

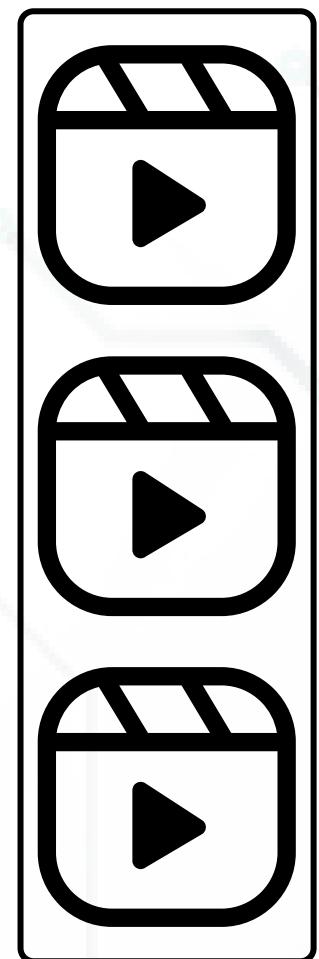


# นับจำนวนและ จำแนกประเภท ของยานพาหนะ

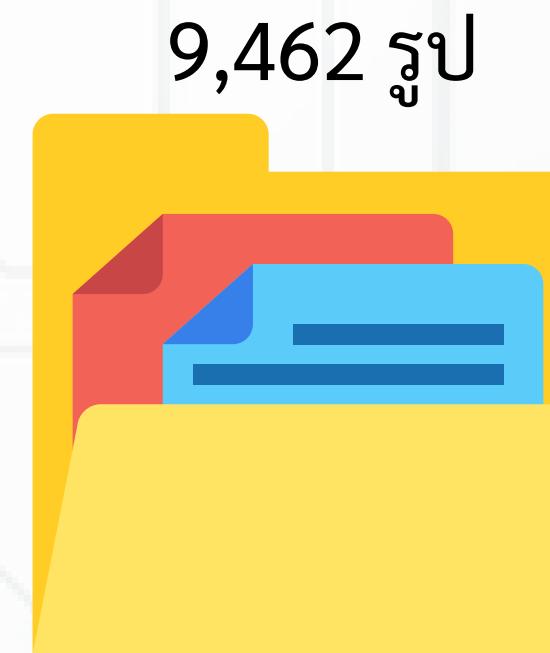
โดย : พิทักษ์ วงศ์



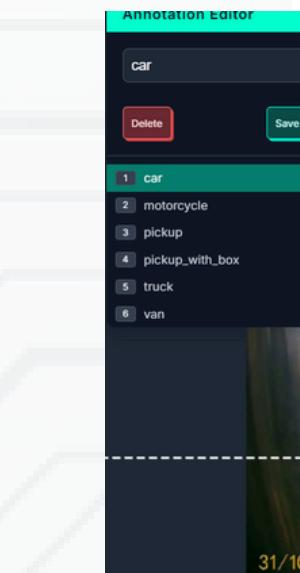
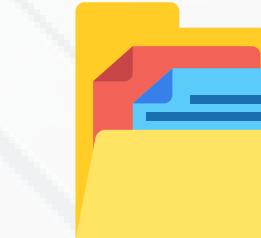
# ขั้นตอนการทำงาน



ดึงเฟรมภาพจากไฟล์วิดีโอ(1 fps)



สุ่ม 500 รูป เพื่อนำไป label ด้วยมือ



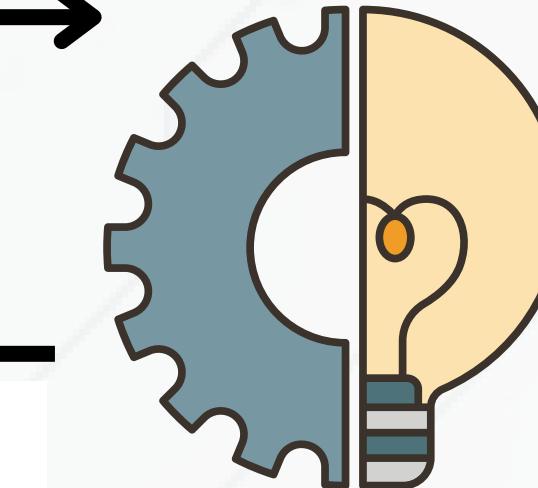
roboflow



train model เพื่อนำไปช่วย label เพิ่ม

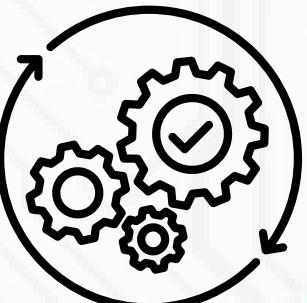


ultralytics  
YOLOv11n



รวม 2,000 รูป

Auto Labeling...



นำ model ไปใช้นับรถแต่ละประเภท

นำไป train model เพื่อใช้งาน

ultralytics  
YOLOv11s

ข้อมูลที่พร้อมนำไปเทรน กับโมเดลที่จะใช้จริงๆ

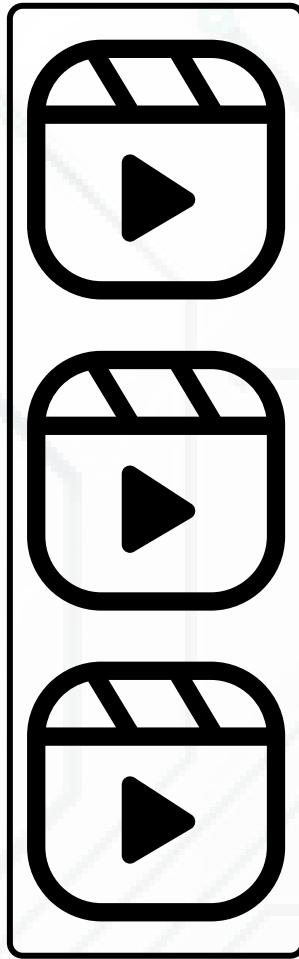


รวม 2,000 รูป

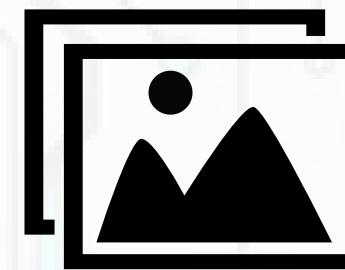
นำไปตรวจสอบความถูกต้อง

roboflow

# การทำงานดึงข้อมูล

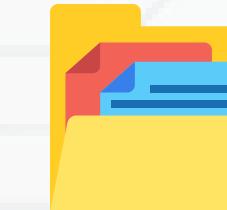
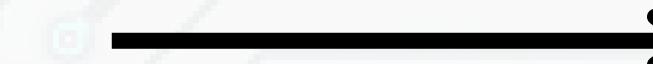


ดึงเฟรมภาพจาก  
ไฟล์วิดีโอ(1 fps)

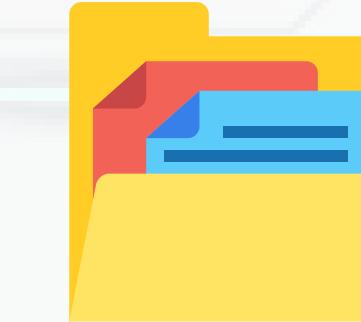


9,462 รูป

สุ่ม 500 รูป เพื่อนำ  
ไป label ด้วยมือ

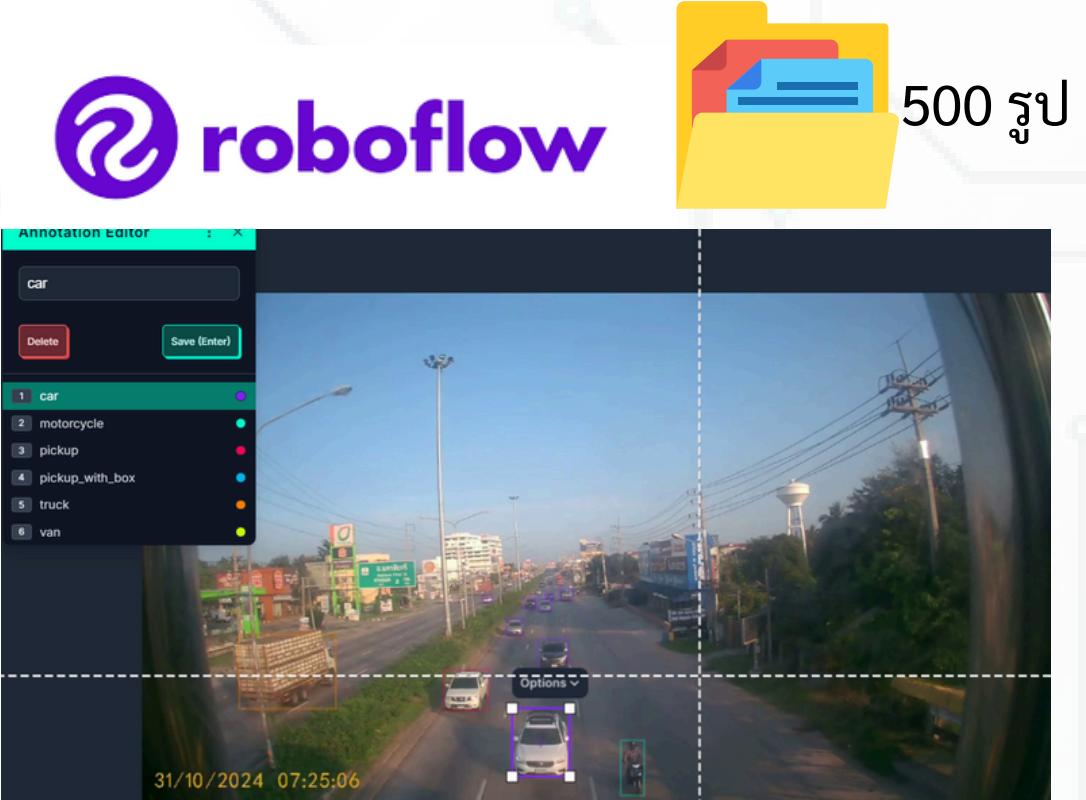


สุ่ม 1,500 รูป



เพื่อทำการดึงรูปภาพจากวิดีโอด้วยจะดึงเป็น 1 เฟรมต่อวินาที แล้วแบ่งข้อมูล 2 ชุด คือ 500 รูป และ 1500 รูป เพื่อนำข้อมูลชุดแรก (500รูป) ไปทำการ label ข้อมูลด้วยมือก่อน แล้วนำไปเทรนโมเดลสำหรับการ auto labeling ข้อมูลที่เหลือเพื่อลดเวลาการ label ข้อมูล

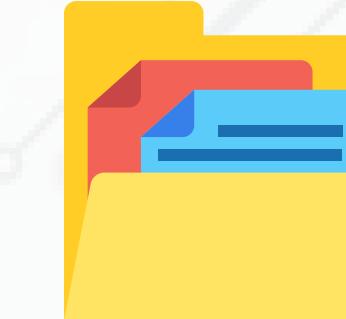
# การ label ข้อมูล



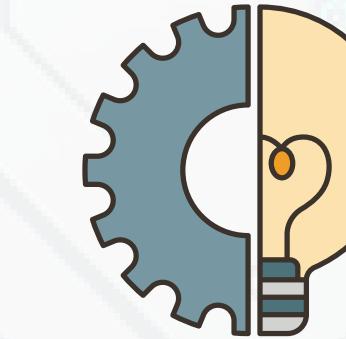
ultralytics

YOLOv11n

train model เพื่อนำ  
ไปช่วย label เพิ่ม



1,500 รูป

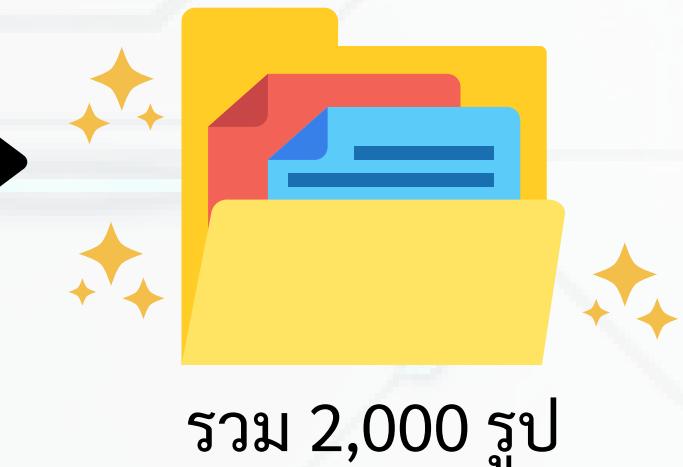


Auto Labeling...

roboflow

ตรวจสอบความ  
ถูกต้อง

ข้อมูลที่พร้อมนำไปเทรน  
กับโมเดลที่จะใช้จริงๆ

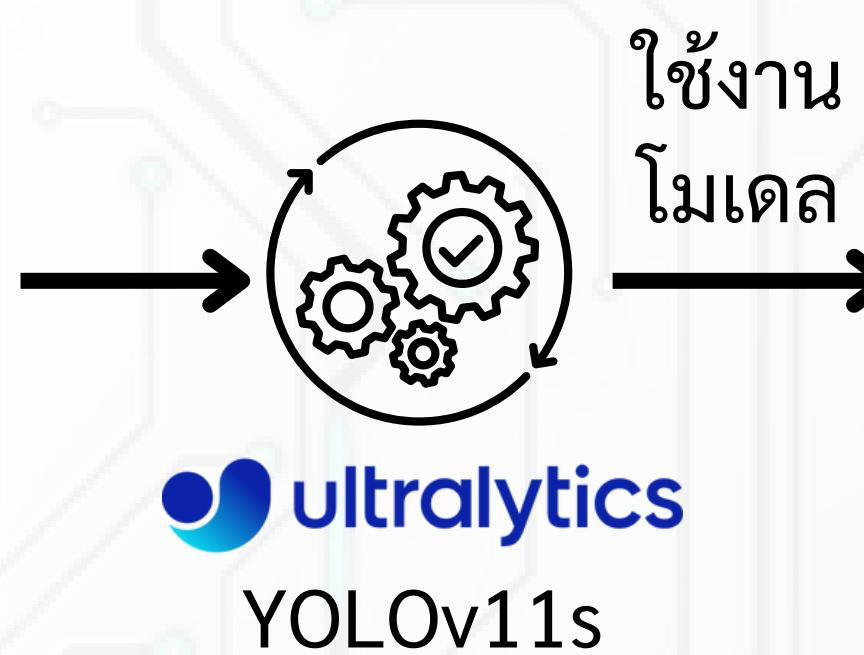


รวม 2,000 รูป

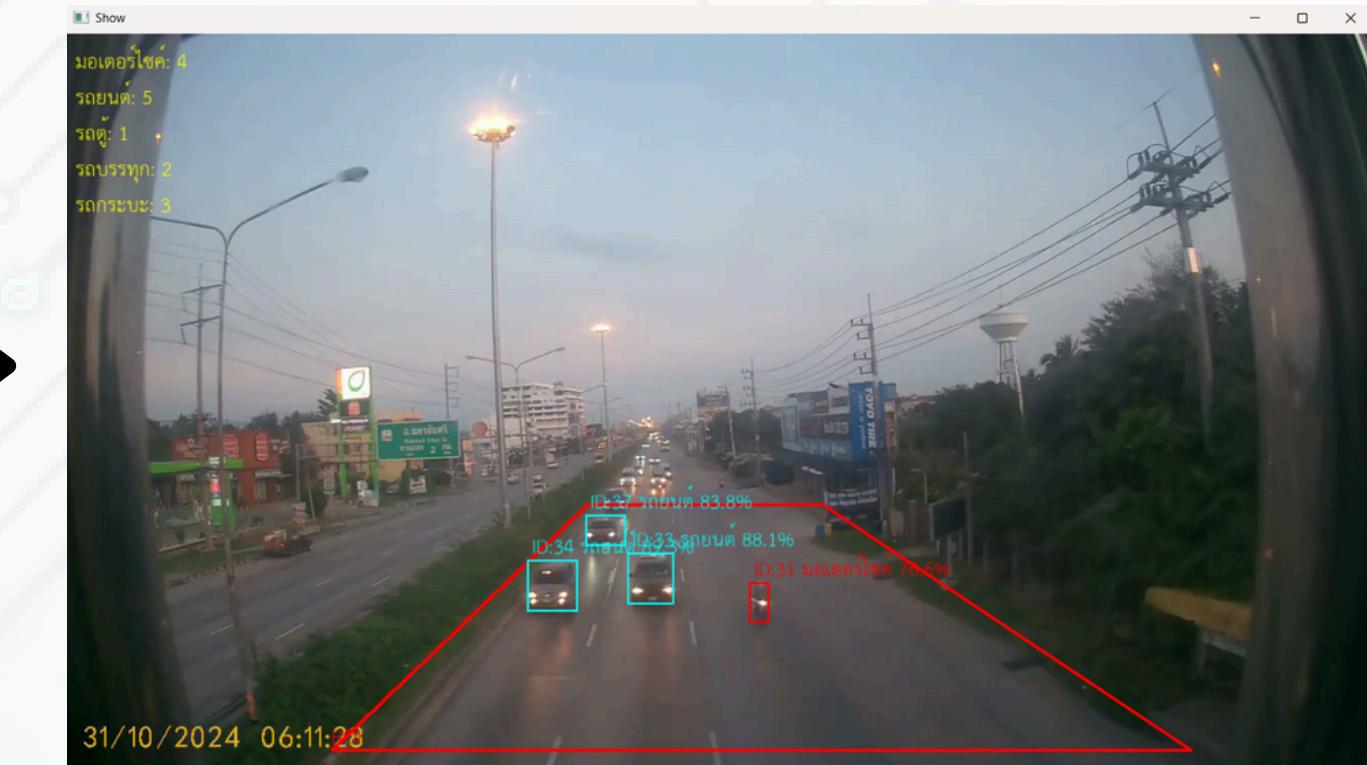
นำข้อมูล 500 รูป ไปทำการ label ข้อมูลด้วย Roboflow และนำไปเทรนโมเดล YOLOv11n ที่เป็นโมเดลไม่ใหญ่มากและเร็วเพื่อนำโมเดลไปใช้สำหรับการ auto labeling ข้อมูล 1500 รูป จากนั้นนำไปตรวจสอบความถูกต้องด้วย Roboflow อีกรอบเนื่องจากการ auto อาจมีจุดผิดพลาดเล็กน้อย การเช็คด้วยมืออีกรอบจะทำให้แม่นยำมากกว่า

# model

ข้อมูลที่พร้อมนำไปเทรน  
กับโมเดลที่จะใช้จริงๆ



แสดง  
ผลลัพธ์



ใช้ข้อมูลที่ 2,000 รูป ในการเทรนโมเดล YOLOv11s ซึ่งเป็นโมเดลที่ใหญ่กว่า YOLOv11n ทำให้  
แม่นยำกว่าแต่ก็จะประมวลผลช้ากว่า เช่นกัน โดยทำโมเดลที่ได้ไปเขียน script เพื่ออ่านวิดีโอแล้วนับ  
จำนวนของยานพาหนะแต่ละประเภทแล้วบันทึกเป็นวิดีโอหรือแสดงผลในรูปแบบ GUI

# ผลลัพธ์สุดท้าย

จำนวน	รถ มอเตอร์ไซค์	รถยนต์	รถตู้	รถกระบะ	รถบรรทุก	รวม
MB4 IN 13	824	2744	632	481	1176	5857
MB4 IN 14	1147	3781	788	1853	692	8261
MB4 IN 15	335	2004	448	1459	545	4791
รวม	2306	8529	1868	3793	2413	18909

# ความท้าทายหลักที่พบ และแนวทางการแก้ไขเบื้องต้น

## ความท้าทายหลักที่พบ

1. ความสับสนของการจำแนกรถกระยะและรถบรรทุก เนื่องจากปัจจุบันรถกระยะมีการใส่กล่องไว้ที่กระยะตัวอย่างเช่นรถขนส่งพัสดุหลายๆ เจ้า ทำให้มีลักษณะคล้ายกับรถบรรทุกในบางมุมมอง
2. การนับจำนวนยานพาหนะแต่ละประเภทและแสดงผลแบบเรียลไทม์ อาจมีความช้าและไม่ลื่นไหล เนื่องจากการประมวลผลตรวจจับและติดตามวัตถุใช้เวลาค่อนข้างมาก โดยเฉพาะบนฮาร์ดแวร์ที่ไม่มี GPU

## แนวทางการแก้ไขเบื้องต้น

1. เพิ่ม class ของรถกระยะที่ติดกล้องเข้าไปตอน Label และค่อยนับรวมกับรถกระยะปกติภายหลัง
2. เพื่อเพิ่มประสิทธิภาพและลดความหน่วงในการแสดงผล จึงมีการใช้เทคนิคข้ามเฟรม (frame skipping) ในการลดจำนวนเฟรมที่ประมวลผล ช่วยเพิ่มความเร็วโดยแยกกับความละเอียดในการติดตามวัตถุ

# ข้อจำกัดของระบบที่ทำ และแนวทางการพัฒนาต่อในอนาคต

## ข้อจำกัดของระบบ

1. ประมวลผลช้าหากไม่มี GPU เมื่อมีการปรับปรุงด้วย frame skipping และลดขนาดภาพ (image resizing) แล้วก็ตาม หากใช้งานบนอุปกรณ์ที่มีเฉพาะ CPU เช่น โน้ตบุ๊กทั่วไป การประมวลผลยังใช้เวลานานมาก เช่น การประมวลผลเฟรมละ ~ 80ms จะใช้เวลากว่า 1.5 ชั่วโมงสำหรับวิดีโอ 1 ชั่วโมง
2. การทำงานแบบ Real-time อาจไม่ลื่นไหล หากต้องการใช้กับกล้องวงจรปิดหรือ feed สด ระบบอาจทำงานได้ไม่ทันเวลา เนื่องจาก latency จากการตรวจจับ + ติดตาม + เขียนวิดีโอ output ยิ่งถ้ามีหลายกล้อง หรือใช้บน embedded device ที่ไม่มีการ์ดจอ (GPU) จะยิ่งเกิดการหน่วง

## แนวทางการพัฒนาต่อในอนาคต

1. ใช้โมเดลที่เบากว่า เช่น YOLOv8n หรือ YOLOv5n ที่ใช้ inference time ต่ำกว่า 15ms/เฟรม
2. ใช้เทคนิค batch inference กับ async processing เพื่อให้ CPU ทำงานต่อเนื่องไม่ว่างเปล่า
3. ติดตั้งระบบบนเครื่องที่มี GPU หรือเลือกใช้บริการ Cloud เช่น Google Colab Pro, AWS EC2, Azure, หรือ Jetson Nano
4. สำหรับการใช้งาน Real-time สามารถแยกการทำงานของ detection และ visualization ออกจากกัน ด้วย multiprocessing หรือ gRPC/REST API

Source Code

<https://github.com/martiie/Faco-test>

Show Result

File video in Google Drive