Introduction

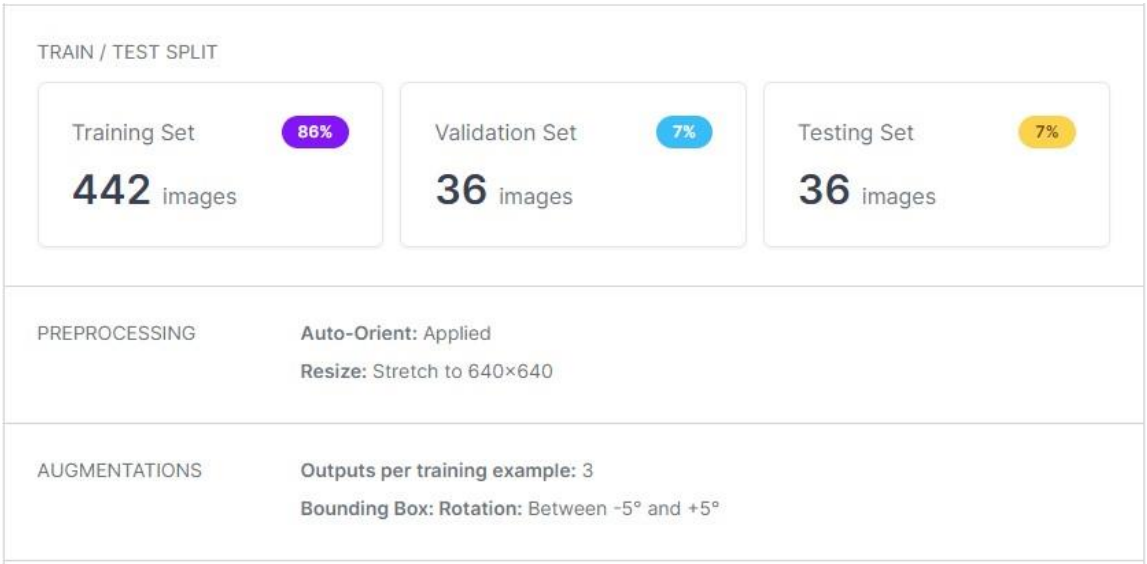
งานชิ้นนี้เป็นการศึกษา Convolutional Neural Network ในการทำ Objective Detection ตรวจจับภาพพนักงานขนส่งอาหารและสิ่งของต่าง ๆ หรือ ไรเดอร์ โดยใช้โมเดล RetinaNet และ YOLO

Data

ข้อมูลที่ใช้ในงานชิ้นนี้จะเป็นภาพของพนักงานรับส่งอาหาร และสิ่งของ หรือไรเดอร์ โดยภาพที่หามาทั้งหมดจะนำเข้าไปทำการ Labeling ใน Roboflow โดยจำแนกข้อมูลได้เป็น 6 ชนิด มีรายละเอียดดังต่อไปนี้

- Foodpanda rider (Class: foodpanda)
- Grabfood & Grabexpress rider (Class: grab)
- Lazada rider (Class: lazada)
- Lineman rider (Class: lineman)
- Robinhood rider (Class: robinhood)
- ShopeeFood rider (Class: shopee)

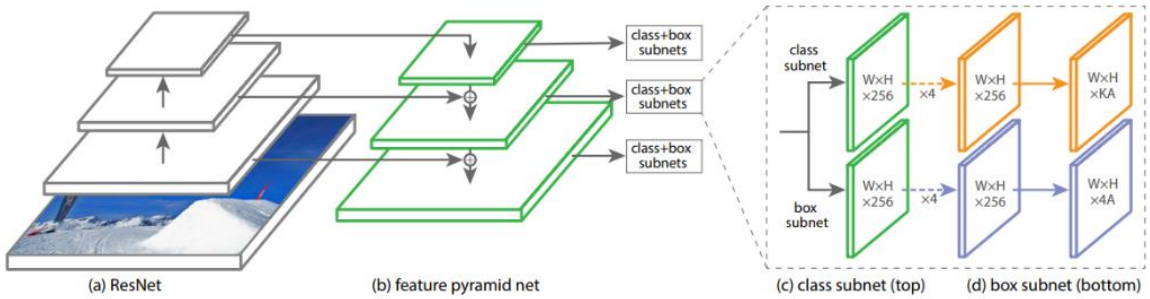
จากนั้นทำการ Split ข้อมูลสำหรับ Training, Validation และ Testing และทำ Data Preprocessing และ Data Augmentation โดยมีรายละเอียดดังต่อไปนี้



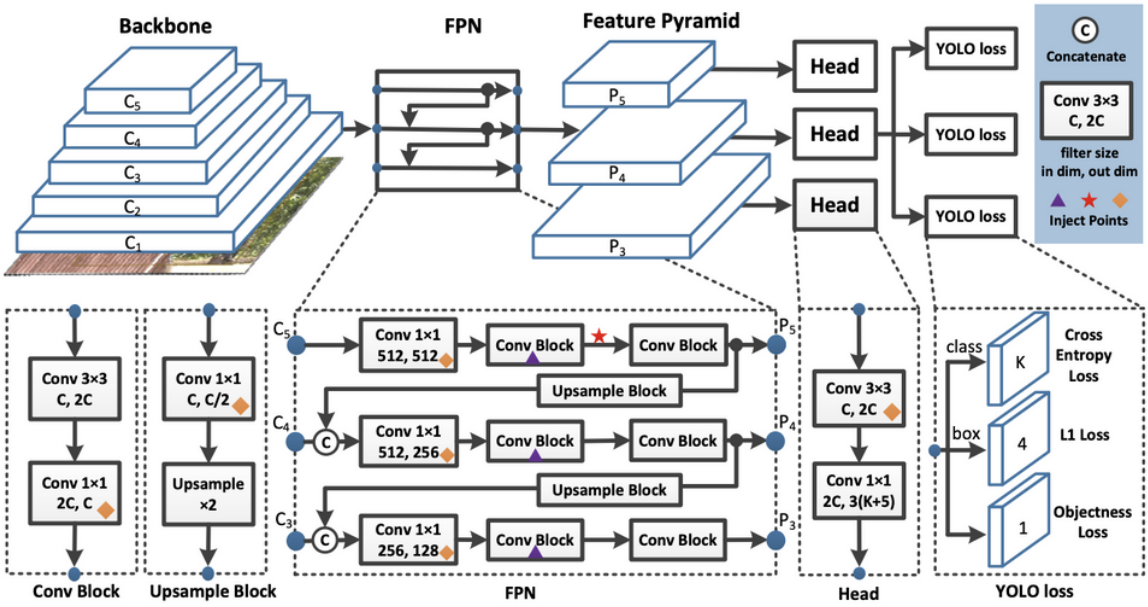
ภาพที่ 1 แสดงรายละเอียด Dataset

Network architecture

โมเดล RetinaNet และ YOLOv7 เป็นโมเดลของ Convolutional Neural Network (CNN) หรือโครงข่ายประสาทแบบคอนโวลูชั่น ประเภท One-stage object detection โดยมี architecture ดังต่อไปนี้



ภาพที่ 2 แสดง RetinaNet model architecture



ภาพที่ 3 แสดง YOLO model architecture ใน PP-YOLO

Training and Results

ในการรัน RetinaNet และ YOLOv7 นั้นรันบน Tesla T4 GPU ที่ Colab ด้วยภาษา Python ได้ให้ผลการทดลองเป็นดังนี้

ผลการทดลองของ Retinanet

รัน RetinaNet บน Keras ซึ่งถูกพัฒนาโดย FIZYR ผ่าน GitHub ใช้ training data ทั้งหมด 552 รูปภาพ และไม่ใช้ validating data เนื่องจากไม่มีตัวเลือกดังกล่าวในชุดคำสั่ง train ใช้ batch size เท่ากับ 8 และจำนวน epochs เท่ากับ 2 ทั้งนี้ RetinaNet ใช้ทั้งหมด 36,507,427 parameters โดยแบ่งออกเป็น trainable กับ non-trainable และให้ค่า loss ที่มาจาก regression และ classification ดังนี้

ตารางที่ 1 แสดง parameter ของ RetinaNet

Attribute		**Freeze extractor**	**Unfreeze all**
Trainable parameters		12,946,275	36,401,187
Non-trainable parameters		23,561,152	106,240
Epoch 1	Regression loss	1.3736	1.0930
	Classification loss	1.1097	1.0508
	Total loss	2.4833	2.1439
Epoch 2	Regression loss	1.0646	0.7565
	Classification loss	0.8126	0.7263
	Total loss	1.8773	1.4828

เมื่อทดสอบ Object detection ที่ได้กับ testing data ทั้งหมด 36 รูปภาพ และพิจารณา IOU threshold ตั้งแต่ 0.1 ถึง 0.9 โดยเพิ่มค่าทีละ 0.1 พบว่า Mean Average Precision (mAP) กรณี freeze ส่วน feature extractor (backbone) มีค่าน้อยกว่ากรณี unfreeze ทั้งส่วน feature extractor และ classifier เล็กน้อย ดังตาราง อาจเป็นเพราะจำนวนของ trainable parameters ที่น้อยกว่า

ตารางที่ 2 แสดงผลลัพธ์ (ค่า Mean Average Precision) จากการรัน RetinaNet

Class	mAP (Freeze feature extractor)	mAP (Unfreeze all)
all	0.106	0.116
foodpanda	0.500	0.500
grab	0.060	0.070
lazada	0.000	0.000
lineman	0.000	0.000
robinhood	0.186	0.242
shopee	0.000	0.000

ผลการทดลองของ YOLOv7 (Pretrained Model)

ในการรัน YOLOv7 มีการตั้งค่า arguments เป็น batch sizes = 16, epochs = 55 และ confident threshold = 0.1 ได้ผลค่า Precision, Recall, mAP ดังนี้

ตารางที่ 3 แสดงผลลัพธ์ (ค่า Mean Average Precision) จากการรัน YOLOv7 แบบ Pretrained model

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	36	44	0.192	0.534	0.234	0.152
foodpanda	36	7	0.0957	0.286	0.101	0.0408
grab	36	12	0.192	0.833	0.335	0.199
lazada	36	6	0.636	0.333	0.325	0.243
lineman	36	11	0.1741	1	0.359	0.198
robinhood	36	4	0	0	0.206	0.174
shopee	36	4	0.0515	0.75	0.0779	0.0586

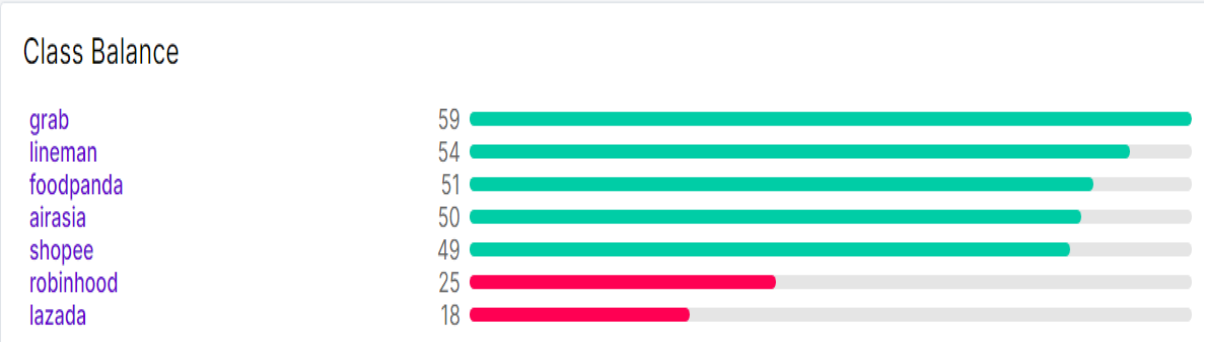
ผลการทดลองของ YOLOv7 (Finetune)

ได้ผลค่า Precision, Recall, mAP ดังนี้

ตารางที่ 4 แสดงผลลัพธ์ (ค่า Mean Average Precision) จากการรัน YOLOv7 แบบ freeze backbone

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	36	44	0.677	0.719	0.781	0.611
foodpanda	36	7	1	0.399	0.767	0.432
grab	36	12	0.843	0.896	0.923	0.741
lazada	36	6	0.429	0.5	0.456	0.402
lineman	36	11	0.903	0.818	0.929	0.752
robinhood	36	4	0.475	1	0.995	0.79
shopee	36	4	0.41	0.7	0.619	0.552

จากผลการทดลอง YOLOv7 ได้ผลค่า Precision, Recall, mAP หลังจาก Freeze backbone ผลดีขึ้นอย่างชัดเจน ซึ่งมากกว่า Pretrain Model เพื่อปรับปรุง Perfomance ของโมเดล อาจต้องเพิ่มเติมเกี่ยวกับความครอบคลุมของชุดข้อมูลโดยการเพิ่มจำนวน datasets ในแต่ละ class ให้มากขึ้นและ และคุณภาพของภาพในการเทรนโมเดล และต้องทำการทดลองมากขึ้น เพื่อประเมินได้ว่าจะสามารถปรับค่าอะไรเพิ่มเติมได้บ้างเพื่อที่จะได้ค่า optimal ของโมเดลนี้ จากข้อสังเกตเห็นว่าใน class foodpanda, lazada, shopee นั้นค่า mAP ที่ได้ยังมีค่าที่ต่ำกว่าหรือใกล้เคียง 0.5 อยู่ซึ่งอาจเกิดจาก dataset ในคลาสนี้ที่น้อยเกินไปจาก คลาสอื่นๆ ดังรูปข้างล่าง



ภาพที่ 4 แสดง balance ของ class ใน dataset

โดยปกติแล้วค่า detect ภาพที่ใช้ Model ในการทำนายจะประกอบด้วย confident score และ bounding box ซึ่งบอกตำแหน่งในรูปของค่า x,y ซึ่งเราสามารถ tune ค่า confident threshold เพื่อเพิ่มประสิทธิภาพของโมเดลได้นอกจากการทำ preprocess หรือ เก็บจำนวนภาพที่มากขึ้น

Discussion and Conclusion

จากผลการทดลองสามารถสรุปได้เป็นดังนี้

ตารางที่ 5 สรุปผลลัพธ์จากการรันโมเดล RetinaNet และ YOLOv7

Model	mAP (Pretrained model)	YOLOv7 (Freeze backbone)
RetinaNet	0.116	0.106
YOLOv7	0.152	0.611

จากตารางสรุป พบว่าการทดลอง Objective Detection โดยใช้โมเดล RetinaNet และ YOLOv7 ในการตรวจจับพนักงานขนส่งทั้งหมด 6 class พบว่า เมื่อทำการปรับ feature extractor โดยการ freeze backbone แล้ว ผลลัพธ์ที่ได้ของทั้ง 2 โมเดลเป็นดังนี้ ก่อนทำการ freeze backbone (pretrained model) โมเดล RetinaNet และ YOLOv7 ให้ค่า Mean Average Precision (mAP) หลังการตรวจจับเป็น 0.116 และ 0.152 ตามลำดับ พอหลังจากทำการ freeze backbone พบว่า ค่า mAP ที่ได้เป็น 0.106 และ 0.611 ตามลำดับ ซึ่งในที่นี้อาจจะไปทดลองซ้ำๆเพื่อสรุปผลในรูปแบบ mean+-sd เนื่องจากทรัพยากรของ google colab ที่มี และ starting point ที่ใช้ในการเทรนโมเดล transfer learning นั้นที่ test size = 640 นั้นมีแค่ starting weights 2 อันให้เลือก yolov7_training และ yolov7x_training ซึ่งจากการทดลองทั้งสองผลที่ได้ใกล้เคียงกัน

สรุปได้ว่า โมเดล RetinaNet เมื่อ pretrained model นั้นจะให้ค่า mAP ที่ไม่แตกต่างจากหลังทำการ freeze backbone มากนัก ส่วนโมเดล YOLOv7 หลังทำการ freeze backbone นั้นจะให้ค่า mAP ที่สูงกว่าเมื่อ pretrained model คาดว่า หากมีการเพิ่มจำนวน dataset ให้มากขึ้น และครอบคลุมในแต่ละ class หรือมีระยะเวลาในการศึกษาการวิเคราะห์การปรับค่า parameter ต่าง ๆ และการทดลองที่มากขึ้น จะสามารถเพิ่มประสิทธิภาพการทำ Objective Detection ของโมเดลให้ดียิ่งขึ้น

Reference

Evaluating Object Detection Models Using Mean Average Precision: https://www.kdnuggets.com/2021/03/evaluating-object-detection-models-using-mean-average-precision.html

How RetinaNet works?: https://developers.arcgis.com/python/guide/how-retinanet-works

Keras RetinaNet: https://github.com/fizyr/keras-retinanet

Object Detection with RetinaNet: https://keras.io/examples/vision/retinanet

Official YOLOv7: https://github.com/WongKinYiu/yolov7

PP-YOLO Surpasses YOLOv4 - State of the Art Object Detection Techniques: https://blog.roboflow.com/pp-yolo-beats-yolov4-object-detection

Roboflow: https://app.roboflow.com

Seven Tips for Labeling Images for Computer Vision: https://blog.roboflow.com/tips-for-how-to-label-images

YOLOv7 Breakdown: https://blog.roboflow.com/yolov7-breakdown

End credit

งานชิ้นนี้เป็นส่วนหนึ่งของรายวิชา วขวข 7202 การเรียนรู้เชิงลึก (DADS 7202 Deep Learning) หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการวิเคราะห์ข้อมูลและวิทยาการข้อมูล (Master of Science Program in Data Analytics and Data Science) คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์ (Graduate School of Applied Statistics, National Institute of Development Administration)

จัดทำโดย 6310432002 ทรงคมกฤช ไชยกาล

6410412005 เดโช ศรีสวัสดิ์

6410412015 คุณานนต์ กลิ่นจันทร์หอม