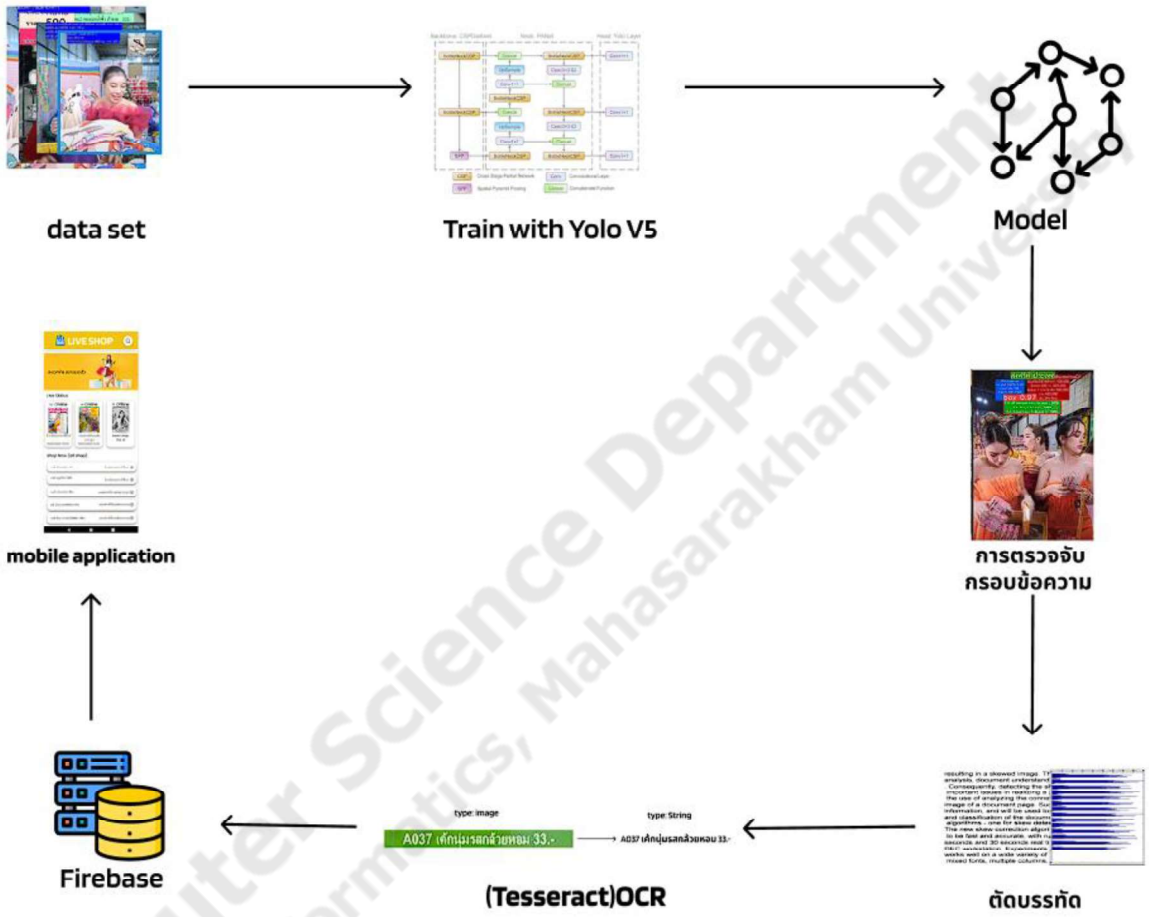
บทที่ 3

ขั้นตอนการดำเนินงาน



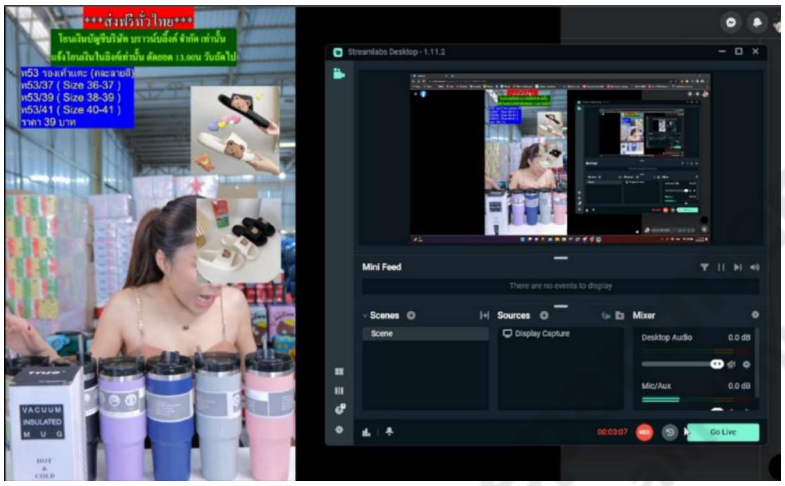
ภาพประกอบที่ 3.1 ขั้นตอนการดำเนินงานของระบบ

สำหรับในบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานของโครงงานปริญญานิพนธ์ซึ่งจะทำให้ทราบถึงการวิเคราะห์และการออกแบบแอพพลิเคชั่นโดยละเอียดว่ามีแนวทางในการดำเนินงานหรือมีขั้นตอนในการทำงานของแอพพลิเคชั่นอย่างไรบ้างโดยขั้นตอนในการดำเนินงานมีรายละเอียดดังนี้ การรวบรวมข้อมูล การเทรนโมเดลด้วย YoloV5 การตัดบรรทัดข้อความ การแปลงรูปภาพเป็นตัวอักษร การเก็บข้อมูลลงในฐานข้อมูล และการนำข้อมูลไปแสดงผลใน mobile application

3.1 เก็บรวบรวมข้อมูล

เก็บข้อมูลโดยการบันทึกคลิปวิดีโอโดยใช้ Streamlabs Desktop ในการบันทึกคลิปวิดีโอแล้วนำวิดีโอมาเปิดและบันทึกเป็นภาพนิ่ง ทำการบันทึกวิดีโอจำนวน 3 ร้านค้าได้แก่ 1.ร้าน FIRST SHOP V2 2.ร้านKANYA SHOP ขายถูกทุกอย่าง 3.ร้านมหัศจรรย์"วันของ AuuM เงื่อนไขในการเก็บ

โดยมีการเก็บวิดีโอเป็น 1 ร้านจะมี 3 วิดีโอไลฟ์สด 1 วิดีโอจะแบ่งเป็น 6 คลิปวิดีโอ 1 คลิปวิดีโอ จะแบ่งเป็น คลิปละ 3 นาที



ภาพประกอบที่ 3.2 Streamlabs Desktop

ตารางที่ 3.1 ตารางจำนวนและเวลาเฉลี่ยของวิดีโอในแต่ละร้านค้า

ชื่อร้านค้า	เวลาเฉลี่ยของวิดีโอ	จำนวน
ร้าน FIRST SHOP V2	3 นาที	18 วิดีโอ
ร้าน KANYA SHOP ขายถูกทุกอย่าง	3 นาที	18 วิดีโอ
ร้านมหัศจรรย์"วันของAuuM	3 นาที	18 วิดีโอ

3.2 การเตรียมข้อมูล

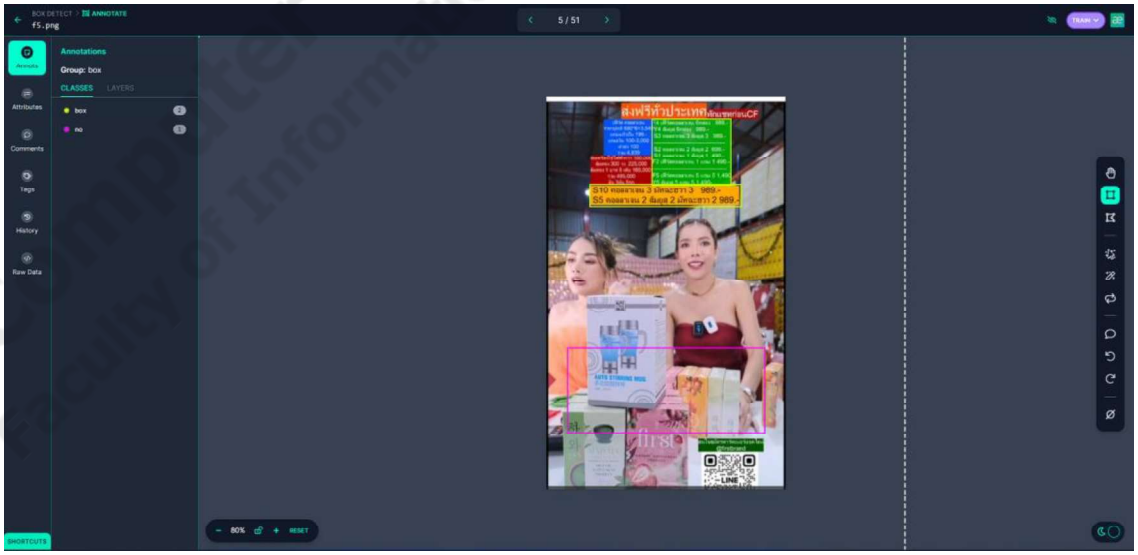
3.2.1 การเตรียมข้อมูลในการ training

เตรียมข้อมูลในการ training โดยบันทึกรูปภาพจาก Video ไลฟ์สดที่เก็บมาได้โดยจะทำการบันทึกเฉพาะช่วงที่มีการเปลี่ยนแปลงของข้อความบนหน้าจอไลฟ์สดดัง โดยจะได้จาก ร้าน FIRST SHOP V2 จำนวน 33 ภาพ จากร้าน KANYA SHOP ขายถูกทุกอย่าง จำนวน 28 ภาพ จากร้านมหัศจรรย์"วันของAuuM" จำนวน 33 ภาพ รวมกันทั้งหมด 92 รูปภาพ

ภาพประกอบที่ 3.3 ตัวอย่างรูปภาพที่ทำการเก็บ

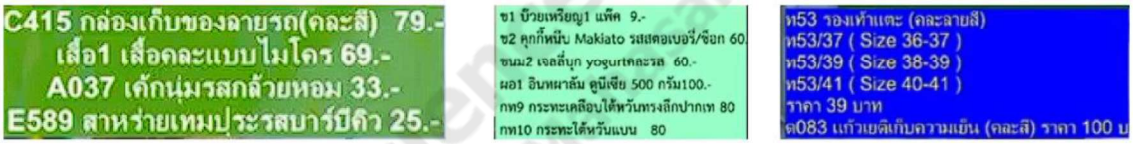
3.2.1 การวาดภาพผลเฉลย

ทำการวาดภาพผลเฉลยรูปภาพผ่าน Roboflow framework สำหรับใช้จัดเก็บ เตรียมชุดข้อมูล และสร้างแบบจำลองต่างๆ ที่สามารถใช้งานผ่าน web browser โดยจะทำการวาดภาพผลเฉลยในรูปภาพที่เตรียมมาทั้งหมด โดยจะจัดหมวดหมู่ในที่ที่สนใจเป็น “box”และในพื้นที่ที่ไม่สนใจเป็น “no”



ภาพประกอบที่ 3.4 การวาดภาพผลเฉลยรูปภาพใน Roboflow

ภาพประกอบที่ 3.5 ผลลัพท์การวาดภาพผลเฉลยรูปภาพใน Roboflow



ภาพประกอบที่ 3.6 ภาพผลเฉลยที่ถูกต้องจากทั้ง 3 ร้านค้า

ตารางที่ 3.2 จำนวนของข้อมูลในแต่ละส่วน

Training Set	Validation Set	Testing Set
62	17	13

เมื่อทำการวาดภาพผลเฉลยรูปภาพทั้งหมดจะทำการแบ่งส่วนของข้อมูลออกเป็นดังนี้โดย จำนวนของ Training Set จะแบ่งเป็น 62 รูปภาพ Validation Set 17 รูปภาพและ Testing Setมีจำนวน 13 รูปภาพ โดยจำนวนของ Training Set จะมีจำนวนที่มากกว่าเนื่องจาก ผลเฉลยที่สนใจมีรูปแบบที่คล้ายกัน

3.2.1 การทำ Data Augmentation

ในส่วนนี้จะนำข้อมูลไปทำการ Generate รูปภาพเพิ่มเพื่อเพิ่มจำนวนของข้อมูลที่จะทำไป train โดยจะทำการ หมุนภาพ 90 องศา,ทำให้เป็นภาพสีเทา,เพิ่มแสงสดแสง,เพิ่มสิ่งรบกวนในรูปภาพ และปรับขนาดของรูปภาพเป็น 416 x 416 pixel ทุกรูปภาพ

ภาพประกอบที่ 3.7 การเพิ่มจำนวนรูปภาพใน Traing Set

จะทำการ Generate ทั้งหมด 3 ครั้ง โดยจะได้รูปภาพจากการ Generate ทั้งหมด 186 ภาพจะมีรูปภาพที่ใช้ในการ train ทั้งหมด 558

ตารางที่ 3.3 จำนวนของข้อมูลก่อนและหลังทำการ Generate

	Training Set	Validation Set	Testing Set
ก่อน Generate	62	17	13
หลัง Generate	186	17	13
หลัง Generate 3 ครั้ง	558	17	13

ภาพประกอบที่ 3.8 ตัวอย่างรูปภาพที่ Generate ออกมา

```
0 0.6838942307692307 0.12259615384615384 0.4951923076923077 0.07211538461538461
1 0.28846153846153844 0.8161057692307693 0.5 0.1875
```

ภาพประกอบที่ 3.9 ผลเฉลยของรูปภาพที่ Generate ออกมา

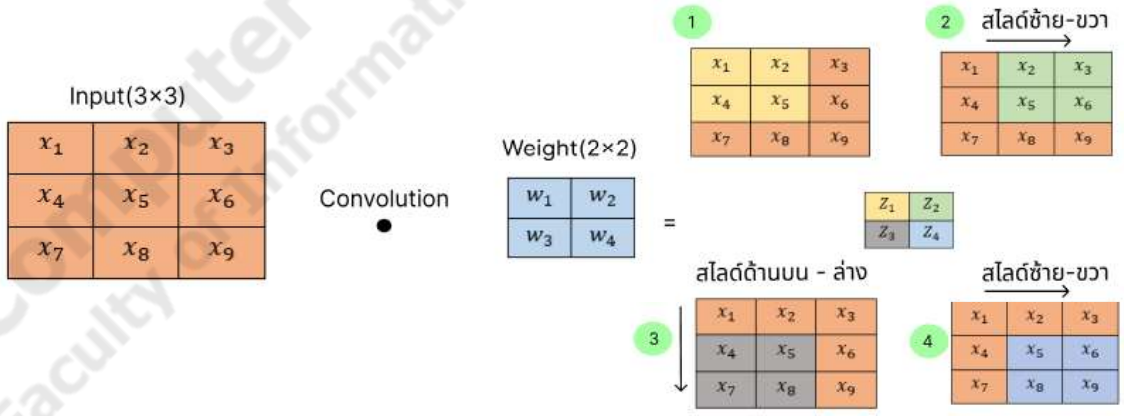
โดยในสิ่งที่ผลเฉลยอธิบายจะมีดังนี้ 1.คลาสของสิ่งที่สนใจในรูป 2.ตำแหน่งเริ่มต้นของกรอบสิ่งที่ detect เจอ 3.ตำแหน่งสิ้นสุดของกรอบสิ่งที่ detect เจอ โดยจะนำตำแหน่งที่ได้มามาหารกับค่า ความกว้างและความสูงของรูปราพ คือ 416*416 จึงได้ค่าออกมาเป็นทศนิยมเพื่อจะได้นำไปปรับใช้กับหน้าจอได้หลายขนาด

3.3 การสร้างโมเดลโดยใช้ yoloV5

ในส่วนนี้จะเป็นการอธิบายถึงโครงสร้างของ yolov5 และการเทรนข้อมูลโดยจะมีขั้นตอนการทำงานแบ่งเป็นส่วนหลักๆดังนี้ ในส่วนที่ 1 จะเป็นสำหรับการคัดกรองภาพเพื่อดึงลักษณะเด่นของรูปภาพออกมาโดยจะมีขั้นตอนการทำงานดังนี้

3.3.1 Convolution Layer

การทำ Convolution Layer เพื่อสกัดเอาส่วนต่างๆ ของภาพออกมา เช่น เส้นขอบของวัตถุต่างๆ เพื่อให้โมเดลสามารถเรียนรู้ลักษณะของภาพได้อย่างมีประสิทธิภาพและแม่นยำโดย ขั้นตอนจะมีรูปภาพที่รับเข้ามาเป็น Matrix input ขนาด 3x3 เป็นรูปภาพของเราและมี Filter ขนาด 2x2



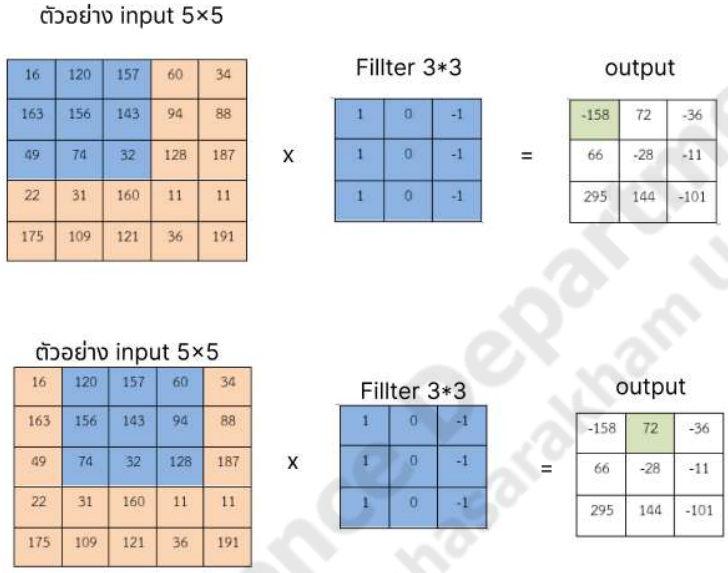
ภาพประกอบที่ 3.10 ขั้นตอนการทำ Convolution

ขั้นตอนการคำนวณคือ หาผลรวมของการคูณระหว่าง Input กับ weight โดยใช้ weight ชุดเดิมแล้ว สแกน ไปทั้ง Input จากซ้ายไปขวา และบนลงล่าง (1)-(4) ผลลัพธ์ (Z) ที่ได้คือ

$$z1 = (w1 * x1) + (w2 * x2) + (w3 * x4) + (w4 * x5)$$

$$z2 = (w1 * x2) + (w2 * x3) + (w3 * x5) + (w4 * x6)$$
$$z3 = (w1 * x4) + (w2 * x5) + (w3 * x7) + (w4 * x8)$$
$$z4 = (w1 * x5) + (w2 * x6) + (w3 * x8) + (w4 * x9)$$

ตัวอย่างการทำงานของ Convolution ของ inputขนาด 5x5 กับ Filter ขนาด3x3



ภาพประกอบที่ 3.11 ตัวอย่างการคำณวนของ Convolution

$$z1 = (16 * 1) + (120 * 0) + \big(157 * (-1)\big) + (163 * 1) + (156 * 0) + \big(143 * (-1)\big) + (49 * 1) + (74 * 0) + \big(32 * (-1)\big)$$
$$= -158$$

$$z2 = (120 * 1) + (157 * 0) + \big(60 * (-1)\big) + (156 * 1) + (143 * 0) + \big(94 * (-1)\big) + (74 * 1) + (32 * 0) + \big(128 * (-1)\big)$$
$$= 72$$

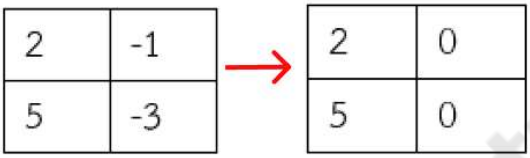
3.3.1 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) เป็นฟังก์ชัน Activation Function ที่นิยมใช้ใน Deep Learning เพื่อปรับค่าผลลัพธ์ให้เป็นค่าบวก โดยถ้า x มีค่าเป็นลบจะแทนค่า x ตัวนั้นกลายเป็น 0 โดยหาได้จากสมการต่อไปนี้

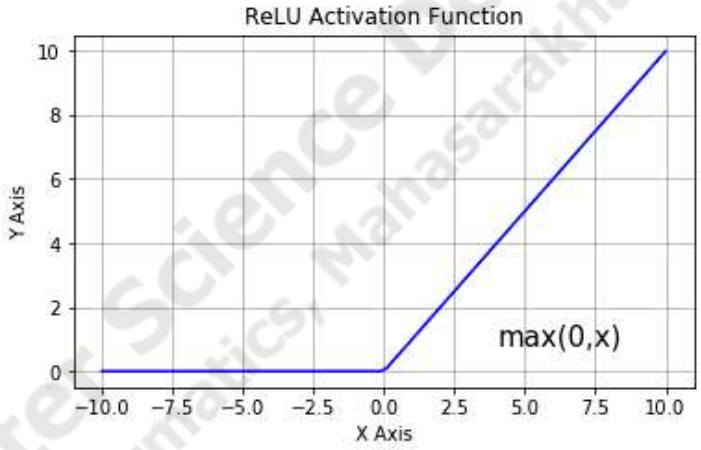
$$f(x) = \max(0, x)$$

ตัวอย่างการคำนวณ ReLU

$$x = 2: f(2) = \max(0, 2) = 2$$
$$x = -1: f(-1) = \max(0, -1) = 0$$
$$x = 5: f(5) = \max(0, 5) = 5$$
$$x = -3: f(-3) = \max(0, -3) = 0$$



ภาพประกอบที่ 3.12 การคำนวน ReLU



ภาพประกอบที่ 3.13 กราฟของ ReLU

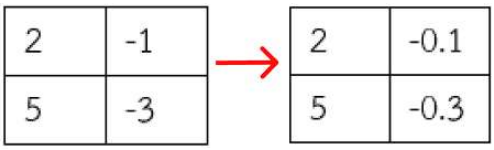
3.3.2 ReLU leaky

ฟังก์ชัน ReLU นั้นมีจุดขาดที่เรียกว่า ReLU leaky โดยจุดขาดนี้เกิดจากเมื่อค่านิพจน์ที่ผ่านมามีค่าติดลบมาก ๆ จะทำให้ gradient หายไปและไม่สามารถปรับค่าได้ จึงมีการพัฒนา ReLU leaky เพื่อแก้ไขปัญหานี้ ReLU leaky จะไม่ให้ค่าลบเป็น 0 แต่จะแทนที่ด้วยค่าที่เล็กๆ โดยที่ค่านี้จะกำหนดได้ โดยทั่วไปจะใช้ค่าเดียวกับ alpha ซึ่งเป็นค่าเล็กๆ อยู่ในช่วง 0 ถึง 1 ซึ่งช่วยให้ gradient ยังคงมีค่าได้ และช่วยให้โมเดลมีความสามารถในการเรียนรู้ได้ดีขึ้นโดยมีสมการดังต่อไปนี้

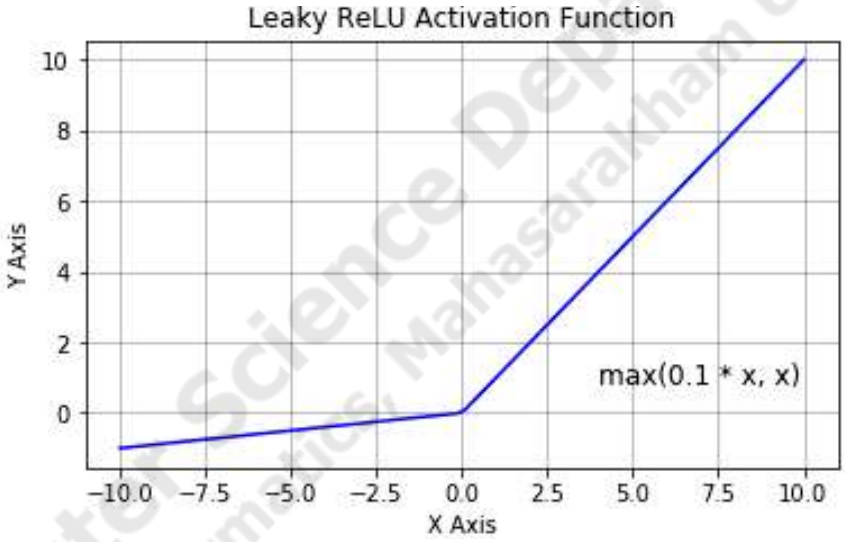
$$f(x) = \max(alpha * x, x)$$

ตัวอย่างการคำนวณ ReLU leaky
โดยจะกำหนดให้ $alpha = 0.1$

$$x = 2: f(2) = \max(0.1 * 2, 2) = 2$$
$$x = -1: f(-1) = \max(0.1 * -1, -1) = -0.1$$
$$x = 5: f(5) = \max(0.1 * 5, 5) = 5$$
$$x = -3: f(-3) = \max(0.1 * -3, -3) = -0.3$$

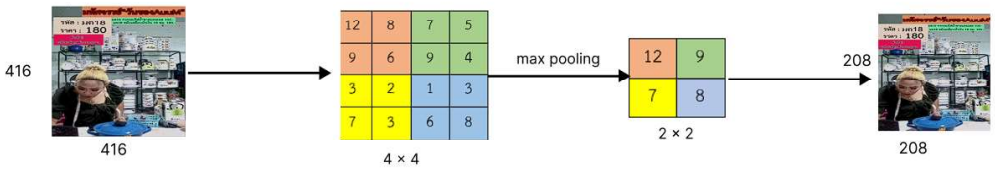


ภาพประกอบที่ 3.14 การคำนวณ ReLu Leaky



ภาพประกอบที่ 3.15 กราฟของ ReLU Leaky

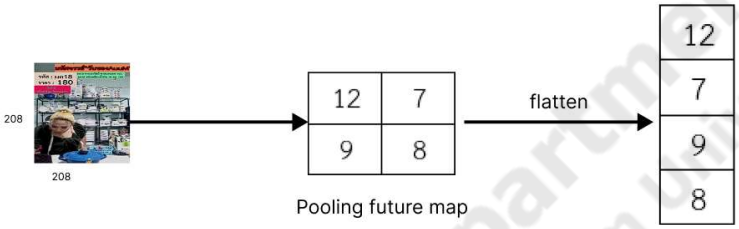
หลังจากทำการ ReLU เพื่อทำให้ค่าเป็นบวกแล้วจะทำการ Pooling layer เพื่อทำการการสกัดหลือลดขนาดของข้อมูล เพื่อเอาส่วนที่สำคัญที่สุดของข้อมูลโดยทั่วไปมักจะเลือกใช้ max pooling หรือ average pooling



ภาพประกอบที่ 3.16 Pooling layer

3.3.1 Pooling layer

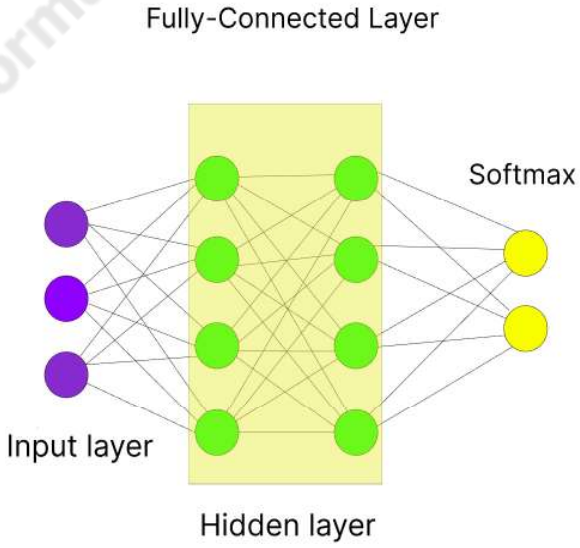
การทำ Pooling layer คือการสกัดหลือลดขนาดของข้อมูลfuture mapที่ได้จากการทำ Convolution เพื่อเอาส่วนที่สำคัญที่สุดของข้อมูล และเพิ่มประสิทธิภาพการประมวลผลให้รวดเร็วยิ่งขึ้น โดย Max Pooling layer คือการสกัดเอาเฉพาะค่าสูงสุดของ Matrix เก็บไว้ใน Output เช่นจากภาพ Max Pooling layer ที่มีขนาด 2x2



ภาพประกอบที่ 3.17 การทำ Flatten

3.3.2 Flatten

จะทำการแปลงข้อมูลจากชั้น Pooling layer ที่ได้มา ให้กลายเป็น vector โดยที่ข้อมูลทุกตัวจะถูกวางต่อกันเป็นแถวเดียวกัน โดยที่ไม่มีการเปลี่ยนแปลงค่าข้อมูล เช่น ถ้ามี matrix ขนาด 2x2 แล้วทำการ flatten จะได้ vector ขนาด 4 (2 x 2 =4) โดยที่ข้อมูลจะเรียงตามลำดับของแถวและคอลัมน์ของเมทริกซ์เดิม ก่อนจะนำไปทำในขั้นตอนของ Fully Connected Layer



ภาพประกอบที่ 3.18 Fully Connected Layer

3.3.1 Fully Connected Layer

Fully Connected Layer คือ การรวมผลลัพธ์ของตัวแปร input จาก Layer ก่อนหน้าทั้งหมด และคูณด้วยน้ำหนัก (weight) ของแต่ละโหนด (neuron) ใน Fully-Connected Layer นี้ โดยที่แต่ละโหนดจะมี weight และ bias ของตัวเองที่แตกต่างกันไป การคูณนี้จะทำให้ได้ผลลัพธ์เป็นเวกเตอร์หรือเมทริกซ์ขนาดเล็กกว่า input และเมทริกซ์นี้จะถูกส่งต่อไปยัง Layer ถัดไป

3.3.2 Softmax

Softmax คือฟังก์ชันทางคณิตศาสตร์ที่มักถูกใช้ในการแปลงค่าของหลายอินพุท (input) เป็นค่าความน่าจะเป็น (probability) โดยที่ผลรวมของค่าความน่าจะเป็นทั้งหมดเท่ากับ 1 นี่มักถูกใช้ในการคลาสสิฟิเคชัน (classification) และมาสเตอร์ไลบรารีของประสานสัมพันธ์ (Neural Networks) เมื่อต้องการคำนวณความน่าจะเป็นของแต่ละคลาสต่อจากค่าผลลัพธ์หลาย ๆ คลาส

ฟังก์ชัน Softmax คำนวณผลรวมของค่าเอ็นเทรี่ (input) และหลังจากนั้นแปลงแต่ละค่าให้อยู่ในช่วงระหว่าง 0 ถึง 1 โดยใช้สูตรดังนี้

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{N} e^{z_j}}$$

โดยที่

$\sigma(z_j)$ คือผลลัพท์ที่ได้จากการคำนวณ softmax ของ

e คือค่าของเลขอิสระ (constant) ซึ่งเป็นค่าประมาณ 2.71828

z_i คือค่า input หรือ zi ที่เราต้องการคำนวณ softmax สำหรับมัน

N คือจำนวนข้อมูล input ทั้งหมด

ตัวอย่างการคำนวณของ softmax

ขั้นตอนที่ 1 ยกตัวอย่างค่า $\mathrm{z_i} = (4.59, 1.00)$

ขั้นตอนที่ 2 ทำการแทนค่า $\mathrm{z_i}$ ลงในสมการ

$$\sigma(4.59, 1.00) = \left(\frac{\mathrm{e}^{4.59}}{\mathrm{e}^{4.59} + \mathrm{e}^{1.0}}, \frac{\mathrm{e}^{1.0}}{\mathrm{e}^{4.59} + \mathrm{e}^{1.0}}\right)$$

ขั้นตอนที่ 3 ทำการแทนค่า e ลงในสมการ

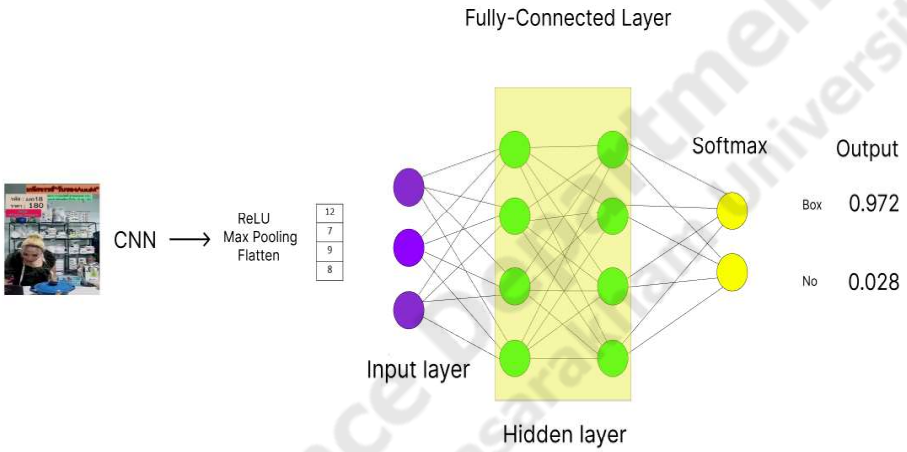
$$\sigma(4.59, 1.00) = \left(\frac{2.71828^{4.59}}{2.71828^{4.59} + 2.71828^{1.0}}, \frac{2.71828^{1.0}}{2.71828^{4.59} + 2.71828^{1.0}}\right)$$

ขั้นตอนที่ 4 ทำการคำนวณเพื่อหาผลลัพท์

$$\sigma(4.59,1.00) = (\frac{98.7}{98.7 + 2.72}, \frac{2.72}{98.7 + 2.72})$$

$$\sigma(4.59,1.00) \approx (0.972,0.028)$$

ดังนั้นค่า softmax เมื่อ zi=(4.59,1.00) คือ (0.972, 0.028) โดยประมาณ แบ่งออกเป็นค่าแรกเป็นประมาณ0.972 และค่าที่สองเป็นประมาณ 0.028



ภาพประกอบที่ 3.19 ตัวอย่างการคำนวณหาค่า softmax

ส่วนที่ 2 เป็นส่วนที่ทำหน้าที่ Detect Object โดยใช้การสร้าง bounding box โดยการคำนวณพิกัดของ object ที่ Convolution Layer เพื่อทำนายความน่าจะเป็นของแต่ละ Object Class และ bounding box coordinates

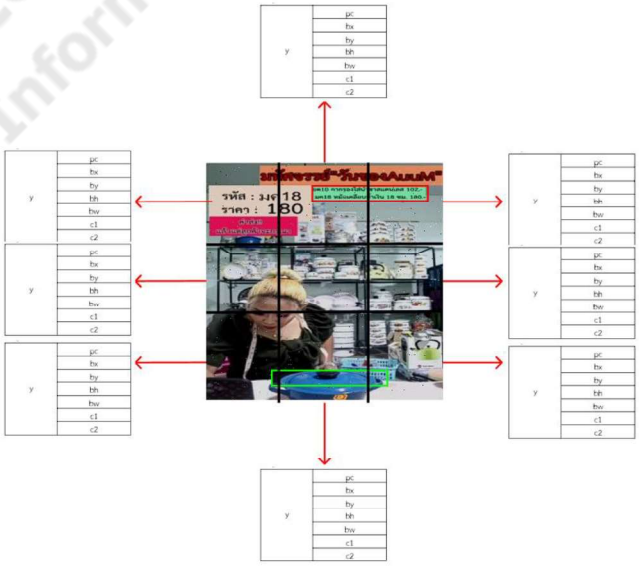
ภาพประกอบที่ 3.20 รูปภาพ input ที่ใส่ grid

การกำหนดข้อมูลเทรนจะมีขั้นตอนดังต่อไปนี้ เราจะต้องส่งข้อมูลที่วาดภาพผลเฉลยแล้วไปยังโมเดลเพื่อฝึกฝน และจะแบ่งภาพออกเป็นตารางขนาด 3 X 3 และมีทั้งหมด 2 คลาสที่ต้องการให้วัตถุถูกจัดประเภท ซึ่งใน 2 คลาสจะมี คลาส’box’ที่เป็นจุดที่เป็นกรอบของข้อความและคลาส ’no’ เป็นจุดที่ไม่สนใจ ดังนั้นสำหรับแต่ละเซลล์ใน grid จะมี ค่าผลเฉลย y เจ็ดค่า ดังต่อไปนี้

ตารางที่ 3.4 ค่า parameter ใน ค่าผลเฉลย y ทั้ง 7

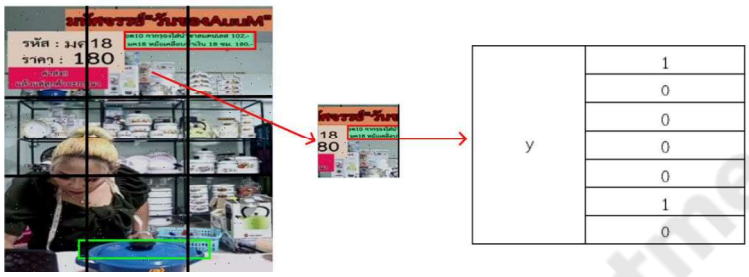
y	pc
	bx
	by
	bh
	bw
	c1
	c2

pc คือ ค่าที่กำหนดว่าวัตถุมีอยู่ในตารางหรือไม่
bx คือ ค่าที่ระบุตำแหน่งในแกน x ของ Bounding Box ของวัตถุเมื่อมีวัตถุภายในภาพ
by คือ ค่าที่ระบุตำแหน่งในแกน y ของ Bounding Box ของวัตถุเมื่อมีวัตถุภายในภาพ
bh คือ ค่าที่ระบุตำแหน่งความสูงของ Bounding Box ของวัตถุเมื่อมีวัตถุภายในภาพ
bw คือ ค่าที่ระบุตำแหน่งความกว้างของ Bounding Box ของวัตถุเมื่อมีวัตถุภายในภาพ
c1 คือ คลาสของวัตถุที่เจอในภาพที่เจอโดยคลาสนี้จะมีชื่อว่า ‘box’ ถ้าเจอวัตถุจะดังกล่าวค่าจะเป็น 1 ถ้าไม่ใช่จะเป็น 0
c2 คือ คลาสของวัตถุที่เจอในภาพที่เจอโดยคลาสนี้จะมีชื่อว่า ‘no’ ถ้าเจอวัตถุจะดังกล่าวค่าจะเป็น 1 ถ้าไม่ใช่จะเป็น 0



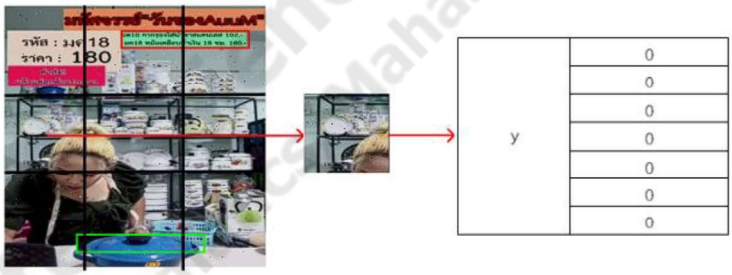
ภาพประกอบที่ 3.21 ค่าผลเฉลย y ในแต่ละ grid cell

การกำหนด input ของรูปภาพสำหรับ grid นี้ YOLO จะตัดสินใจว่ามีวัตถุอยู่ใน grid จริงหรือไม่โดย YOLO จะใช้จุดกึ่งกลางของวัตถุและวัตถุเหล่านั้นจะถูกกำหนดให้กับ grid ที่มีจุดกึ่งกลางของวัตถุนั้น มี ค่าผลเฉลย y สำหรับ grid ช่องนั้นๆ ถ้าเจอวัตถุในภาพ



ภาพประกอบที่ 3.22 ตัวอย่าง grid ที่มีวัตถุในภาพและการแทนค่า

ในกรณีที่เจอวัตถุใน grid ในตารางนี้ pc = 1 และในส่วนbx, by, bh, bw จะถูกคำนวณเทียบกับเซลล์ grid ที่ทำการคำนวณอยู่ในภายหลัง และเนื่องจากเจอกรอบข้อความในรูปภาพ ค่า c1 = 1 และ c2 = 0



ภาพประกอบที่ 3.23 ตัวอย่าง grid ที่ไม่มีวัตถุในภาพและการแทนค่า

ในกรณีที่ไม่เจอวัตถุใน grid ในตารางนี้ pc = 0 เมื่อมีค่า pc = 0 เท่ากับว่าไม่มีวัตถุที่สนใจภายในภาพเราจะไม่แทนค่าต่อใน grid cell นี้

ภาพประกอบที่ 3.24 จุดกึ่งกลางของ Bounding Box

3.3.1 การแทนค่าความสูงแความกว้างของ Bounding Box

Bounding Box คือการวาดกล่องรอบวัตถุ ดังที่ได้กล่าวไว้ก่อนหน้านี้โดย bx, by, bh,bw จะถูกคำนวณเทียบกับเซลล์ grid ที่ทำการคำนวณอยู่ พิจารณา grid ตรงที่มีกรอบของตัวอักษร
bx คือพิกัด x ของจุดกึ่งกลางของวัตถุที่อยู่ใน grid นี้ ในกรณีนี้ bx = 0.9 โดยจะวัดจากจุดกึ่งกลาง
by คือพิกัด y ของจุดกึ่งกลางของวัตถุที่อยู่ใน grid นี้ ในกรณีนี้ bx = 0.4 โดยจะวัดจากจุดกึ่งกลาง
bh คืออัตราส่วนของความสูงของ Bounding Box กับความสูงของเซลล์ grid ที่เกี่ยวข้องซึ่งในกรณีของรูปภาพในตัวอย่างจะมีค่าคือประมาณ bh = 0.3 โดยประมาณ
bw คืออัตราส่วนของความกว้างของ Bounding Box กับความกว้างของเซลล์ grid ที่เกี่ยวข้องซึ่งในกรณีของรูปภาพในตัวอย่างจะมีค่าคือประมาณ bw = 1.5 โดยประมาณ
โดยค่า y สำหรับ grid นี้จะมีค่าดังนี้

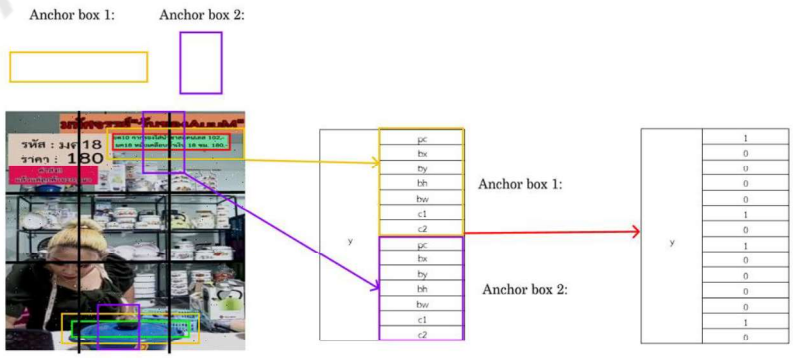
ตารางที่ 3.5 การแทนค่า bx, by, bh, bw

y	1
	0.9
	0.4
	0.3
	1.5
	1
	0

bx และ by จะอยู่ในช่วงระหว่าง 0 ถึง 1 เสมอเนื่องจากจุดกึ่งกลางจะอยู่ภายใน grid เสมอ ขณะที่ bh และ bw สามารถมากกว่า 1 ในกรณีที่ Bounding Box มากกว่าขนาดของ grid

3.3.2 การทำ anchor box

เป็นการสร้าง anchor box ที่มีรูปร่างต่างกันเพื่อเพิ่มความแม่นยำในการตรวจจับของโมเดลโดยถ้ามี anchor box 2 กล่อง ค่า label y โดยที่ค่า parameter 7 แถวแรกเป็นของ anchor box ที่ 1 และอีก 7 แถวที่เหลือเป็นของ anchor box ที่ 2 โดยจะเปลี่ยนแปลงไปดังนี้



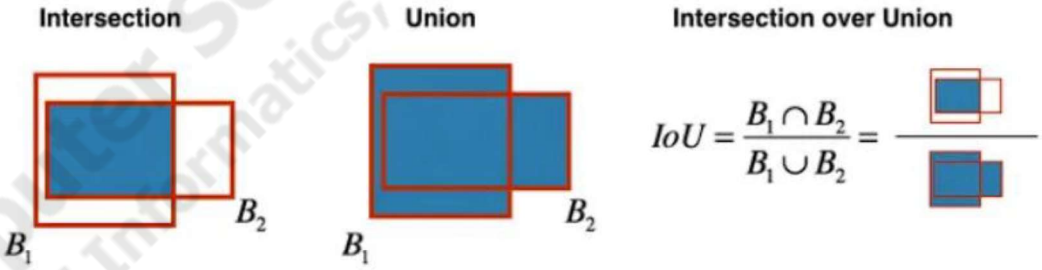
ภาพประกอบที่ 3.25 การแทนค่าใน anchor box และภาพรวมของ anchor box

ตารางที่ 3.6 parameter ของการทำ anchor box

y	pc
	bx
	by
	bh
	bw
	c1
	c2
	pc
	bx
	by
	bh
	bw
	c1
	c2

3.3.3 Intersection over Union

Intersection over Union คือ วิธีการวัดความเหมือนหมายถึงการเปรียบเทียบระหว่างกรอบสี่เหลี่ยมที่รอบตัวของวัตถุ (bounding box) ที่ระบุโดยโมเดลกับกรอบสี่เหลี่ยมที่ถูกต้อง (ground truth bounding box) โดยการคำนวณ IoU จะได้ผลลัพธ์ออกมาเป็นเลขระหว่าง 0 ถึง 1 โดยจะมีค่ามากเมื่อกรอบสี่เหลี่ยมที่ระบุโดยโมเดลมีการครอบคลุมวัตถุที่ถูกต้องมากขึ้น



ภาพประกอบที่ 3.26 IOU

box 1

1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1

box 2

1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

ภาพประกอบที่ 3.27 ตัวอย่างการคำนวณค่า IoU

ตัวอย่างของการหา IOU โดยภาพ box 1 คือ ภาพ Matrix แรก และภาพ box 2 คือ ภาพ Matrix ที่สอง และในภาพ box 1 union box 2 คือสีเขียว ส่วน box 1 intersection box 2 คือสีเขียว สีแดงคือทายขาด โดยสมาการดังนี้

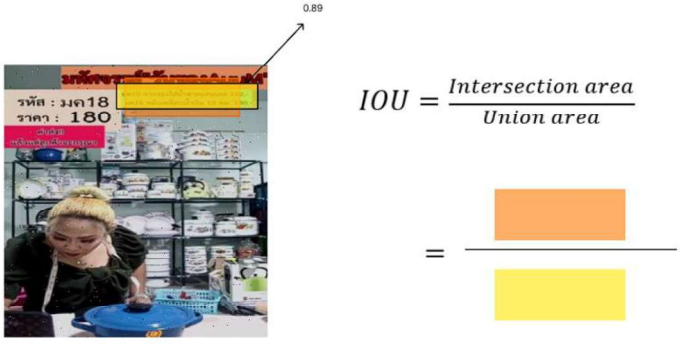
$$\text{Union area} = \text{box1} + \text{box2} - \text{intersection } area$$

$$\text{Union area} = 54 + 54 - 51 = 57$$

โดยค่า IoU จะหาได้จากสมการดังนี้

$$\text{IOU} = \frac{\text{Intersection area}}{\text{Union area}}$$

$$\text{IOU} = \frac{51}{57} = 0.89$$



ภาพประกอบที่ 3.28 ตัวอย่างของ ค่า IoU ในรูปภาพ

3.3.1 Non-Max Suppression

Non-Max Suppression เป็นเทคนิคหนึ่งที่ใช้ในการลดจำนวนของ bounding boxes หรือ detections ที่ซ้อนทับกันในการตรวจจับวัตถุ ซึ่งเป็นส่วนสำคัญในการประมวลผลภาพการฝึกฝนข้อมูล วิธีการทำงานของ Non-Max Suppression คือ การเลือก bounding box ที่มีค่า confidence score สูงสุด และกำจัด bounding box อื่น ๆ ที่ซ้อนทับอยู่ในพื้นที่เดียวกัน โดยมีขั้นตอนดังนี้
1.จัดเรียง bounding box ตามค่า confidence score จากมากไปน้อย
2.เลือก bounding box ที่มีค่า confidence score สูงสุด และเก็บไว้
3.ลบ bounding box ที่มีค่า IoU หรือ threshold น้อยกว่าหรือเท่ากับค่าที่กำหนด
4.ทำขั้นตอนที่ 2 และ 3 จนกว่าจะไม่มี bounding box ที่เหลือให้ตรวจสอบ

ตัวอย่างเมื่อกำหนดหนดค่า IoU ให้มากกว่าหรือเท่ากับ 0.5

กล่องที่	confidence score
1	0.9
2	0.6
3	0.75

→

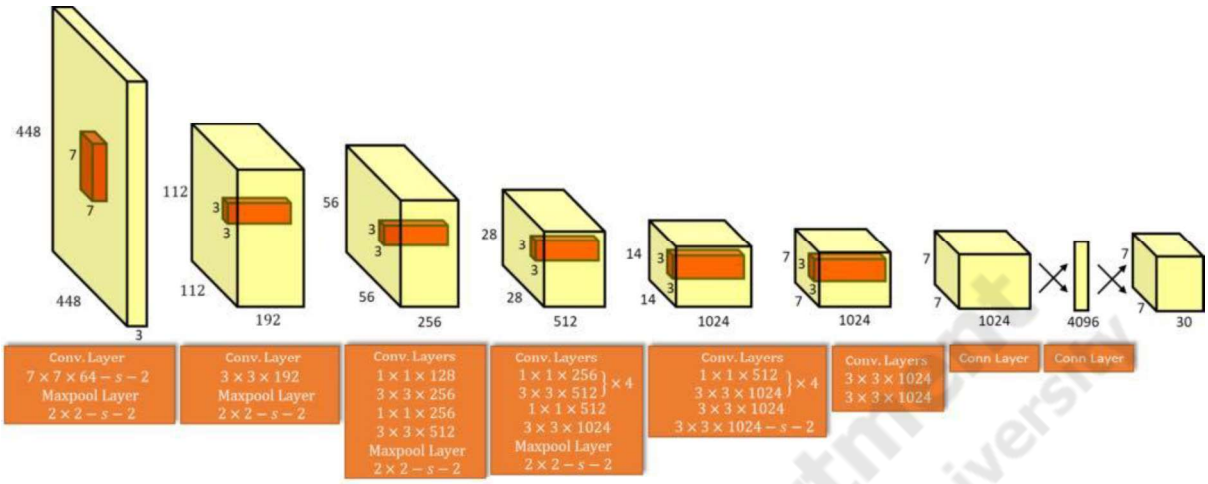
กล่องที่	confidence score
1	0.9

ภาพประกอบที่ 3.29 ตัวอย่างการทำ Non-Max Suppression

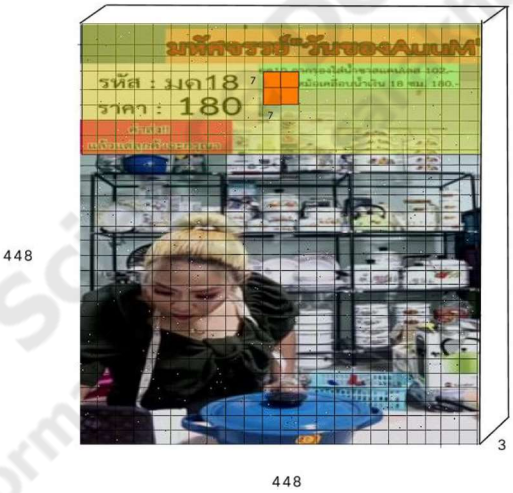
ภาพประกอบที่ 3.30 ตัวอย่างการทำ Non-Max Suppression

3.3.1 โครงสร้างของ Yolov5

ในโครงสร้างของ yolov5 นั้นเป็นการแสดงรูปแบบของรูปภาพเมื่อผ่าน layer ต่างๆใน โครงสร้างของ yolov5 จะมีขั้นตอนการดำเนินงาน 9 ขั้นตอนโดยจะมีวิธีการดำเนินการดังต่อไปนี้

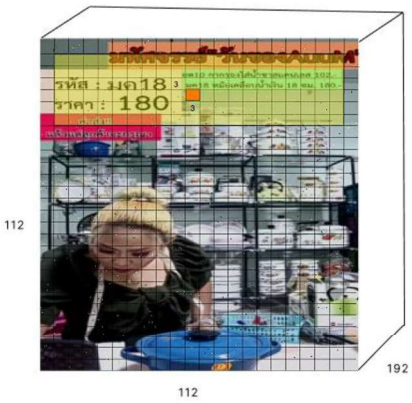


ภาพประกอบที่ 3.31 Layer ของ Yolov5



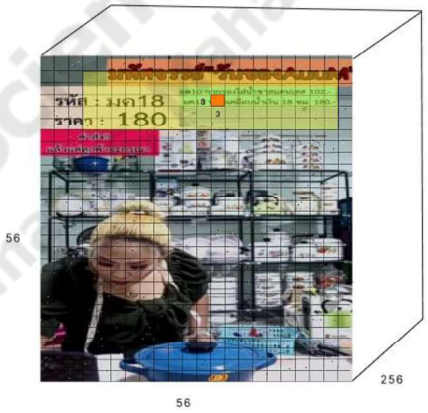
ภาพประกอบที่ 3.32 รูปภาพผลลัพท์ในขั้นตอนที่ 1

ขั้นตอนที่ 1 จะทำการรับรูปภาพขนาด 448*448 pixels เข้ามาและทำการ ผ่านการทำ Convolution ที่มี kernel ขนาด 7*7 และมี feature map ใหม่โดยใช้ filters จำนวน 64 ตัว และkernel จะเคลื่อนที่ทีละ 2 ช่องและทำการผ่าน Maxpool Layer เพื่อทำการลดขนาดของ feature map โดย Maxpool Layer จะมีขนาด2*2และจะทำการขยับไปทีละ 2 ช่อง



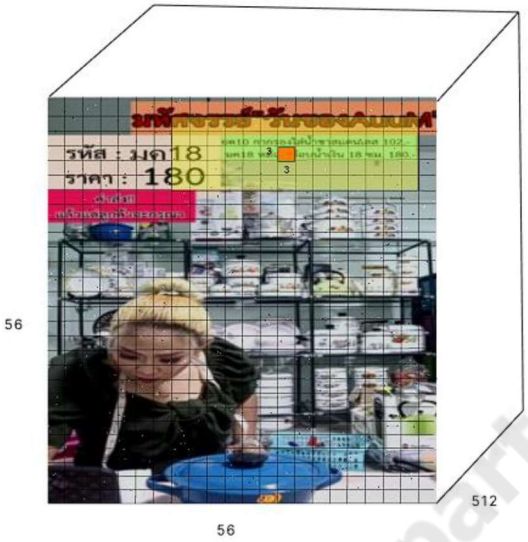
ภาพประกอบที่ 3.33 รูปภาพผลลัพท์ในขั้นตอนที่ 2

ขั้นตอนที่ 2 จะทำการนำ feature map เดิมเข้ามาและทำการ ผ่านการทำ Convolution ที่มี kernel ขนาด 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 129 ตัวและ kernel จะเคลื่อนที่ทีละ1 ช่องและทำการผ่าน Maxpool Layer เพื่อทำการลดขนาดของ feature map โดย Maxpool Layer จะมีขนาด2*2และจะทำการขยับไปทีละ 2 ช่อง



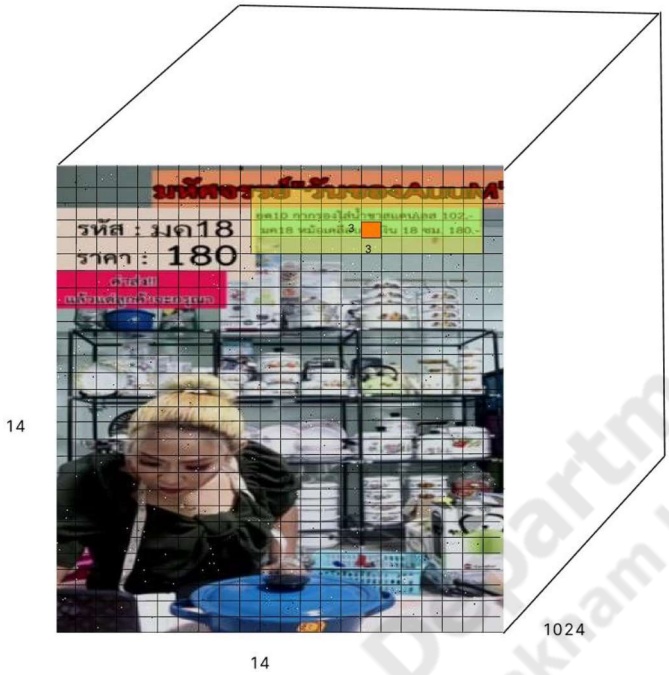
ภาพประกอบที่ 3.34 รูปภาพผลลัพท์ในขั้นตอนที่ 3

ขั้นตอนที่ 3 จะนำรูปภาพเข้ามาผ่านการทำ Convolution 4 ชั้น โดย
ชั้นที่ 1 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 128
ชั้นที่ 2 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 3 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 4 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 512
และ kernel ของชั้นที่ 1 - 4 จะเคลื่อนที่ทีละ 1 ช่องและทำการผ่าน Maxpool Layer เพื่อทำการลดขนาดของ feature map โดย Maxpool Layer จะมีขนาด2*2และจะทำการขยับไปทีละ 2 ช่อง



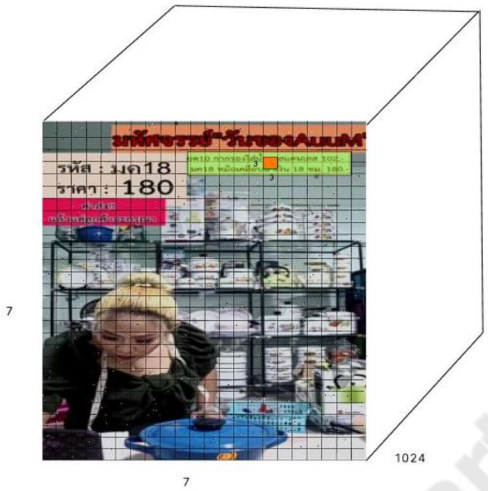
ภาพประกอบที่ 3.35 รูปภาพผลลัพท์ในขั้นตอนที่ 4

ขั้นตอนที่ 4 จะนำรูปภาพเข้ามาผ่านการทำ Convolution 10 ชั้น โดย
ชั้นที่ 1 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 2 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 3 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 4 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 5 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 6 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 7 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 256
ชั้นที่ 8 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 9 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 10 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
และ kernel ของชั้นที่ 1 – 10 จะเคลื่อนที่ทีละ1 ช่องและทำการผ่าน Maxpool Layer เพื่อทำการลดขนาดของ feature map โดย Maxpool Layer จะมีขนาด2*2และจะทำการขยับไปทีละ 2 ช่อง



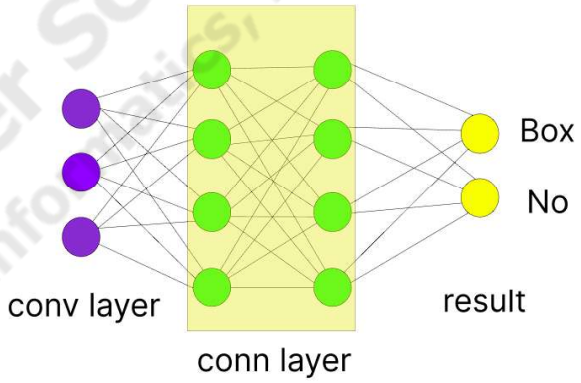
ภาพประกอบที่ 3.36 รูปภาพผลลัพท์ในขั้นตอนที่ 5

ขั้นตอนที่ 5 จะนำรูปภาพเข้ามาผ่านการทำ Convolution 10 ชั้น โดย
ชั้นที่ 1 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 2 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
ชั้นที่ 3 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 4 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
ชั้นที่ 5 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 6 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
ชั้นที่ 7 จะเป็นแบบ 1*1 และมี feature map ใหม่โดยใช้ filters จำนวน 512
ชั้นที่ 8 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
ชั้นที่ 9 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
และ kernel ของชั้นที่ 1 - 9 จะเคลื่อนที่ทีละ 1 ช่องและทำการผ่าน
ชั้นที่ 10 โดยจะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
และ kernel ของช่องที่ 10 จะเคลื่อนที่ทีละ 2 ช่อง



ภาพประกอบที่ 3.37 รูปภาพผลลัพท์ในขั้นตอนที่ 6

ขั้นตอนที่ 6 จะนำรูปภาพเข้ามาผ่านการทำ Convolution 2 ชั้น โดย
ชั้นที่ 1 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
ชั้นที่ 2 จะเป็นแบบ 3*3 และมี feature map ใหม่โดยใช้ filters จำนวน 1024
และ kernel ของช่องที่ 1 – 2 จะเคลื่อนที่ทีละ 1 ช่องและทำการผ่าน Maxpool Layer เพื่อทำการลดขนาดของ feature map โดย Maxpool Layer จะมีขนาด2*2และจะทำการขยับไปทีละ 2 ช่อง

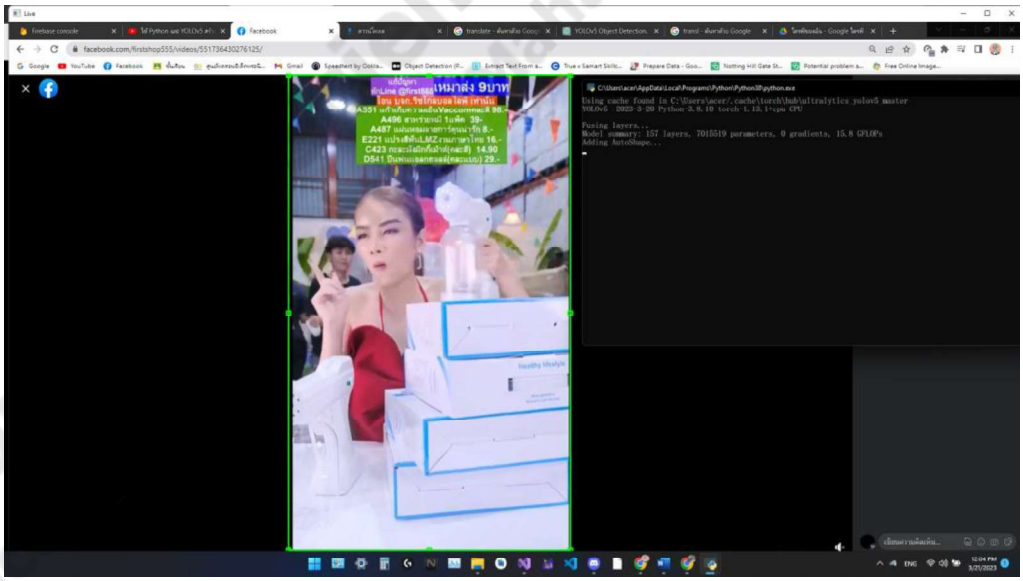


ภาพประกอบที่ 3.38 connection layer

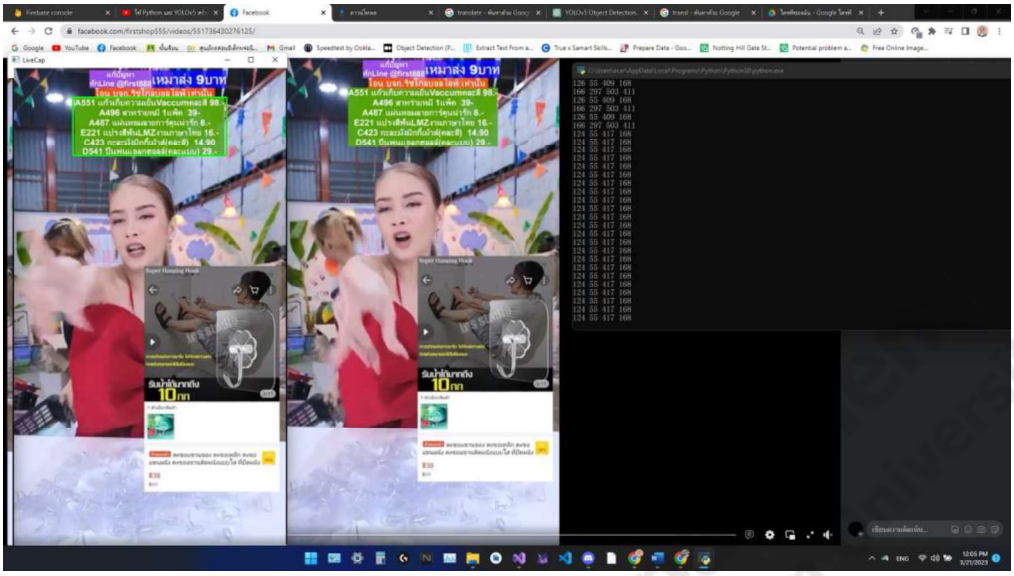
ขั้นตอนที่ 7 และ 8 จะผ่านการ connection layer โดยจะทำการเชื่อมต่อชั้น convolutional layers เข้าด้วยกันเพื่อปรับปรุงความแม่นยำของโมเดล โดยที่แต่ละชั้น convolutional layers จะมี feature maps ที่มีขนาดและลักษณะการเข้ารหัสที่แตกต่างกัน การเชื่อมต่อหลาย feature maps นี้จะช่วยให้โมเดลมีความสามารถในการจำแนกวัตถุที่ซับซ้อนและติดตามได้ดีขึ้น

ภาพประกอบที่ 3.39 ขั้นตอนการทำ NMS และแสดงผลลัพท์

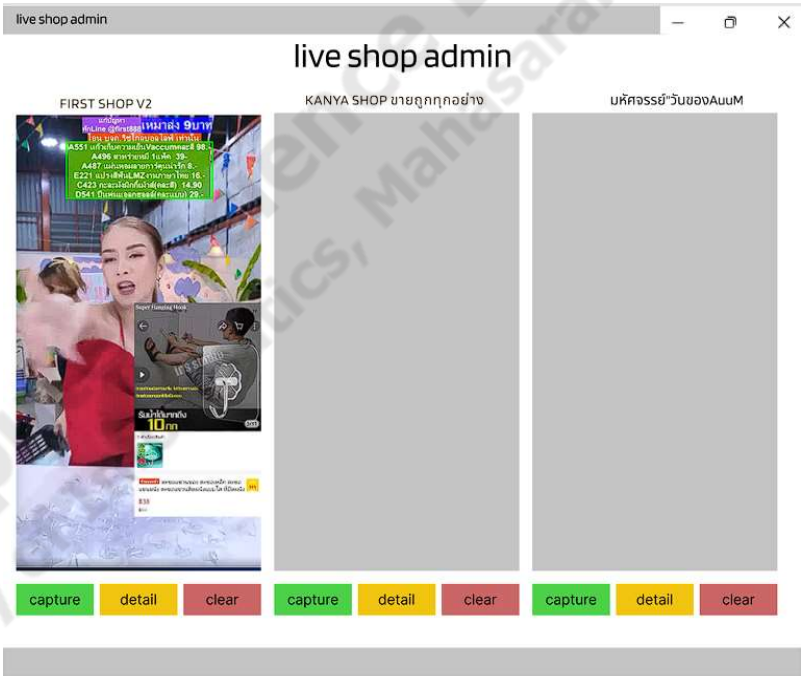
ขั้นตอนที่ 9 จะเป็น การทำ Non-Max Suppression เพื่อการลดจำนวนของ bounding boxes ที่ซ้อนทับกันในการตรวจจับวัตถุ ที่มีความน่าจะเป็นสูงที่สุด โดยทำการคัดลอก bounding box ที่มี IoU (Intersection over Union) มากกว่าหรือเท่ากับ threshold ที่กำหนด และเลือก bounding box ที่มีคะแนนความเชื่อมั่น (confidence score) สูงที่สุดเพื่อใช้เป็นผลลัพธ์สุดท้ายของโมเดล สุดท้ายจึงแสดงผลลัพธ์ออกมาในรูปแบบของ bounding box ที่วาดรอบ object และคำแนะนำว่า object นั้นมีชื่อว่าอะไร (class label) พร้อมกับคะแนนความเชื่อมั่นของการตรวจจับ (confidence score)



ภาพประกอบที่ 3.40 การทำงานในส่วนของการเลือกส่วนที่ต้องการ



ภาพประกอบที่ 3.41 ผลลัพท์ในการ detect และการเลือกพื้นที่ที่ต้องการ



ภาพประกอบที่ 3.42 ผลลัพท์ในการ detect ใน desktop appliction

3.4 การตัดบรรทัดด้วยเทคนิค projection profile

ในส่วนของงานในส่วนนี้จะเป็นการนำรูปภาพจากที่ทำการ detect มาได้มาทำการแปลงเป็นภาพ binary และทำการใช้เทคนิค horizontal projection profile มาทำการแบ่งบรรทัดเพื่อทำให้สินค้าแยกชิ้นกันอย่างชัดเจน

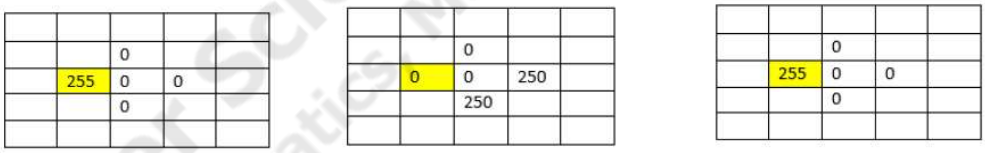
3.4.1 การเตรียมภาพ

ภาพประกอบที่ 3.43 ตัวอย่างภาพนำเข้า

การเตรียมพร้อมภาพ(Pre-Processing) เป็นขั้นตอนการนำเอาภาพเข้าเพื่อให้พร้อมต่อการนำไปประมวลผลต่อไป โดยประกอบด้วย ขั้นตอนการแปลงเป็นภาพระดับเทา และ การแปลงให้เป็นไบนารี ขาว-ดำ และตัดรูปภาพให้มีเฉพาะตรงส่วนที่มีข้อความอยู่ โดยขั้นตอนต่อไปนี้

3.4.2 แปลงภาพสี RGB ไปเป็นภาพระดับเทา

ขั้นตอนการแปลงภาพทำได้โดยอาศัยค่าของ RGB ที่อยู่ในแต่ละพิกเซลของภาพต้นฉบับมาแปลงเป็นภาพระดับเทา โดยช่องสี R จะมีค่าอยู่ที่ 0 – 255 ช่องสี G มีค่าอยู่ที่ 0-255 และช่องสี B จะมีค่าอยู่ที่ 0-255



ภาพประกอบที่ 3.44 ตารางค่าสี R G B ตามลำดับ

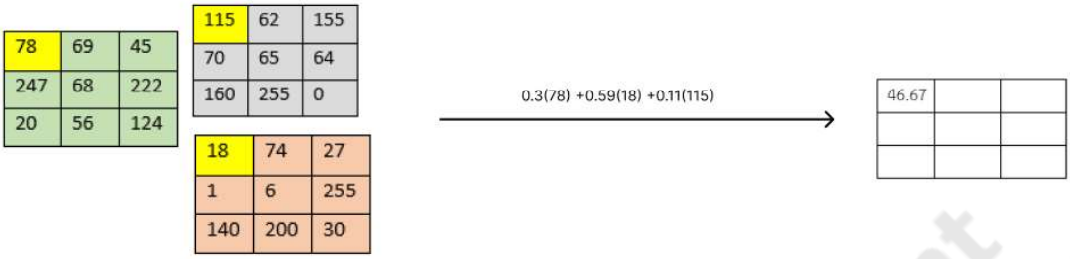
วิธีการถ้านำพิกเซลตำแหน่งที่(3,4)จากตารางมาแปลงเป็นภาพระดับเทาสามารถทำได้ดังนี้

จากสูตร Gray = 0.3R + -0.59 G + 0.11B

แทนค่า Gray = 0.3(255)+0.59(0)+0.11(255)

จะได้ค่า Gray = 76.5+ 0+ 28.5 = 104.55

โดยค่าคำตอบของ Gray จะถูกแทนลงในตำแหน่ง (3,4)ของภาพระดับเทา ตัวอย่างการแปลงภาพสี RGB เป็นภาพระดับเทา ของภาพที่มีขนาด 3x3 พิกเซล



ภาพประกอบที่ 3.45 การแปลงภาพสี RGB ไปเป็นระดับเทา

46.67	71.18	46.48
82.39	31.09	224.09
106.2	162.85	54.9

ภาพประกอบที่ 3.46 ตัวอย่างผลลัพท์หลังจากการแปลงเป็นภาพระดับเทา

ภาพประกอบที่ 3.47 ตัวอย่างหลังจากการแปลงเป็นภาพระดับเทา

3.4.3 การแปลงภาพระดับเทา ไปเป็นภาพขาวดำ (Binary image)

เป็นการแปลงค่าสีจากระดับเทาที่มีค่าอยู่ระหว่าง 0 – 255 ให้เป็นภาพแบบไบนารีที่มีค่า 0 และ 255 โดยจะหาค่าขีดแบ่ง (Threshold) ที่เหมาะสม โดยใช้ Otsu จะคำนวณหาความแปรปรวนระหว่างกลุ่มเพื่อหาค่าขีดแบ่งที่ดีที่สุดในการแยกวัตถุออกจากพื้นหลังออกจากพื้นหลัง โดยพิจารณาจากค่า t ที่เป็นไปได้ทั้งหมดโดยทั่วไป ในสมการ Otsu จะใช้ตัวแปรดังนี้

Ni จำนวนพิกเซลในระดับความเทา i

L จำนวนระดับความเทาทั้งหมดในภาพ

μ_T ค่าคาดหวังของระดับความเทาทั้งหมดในภาพ.

$\mu(i)$ ค่าคาดหวังของระดับความเทา i

$\sigma_B^2(i)$ ความแปรปรวนระหว่างกลุ่ม B และ F สำหรับระดับความเทา i

σ_B^2 ความแปรปรวนระหว่างกลุ่ม B และ F สำหรับทั้งภาพ

t ค่าขีดแบ่ง (Threshold) ที่เราค้นหาเพื่อแบ่งวัตถุออกจากพื้นหลังในภาพ

$arg\ max$ ตัวแปรนี้ใช้ในการหาค่า i ที่ทำให้ $\sigma_B^2(i)$ มากที่สุด ในที่นี้คือ i ที่ให้ค่าขีดแบ่ง t ที่ดีที่สุดในการแยกวัตถุออกจากพื้นหลัง

โดยมีวิธีการคำนวณดังนี้

ขั้นตอนที่ 1 กำหนดค่า $\mathrm{L} = 4$ และ $\mathrm{n_i} = (50,30,40,25)$

ขั้นตอนที่ 2 คำนวณค่า N โดยบวกทุกๆ ค่า Ni ของทุกระดับความเทา i และแสดงค่าไว้ที่ N

$$\mathrm{N} = \sum N_i = 50 + 30 + 40 + 25 = 145$$

ขั้นตอนที่ 3 คำนวณค่า L ซึ่งคือจำนวนระดับความเทาทั้งหมดในภาพ

$$L = 4$$

ขั้นตอนที่ 4 คำนวณค่าคาดหวังทั้งหมดของระดับความเทา μ_T โดยใช้สมการ

$$\mu_T = \sum \frac{N_i}{N}\ i$$

$$\mu_T = \frac{50}{145} * 0 + \frac{30}{145} * 1 + \frac{40}{145} * 25\ \frac{25}{145} * 3$$

$$\mu_T = \frac{0 + 30 + 80 + 75}{145} = \frac{185}{145}$$

ขั้นตอนที่ 5 คำนวณค่าคาดหวังของระดับความเทาแต่ละระดับ $\mu(\mathrm{i})$

$$\mu(0) = \frac{50}{145} * 0 = 0$$

$$\mu(1) = \frac{30}{145} * 1 = \frac{30}{145}$$

$$\mu(2) = \frac{40}{145} * 2 = \frac{80}{145}$$

$$\mu(3) = \frac{25}{145} * 3 = \frac{75}{145}$$

ขั้นตอนที่ 6 คำนวณความแปรปรวนระหว่างกลุ่ม B และ F สำหรับทุกระดับความเทา i โดยใช้สมการ

$$\sigma_B^2(i) = \frac{N_i}{N} * (\mu(i) - \mu_T)^2$$

$$\sigma_B^2(1) = \frac{50}{145} * (0 - \frac{185}{145})^2$$

$$\sigma_B^2(2) = \frac{30}{145} * (\frac{30}{145} - \frac{185}{145})^2$$

$$\sigma_B^2(2) = \frac{40}{145} * (\frac{80}{145} - \frac{185}{145})^2$$

$$\sigma_B^2(3) = \frac{25}{145} * (\frac{75}{145} - \frac{185}{145})^2$$

ขั้นตอนที่ 7 คำนวณความแปรปรวนระหว่างกลุ่ม B และ F สำหรับทั้งภาพ σ_B^2

$$\sigma_B^2 = \sum \sigma_B^2(i) = \sigma_B^2(0) + \sigma_B^2(1) + \sigma_B^2(2) + \sigma_B^2(3)$$

$$\sigma_B^2 = \frac{50}{145} * \left(0 - \frac{185}{145}\right)^2 + \frac{30}{145} * \left(\frac{30}{145} - \frac{185}{145}\right)^2 + \frac{40}{145} * \left(\frac{80}{145} - \frac{185}{145}\right)^2 + \frac{25}{145} * \left(\frac{75}{145} - \frac{185}{145}\right)^2$$

$$\sigma_B^2 = \frac{1}{145} * \left(0 - \frac{185}{145}\right)^2 + \frac{2}{145} * \left(\frac{30}{145} - \frac{185}{145}\right)^2 + \frac{8}{145} * \left(\frac{80}{145} - \frac{185}{145}\right)^2 + \frac{3}{145} * \left(\frac{75}{145} - \frac{185}{145}\right)^2$$

$$\sigma_B^2 = \frac{140625}{21025 * 145}$$

$$\sigma_B^2 = 0.2763$$

ดังนั้น $t \approx 0.2763$ คือค่าที่เหมาะสมที่จะนำไปใช้เป็น threshold ในการแปลงภาพระดับเทาให้เป็นภาพ binary โดยให้ค่า pixel ที่มีความสว่างต่ำกว่า t เป็นพื้นหลัง (ค่า 0) และค่า pixel ที่มีความสว่างสูงกว่าหรือเท่า t เป็นวัตถุ (ค่า = 255) ดังนี้

ภาพประกอบที่ 3.48 ตัวอย่างการแปลงภาพระดับ เทา เป็นภาพ Binary

3.4.4 การวาดเส้นในจุดที่เป็นที่ว่าง

ในขั้นตอนนี้จะทำการวาดเส้นในส่วนที่เป็นช่องว่างระหว่างข้อความเพื่อแบ่งสินค้าออกเป็นแต่ละชิ้นโดยทำการหาพิกัด (x,y) เริ่มต้นของภาพและไล่เช็คไปทีละพิกเซลจากซ้ายไปขาวและเมื่อเจอจุดที่เป็นช่องว่างจะนำมาทำการวาดเส้นแบ่งไว้โดยมีรายละเอียดและขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 เริ่มสแกนรูปภาพจากซ้ายไปขวาจนกว่าจะพบพิกเซลวัตถุ โดยหากค่าพิกเซลที่พบมีค่ามากกว่าหรือเท่า 1 แปลว่าพิกเซลที่พบมีข้อความอยู่

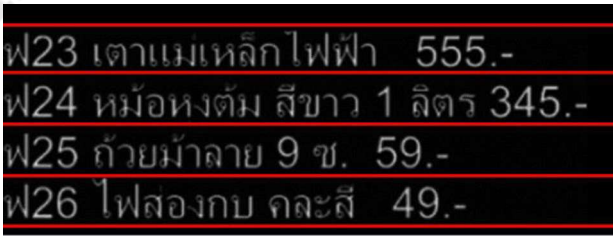
0	0	0	0	0
1	0	0	0	0
1	1	1	1	1
1	0	0	1	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0

ภาพประกอบที่ 3.49 ตัวอย่างสแกนรูปภาพจากซ้ายไปขวา

ขั้นตอนที่ 2 หลังจากได้จุดที่น่าจะเป็นขอบข้อความแล้วจะทำตามมาร์คเส้น ใน index ที่ว่างแล้วใส่เลข 5 ลงไปแทน แล้วเริ่มสแกนไปยังแถวถัดไป โดยในตำแหน่งที่เป็นเลข5จะกำหนดให้เป็นสีแดงและจะทำการวาดเส้นในตำแหน่งที่เป็นเลข 5 เพื่อทำการระบุว่าจุดตรงนี้คือช่องว่างระหว่างบรรทัด

0	0	0	0	0
1	0	0	0	0
1	1	1	1	1
1	0	0	1	0
5	5	5	5	5
1	1	1	1	1
5	5	5	5	5

ภาพประกอบที่ 3.50 ตัวอย่างการมาร์คเส้น



ภาพประกอบที่ 3.51 ผลลัพท์ในการวางเส้นแยกบรรทัด

3.5 ขั้นตอนการเรียกใช้ Tesseract OCR

ในกระบวนการ Optical Character Recognition (OCR) นั้นจำเป็นจะต้องมีข้อมูลลักษณะ (Feature) ของตัวอักขระนั้นๆก่อน เพื่อนำมาประมวลผลเทียบเคียงกับข้อมูลที่ได้จากภาพ ข้อมูล

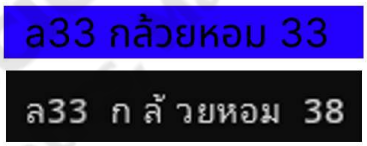
Feature ที่ได้มาจาการ (Train) ซึ่งค่อนข้างมีความซับซ้อนในการพัฒนา ดังนั้นในโครงงานนี้จะใช้เครื่องมือที่ช่วยลดความยุ่งยาก ที่มีชื่อว่า Tesseract OCR เพื่อนำมาเปรียบเทียบผลลัพธ์ที่ได้

วิธีการดำเนินการ การสกัดตัวหนังสือภาษาไทยและภาษาอังกฤษ โดยขั้นตอนแรกนำภาพเข้าไปจากนั้นเขียน Code เพื่อทดสอบการทำงานและตรวจสอบผลลัพธ์

ตารางที่ 3.7 การแปลงรูปเป็นตัวอักษรด้วย tesserract

```
import pytesseract as tess
from PIL import Image
tess.pytesseract.tesseract_cmd = r'D:\ocr\tesseract.exe'
image = Image.open('D:\ocr\kan2.png')
text = tess.image_to_string(image, lang='tha+eng')
print(text)
```

บรรทัดที่ 1 import flies tesseract
บรรทัดที่ 2 import image เพื่อใช้ในการเพิ่มรูปภาพเข้ามา
บรรทัดที่ 3 คือการอ่าน path files ของ tesseract
บรรทัดที่ 4 การอ่าน files รูปภาพแล้วกับไว้ในตัวแปล image เพื่อนำไปประมวลผล
บรรทัดที่ 5 การเอารูปภาพมาแปลงเป็น text จะแปลงเป็นภาษาไทยและภาษาอังกฤษ

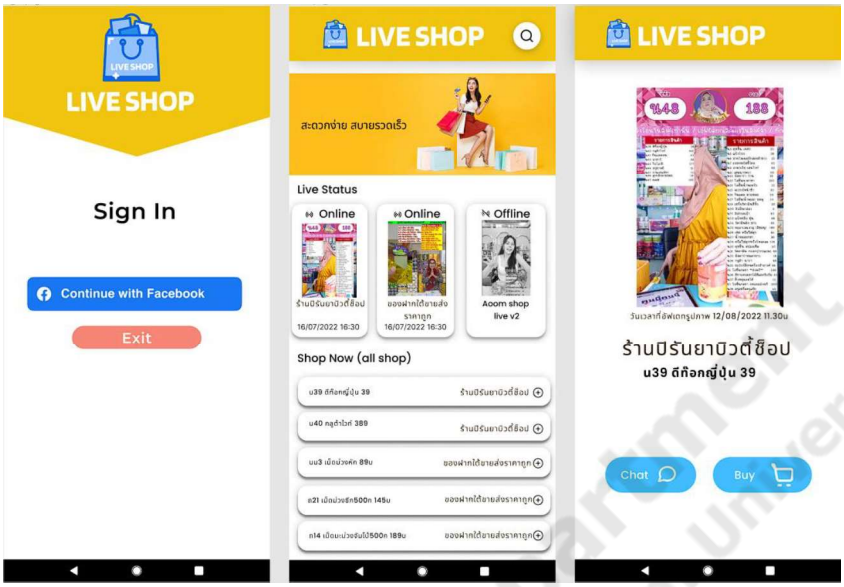


ภาพประกอบที่ 3.52 ตัวอย่างของผลลัพท์ของการแปลงที่ผิดพลาด

การใช้ Training Dataset ในการสกัดตัวอักษรไทยและอังกฤษจากภาพ จะพบว่าสามารถสกัดข้อความออกมาได้ถูกบางคำ เพื่อปกป้องกันข้อผิดผลาดโครงงานนี้จึงจะส่งรูปภาพให้ผู้ใช้ด้วย เพื่อลดข้อผิดพลาดลง

3.6 Mobile application

ในส่วนของ mobile application จะใช้ Flutter ในการสร้าง application ที่ทำการล็อกอินด้วย facebookได้และเชื่อมต่อกับ firebase และดึงข้อมูลจาก filebase มาแสดงผลได้และเพื่อลดความผิดพลาดในการแสดงผลข้อมูลจะมีรูปภาพของร้านค้าในขนาดที่ทำการไลฟ์สดขึ้นโชว์ในหน้ารายละเอียดสินค้าด้วย

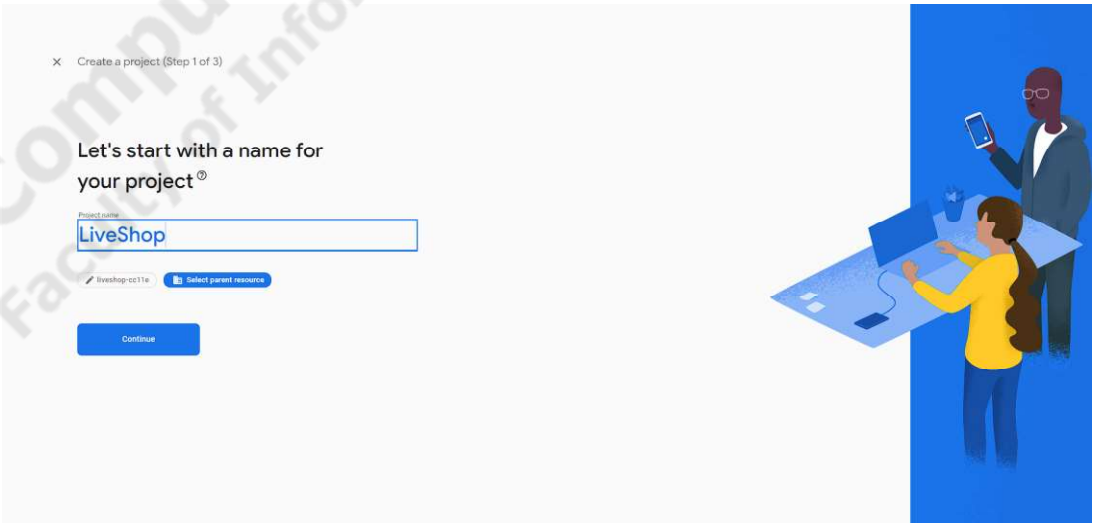


ภาพประกอบที่ 3.53 ตัวอย่างหน้าตา UI mobile application

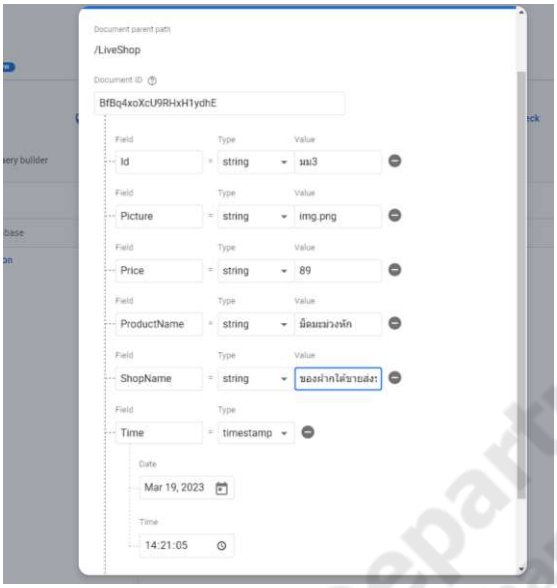
3.7 การจัดเก็บข้อมูล

ในส่วนของการจัดการข้อมูลจะใช้ Cloud Firestore จัดเก็บข้อมูล โดยใช้ฐานข้อมูล NoSQL ที่โฮสต์บนคลาวด์ Cloud Firestore โดยโครงสร้างจะมี 3 ส่วนคือ

1. Collection เป็น Folder ที่ไว้เก็บเอกสาร และมีชื่อบอกว่าเก็บเอกสารเกี่ยวกับอะไร
2. Document เป็นกระดาษไว้สำหรับเก็บข้อมูล และมีชื่อบอกว่าเก็บข้อมูลเกี่ยวกับอะไร
3. Data เป็นที่เก็บข้อมูล



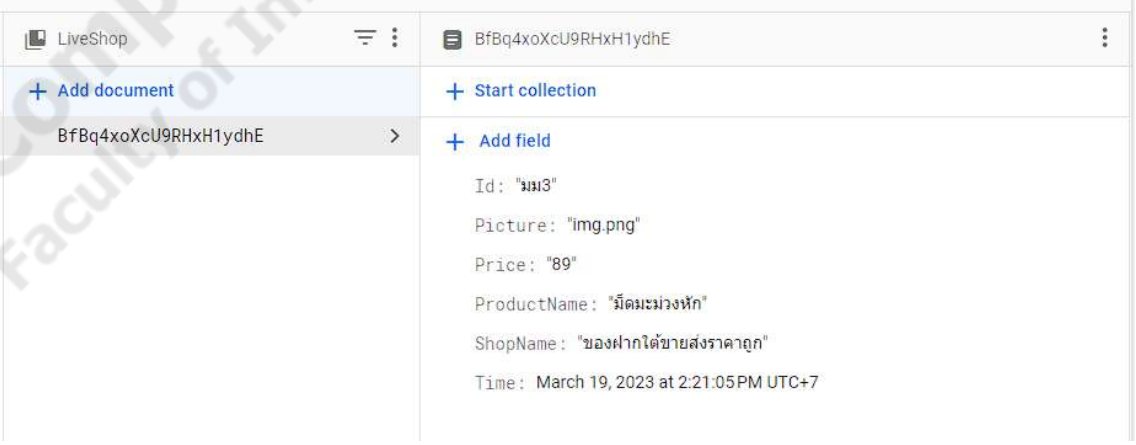
ภาพประกอบที่ 3.54 การสร้างโปรเจ็ค firebase



ภาพประกอบที่ 3.55 การสร้าง Document ใน firebase

โดยข้อมูลที่ทำการส่งไปเก็บในฐานข้อมูลจะประกอบไปด้วย

(1) ชื่อร้านค้า
(2) รหัสสินค้า
(3) ชื่อสินค้า
(4) ราคาสินค้า
(5) รูปภาพในการไลฟ์สด
(6) วันเวลาที่ capture รูปภาพ



ภาพประกอบที่ 3.56 ตัวอย่างข้อมูลและประเภทของข้อมูล

3.8 วัดประสิทธิภาพ

การวัดประสิทธิภาพการ Detect วัดด้วย Mean Average Precision (mAP)

ค่าเฉลี่ยความแม่นยำเฉลี่ย Mean Average Precisio (mAP) เป็นเมตริกที่ใช้ในการประเมินแบบจำลองการตรวจจับวัตถุ เช่น Fast R-CNN, YOLO, Mask R-CNN เป็นต้น ค่าเฉลี่ยของค่าความแม่นยำเฉลี่ย (AP) จะคำนวณจากค่าการเรียกคืนตั้งแต่ 0 ถึง 1 mAP คำนวณได้จากค่าดังนี้

Confusion Matrix ถือเป็นเครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนาย หรือ Prediction ที่ทำนายจาก Mode lที่เราสร้างขึ้น ในMachine learning โดยมีไอเดียจากการวัดว่า สิ่งที่เราคิด (Model ทำนาย) กับ สิ่งที่เกิดขึ้นจริง มีสัดส่วนเป็นอย่างไร

	Actually Positive (1)	Actually Negative (0)
Predicted Positive(1)	True Positives (TPs)	False Positives (TPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (FNs)

ภาพประกอบที่ 3.57 Confusion Matrix

True Positive (TP)=สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริงในกรณีทำนายว่าจริงและสิ่งที่เกิดขึ้นก็คือ จริง
True Negative (TN)=สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณีทำนายว่าไม่จริงและสิ่งที่เกิดขึ้นก็คือไม่จริง
False Positive (FP)=สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง
False Negative (FN)=สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้น คือ จริง

Intersection over Union(IoU) เป็นวิธีทางสถิติที่ใช้วัดความสอดคล้องของข้อมูลสองชุด โดยมีข้อมูลสองเซ็ตคือ P ซึ่งแทนเซตของพิกเซลในกรอบที่โมเดลเลือกมา และ G คือเซตของพิกเซลในกรอบที่เป็นเฉลย ใช้ P แทนคำว่า Predicted ส่วน G คือ Ground truth

$$IoU(P,G) = \frac{|P \cap G|}{|P \cup G|}$$

ภาพประกอบที่ 3.58 สมการหาค่า IoU

1	1	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0
0	0	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	1
0	0	1	1	0	0	0	0	1

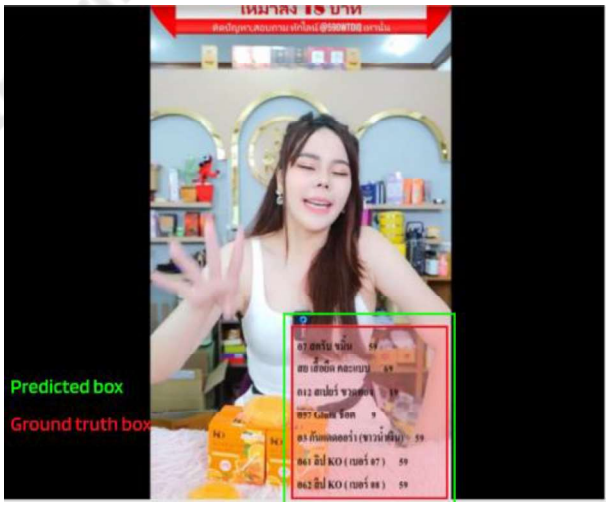
0	0	1	1	0	0	0	0	0
0	0	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	1
0	0	0	0	0	0	0	0	0

1	1	1	1	0	0	0	0	0
0	0	1	1	1	0	1	1	0
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	1	1	1	1	0	0	1
0	0	1	1	0	0	0	0	1

ภาพประกอบที่ 3.59 ตัวอย่างการหาค่า IoU

โดยภาพ P คือ ภาพ Matrix แรก และภาพ G คือ ภาพ Matrix ที่สอง และในภาพ P union G คือสีเทา ส่วน P intersection G คือสีเขียว สีเหลืองคือทายเกิน สีแดงคือทายขาด โดยสมาการดังนี้

$$\text{union} = 54 + 54 - 31 = 77$$
$$\text{iou} = \frac{31}{77} = 0.4$$



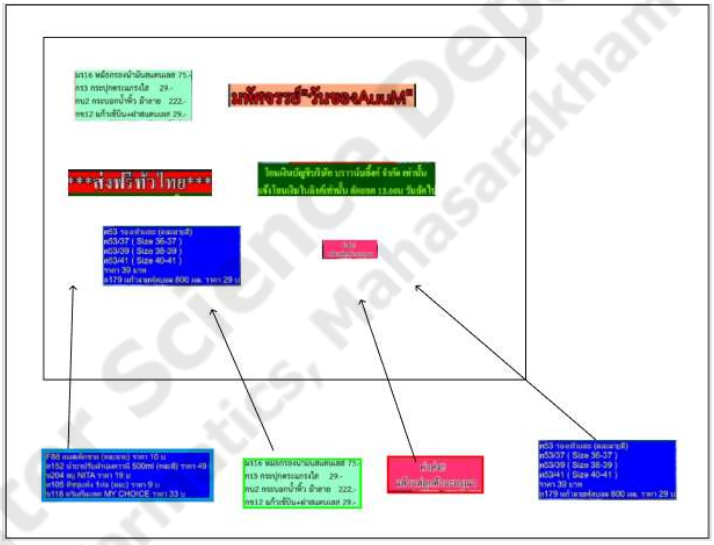
ภาพประกอบที่ 3.60 Predicted box และ Ground truth box

Precision คือ ค่าความแม่นยำจะวัดว่าสามารถค้นหาผลบวกที่แท้จริง (TP) จากการคาดการณ์เชิงบวกทั้งหมดได้ดีเพียงใด (TP+FP)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

ภาพประกอบที่ 3.61 การหาค่า Precision

Recall คือ จำนวนที่ทายถูกต่อจำนวนของ GroundTruth ทั้งหมด ตัวอย่าง Model เราทายมาเป็นลักษณะนี้ เราจะให้ Model เราทายเฉพาะแค่ กรอบข้อความ หมายถึง เอาแต่ข้อความ มาให้เราเอาอย่างอื่นมาให้ถือว่าผิด

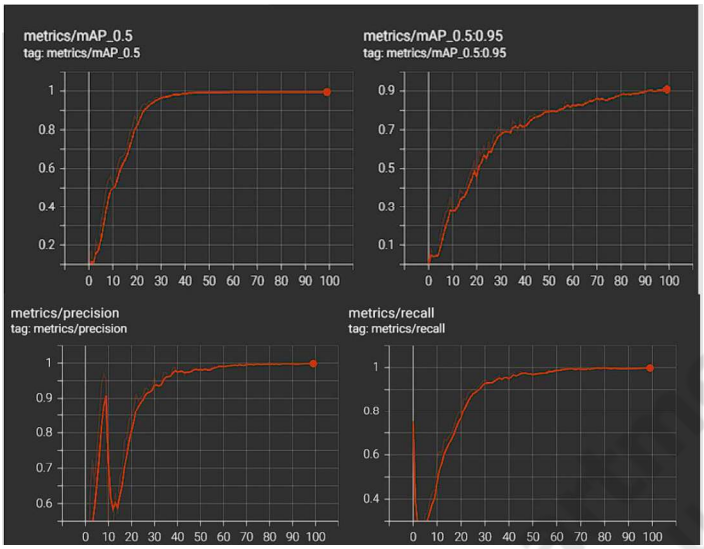


ภาพประกอบที่ 3.62 ตัวอย่างผลลัพท์ในการทำนาย

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

ภาพประกอบที่ 3.63 การหาค่า recall

จาก model ของเราทำการเลือกภาพมาให้เราดังภาพที่อยู่นอกกรอบ จะเห็นว่า ได้รูปกรอบข้อความ ได้มา 3 รูป ส่วน สี่เหลี่ยม รูป ลองคำนวน Precision และ Recall ดูจากตัวอย่างจะได้ 3 TP และ 1 FP ดังนั้น precision คือ 3/4 = 0.75 และ recall คือ 3/6 = 0.5



ภาพประกอบที่ 3.64 กราฟการเปลี่ยนแปลงค่า mAP precision และ recall ของ model

การวัดประสิทธิภาพการ OCR วัดด้วย CER(Character Error Rate)

ค่า CER จะวัดเป็นเปอร์เซ็นต์ สังเกตว่ายิ่งค่า CER เยอะ ประสิทธิภาพของโมเดลก็จะยิ่งแย่ค่าความเหมาะสมของค่า CER ที่เหมาะสมสำหรับงานที่กำหนดไว้คือค่า CER ต้อง น้อยกว่า 10% โดยค่า CER สามารถหาค่าได้ดังนี้

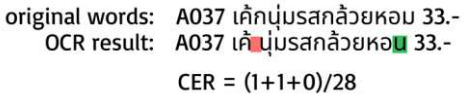
$$CER = \frac{S + D + I}{N}$$

ภาพประกอบที่ 3.65 สมการ CER

I (inserted words) คือ จำนวนตัวอักษรที่ถูกแทรกขึ้นมาจากข้อความเดิม
D (deleted words) คือ จำนวนตัวอักษรที่หายไปจากข้อความเดิม
S (substituted words) คือ จำนวนตัวอักษรที่ถูกแทนที่ไปจากคำเดิม
N คือ จำนวนตัวอักษรทั้งหมด

A037 เค้กนุ่มรสกล้วยหอม 33.-	original words
A037 เค้นุ่มรสกล้วยหอม 33.-	deleted words
A037 เค้กคนุ่มรสกล้วยหอม 33.-	inserted words
A037 เค้กนุ่มรสกล้วยหอน 33.-	substituted words

ภาพประกอบที่ 3.66 ตัวอย่างตัวแปรในสมการ CER



ภาพประกอบที่ 3.67 ตัวอย่างการหาค่า CER

จาก ภาพประกอบที่ 3.67 ตัวอย่างการหาค่า CER เมื่อนำค่าจากผลลัพท์จากการ OCR มาแทนค่าในตัวแปรทั้ง 4 จะมีผลลัพท์ดังนี้

S (substituted words) จะมีค่าเท่ากับ 1

D (deleted words) จะมีค่าเท่ากับ 1

I (inserted words) จะมีค่าเท่ากับ 0

N จะมีค่าเท่ากับ 28

เมื่อแทนค่าทั้งหมดลงในสมการCER=(S,D,I)/N จะได้ (1+1+0)/28= 0.07

เมื่อนำมาคิดเป็นเปอร์เซ็นต์จะได้ค่า CER