอกจากนี้อัตราความเร็วในการลดลงของ Loss ยังชี้ให้เห็นว่า YOLO11 สามารถเรียนรู้ลักษณะของวัตถุได้เร็วและมีประสิทธิภาพมากที่สุด

ดังนั้น สำหรับงานตรวจจับยานพาหนะในชุดข้อมูลนี้ YOLO11 เป็นตัวเลือกที่มีประสิทธิภาพสูงสุด