Weapon Detection

จัดทำโดย

63310687 จันทรกานต์ วิทยกิจ

63311936 ธัญญารัตน์ ผูกฤทัย

เสนอ

ผศ.ดร.สุธาสินี จิตต่อนันต์

รายวิชา 278488 Digital image Processing

มหาวิทยาลัยนเรศวร

คำนำ

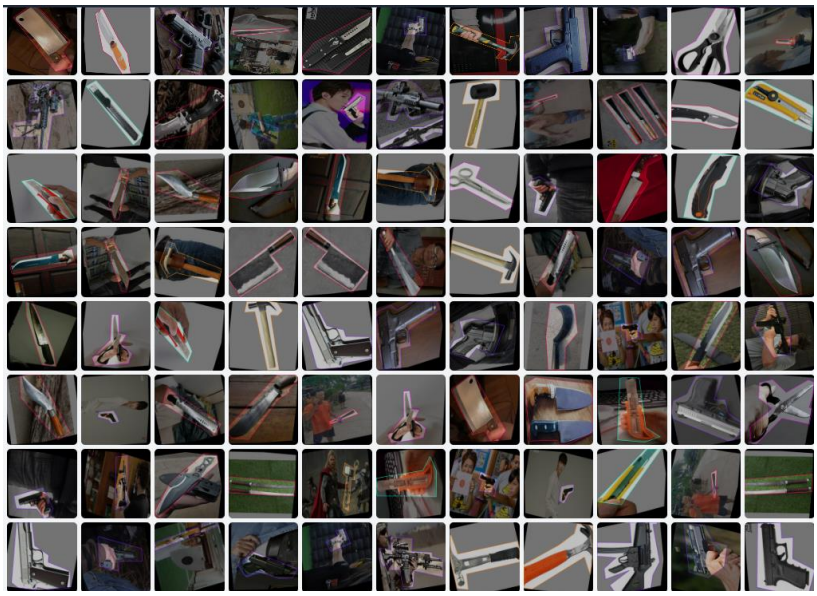
รายงานเล่มนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของรายวิชา 2783488 Digital image Processing คณะผู้จัดทำได้จัดทำ Weapon Detection ขึ้นมาโดยมีวัตถุประสงค์ เพื่อพัฒนาและทดลองการตรวจจับอาวุธที่ไม่พึงประสงค์ผ่านวิดิโอหรือรูปภาพ เพื่อช่วยลดเวลาในการรวบรวมหลักฐาน ในกรณีเกิดเหตุการณ์รุนแรงขึ้นและช่วยสนับสนุนเจ้าหน้าที่ในการปฏิบัติงาน

คณะผู้จัดทำขอขอบพระคุณ ผศ.ดร.สุธาสินี จิตต์อนันท์ ผู้ให้ความรู้และแนวทางในการศึกษา คณะผู้จัดทำหวังว่ารายงานเล่มนี้จะเป็นประโยชน์อย่างมากสำหรับผู้ที่กำลังพัฒนาหรือกำลังศึกษาแนวทางในเนื้อหาเหล่านี้และช่วยในการศึกษารายวิชา 273488 Digital image Processing ได้อย่างเต็มที่และเกิดประโยชน์อย่างสูงสุด หากเกิดผิดพลาดประการใด ผู้จัดทำขออภัยมาใน ณ ที่นี้ด้วย

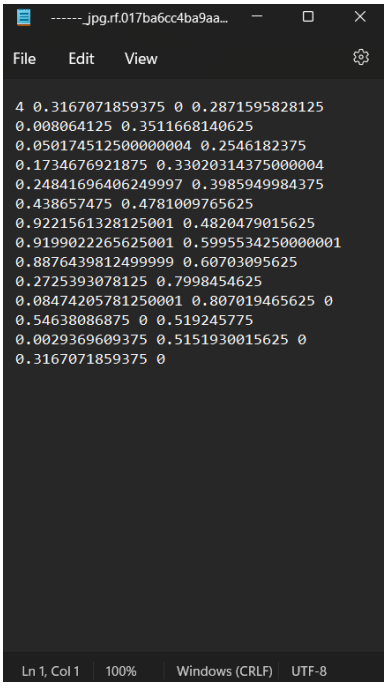
คณะผู้จัดทำ

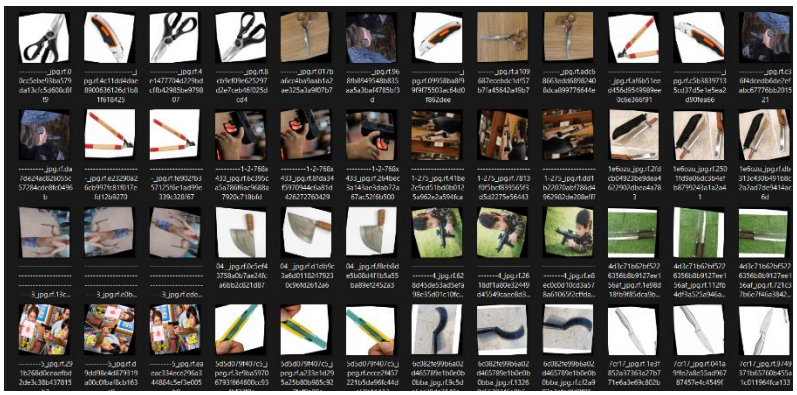
สารบัญ

ข้อมูลนำเข้า



ข้อมูลการนำเข้า ได้จาการนำรูปภาพอาวุธต่างๆ ทั้ง 5 ชนิด ไปทำการวาดกรอบในโปรแกรม roboflow

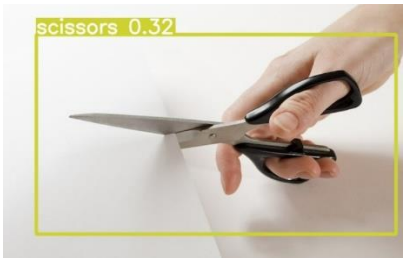




เมื่อทำการวาดกรอบรูปเสร็จแล้วจะได้รูปภาพและlabel ออกมาดังภาพ เพื่อนำข้อมูลดังกล่าวไปทำการ Train ต่อไป

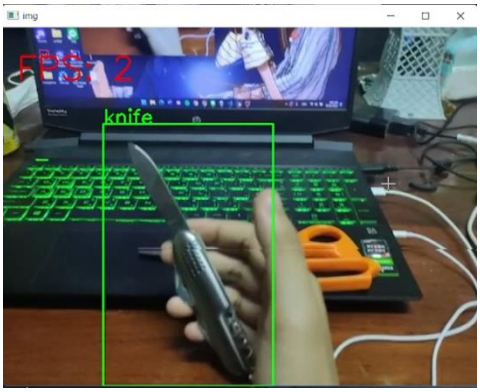
ผลลัพธ์

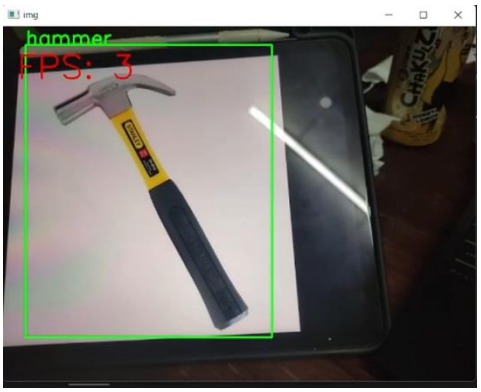
ทำการทดสอบโดยใช้รูปภาพ

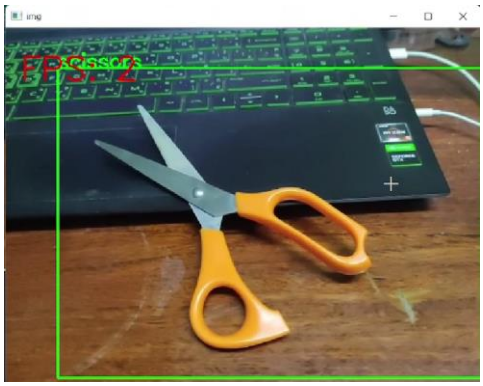


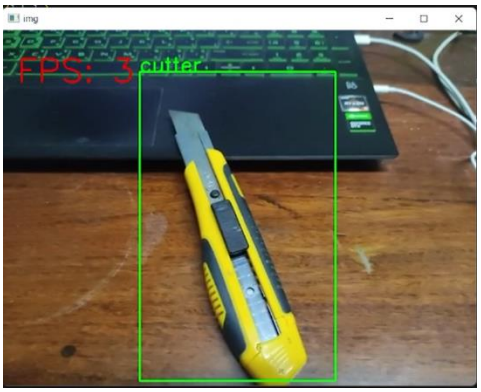
ทำการทดสอบโดยใช้คลิปวิดิโอ

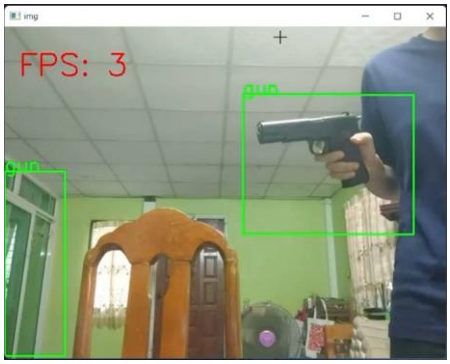
ทำการทดสอบแบบเรียบ real time



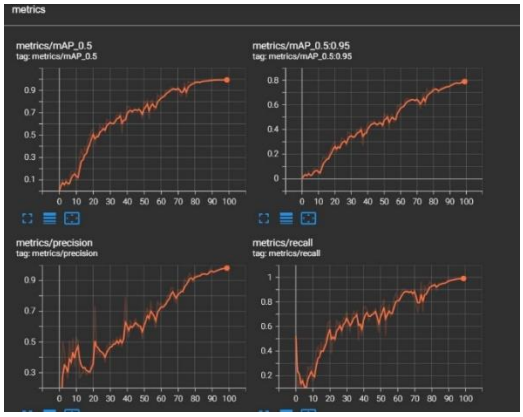


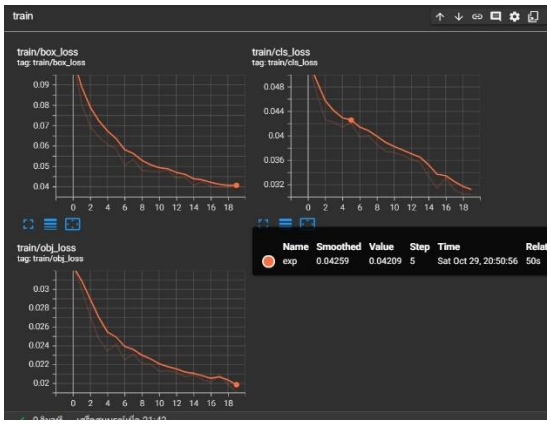


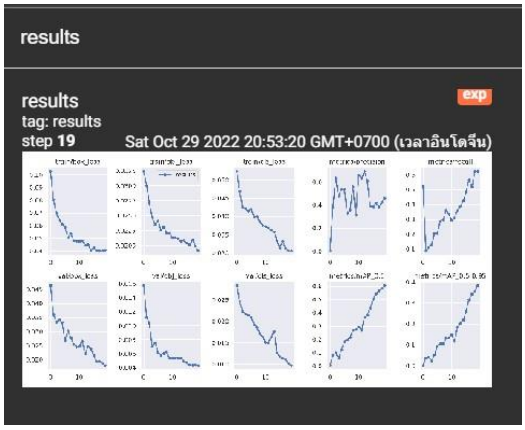


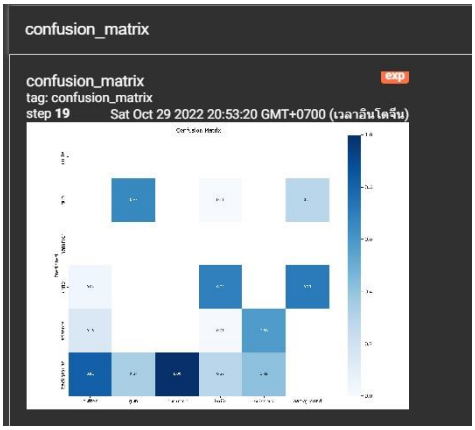


ค่าความแม่นยำ

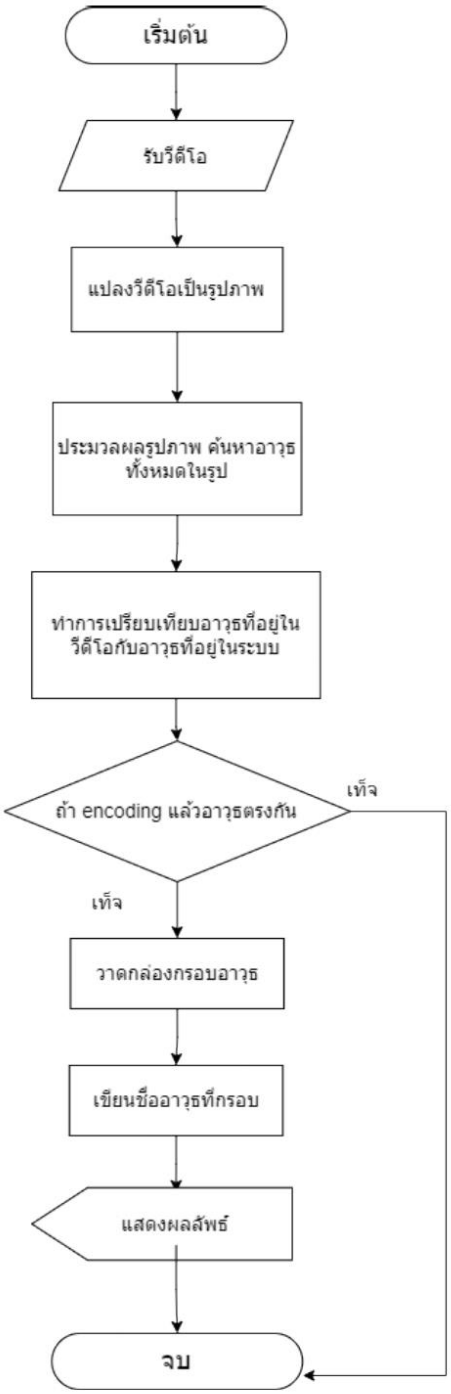








ขั้นตอนการทำงานของโปรแกรม



1.รับวีดีโอที่ต้องการจะตรวจสอบเข้าสู่โปรแกรม

2.แปลงวีดีโอเป็นรูปภาพตามเฟรมของวีดีโอ

3.ระบบทำการค้นหาอาวุธทั้งหมดในภาพ

4.ทำการเปรียบเทียบอาวุธที่อยู่ในวีดีโอกับอาวุธที่อยู่ในระบบ

5.ถ้าเจออาวุธจะทำวาดกรอบรอบอาวุธนั้น

6.เขียนชื่ออาวุธนั้นว่าคืออาวุธชนิดไหน

ทฤษฎีและหลักการที่เกี่ยวข้อง

1.R-CNN

R-CNN เป็นงานวิจัยที่ถูกเขียนด้วย Girshick R. et al. ซึ่งถือได้ว่าเป็นความพยายามแรกๆที่ทำให้ Region Proposal Network มีชื่อเสียงขึ้นมา โดยรวมแล้ว

R-CNN จะประกอบด้วยกัน 4 ขั้นตอนดังนี้

1.1 การเสนอพื้นที่ในภาพที่อาจจะมีวัตถุที่สนใจด้วย Selective Search

1.2 การฝึกและปรับแต่งอย่างละเอียดบนตัวแบบ CNN

1.3 การฝึกตัวจำแนกประเภทด้วย SVM แบบแยกทีละคลาส

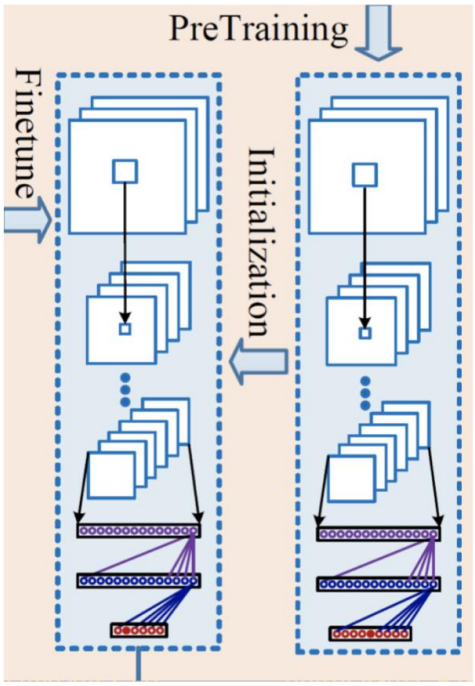
1.4 การฝึกการตีกรอบวัตถุให้แม่นยำ

ใช้ CNN ในการสกัดฟีทเจอร์ ไม่ใช่ระบุประเภท

จริงๆแล้ว RCNN นั้นนำ CNN มาใช้ก็จริง แต่นำมาใช้แค่การทำการสกัดคุณลักษณะ (Features Extraction) แล้วหลังจากนั้นเราจะใช้ SVM (Support Vector Machine) ในการแยกประเภทอีกทีหนึ่ง

Pre-Trained model and Fine tuning

- วิธีการที่ว่ามานี้จะเริ่มจากการดาวน์โหลดตัวแบบที่ฝึกมาล่วงหน้า (Pre-Trained model) ซึ่งค่าน้ำหนักในตัวแบบได้รับการปรับแต่มาเป็นอย่างดี



Deep Learning for Generic Object Detection: A Survey [1]

- ซึ่งการปรับแต่งนี้ทำให้เราได้ค่าน้ำหนักที่เหมาะสมในการแยกประเภทสิ่งของในภาพถึงแม้มันอาจจะไม่ได้เหมาะสมที่สุดในงานของเรา แต่ค่าน้ำหนักสุดท้ายมักจะไม่ต่างจากเดิมมากนัก
- วิธีนี้จะฝึกโครงข่ายได้เร็วมาก ขั้นตอนการใช้ค่าน้ำหนักจากโมเดลที่ฝึกมาล่วงหน้าคือ **(Weight) Initialization** ส่วนการมาฝึกต่อในงานของเราคือ **Fine tuning**

2.mAP Evaluation

True Positive (TP) คือ สิ่งที่ทาย(prediction) ว่ามันมีจริง และ เฉลย (Ground Truth) บอกว่ามีจริง

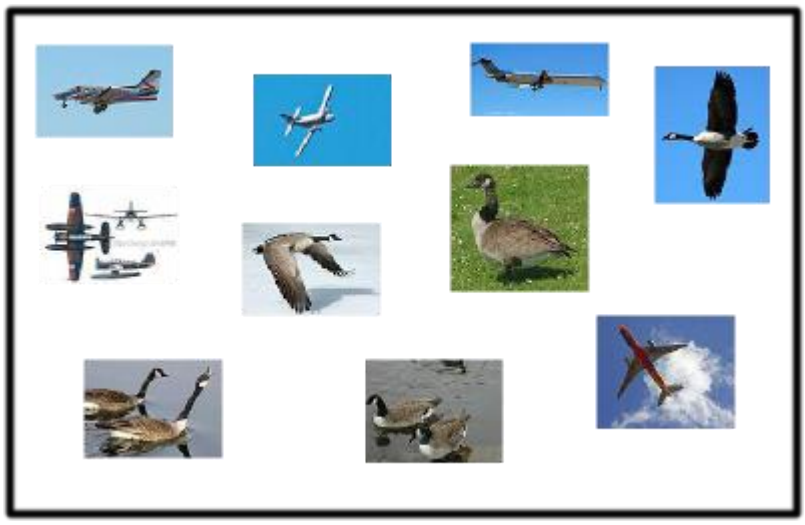
True Negative (TN) คือ สิ่งที่ทายว่ามันไม่มีจริง และเฉลยก็บอกว่ามันไม่มีจริง

False Positive (FP) คือ สิ่งที่ทายว่ามันมีจริง แต่เฉลยก็บอกว่ามันไม่จริง(มันไม่ถูก)

False Negative (FN) คือ สิ่งที่ทายว่ามันมีไม่จริง แต่เฉลยก็บอกว่ามันมีจริง (นี่ก็ไม่ถูก)

Precision and recall

การวัดประสิทธิภาพของ การค้นหา Class ใด Class หนึ่งของ Model เราตัวอย่างในที่นี้ขอเป็น ค้นหาเครื่องบิน สมมุติว่าเรามีรูปที่สามารถค้นหาได้ดังภาพ ในภาพนี้มีรูปเครื่องบิน และรูป ห่าน



https://sanchom.wordpress.com/tag/average-precision/[2]

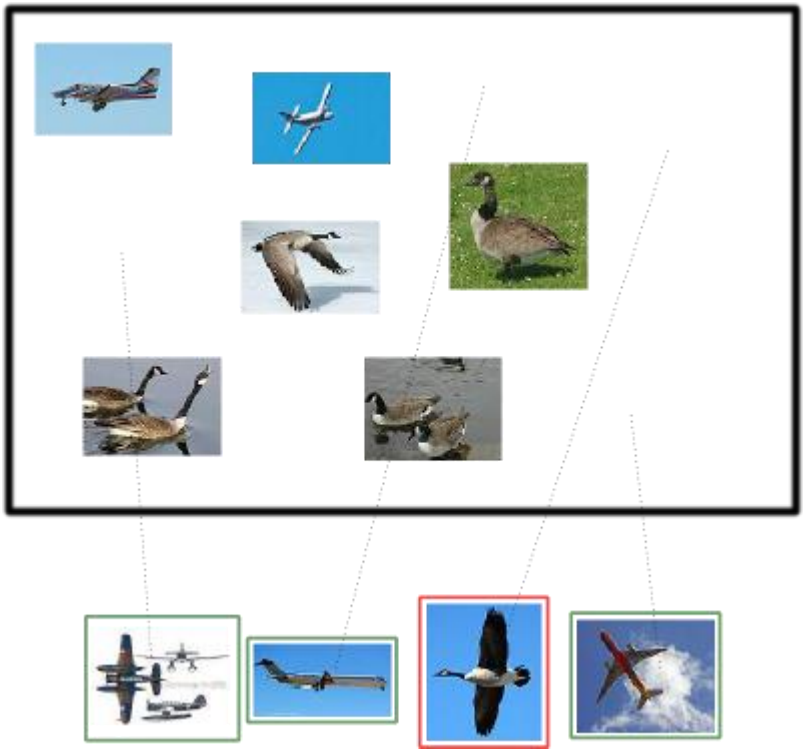
$$precision = \frac{tp}{tp + fp} = \frac{tp}{n}$$

Precision คือ อัตราการทายถูกต่อการทายทั้งหมด โดย n คือจำนวน ที่ Model พยายามทายทั้งหมด เช่นทายมา 2 ตัว โอกาส ก็มี ถูกทั้งหมด ก็ TP = 2 หรือ ทายถูก 1 ทาย ผิด 1, TP =1 FP =1 หรือ ทายผิดหมดเลย ก็เป็นได้

$$recall = \frac{tp}{tp + fn}$$

Recall คือ จำนวน ที่ทายถูกต่อจำนวนของ Ground Truth ทั้งหมด

สมมุติว่า Model เราทายมาเป็นลักษณะนี้ เราจะให้ Model เราทายเฉพาะแค่ เครื่องบิน หมายถึง นำแต่เครื่องบินมาให้เรา นำอย่างอื่นมาให้ถือว่าผิด



https://sanchom.wordpress.com/tag/average-precision/[3]

จากmodel ของเรา ทำการเลือกภาพมาให้เราดังภาพที่อยู่นอกกรอบครับ จะเ้ว็นว่า ได้รูปเครื่องบิน ได้มา 3 รูป ส่วน ห่านติดมา 1 รูปครับ ลองคำนวน Precision และ Recall ดู

จากตัวอย่าง เราจะได้ 3 TP และ 1 FP ดังนั้น precision คือ 3/4 = 0.75 และ recall คือ 3/5 = 0.6

Truning the threshold

ในกรณีที่เรา ไม่พอใจใน ผลลัพธ์ของ Model เราเช่น ค้นหา object ในภาพ ทั้งหมดไม่เจอ ทายภาพออกมาได้ไม่หมด เราอาจจะใช้ ปรับ Thresold ให้ model เราสามารถทายผลลัพธ์ ให้มาก ขึ้น ซึ่งแน่นอน อาจจะได้มาทั้งตัวที่ ทายถูกและตัวที่ทายผิดด้วยเช่นกัน ทีนี้เราจะปรับ threshold ไปเรื่อยๆ จนกว่าจะได้ค่าที่ดีที่สุด สำหรับเรา ว่า ตำแหน่งนี้และ Threshold ได้ให้ ค่า Precision ดีที่สุดแล้ว ซึ่งในความเป็นจริง เราอาจจะทราบเพียงจุดที่ดีที่สุด และนำเสนอเพียงจุดนั้นจุดเดียว ปกปิด ค่าตำแหน่ง Threshold ตำแหน่งอื่นๆ ที่ให้ผลลัพธ์ไม่ดีก็เป็นได้ ซึ่งการนำเสนอจุดที่ดีที่สุดเพียงจุดเดียว ไม่ได้บ่งบอกประสิทธิภาพโดยรวมของ Model ของเรา (ไม่สามารถเป็นตัวแทนประสิทธิภาพโดยรวมได้) ดังนั้น จึงมีวิธีการวัดประสิทธิภาพโดยรวม ซึ่งเรียกว่า

mAP ซึ่งก่อนอื่นเราจะต้องทำการ Rank ลำดับของ Object ที่เราจะทายก่อน อาจจะ Rank ด้วย ค่าสูงสุดของ Probability ค่า Confidence ก็ได้

จากตัวอย่าง รูปก่อนหน้า เราสามารถปรับค่า ขึ้น หรือค่าลงเพื่อทำให้ได้การทำนายค่าที่ดีที่สุดของ Model ในที่นี้สมมุติว่ามีการ Rank ลำดับมาให้แล้วดังภาพด้านล่าง

จาก รูปเราอาจจะปรับ Threshold (เส้นสีน้ำเงิน)ให้เหลือ เพียง รูปเครื่องบิน อันแรกสุดอันเดียวก็ได้ หรือ เลื่อน Threshold ให้อยู่อันสุดท้ายก็ได้ กรณีที่ เลื่อนค่า Threshold จนเหลือ เครื่องบินอันเดียว แน่นอน ค่า Precision จะได้ 1 แต่ในทางกลับกันยังมี เครื่องบินที่ยังไม่ได้ถูกเลือกออกมา เพราะค้นหาได้ไม่หมดดังนั้นค่า Recall จะต่ำเหลือ 0.2หาก model ใดมีประสิทธิภาพที่ดี ค่า Precision และ Recall ควรจะอยู่ที่ 1 นั่นคือ ทุกตัวที่ทายถูกทั้งหมด(precision) และ ค้นหาได้ครบทุกตัว(recall)

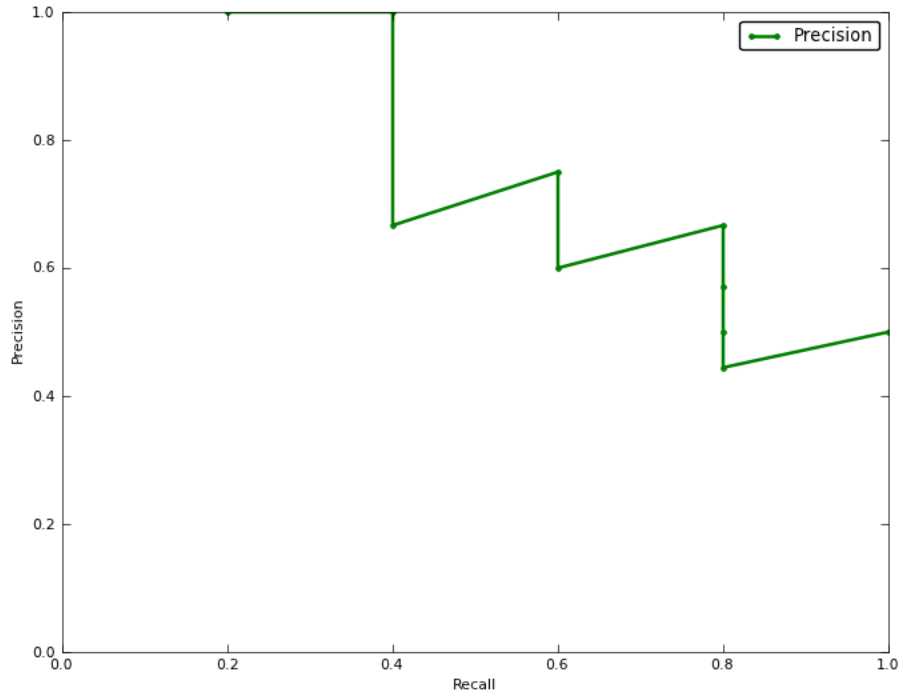
กลับมาดู Threshold ดังนั้นหากเราเลือก Threshold แตกต่างกันค่า Precision และ Recall ก็จะถูกเปลี่ยนไปตามค่า Threshold ที่เรา

เลือกด้วยเช่นกัน ซึ่งสามารถเขียนได้ตามตารางด้านล่าง

Retrieval cutoff	Precision	Recall	delta recall
Top 1 image	1	0.2	0.2
Top 2 images	1	0.4	0.2
Top 3 images	0.66	0.6	0.2
Top 4 images	0.75	0.6	0
Top 5 images	0.6	0.6	0
Top 6 images	0.6	0.8	0.2
Top 7 images	0.57	0.8	0
Top 8 images	0.5	0.8	0
Top 9 images	0.44	0.8	0
Top 10 images	0.5	1	0.2

จากตารางจะเห็นว่า หากเรา ตั้ง threshold ไว้ที่ 1 รูป Precision จะเป็น 1 แต่ recall จะค้นไม่หมด ได้เพียง 0.2 Precision ดี แต่ Recall ไม่ดี งั้นลงลด threshold ลง มาที่ 2 รูป ได้ Recall 0.4 แต่พอลงมาที่ 3 รูป

Precision ลดลงไปที่ 0.66 เมื่อเราปรับ Threshold ลงไปเรื่อยๆ เราจะได้กราฟระหว่าง Precision กับ Recall ดังรูป



ซึ่งจะเห็นว่า บางช่วง Precision ก็เพิ่มขึ้น และบางช่วง Precision ก็ลดลง ซึ่งปกติแล้วเวลาที่เรานำเสนอ ข้อมูลก็จะใช้กราฟ ระหว่าง Precision กับ Recall นี้แหละ เป็นตัวนำเสนอข้อมูล

Average precision

หากเราต้องการจะวัดค่าประสิทธิภาพของ Model เราจริงๆ เราจำเป็นต้องได้ค่ามา 1 ค่าเพื่อเป็นตัวแทนประสิทธิภาพของ Model เรา โดยปกติแล้วจะใช้ค่าของ *average precision* เป็นตัวแทนของประสิทธิภาพของ Model เรา

ตัว Average Precision คำนวนได้จริง พื้นที่ ใต้กราฟของ Precision กับ Recall ครับ โดยจะมีค่าอยู่ระหว่าง 0 ถึง 1 ถ้าเป็น 1 นี่ Perfect เลยกราฟ ของ Precision จะหน้าตาเป็น สี่เหลี่ยม คือมี Precision เป็น

1 และ Recall เป็น 1 หาพื้นที่ ใต้กราฟก็ได้ 1 Perfect แต่ไม่ได้เรียบง่ายแบบที่คิดลองดูตัวอย่างกราฟ ด้านบน จะหาพื้นที่ใต้กราฟโดยสมการดังนี้

$$\int_0^1 p(r)dr$$

แต่กราฟเราหน้าตาคล้ายๆ Discrete เลยงั้นแปลงให้เป็นสมการ Summation ได้ดังนี้

$$\sum_{k=1}^{N} \mathrm{P(k)} \triangle r(k)$$

โดยที่ N คือ จำนวน รูปทั้งหมดที่เราทายออกมาทั้งถูกและไม่ถูก ส่วน P(k) คือ Precision ที่ตำแหน่ง ที่เกิน thresold ณ ที่ค่า k นั้นๆ ส่วน delta r(k) คือการเปลี่ยนแปลงของ ค่า recall ระหว่าง k-1 และ k จากตัวอย่างเราจะได้

Cutoff ที่ 1 จะได้ Precision 1 และ delta recall 0.2 = (1*0.2)

Cutoff ที่ 2 จะได้ Precision 1 และ delta recall 0.2 = (1*0.2)

Cutoff ที่ 3 จะได้ Precision 0.66 และ delta recall 0.0 = (0.66*0)

Cutoff ที่ 4 จะได้ Precision 0.75 และ delta recall 0.2 = (0.75*0.2)

Cutoff ที่ 5 จะได้ Precision 0.60 และ delta recall 0.0 = (0.6*0)

Cutoff ที่ 6 จะได้ Precision 0.66 และ delta recall 0.2 = (0.66*0.2)

Cutoff ที่ 7 จะได้ Precision 0.57 และ delta recall 0.2 = (0.57*0)

Cutoff ที่ 8 จะได้ Precision 0.50 และ delta recall 0.0 = (0.5*0)

Cutoff ที่ 9 จะได้ Precision 0.44 และ delta recall 0.0 = (0.44*0)

Cutoff ที่ 10 จะได้ Precision 0.5 และ delta recall 0.2 =(0.5*0.2)

หาผลรวมจะมีค่าเท่ากับ AP= 0.782

โดยซึ่งการคำนวนหา AP นี้เป็นเพียงเฉพาะ หา AP เพียง 1 Class เท่านั้น ในที่นี้เราสนใจเพียง Class เครื่องบิน แต่เราจะเป็นต้องทำการหา ในทุกๆ Class แล้วจากนั้น นำค่า AP มาทำการเฉลี่ย จึงจะได้ ค่า mAP ของระบบของเรา

Interpolated average precision

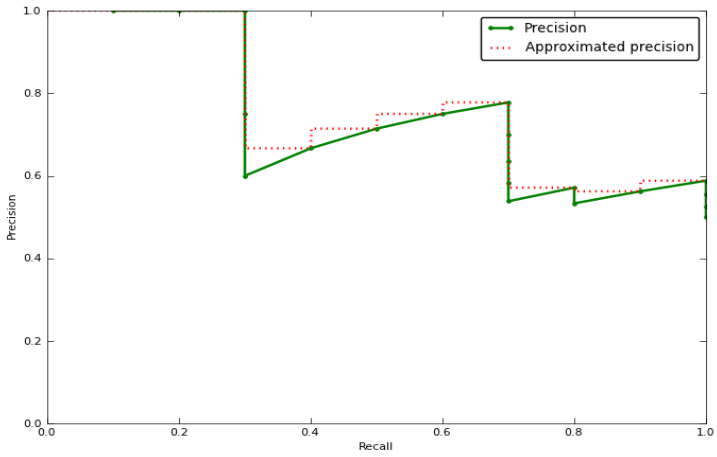
เป็นอีกเทคนิคนึงสำหรับการประมาณค่า AP ผู้เขียนงานวิจัยบางคนใช้วิธีการเติมเต็มค่าประมาณของ recision ลงไปเนื่องจากทำให้สามารถคำนวนได้ง่ายกว่า โดยทำการเปรียบเทียบค่า Precision nที่อยู่ทางด้านหน้าว่า มีค่า Precision ที่มากกว่าไหม ถ้าหากมี ก็จะทำการเติมค่า Precision ให้เท่ากับค่าที่อยู่ด้านหน้าเพื่อให้การคำนวนที่ง่ายขึ้น ดังสมการ

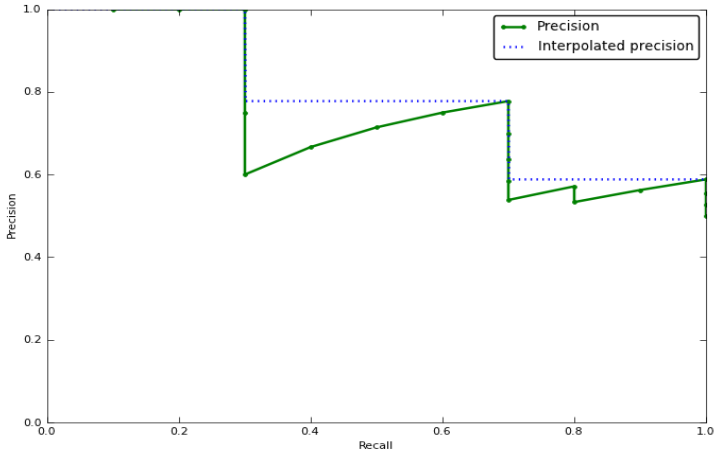
$$max_{k' \geq k} P(k')$$

จากนั้นจึงทำการหาค่า AP ใหม่ ซึ่งสามารถเขียนได้เป็นสมการดังนี้

$$\sum_{k=1}^{N} max_{k' \geq k} \mathrm{P}(\mathrm{k}') \triangle r(k)$$

โดยกราฟ Interpolated average precision ที่ได้จะมีลักษณะคล้ายคลึงกับกราฟเดิมแต่จะเติมค่า Precision ให้ทำการคำนวนได้ง่ายขึ้นดังกราฟด้านล่าง



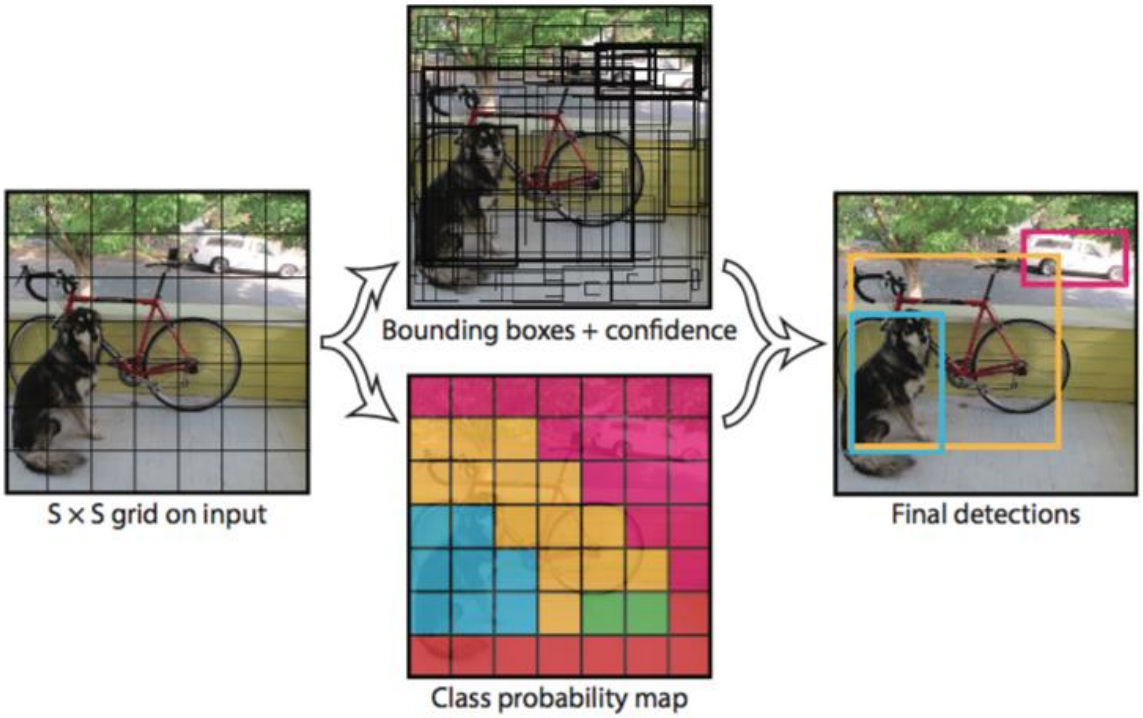


จะเห็นได้ว่ากราฟของ Interpolation Precision จะคำนวนได้ง่ายกว่าเพราะมีลักษณะเหมือน สี่เหลี่ยม ทำให้คำนวนค่าได้ง่ายขึ้น

3. YOLO (You Only Look Once)

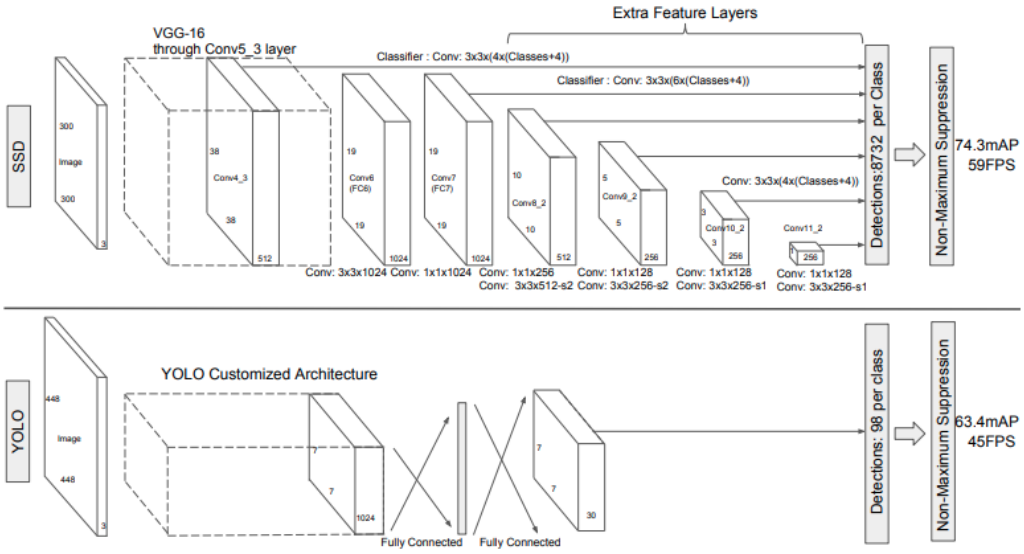
YOLO นั้นมีแนวความคิดที่ต่างไปจาก Faster R-CNN นั่นคือ 1) แทนที่จะทาย box แล้วจึงส่งไป classify ต่อ YOLO นั้นทายทั้ง box และความน่าจะเป็นของคลาสต่าง ๆ ออกมาพร้อมกันเลย และ 2) แทนที่เราทายค่า

ต่าง ๆ จากทั้งภาพ เราจะแบ่งภาพออกเป็นส่วน ๆ สำหรับแต่ละส่วนเราจะทายทั้ง box และคลาส ซึ่งเราสามารถนำมารวมกันเพื่อเลือกคู่คลาส/box ที่คะแนนสูงสุดเป็นคำตอบ



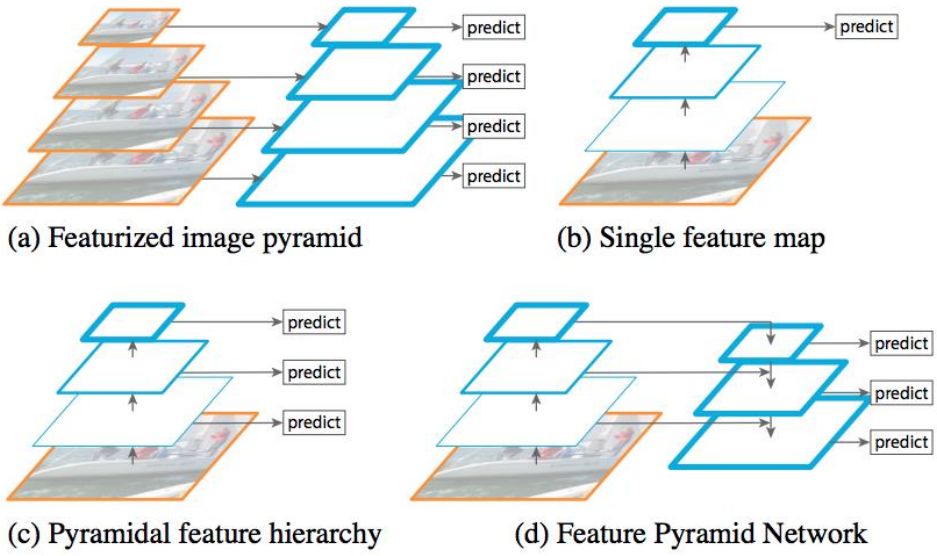
ส่วนหัวของ YOLO จาก [4]

อีกโครงสร้างหนึ่งที่มองได้ว่าเป็นการต่อยอดจาก YOLO คือ SSD (Single Shot multi-Box Detector) ที่เสนอให้ทาย box ต่าง ๆ จากหลาย scale แทนที่จะทาย ณ scale เดียวแบบ YOLO วิธีนี้ช่วยให้รับมือกับภาพที่มีวัตถุ scale ต่างกันมากได้



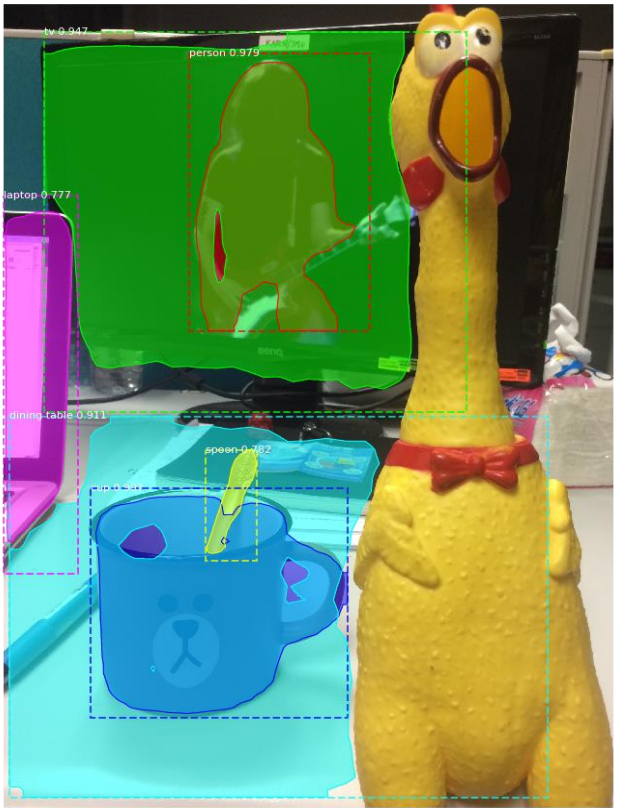
YOLO และ SSD จาก [5]

โครงสร้างสุดท้ายที่ดูน่าสนใจคือ Feature Pyramid Network (FPN) ที่นอกจะทาย box จากหลาย scale แล้วยังมีการส่งข้อมูลจาก scale ใหญ่ไปเล็ก (ขาลง) และจากเล็กกลับมาใหญ่ (ขาขึ้น) นอกจากนี้ยังมีการส่งข้อมูลข้ามใน scale เดียวกันอีกด้วย โครงสร้างนี้ดูแล้วเหมือนกับ U-Net ที่ใช้ในงาน image segmentation โดย FPN นั้นอิงแนวความคิดว่าการนำเอาข้อมูลจากภาพใหญ่ แต่ได้จากการสร้างขาขึ้น นั้นทำให้ได้ข้อมูลที่มี high semantic content (เพราะจริง ๆ แล้วมันนับได้ว่าอยู่ใน layer ชั้นบน ๆ) ใน high resolution (เพราะเป็นภาพใหญ่) ใน CNN ปกติข้อมูล high semantic content มักอยู่ที่ low resolution นั่นคือ FPN สามารถดึงข้อมูลที่มี high semantic content จากทุก ๆ resolution ได้



FPN ตรงกับ (d), YOLO ตรงกับ (b) และ SSD ตรงกับ (c) จาก [6]

สังเกตุได้คือโครงสร้างเหล่านี้สามารถแบ่งได้คร่าว ๆ เป็น 2 ส่วนคือ 1) ส่วนสกัด feature ผมเห็นบางคนก็เรียกส่วนนี้ว่าเป็น backend หรือ backbone ของ object detector และ 2) ส่วนที่นำเอา feature เหล่านี้ไปตรวจจับวัตถุในภาพ ส่วน backend นี้มักต้องถูก pre-trained มาก่อน ส่วนมากก็ทำบน Imagenet จากนั้นจึงเอามาต่อยอดส่วนที่ 2 และ re-train ใหม่ หลาย ๆ ครั้งโครงสร้างของส่วนที่ 1 ก็เป็นพวกที่ใช้ซ้ำ ๆ กันเช่น VGG, ResNet, MobileNet เป็นต้น ในปัจจุบันคนที่ทำงานคอมพิวเตอร์วิทัศน์คงต้องเริ่ม update พวก keywords ใหม่โดยเพิ่มชื่อพวก R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD, FPN เข้าไปด้วยแล้ว



ตัวอย่างการตรวจจับวัตถุ [7]

4. Confusion Matrix

Confusion Matrix คือตารางสำคัญในการวัดความสามารถของ machine learning ในการแก้ปัญหา classification

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

ตัวอย่างตาราง Confusion Matrix ขนาด 2x2 [8]

1. Precision เป็นการวัดความแม่นยำของข้อมูล โดยพิจารณาแยกทีละคลาส

$$\frac{TP}{TP + FP}$$

สมการหาค่า Precision

2.Recall เป็นการวัดความถูกต้องของ Model โดยพิจารณาแยกทีละคลาส

$$\frac{TP}{TP + FN}$$

สมการหาค่า Recall

3.Accuracy เป็นการวัดความถูกต้องของ Model โดยพิจารณารวมทุกคลาส

$$\frac{TP + TN}{TP + TN + FP + FN}$$

สมการหาค่า Accuracy

Code python

DetecrWeapon.py > ...

```python
import torch
import numpy as np
import cv2
import time


class ObjectDetection:

    def __init__(self, capture_index, model_name):

        self.capture_index = capture_index
        self.model = self.load_model(model_name)
        self.classes = self.model.names
        self.device = 'cuda' if torch.cuda.is_available() else 'cpu'
        print("Using Device:",self.device)


    def load_model(self,model_name):
        """
        โหลดโมเดล Yolo5 จากฮับ pytorch
        :return: โมเดล Pytorch ที่ผ่านการฝึกอบรม
        """
        if model_name:
            model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_name, force_reload=True)
        else:
            model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
        return model
```

```python

    def score_frame(self, frame):
        """
        รับเฟรมเดียวเป็นอินพุต และให้เฟรมเรทโดยใช้โมเดล yolo5
        :param frame: ใส่เฟรมในรูปแบบ numpy/list/tuple
        :return: ป้ายกำกับและพิกัดของวัตถุที่ตรวจพบโดยโมเดลในเฟรม
        """
        self.model.to(self.device)
        frame = [frame]
        results = self.model(frame)

        labels, cord = results.xyxyn[0][:, -1], results.xyxyn[0][:, :-1]
        return labels, cord


    def class_to_label(self, x):
        """
        สำหรับค่าป้ายกำกับที่กำหนด ให้ส่งคืนป้ายกำกับสตริงที่เกี่ยวข้อง
        :param x: ป้ายตัวเลข
        :return: ป้ายกำกับสตริงที่เกี่ยวข้อง
        """
        return self.classes[int(x)]
```

```
    def plot_boxes(self, results, frame):
        """
        รับเฟรมและผลลัพธ์เป็นอินพุต และพล็อตกล่องที่มีขอบเขตและป้ายกำกับไปที่เฟรม
        :param results: มีป้ายกำกับและพิกัดที่คาดการณ์โดยโมเดลในเฟรมที่กำหนด
        :param frame: เฟรมที่ได้คะแนนแล้ว
        :return: เฟรมที่มีกรอบล้อมรอบและป้ายกำกับที่วาดไว้
        """
        labels, cord = results
        n = len(labels)
        x_shape, y_shape = frame.shape[1], frame.shape[0]
        for i in range(n):
            row = cord[i]
            if row[4] >= 0.3 :
                x1, y1, x2, y2 = int(row[0]*x_shape), int(row[1]*y_shape), int(row[2]*x_shape), int(row[3]*y_shape)
                bgr = (0, 255, 0)
                cv2.rectangle(frame, (x1, y1), (x2, y2), bgr, 2)
                cv2.putText(frame, self.class_to_label(labels[i]), (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.9, bgr, 2)

        return frame
```

```
    def __call__(self):
        """
        ฟังก์ชันนี้ถูกเรียกเมื่อคลาสทำงาน มันรันลูปเพื่ออ่านวิดีโอทีละเฟรม
        และเขียนผลลัพธ์ลงในไฟล์ใหม่
        :return: void
        """
        cap = cv2.VideoCapture(0)

        while cap.isOpened():

            start_time = time.perf_counter()
            ret, frame = cap.read()
            if not ret:
                break
            results = self.score_frame(frame)
            frame = self.plot_boxes(results, frame)
            end_time = time.perf_counter()
            fps = 1 / np.round(end_time - start_time, 3)
            cv2.putText(frame, f'FPS: {int(fps)}', (20,70), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0,0,255), 2)
            cv2.imshow("img", frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

#สร้างวัตถุใหม่และดำเนินการ
detection = ObjectDetection(capture_index=0,model_name='best.pt')
detection()
```

เอกสารอ้างอิง

[1] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." In CVPR 2016

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C.-Y. Fu and A.C. Berg. "SSD: Single Shot MultiBox Detector." In ECCV 2016

[3] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S.J. Belongie "Feature Pyramid Networks for Object Detection." In CVPR 2017

[4] https://sanchom.wordpress.com/tag/average-precision/

[5] The PASCAL Visual Object Classes Challenge (VOC2012). http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2012/index.html.

[6] Python example code https://github.com/Cartucho/mAP