บทที่ 3

ขั้นตอนการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานของโครงงานปริญญานิพนธ์ ซึ่งจะทำให้ทราบถึงการวิเคราะห์ระบบ กระบวนการประมวลผลของระบบ และขั้นตอนการทำงานของระบบเป็นอย่างไร โดยขั้นตอนในการดำเนินงานวิจัยมีรายละเอียดดังนี้

1. กรอบการดำเนินงาน
2. การเตรียมชุดข้อมูล (Dataset)
3. ขั้นตอนการทำงานของ YOLO (You Only Look Once)
4. การระบุตำแหน่งวัตถุ
5. ขั้นตอนในการพัฒนา Mobile application
    a. นำ model มาใช้ในแอปพลิเคชัน
    b. Text to speech
    c. Speech to text
    d. การคำนวณหาตำแหน่งวัตถุ
    e. หน้าหลัก Application
    f. การใช้งาน API ภายนอก
6. การพัฒนาโมเดล (Model)

3.1 กรอบการดำเนินงาน

จากภาพ เป็นภาพรวมในการดำเนินงานของโครงงาน โดยแบ่งเป็นสองส่วนโดย ส่วนแรกคือ Deep learning จะเป็นส่วนที่ใช้ในการสร้าง Model ที่ใช้ในการจำแนกสิ่งของ และส่วนที่สองคือ Mobile application เป็นส่วนในการสร้าง Application โดยแต่ละส่วนมีรายละเอียดดังต่อไปนี้

3.1.1 Deep learning

ในส่วนของ Deep learning จะเป็นขั้นตอนในการสร้าง House model ที่ใช้ในการ classification ที่จะนำไปใช้ใน mobile application ก่อนที่จะสร้าง House model จะต้องมี dataset ของสิ่งของก่อน โดย dataset ที่ได้จะรวบรวมจากหลากหลายเว็บไซต์ หลังจากนั้นจะนำ dataset ที่ได้ไปใช้ในการ train model โดยใช้ pre-train model ของ YOLOv5 พอ train model ได้ตามที่ต้องการ จากนั้นจะบันทึก model เป็นไฟล์ tflite เพื่อนำไปใช้บน moblie application

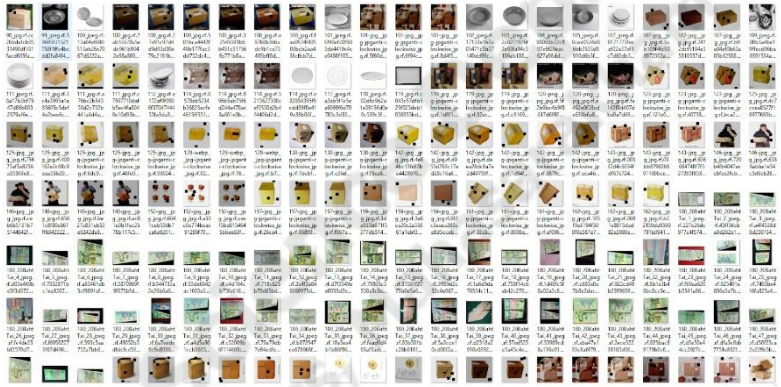
3.1.2 Mobile Application

ในส่วนของ Mobile application โดยขั้นตอนในการใช้งาน House model คือผู้ใช้ต้องนำมือถือส่องไปรอบๆ บริเวณห้อง จากนั้น mobile application จะรับภาพจากกล้องมือถือ model จะทำการ classification และส่งผลลัพธ์กลับมาให้ผู้ใช้

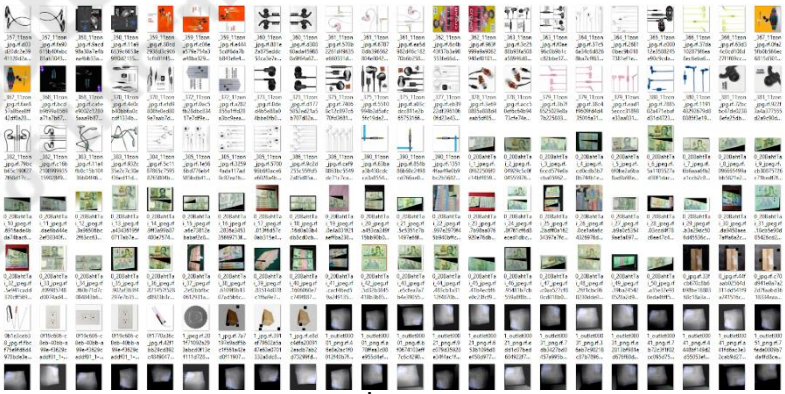
3.2 การเตรียมชุดข้อมูล (Dataset)

3.2.1 ชุดข้อมูล

โครงงานนี้ได้ใช้ชุดข้อมูลสิ่งของ 24 ชนิด ในการสร้างโมเดล ซึ่งจะมีรูปภาพทั้งหมด 8,959 รูป ประกอบด้วยชุดข้อมูลสำหรับการ Train 6,268 รูป Valid 1,788 รูป และ Test 903 รูป ดังตารางที่ 3.1 รูปภาพ Input ทั้งหมดจะถูกปรับขนาด 640*640 รูปภาพทั้งหมดได้ทำการรวบรวมมาจาก Roboflow



ภาพประกอบที่ 3.1 Dataset



ภาพประกอบที่ 3.2 Dataset test

ตารางที่ 3.1 รายละเอียดข้อมูลภาพ

ลำดับที่	สิ่งของ	สัดส่วนข้อมูล		
		Train(70%)	Validation (20%)	Test (10%)
1	1000Baht	140	40	20
2	100Baht	140	40	20
3	20Baht	146	25	12
4	500Baht	140	40	20
5	50Baht	140	40	20
6	Backpack	761	212	117
7	Book	571	163	83
8	Box	152	41	23
9	Chair	245	74	39
10	Charger cable	68	19	10
11	Earphone	382	109	54
12	Fan	386	110	54
13	Fork	481	139	67
14	Glass	164	56	29
15	Key	151	43	22

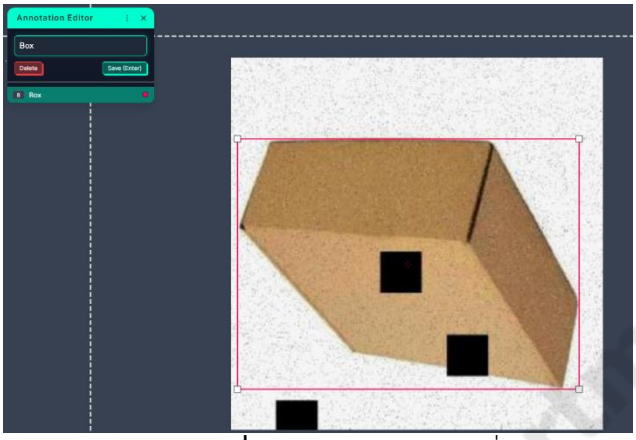
ตารางที่ 3.1 รายละเอียดข้อมูลภาพ(ต่อ)

ลำดับที่	สิ่งของ	สัดส่วนข้อมูล		
		Train(70%)	Validation (20%)	Test (10%)
16	Knife	472	144	67
17	Mouth Mask	196	56	28
18	Outlet	381	112	58
19	Plate	231	65	27
20	Spoon	409	119	57
21	TV Remote	247	71	35
22	Table	293	80	39
23	Vase	180	60	28
24	Walking stick	44	13	6
รวม		6268	1788	903

3.2.2 Roboflow

Roboflow เป็น Computer Vision Developer Framework สำหรับใช้จัดเก็บเตรียมชุดข้อมูล และสร้างแบบจำลองต่างๆ ที่สามารถใช้งานผ่าน web browser ได้

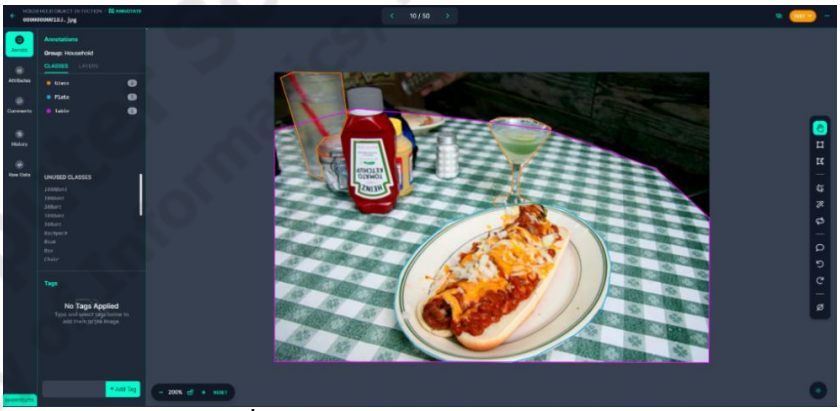
3.2.2.1 การเตรียมข้อมูลใน Roboflow



ภาพประกอบที่ 3.3 การตีกรอบภาพสิ่งของ

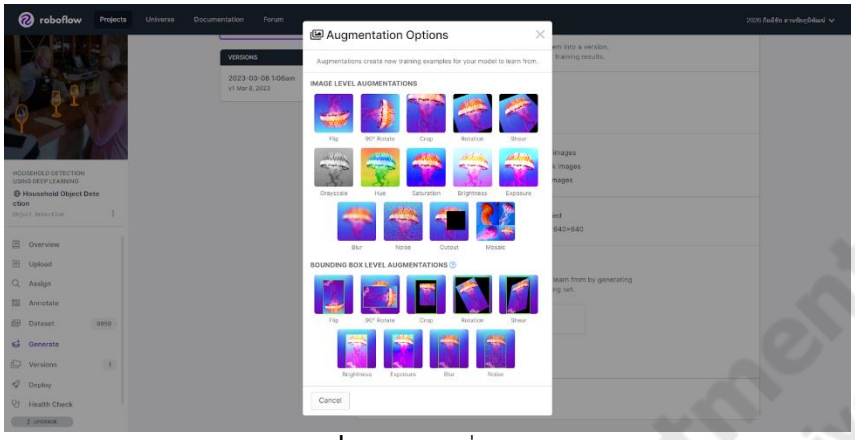
จากภาพตัวอย่าง 3.3 เป็นการตีกรอบตำแหน่งที่เป็นกล่อง และกำหนด class ของวัตถุที่อยู่ในกรอบ

การตีกรอบแบบ Polygon annotation ช่วยให้ระบบตรวจจับวัตถุได้อย่างแม่นยำมากขึ้น เนื่องจากสามารถระบุรูปร่างและขอบเขตของวัตถุได้อย่างละเอียด โดยการนำตำแหน่ง x และ y แต่ละจุดมาเชื่อมกันตามลำดับในการกำหนดขอบเขตของวัตถุในภาพ เช่น x1y1, x2y2, x3y3 เป็นต้น

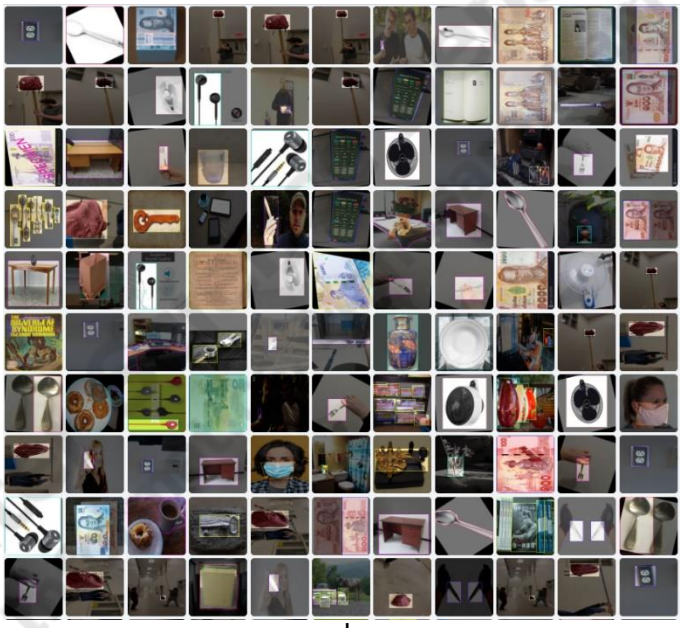


ภาพประกอบที่ 3.4 การตีกรอบแบบ Polygon annotation

3.2.2.2 การเพิ่มปริมาณ dataset

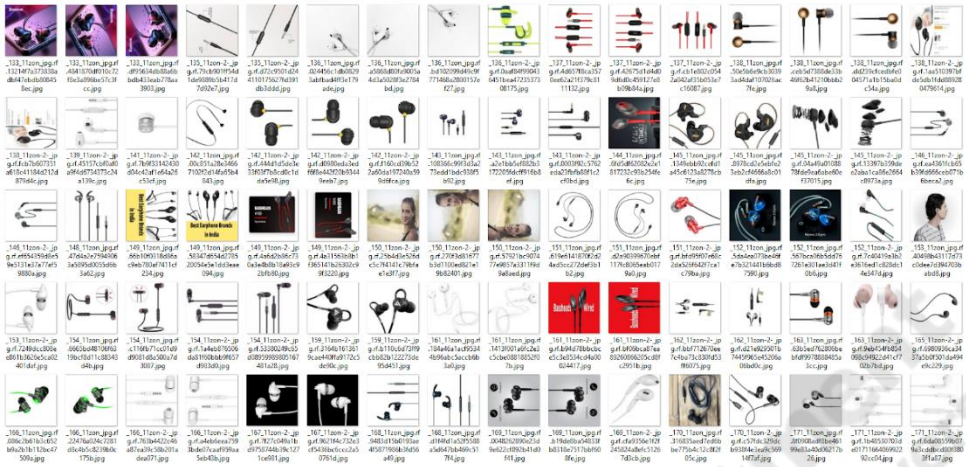


ภาพประกอบที่ 3.5 การเพิ่มปริมาณ dataset



ภาพประกอบที่ 3.6 dataset

การนำ dataset ที่เตรียมใน Roboflow มาใช้เมื่อเรา generate รูปภาพเสร็จทำการ Export ไฟล์เราจะได้สองไฟล์คือ images และ labels



ภาพประกอบที่ 3.7 ตัวอย่างไฟล์ images

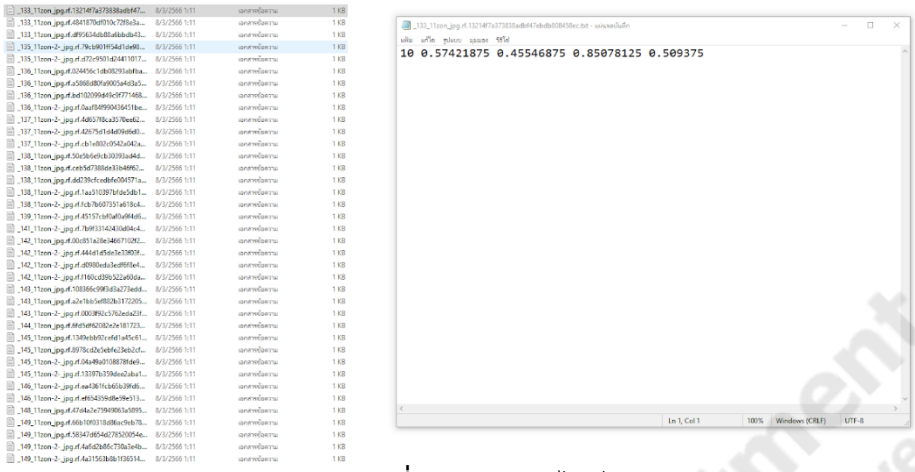
ค่าในไฟล์ label ใช้ normalized xywh format แทนที่จะเป็นค่าพิกเซลเพราะ ขนาดของภาพมีขนาดที่แตกต่างกัน การใช้ค่าพิกเซลในรูปแบบเดียวกันสำหรับภาพที่มีขนาดต่างกันอาจทำให้ข้อมูลของคุณมีความแปรปรวนและความไม่สมดุล และความเปรียบเทียบระหว่างภาพ การทำให้ค่าพิกเซลของกล่องขอบเขตเป็นมาตรฐานของ 0-1 ในรูปแบบ normalized xywh format ทำให้ง่ายต่อการเปรียบเทียบวัตถุในภาพที่มีขนาดและสัดส่วนที่แตกต่างกัน ซึ่งเป็นสิ่งสำคัญในการตรวจจับวัตถุหรือเรียนรู้

รูปแบบ normalized xywh มีรายละเอียดดังนี้:

- x_center คือตำแหน่ง x ของจุดกึ่งกลาง (center) ของกล่อง โดยที่ 0 คือขอบซ้ายและ 1 คือขอบขวาของภาพ.
- y_center คือตำแหน่ง y ของจุดกึ่งกลาง (center) ของกล่อง โดยที่ 0 คือขอบบนและ 1 คือขอบล่างของภาพ.
- width คือความกว้างของกล่องที่เรากำหนดในหน่วยเปอร์เซ็นต์ของความกว้างของภาพ (จาก 0 ถึง 1).
- height คือความสูงของกล่องที่เรากำหนดในหน่วยเปอร์เซ็นต์ของความสูงของภาพ (จาก 0 ถึง 1).

ยกตัวอย่าง

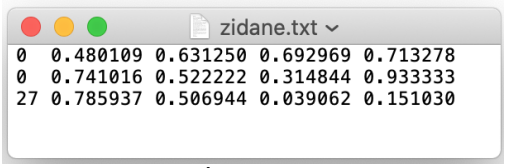
หากความกว้างของภาพเป็น 1000 พิกเซลและ x_center ของกล่องเป็น 500 พิกเซล, ค่า x_center ที่ถูกแปลงให้อยู่ใน normalized xywh format คือ 0.5 (500/1000).



ภาพประกอบที่ 3.8 ตัวอย่างไฟล์ labels

ค่า label ใน Bounding box

ภาพประกอบที่ 3.9 ตัวอย่างรูปภาพ



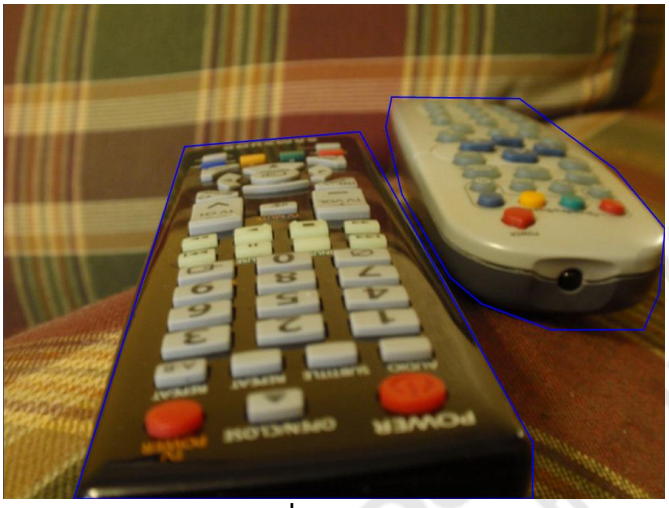
ภาพประกอบที่ 3.10 ค่าในไฟล์ label

ยกตัวอย่างในบรรทัดแรก 0 0.480109 0.631250 0.692969 0.713278

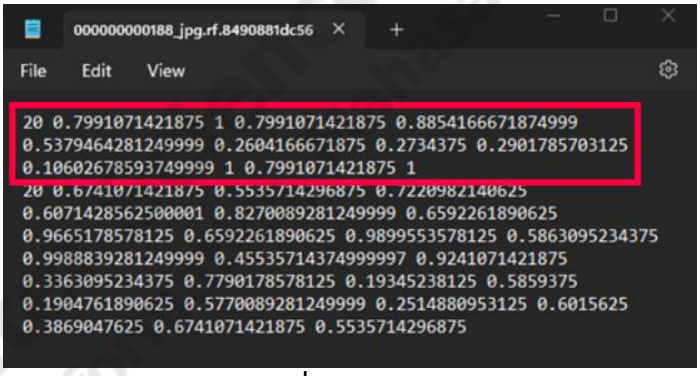
- 0 คือประเภทของ Object แต่ในที่นี่ 0 คือจะเป็น Class คน
- 0.480109 คือตำแหน่งตรงกลางของรูปว่าอยู่จุดไหนในรูปแบบของ X
- 0.631250 คือตำแหน่งตรงกลางของรูปว่าอยู่จุดไหนในรูปแบบของ Y

- 0.692969 คือความกว้างและความสูงของรูปในรูปแบบของ Width
- 0.713278 คือความกว้างและความสูงของรูปในรูปแบบของ Height

ค่า label ใน Polygon Annotation



ภาพประกอบที่ 3.11 ตัวอย่างรูปภาพ



ภาพประกอบที่ 3.12 ค่าในไฟล์ label

ยกตัวอย่างภายในกรอบสีแดงดังภาพประกอบที่ 3.12

- 20 คือประเภทของ Object แต่ในที่นี่ 20 คือจะเป็น Class TV Remote
- 0.7991071421875 และ 1 คือตำแหน่ง X และ Y จุดแรก
- 0.7991071421875 และ 0.8854166671874999 คือตำแหน่ง X และ Y จุดที่สอง
- 0.5379464281249999 และ 0.2604166671875 8854166671874999 คือตำแหน่ง X และ Y จุดที่สาม
- 0.2734375 และ 0.2901785703125 คือตำแหน่ง X และ Y จุดที่สี่
- 0.10602678593749999 และ 1 คือตำแหน่ง X และ Y จุดที่ห้า
- 0.7991071421875 และ 1 คือตำแหน่ง X และ Y จุดที่หก

3.3 ขั้นตอนการทำงานของ YOLO (You Only Look Once)

3.3.1 การเตรียมอินพุต

อัลกอริธึมจะใช้รูปภาพเป็นอินพุต รูปภาพจะถูกปรับขนาดให้เป็นขนาดคงที่โดยยังคงอัตราส่วนไว้ รูปภาพที่ปรับขนาดแล้วเหล่านี้จะถูกป้อนเข้าสู่โมเดล YOLOv5 โดยรูปภาพของเรามีอินพุตมีขนาด 640 x 640 x 3

3.3.2 การทำงานของ Convolution Layer

คือการสกัดคุณลักษณะของรูปภาพอินพุต โดยรูปภาพจะถูกแบ่งเป็นแมทริกซ์ อีกเมทริกซ์คือเซตของพารามิเตอร์ที่สามารถเรียนรู้ได้ที่เรียกว่า Filter โดยการทำ convolution จะใช้ filter มาสแกนภาพเพื่อทำการแยกองค์ประกอบ โดยวิธีดำเนินการ เริ่มจากนำ filter สแกนไปบนภาพ และทำการคำนวณโดยการนำเอาตำแหน่งที่ตรงกันของภาพต้นฉบับ และ filter มาคูณกัน แล้วจึงนำผลรวมของทุกตำแหน่งมาบวกกัน (ตำแหน่งที่ 1 + ตำแหน่งที่ 2 + ... + ตำแหน่งที่ n)

convolution layer filters ที่ใช้ในการดูและแยกแยะคุณลักษณะของข้อมูลภาพในภาพต้นฉบับ มีขนาดและลักษณะที่แตกต่างกันเพื่อใช้ในการค้นหาคุณลักษณะที่แตกต่างกันในข้อมูลภาพ ดังนั้นมีหลายแบบของ filters ตามลักษณะและการใช้งานต่าง ๆ

1. แบบเบื้องต้น (Basic Filters): ซึ่งเป็น filter สีเทาที่ใช้ในการตรวจจับข้อมูลพื้นฐานของภาพ เช่น filter สีเทา 3x3, 5x5, 7x7 ซึ่งช่วยในการค้นหาเส้น, เส้นขอบ, และรายละเอียดพื้นหลังของภาพ
2. แบบสองมิติ (2D Filters): ใช้ในการค้นหาคุณลักษณะสองมิติในภาพ เช่น ขอบแนวนอน, ขอบแนวตั้ง, เส้นทแยงมุม, และลวดลายที่ใช้ในงานต่าง ๆ
3. แบบสีเทา (Grayscale Filters): ใช้ในการค้นหาคุณลักษณะเฉพาะของข้อมูลสีเทา เช่น ฟิลเตอร์ Sobel, Laplacian, และ Gaussian
4. แบบสี (Color Filters): ใช้ในการประมวลผลข้อมูลสี ซึ่งประกอบด้วยช่องสีแบบ RGB, HSV, หรือ LAB และสามารถค้นหาคุณลักษณะสี, เสียง, หรือลวดลายสีได้
5. Depthwise Separable Convolution: ชนิดนี้แยกการคำนวณแบบหลายขั้นตอน โดยคัดลอกการทำงานของ convolution layer ไปยังแต่ละช่องสีแยกกันก่อนรวมผลลัพธ์ทั้งหมด เป็นที่นิยมในการลดความซับซ้อนและทรัพยากรในการฝึก CNNs
6. Dilated Convolution: ใช้การระยะห่างระหว่างพิกเซลในการคำนวณ convolution ซึ่งช่วยในการรับรู้คุณลักษณะที่มีขนาดใหญ่ขึ้นในภาพ
7. Separable Convolution: แบบนี้แยกการคำนวณระหว่างส่วนของการค้นหาคุณลักษณะและการคำนวณค่าตัวแปรส่วนหนึ่ง เพื่อลดความซับซ้อนและการใช้ทรัพยากร.
8. Grouped Convolution: ใช้ในการแบ่งกลุ่มช่องสีหรือชั้นในการคำนวณ convolution ซึ่งช่วยในการปรับปรุงประสิทธิภาพในกรณีที่มีข้อมูลหลายชุดสองมิติ.

9. Transposed Convolution (ConvTranspose): ใช้ในการขยายขนาดภาพเพื่อทำการทำสัญญาณหรือสร้างรูปภาพขนาดใหญ่ขึ้นจากข้อมูลนำเข้าที่มีขนาดเล็ก.

10. Fractionally-Strided Convolution: แบบนี้ใช้ในการขยายขนาดข้อมูลเชิงภาพแบบสัญญาณหรือตัวเลข.

11. Adaptive Convolution: ใช้การคำนวณที่ปรับไปตามขนาดของข้อมูลนำเข้า เพื่อให้ได้ความละเอียดและความซับซ้อนที่ต่างกันตามต้องการ.

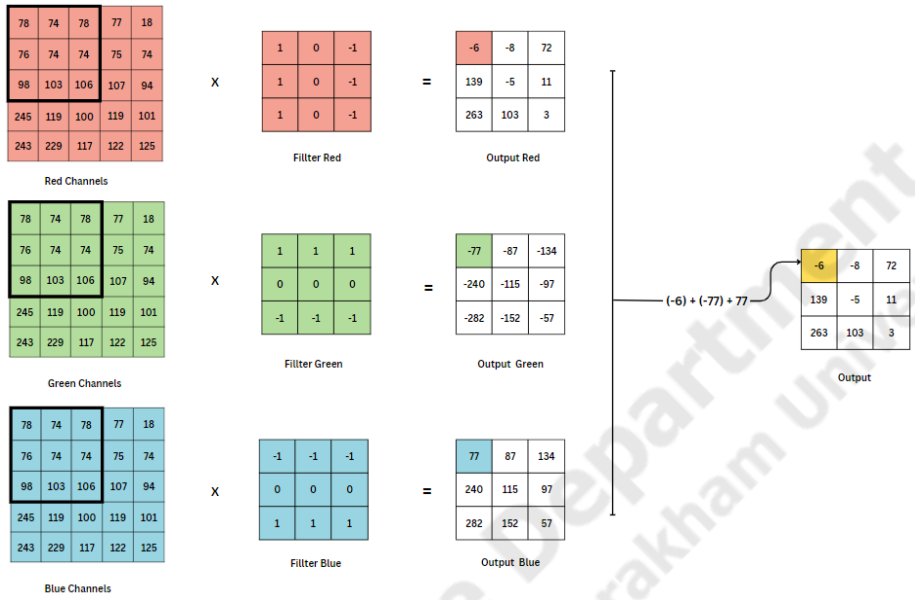
นี่เป็นแค่บางตัวอย่างของแบบของ convolution layer filters ที่มีใน CNNs แต่มีอีกหลายรูปแบบและคุณลักษณะอื่น ๆ

ดังนั้นจะใช้ Depthwise Separable Convolution filter ที่มีขนาด 3x3x3 โดยมีส่วน Depthwise Convolution ที่คำนวณ convolution แยกตามช่องสีและส่วน Pointwise Convolution ที่ใช้ filter ขนาด 1x1x3 ในการรวมข้อมูลสีจากทุกช่องสีเข้าด้วยกัน เพื่อลดความซับซ้อนของการคำนวณและปริมาณพารามิเตอร์ที่จำเป็นต้องเทรน นี่เป็นวิธีที่มีประสิทธิภาพในการประมวลผลข้อมูลภาพที่มี 3 channels RGB

การหาค่า filter

- สุ่มค่า filter เริ่มต้น
- Feedforward จะใช้ filter ที่สุ่มมาในขั้นตอนแรกนี้เพื่อคำนวณผลลัพธ์ (output) จาก convolution ระหว่าง filter และข้อมูล input
- คำนวณค่าคลาดเคลื่อน (Error) ค่าคลาดเคลื่อนหรือค่าความผิดพลาด (loss) จะถูกคำนวณโดยเปรียบเทียบผลลัพธ์ที่คำนวณได้จากโมเดลกับผลลัพธ์ที่เป็นค่าเป้าหมาย (ground truth) ที่ต้องการให้โมเดลคำนวณได้ ค่าความผิดพลาดนี้จะบ่งบอกถึงความแตกต่างระหว่างผลลัพธ์ที่โมเดลคำนวณกับผลลัพธ์ที่เราต้องการให้คำนวณ
- Backpropagation ค่าความผิดพลาดจะถูกนำมาใช้ในกระบวนการ backpropagation เพื่อคำนวณ gradient ของค่าความผิดพลาดตามต่อค่าของ filter แต่ละตัว. Gradient คืออนุพันธ์ของค่าความผิดพลาดตามต่อค่าของ filter แต่ละค่า ซึ่งบ่งบอกถึงว่าถ้าเราเพิ่มหรือลดค่าใน filter แต่ละตัวอย่างไร จะส่งผลให้ค่าความผิดพลาดลดลงหรือเพิ่มขึ้น
- การปรับค่า Filter หลังจากที่คำนวณ gradient แล้ว, จะใช้การอัปเดตค่าของ filter โดยใช้อัลกอริทึมการอัปเดตเชิงอนุพันธ์ (gradient descent) เพื่อปรับค่าของ filter ให้มีค่าที่ดีที่สุดเพื่อลดค่าความผิดพลาด
- วนลูป กระบวนการข้างต้นจะทำซ้ำกันหลายครั้ง (epochs) ซึ่งในแต่ละครั้ง filter จะถูกปรับค่าใหม่ๆ ตาม gradient ที่คำนวณได้. การทำซ้ำเหล่านี้จะช่วยให้โมเดลเรียนรู้และปรับปรุงค่าของ filter ให้สามารถสกัดลักษณะเฉพาะจากข้อมูล input ได้ดียิ่งขึ้น

การทำงานของ Convolution ยกตัวอย่างรูปภาพอินพุต ขนาด 5 x 5 และ filter ขนาด 3 x 3 เพื่อง่ายต่อการเข้าใจ ดังภาพประกอบที่ 3.13



ภาพประกอบที่ 3.13 การทำ Convolution ขนาด 5 x 5 และ filter ขนาด 3 x 3

วิธีการคำนวณ Convolution

Red Channels = (78x1)+(74x0)+(78x(-1))+(76x1)+(74x0)+(74x(-1))+(98x1)+(103x0)+(106x(-1)) = -6

Green Channels = (78x1)+(74x1)+(78x1)+(76x0)+(74x0)+(74x0)+(98x(-1))+(103x(-1))+ (106x(-1)) = -77

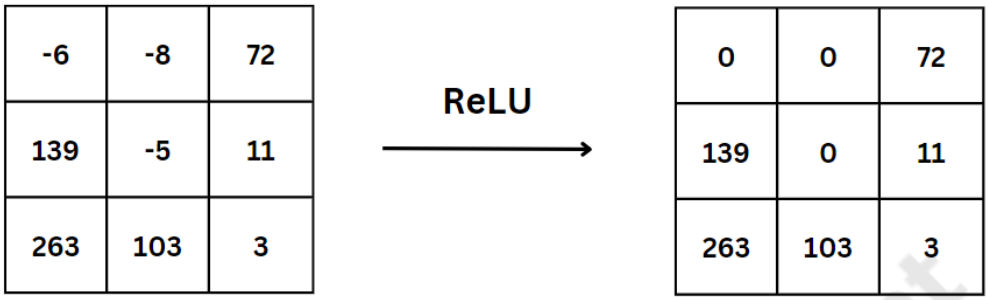
Blue Channels = (78x(-1))+(74x(-1))+(78x(-1))+(76x0)+(74x0)+(74x0)+(98 x1)+(103 x1)+(106x1) = 77

3.3.3 ReLU (Rectified Linear Unit)

ผลลัพธ์ที่ได้จากการทำ Convolution ในแต่ละตำแหน่งจะแปลงค่าด้วยฟังก์ชัน ReLU ที่เป็นการแปลงแบบไม่เป็นเชิงเส้น เพื่อความง่ายในการคำนวณและประเมิน ประสิทธิภาพผลลัพธ์ คือการลบค่าลบทั้งหมดออกจาก Convolution ค่าบวกทั้งหมด ยังคงเหมือนเดิม แต่ค่าลบทั้งหมดจะเปลี่ยนเป็นศูนย์ เนื่องจากค่าลบอาจบ่งบอกถึงข้อมูลที่ไม่เกี่ยวข้องหรือไม่สนใจในการแยกแยะคุณสมบัตินั้น ๆ ดังภาพประกอบที่ 3.14

สูตรคำนวณ ReLU

$$f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$

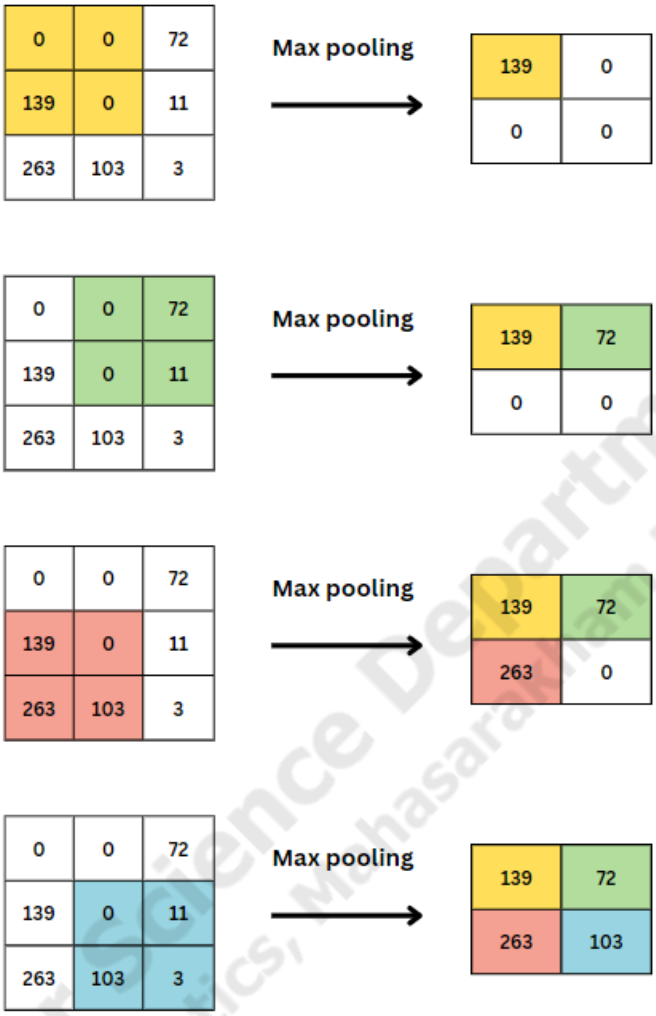


ภาพประกอบที่ 3.14 ตัวอย่างการทำ RELU

3.3.4 Max Pooling Layer

Max Pooling ช่วยให้โมเดลความสามารถในการระบุคุณสมบัติที่สำคัญมากยิ่งขึ้น เนื่องจากมันจะเลือกค่าที่มากที่สุดในบริเวณของ filter ซึ่งหมายความว่ามันจะเน้นการตอบสนองต่อลักษณะที่สำคัญ และช่วยให้โมเดลมีความเป็นอิสระต่อขนาดของข้อมูลนำเข้า เนื่องจากการขยับ filter ด้วยค่า stride ทำให้ขนาดข้อมูลผ่านไปอาจลดลง แต่ Max Pooling ยังคงรักษาข้อมูลที่สำคัญและลดขนาดข้อมูลอย่างมีประสิทธิภาพ

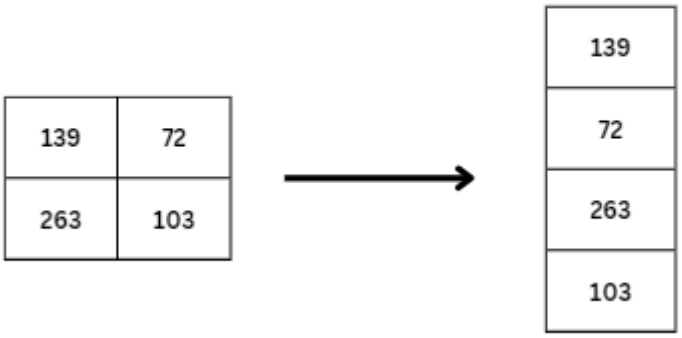
หลักการทำงานของ Max Pooling Layer ทำการกำหนดขนาดของ filter และกำหนดค่า stride จากนั้นวางฟิลเตอร์ที่มุมซ้ายสุดแล้วนำค่าที่มากที่สุดที่อยู่ในบริเวณของ filter มาเป็นค่าของรูปใหม่ จากนั้นขยับ filter ตามค่า stride เช่นค่า stride = 1 ก็ให้ขยับ filter ทีละ 1 ช่อง ดังภาพประกอบที่ 3.15



ภาพประกอบที่ 3.15 ตัวอย่างการทำ Max Pooling Layer

3.3.5 Flatten

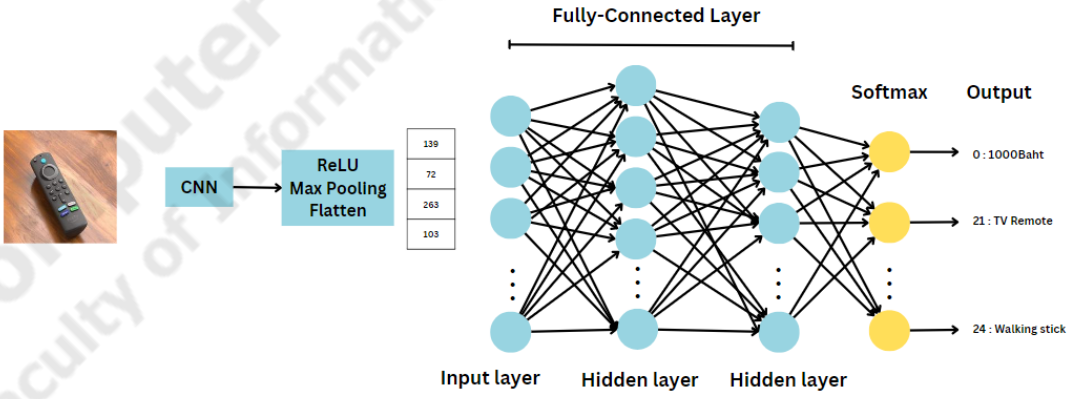
Flatten คือการทำให้ค่าข้อมูลที่เป็นภาพประกอบที่ได้จาก Max pooling 2 มิติ กลายเป็นข้อมูล Vector แบบ 1 มิติ เพื่อเป็นการเตรียมข้อมูลก่อนส่งเข้า Fully-Connected Layer เนื่องจาก Neuron ใน Fully-Connected Layer มักต้องรับข้อมูลในรูปแบบนี้ ดังภาพประกอบที่ 3.16



ภาพประกอบที่ 3.16 ตัวอย่างการทำ Flatten

3.3.6 Fully-Connected Layer

Fully-Connected Layer เป็นชั้นในโครงสร้างของ Neural Network ที่ทุก Neuron ในชั้นนี้เชื่อมโยงกับ Neuron ในชั้นก่อนหน้าแบบสมบูรณ์ หรือกล่าวอีกนัยหนึ่งคือทุก Neuron ใน Fully-Connected Layer นี้มีการเชื่อมต่อกับทุก Neuron ในชั้นก่อนหน้า (ซึ่งอาจเป็นชั้นที่เรียกว่า Flatten Layer หรือชั้น Convolutional Layer ก็ได้) โดยใช้น้ำหนัก (weights) และ bias เพื่อคำนวณผลลัพธ์ที่จะส่งไปยังชั้นถัดไป (Output Layer) สำหรับแต่ละ Neuron ใน Fully-Connected Layer นี้, มันรับข้อมูลจาก Neuron ในชั้นก่อนหน้าทุกตัว นำมาคูณกับน้ำหนักและบวกด้วย bias bias คือพารามิเตอร์เพิ่มเติมที่ใช้ในการปรับค่าออกมาจากการคูณระหว่างน้ำหนัก (weights) และข้อมูล input แล้วนำผลรวมไปผ่านฟังก์ชัน activation เพื่อสร้างผลลัพธ์ ดังภาพประกอบที่ 3.17



ภาพประกอบที่ 3.17 ตัวอย่างการทำงาน Fully-Connected Layer

3.3.7 Softmax

Softmax function เป็นฟังก์ชันที่จะแปลงคะแนนผลลัพธ์ของชั้นสุดท้ายในแต่ละโหนด ของโครงข่ายประสาทเทียมเป็นค่าความน่าจะเป็นตามสัดส่วนของคลาส คลาสไหนที่มีค่าความน่าจะเป็นมากที่สุดโมเดลจะเลือกคลาสนั้นเป็นคำตอบ

$$\sigma(z_i) = \frac{e^{z_i}}{\Sigma_{j=1}^{k} e^{z_i}}$$

โดยที่

- z คือ อินพุตของฟังก์ชัน softmax ซึ่งเป็นค่าของ pixel value (element) คูณกับ weight จนครบทุกโหนดแล้วนำไปบวกกับ bias จะได้ออกมาเป็นค่าของโหนดของแต่ละคลาส

- z_i คือ ค่าของแต่ละโหนด

- e^{z_i} คือ ค่ามาตรฐานเอกซ์โพเนนเชียล(e) ที่นำมายกกำลังโดย Zi แต่ละตัว

- $\Sigma_{j=1}^{k} e^{z_i}$ คือ ผลรวมของ e^{z_i}ทุกตัว

ยกตัวอย่างการคำนวณให้ z_i = (6.45, 1.80)

ขั้นตอนที่ 1 หาค่า e^{z_i} แต่ละค่า โดยค่า e มีค่าประมาณ 2.71828

e6.45 = 636.0489622

e1.80 = 6.049647464

ขั้นตอนที่ 2 หาค่า $\Sigma_{j=1}^{k} e^{z_i}$

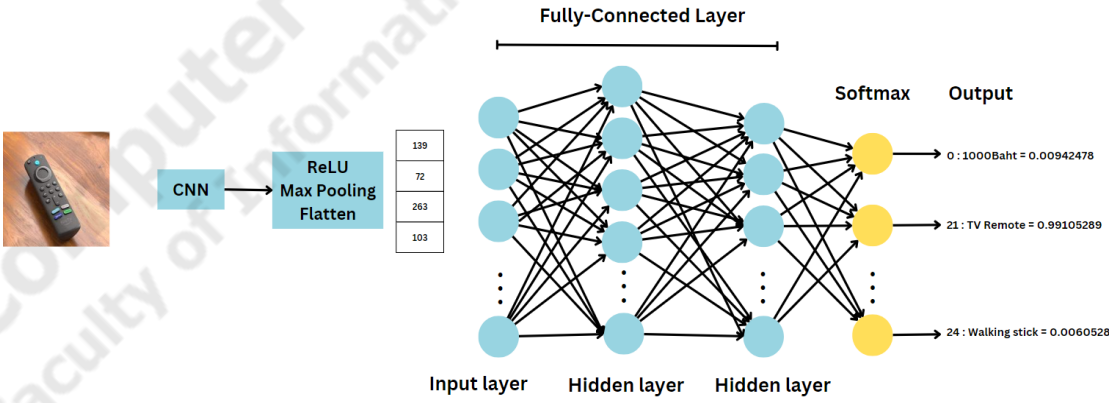
$\Sigma_{j=1}^{k} e^{z_i}$ = 636.0489622 + 6.049647464

= 642.098609664

ขั้นตอนที่ 3 หาค่า SoftMax โดยแทนค่าในสมการ

(6.45) = 636.0489622642.098609664 = 0.99105289

(1.80) = 6.049647464642.098609664 = 0.00942478



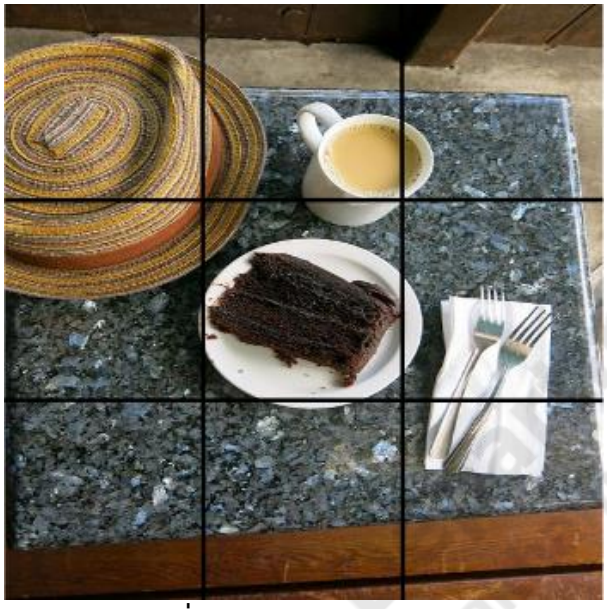
ภาพประกอบที่ 3.18 ตัวอย่างผลลัพธ์การทำ SoftMax

3.4 Detection

ในการค้นหาวัตถุจะยกตัวอย่างภาพนำเข้า 2 แบบคือ ภาพที่มีวัตถุชนิดเดียว และ ภาพที่มีวัตถุสองชนิด โดยมีรายละเอียดแสดงตามลำดับดังนี้

3.4.1 ตัวอย่างภาพที่มีวัตถุชนิดเดียวแสดงขั้นตอนการคำนวณดังนี้

กำหนดขนาดภาพ 640 x 640 และแบ่งภาพเป็นตาราง 3 x 3



ภาพประกอบที่ 3.19 ตัวอย่างการแบ่งตาราง 3 x 3

3.4.1.1 กำหนดค่า y

เพื่อเก็บคำตอบ กำหนดค่า y เพื่อเก็บคำตอบ มีค่าเท่ากับ S x S x A x (5 + จำนวน class) โดยให้ A=2 ซึ่ง A คือ จำนวน anchor ภายในแต่ละ grid ในตัวอย่างกำหนดจำนวน class เท่ากับ 1 คือ Plate ดังนั้น y=3x3x2x(5+1) หรือ 3x3x2x6 การเก็บข้อมูลแสดงดังนี้

$$y = \begin{vmatrix} p(\text{Plate}) \\ bx \\ by \\ bh \\ bw \end{vmatrix}$$

โดย p คือ class ของวัตถุนั้น และ (bx,by,bh,bw) คือตำแหน่งของวัตถุ

3.4.1.2 ขนาดของ anchor

ในแต่ละตารางจะหาได้จาก k-mean หากแต่ละตารางต้องการ 2 anchor แสดงดังภาพประกอบที่ 3.24

3.4.1.3 K-mean

K-Means เป็นวิธีที่นิยมใช้ในการแบ่งกลุ่มข้อมูล โดยเปรียบเทียบความคล้ายคลึง ของข้อมูล กับจุด ศูนย์กลางของแต่ละคลัสเตอร์ (Cluster) หรือค่าเฉลี่ย (Mean) เป็นการแบ่งแบบ Partitional clustering ด้วยการแบ่งข้อมูลออกเป็นส่วน ตามจำนวนกลุ่มที่ระบุ มี 4 ขั้นตอน

1. mark - กำหนดจำนวนกลุ่ม K กลุ่ม และกำหนดจุดศูนย์กลางเริ่มต้นจำนวน K จุดด้วยการสุ่ม

2. distance - นำวัตถุทั้งหมดจัดเข้ากลุ่มที่มีจุดศูนย์กลางที่อยู่ใกล้วัตถุนั้นมากที่สุด โดย คำนวณจากการวัดระยะห่างระหว่างจุดที่น้อยที่สุด

3. center - คำนวณจุดศูนย์กลาง K จุดใหม่ โดยหาจากค่าเฉลี่ยทุกวัตถุที่อยู่ในกลุ่ม

4. repeat - ทำซ้ำในข้อ 2. จนกระทั่งจุดศูนย์กลางไม่เปลี่ยนแปลงสมการสูตร Euclidean

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

ตัวอย่างที่ K-mean นำภาพมาจาก grid ที่ 5 โดยทำการสุ่ม C1 = 41 ,C2 = 132

25	30	105	130	127
75	41	55	150	170
73	27	134	188	199
45	43	56	132	233
53	23	11	209	213

ภาพประกอบที่ 3.20 ภาพตัวอย่างที่นำมาทำ k-mean

จุดที่ 1 ที่ 25

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

dist1 = $\sqrt{(25 - 41)^2}$

= 16

dist2 = $\sqrt{(25 - 132)^2}$

= 107

ดังนั้น 25 จะถูกจัดในกลุ่มข้อมูลเดียวกับ C1=41 เนื่องจากมีระยะห่างที่น้อยกว่า C2=132 แสดงดังภาพที่ 3.21

25	30	105	130	127
75	41	55	150	170
73	27	134	188	199
45	43	56	132	233
53	23	11	209	213

ภาพประกอบที่ 3.21 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูลตัวอย่างที่ 1

จุดที่ 2 ที่ 30

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

dist1 = $\sqrt{(30 - 41)^2}$

= 11

dist2 = $\sqrt{(30 - 132)^2}$

= 102

ดังนั้น 30 จะถูกจัดในกลุ่มข้อมูลเดียวกับ C1=41 เนื่องจากมีระยะห่างที่น้อยกว่า C2=132 แสดงดังภาพที่ 3.22

25	30	105	130	127
75	41	55	150	170
73	27	134	188	199
45	43	56	132	233
53	23	11	209	213

ภาพประกอบที่ 3.22 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูลตัวอย่างที่ 2

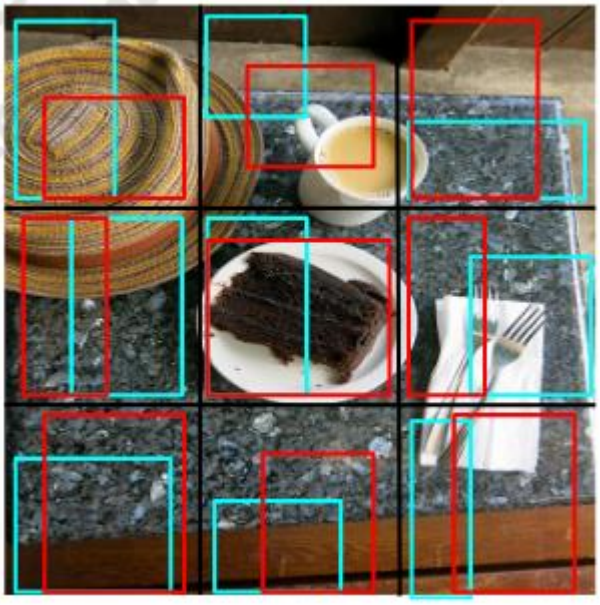
เมื่อคำนวณต่อไปเรื่อยๆจนครบทุกพิกเซลแล้ว จะทำการคำนวณจุดศูนย์กลาง ใหม่ โดยหาจากค่าเฉลี่ยทุกตัวที่อยู่ในกลุ่ม จนกว่าค่าจุดศูนย์กลางจะไม่เปลี่ยนแปลง

25	30	105	130	127
75	41	55	150	170
73	27	134	188	199
45	43	56	132	233
53	23	11	209	213

ภาพประกอบที่ 3.23 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูล

ภาพประกอบที่ 3.24 ตัวอย่างผลลัพธ์การกำหนดขนาด anchor โดยใช้ k-mean

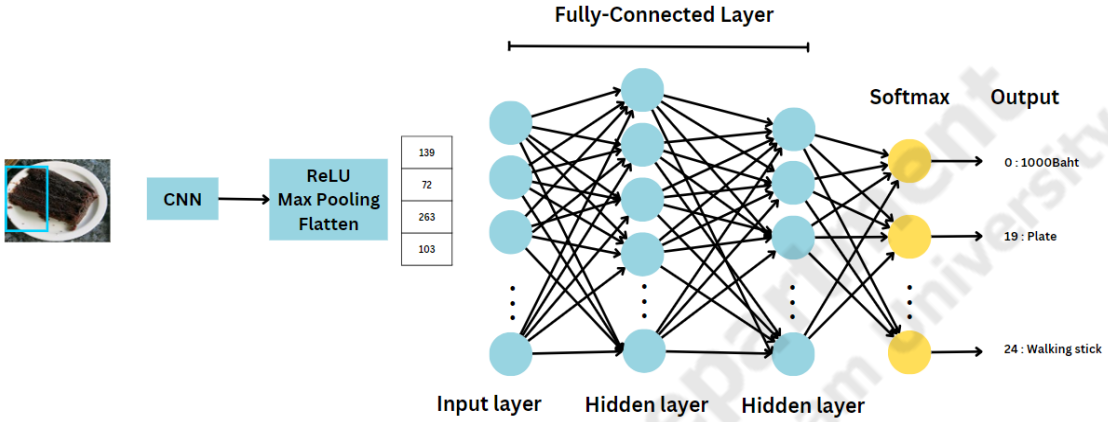
ผลลัพธ์ที่ได้ใน grid อื่นๆดัง ภาพประกอบที่ 3.25



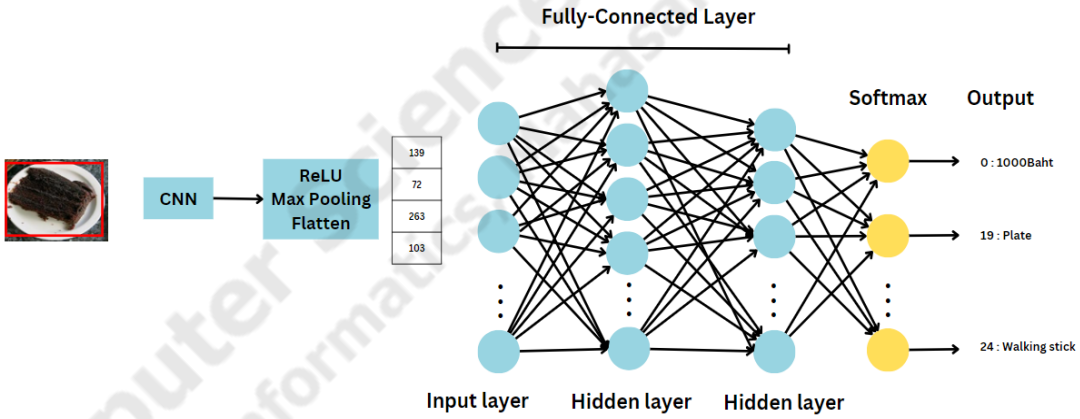
ภาพประกอบที่ 3.25 การแบ่งกลุ่มข้อมูลในแต่ละช่อง โดยแต่ละช่องจะมี 2 anchor

3.4.1.4 นำแต่ละ bounding box สกัด Features ด้วย CNN และเข้าโมเดลเพื่อตรวจสอบว่าอยู่ในคลาสใด

ตัวอย่างนำ grid ที่ 5 bounding box สีน้ำเงินและ bounding box สีแดงแสดงดังภาพประกอบที่ 3.26 และ ภาพประกอบที่ 3.27



ภาพประกอบที่ 3.26 นำแต่ละ bounding box สีน้ำเงินสกัด Features ด้วย CNN และเข้าโมเดล



ภาพประกอบที่ 3.27 นำแต่ละ bounding box สกัด Features ด้วย CNN และเข้าโมเดล

การเก็บค่า Y ของ grid ที่ 5 หลังจากเข้าโมเดลจะทราบว่าวัตถุที่อยู่ใน bounding box นั้นเป็น class อะไร

$$y = \begin{bmatrix} p(Class) \\ bx \\ by \\ bh \\ bw \end{bmatrix}$$

ใน grid อื่นๆก็ทำเช่นเดียวกันจนครบทุก grid จากนั้นต้องหาค่า probity โดยหาจากค่า IOU

3.4.1.5 IOU

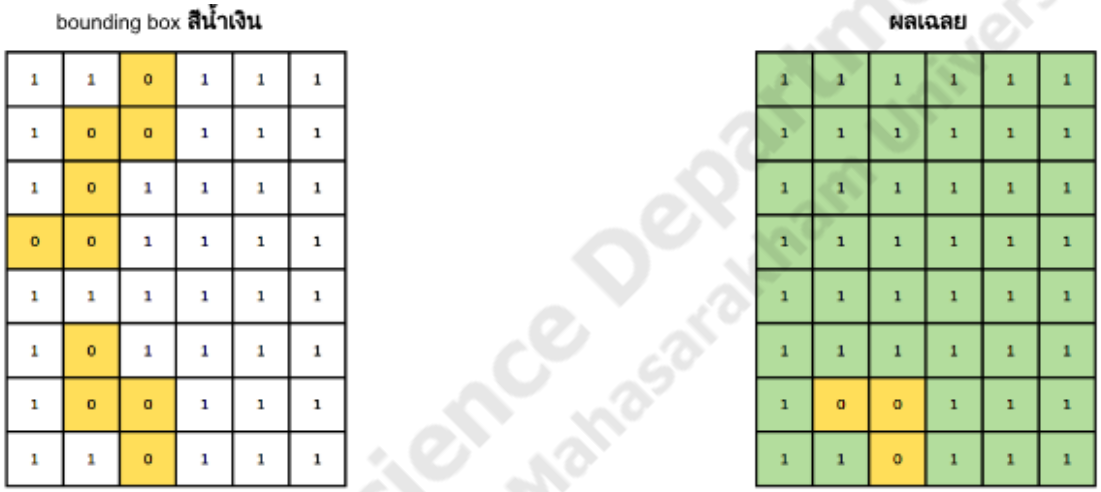
ในขั้นตอนนี้จะเป็นการนำแต่ละ bounding box ไปเปรียบเทียบกับผลเฉลย

สูตรคำนวณ IOU

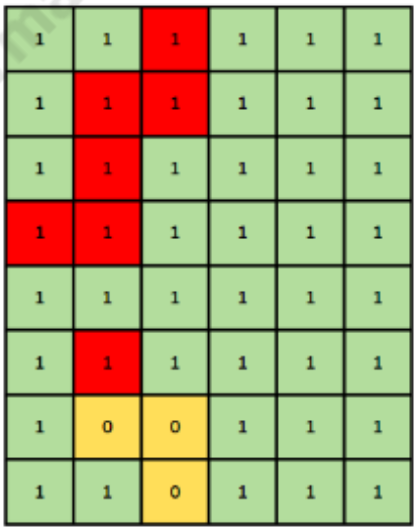
$$IOU = \frac{area\_intersection}{areaUnion} \quad (1)$$

areaunio = area_box1 + area_box2 - area_intersection (2)

ยกตัวอย่างการนำ grid ที่ 5 มาหาค่า IOU



ภาพประกอบที่ 3.28 ตัวอย่างข้อมูลของกรอบสีน้ำเงินและผลเฉลย



ภาพประกอบที่ 3.29 areaOverlap

areaUnion = 54+54-44

= 64

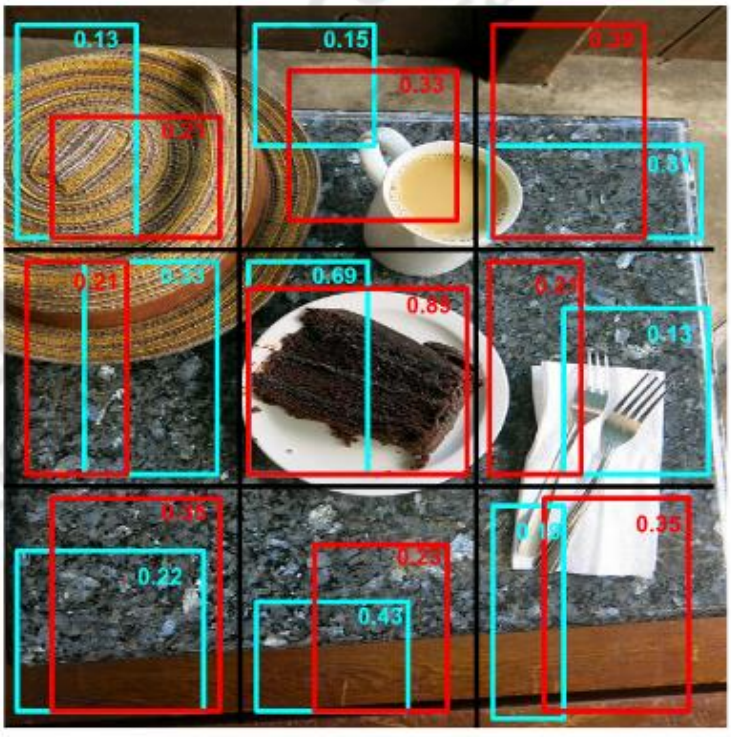
IOU = 4464

= 0.69

ดังนั้นค่า probity ของ bounding box สีน้ำเงินเท่ากับ 0.69 ใน bounding box สีแดงใช้วิธีการเดียว และคำนวณทุก grid การเก็บค่า Y ของ grid ตัวอย่าง bounding box สีน้ำเงินและ bounding box สีแดง หลังจากทำ IOU จะได้ค่า probity

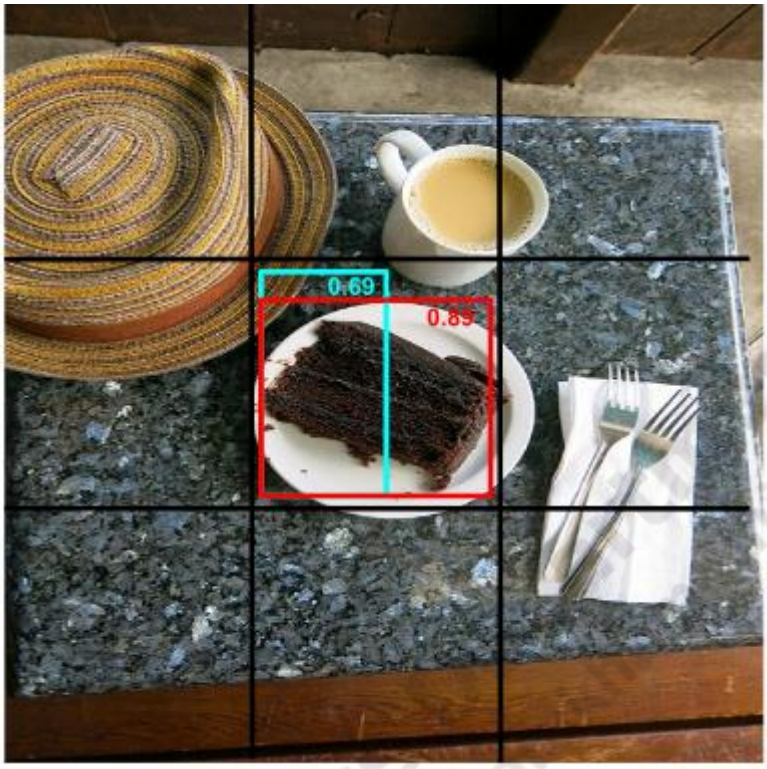
$$y = \begin{vmatrix} p(0.69) \\ 0.107 \\ 0.45 \\ 0.61 \\ 0.37 \\ p(0.81) \\ 0.133 \\ 0.41 \\ 0.72 \\ 0.45 \end{vmatrix}$$

หลังจากหาค่า probity จาก IOU ทั้งหมด แสดงดังภาพตัวอย่างที่ 3.30



ภาพประกอบที่ 3.30 ตัวอย่างหลังจากการหาค่า probity ทั้งหมด

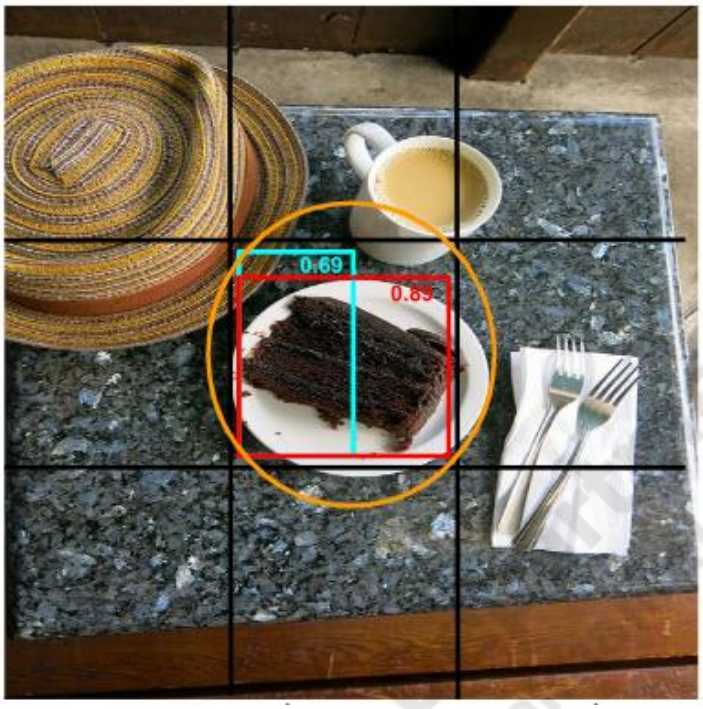
เมื่อหาค่า Y ได้ครบแล้วเราจะกำหนดว่าถ้าค่า probity น้อยกว่า 0.6 จะไม่นำค่านั้นมาคำนวณต่อ



ภาพประกอบที่ 3.31 ตัดข้อมูลที่มีความน่าจะเป็นน้อยกว่าที่กำหนดออก

3.4.1.6 non-max suppression

โดยวิธีการนี้จะทำการคำนวณค่า IOU ซึ่งเป็นการหาอัตราส่วน ของ Intersection area และ Union area ซึ่งเราจะกำหนดว่า ถ้าค่า IOU มากกว่า 0.5 เมื่อค่า IOU สูง แสดงว่ามี bounding box กำลังซ้อนทับกันอยู่ จะใช้วิธีการ non-max suppression ซึ่งจะสนใจ bounding box ที่มีความน่าจะเป็นสูงสุด ถ้าค่า IOU ต่ำกว่า 0.5 เราจะใช้คำตอบของทั้งสอง bounding box



ภาพประกอบที่ 3.32 ตัวอย่าง grid ที่ 5

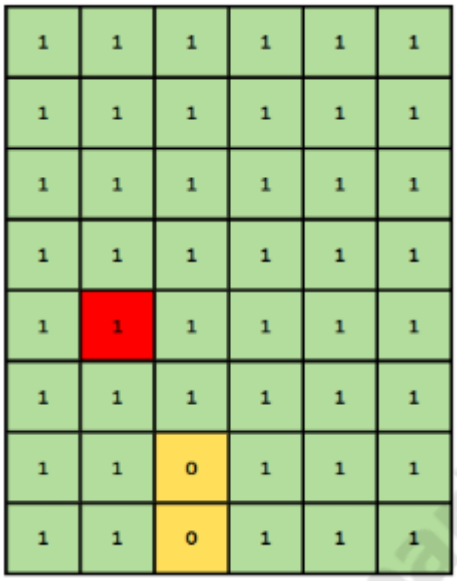
bounding box สีน้ำเงิน

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

bounding box สีแดง

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	0	1	1	1

ภาพประกอบที่ 3.33 ตัวอย่างข้อมูล bounding box สีน้ำเงิน และสีแดง



ภาพประกอบที่ 3.34 intersection_area ของ bounding box สีน้ำเงินและสีแดง

จากสูตรคำนวณ IOU คือ

$$IOU = \frac{area\_intersection}{areaUnion} \quad (1)$$

areaunio = area_box1 + area_box2 - area_intersection (2)

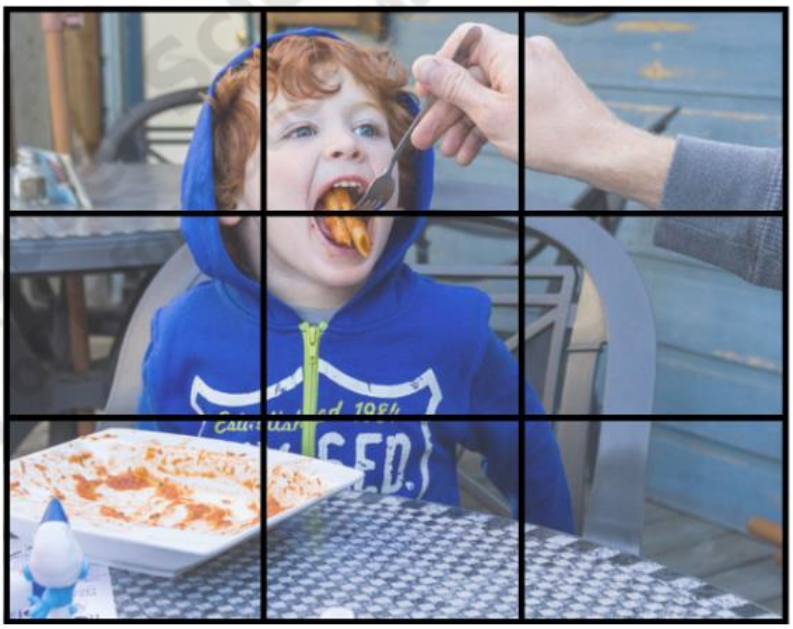
areaunion = 54 + 54 − 51 = 57

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

จากการคำนวณ IOU ของ grid ตัวอย่างได้ค่า IOU เท่ากับ 0.89 ซึ่งมากกว่าที่เรากำหนดไว้ แสดงว่า bounding box สีน้ำเงินและ bounding box สีแดงกำลังครอบวัตถุเดียวกันอยู่ จึงใช้วิธีการ non-max suppression ซึ่งจะสนใจ bounding box ที่มีความน่าจะเป็นสูงสุด ใน gird ที่ 5 ก็ทำเช่นกัน จนครบทุก grid

ภาพประกอบที่ 3.35 ตัวอย่าง bounding box ที่มีความน่าจะเป็นสูงสุด

ตัวอย่างภาพนำเข้าที่มีวัตถุสองชนิด จานและช้อมแสดงขั้นตอนดังนี้
กำหนดขนาดภาพ 640 x 640 และแบ่งภาพเป็นตาราง 3 x 3



ภาพประกอบที่ 3.36 ตัวอย่างการแบ่งตาราง 3x3

3.4.1.1 กำหนดค่า y เพื่อเก็บคำตอบ

กำหนดค่า y เพื่อเก็บคำตอบ มีค่าเท่ากับ S x S x A x (5+จำนวน class) โดยให้ A=2

ซึ่ง A คือ จำนวน anchor ภายในแต่ละ grid ในตัวอย่างกำหนดจำนวน class เท่ากับ 2 คือ plate และ fork ดังนั้น y=3x3x2x(5+2) หรือ 3x3x2x7 การเก็บข้อมูลแสดงดังนี้

$$y = \begin{vmatrix} p(\text{Plate}) \\ bx \\ by \\ bh \\ bw \\ p(\text{Fork}) \\ bx \\ by \\ bh \\ bw \end{vmatrix}$$

โดย p คือ class ของวัตถุนั้น และ (bx,by,bh,bw) คือตำแหน่งของวัตถุ

3.4.1.2 ขนาดของ anchor

ในแต่ละตารางจะหาได้จาก k-mean หากแต่ละ grid ต้องการ 2 anchor แสดงดังภาพประกอบที่ 3.41

3.4.1.3 K-mean

K-Means เป็นวิธีที่นิยมใช้ในการแบ่งกลุ่มข้อมูล โดยเปรียบเทียบความคล้ายคลึง ของข้อมูล กับจุด ศูนย์กลางของแต่ละคลัสเตอร์ (Cluster) หรือค่าเฉลี่ย (Mean) เป็นการแบ่งแบบ Partitional clustering ด้วยการแบ่งข้อมูลออกเป็นส่วน ตามจำนวนกลุ่มที่ระบุ มี 4 ขั้นตอน

1. mark - กำหนดจำนวนกลุ่ม K กลุ่ม และกำหนดจุดศูนย์กลางเริ่มต้นจำนวน K จุดด้วยการสุ่ม
2. distance - นำวัตถุทั้งหมดจัดเข้ากลุ่มที่มีจุดศูนย์กลางที่อยู่ใกล้วัตถุนั้นมากที่สุด โดย คำนวณจากการวัดระยะห่างระหว่างจุดที่น้อยที่สุด
3. center - คำนวณจุดศูนย์กลาง K จุดใหม่ โดยหาจากค่าเฉลี่ยทุกวัตถุที่อยู่ในกลุ่ม
4. repeat - ทำซ้ำในข้อ 2. จนกระทั่งจุดศูนย์กลางไม่เปลี่ยนแปลงสมการสูตร Euclidean

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

ตัวอย่างที่1 K-mean นำภาพมาจาก grid ที่ 2 โดยทำการสุ่ม C1 = 20, C2 = 225

11	15	25	188	155
12	20	190	168	198
40	35	17	250	186
25	59	47	225	242
29	17	13	201	217

ภาพประกอบที่ 3.37 ภาพตัวอย่างที่นำมาทำ k-mean grid ที่ 2

ยกตัวอย่างคำนวณจุดที่ 1

11	15	25	188	155
12	20	190	168	198
40	35	17	250	186
25	59	47	225	242
29	17	13	201	217

ภาพประกอบที่ 3.38 ตัวอย่างการจัดกลุ่มข้อมูลตัวอย่างที่ 1

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

dist1 = $\sqrt{(11-20)^2}$

= 9

dist2 = $\sqrt{(11-225)^2}$

= 244

ดังนั้น 11 จะถูกจัดในกลุ่มข้อมูลเดียวกับ C1=20 เนื่องจากมีระยะห่างที่น้อยกว่า C2=225 แสดงดังภาพประกอบที่ 3.39

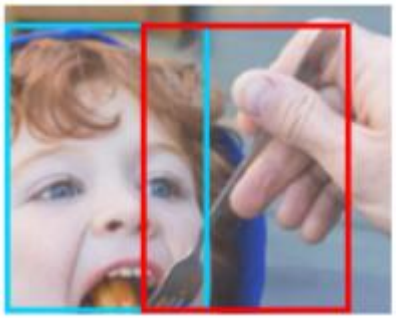
11	15	25	188	155
12	20	190	168	198
40	35	17	250	186
25	59	47	225	242
29	17	13	201	217

ภาพประกอบที่ 3.39 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูลตัวอย่างที่ 2

เมื่อคำนวณต่อไปเรื่อยๆจนครบทุกพิกเซลแล้ว จะทำการคำนวณจุดศูนย์กลาง ใหม่ โดยหาจากค่าเฉลี่ยทุกตัวที่อยู่ในกลุ่ม จนกว่าค่าจุดศูนย์กลางจะไม่เปลี่ยนแปลง

11	15	25	188	155
12	20	190	168	198
40	35	17	250	186
25	59	47	225	242
29	17	13	201	217

ภาพประกอบที่ 3.40 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูล grid ที่ 2



ภาพประกอบที่ 3.41 ตัวอย่างผลลัพธ์การกำหนดขนาด anchor โดยใช้ k-mean

ตัวอย่างที่2 K-mean นำภาพมาจาก grid ที่ 7 โดยทำการสุ่ม C1 = 45 ,C2 = 220

175	189	185	19	57
166	220	19	60	74
170	188	14	82	79
157	168	187	45	40
185	179	18	19	34

ภาพประกอบที่ 3.42 ภาพตัวอย่างที่นำมาทำ k-mean grid ที่ 7

ยกตัวอย่างคำนวณจุดที่ 1

dist = $\sqrt{\sum_{k=1}^{n}(P_k - q_k)^2}$ หรือ $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

dist1 = $\sqrt{(175 - 45)^2}$

= 130

dist2 = $\sqrt{(175 - 220)^2}$

= 45

ดังนั้น 175 จะถูกจัดในกลุ่มข้อมูลเดียวกับ C2= 220 เนื่องจากมีระยะห่างที่น้อยกว่า C1= 45 แสดงดังภาพประกอบที่ 3.43

175	189	185	19	57
166	220	19	60	74
170	188	14	82	79
157	168	187	45	40
185	179	18	19	34

ภาพประกอบที่ 3.43 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูลตัวอย่างที่ 2

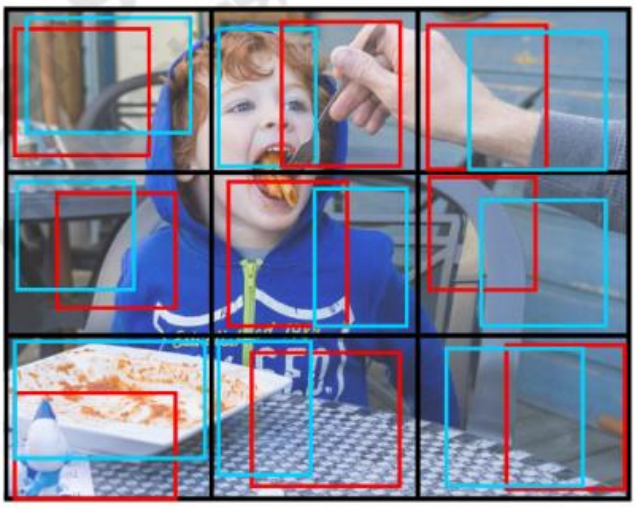
เมื่อคำนวณต่อไปเรื่อยๆจนครบทุกพิกเซลแล้ว จะทำการคำนวณจุดศูนย์กลาง ใหม่ โดยหาจากค่าเฉลี่ยทุกตัวที่อยู่ในกลุ่ม จนกว่าค่าจุดศูนย์กลางจะไม่เปลี่ยนแปลง

175	189	185	19	57
166	220	19	60	74
170	188	14	82	79
157	168	187	45	40
185	179	18	19	34

ภาพประกอบที่ 3.44 ตัวอย่างผลลัพธ์การจัดกลุ่มข้อมูล grid ที่ 7

ภาพประกอบที่ 3.45 ตัวอย่างผลลัพธ์การกำหนดขนาด anchor โดยใช้ k-mean

ผลลัพธ์ที่ได้ใน grid อื่นๆ ดัง ภาพประกอบที่ 3.46

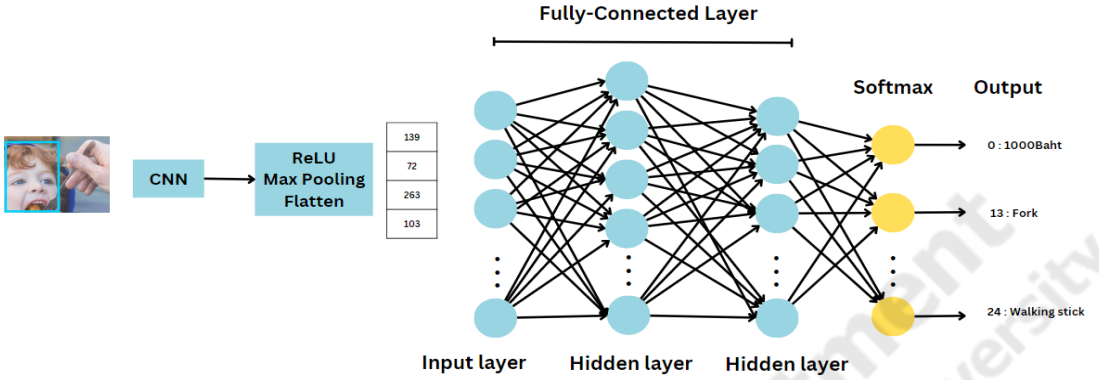


ภาพประกอบที่ 3.46 การแบ่งกลุ่มข้อมูลในแต่ละ grid โดยแต่ละ grid จะมี 2 anchor

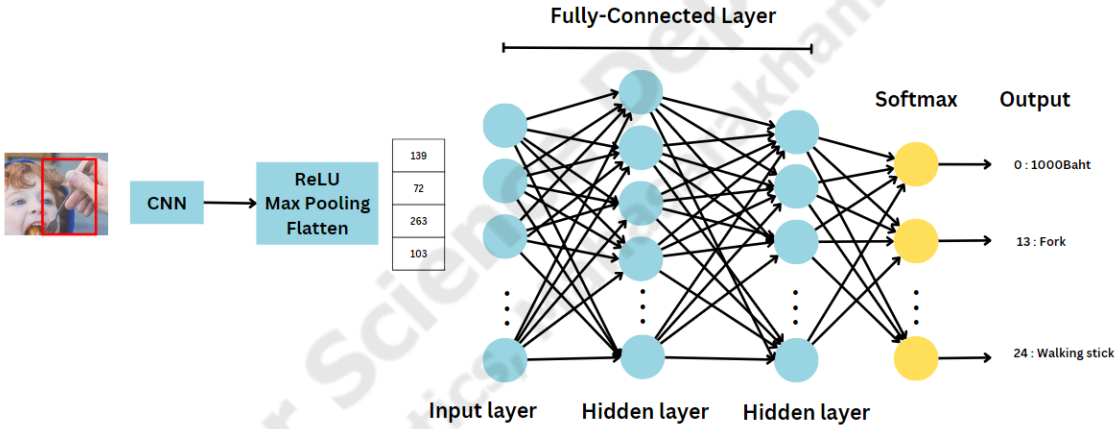
3.4.1.1 นำแต่ละ bounding box สกัด Features ด้วย CNN และเข้าโมเดลเพื่อตรวจสอบว่าอยู่ในคลาสใด

ตัวอย่างนำ grid ที่ 2 bounding box สีน้ำเงินและ bounding box สีแดงแสดงดัง

ภาพประกอบที่ 3.47 และ ภาพประกอบที่ 3.48



ภาพประกอบที่ 3.47 นำแต่ละ bounding box สีน้ำเงิน สกัด Features ด้วย CNN และเข้าโมเดล

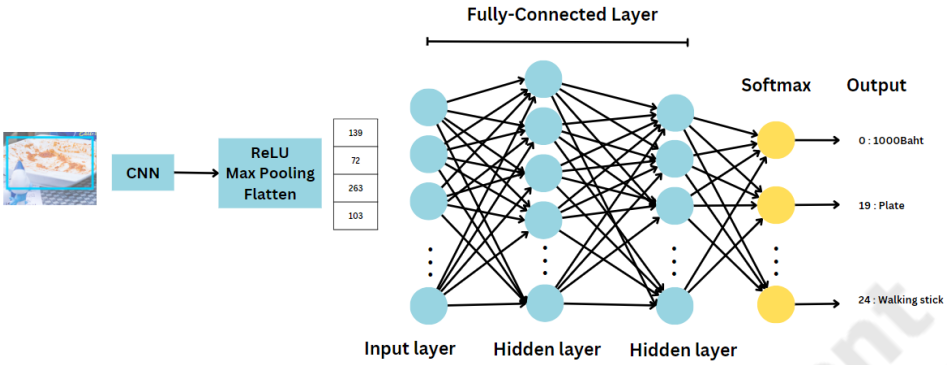


ภาพประกอบที่ 3.48 นำแต่ละ bounding box สีแดง สกัด Features ด้วย CNN และเข้าโมเดล

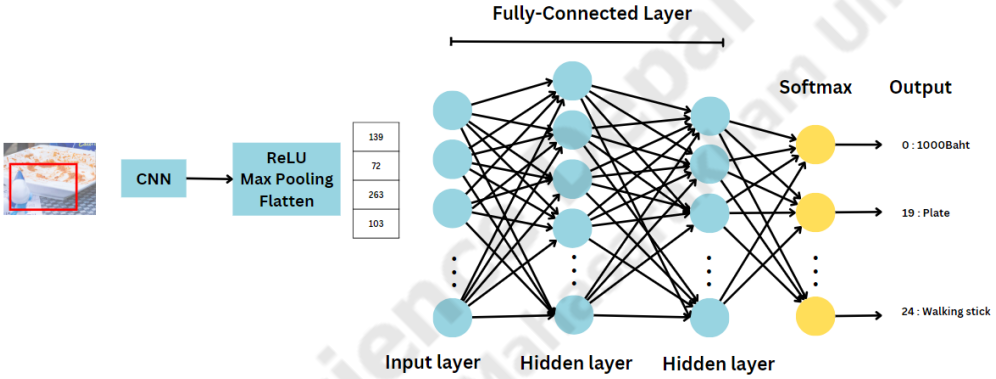
การเก็บค่า Y ของ grid ที่ 2 หลังจากเข้าโมเดล เราจะทราบว่าวัตถุที่อยู่ในกรอบนั้นเป็นคลาสอะไร

$$y = \begin{bmatrix} p(Fork) \\ bx \\ by \\ bh \\ bw \end{bmatrix}$$

ตัวอย่างนำ grid ที่ 7 bounding box สีน้ำเงินและ bounding box สีแดง แสดงดัง ภาพประกอบที่ 3.49 และ ภาพประกอบที่ 3.50



ภาพประกอบที่ 3.49 นำแต่ละ bounding box สีน้ำเงิน สกัด Features ด้วย CNN และเข้าโมเดล



ภาพประกอบที่ 3.50 นำแต่ละ bounding box สีแดง สกัด Features ด้วย CNN และเข้าโมเดล

การเก็บค่า Y ของ grid ที่ 7 หลังจากเข้าโมเดล เราจะทราบว่าวัตถุที่อยู่ใน bounding box นั้นเป็นคลาสอะไร

$$y = \begin{vmatrix} p(Plate) \\ bx \\ by \\ bh \\ bw \end{vmatrix}$$

ใน grid อื่นๆก็ทำเช่นเดียวกันจนครบทุก gird จากนั้นต้องหาค่า probity โดยหาจากค่า IOU

3.4.1.2 IOU

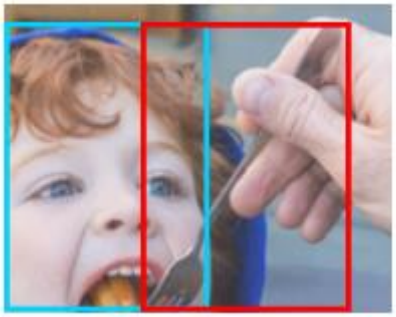
ในขั้นตอนนี้จะเป็นการนำแต่ละ bounding box ไปเปรียบเทียบกับผลเฉลย

สูตรคำนวณ IOU

$$IOU = \frac{area\_intersection}{areaUnion} \quad (1)$$

areaunio = area_box1 + area_box2 - area_intersection (2)

ยกตัวอย่าง นำ grid ที่2 มาหาค่า IOU



ภาพประกอบที่ 3.51 ภาพตัวอย่างการคำนวณ IOU

ตัวอย่างการหาค่า IOU ของ bounding box สีน้ำเงิน

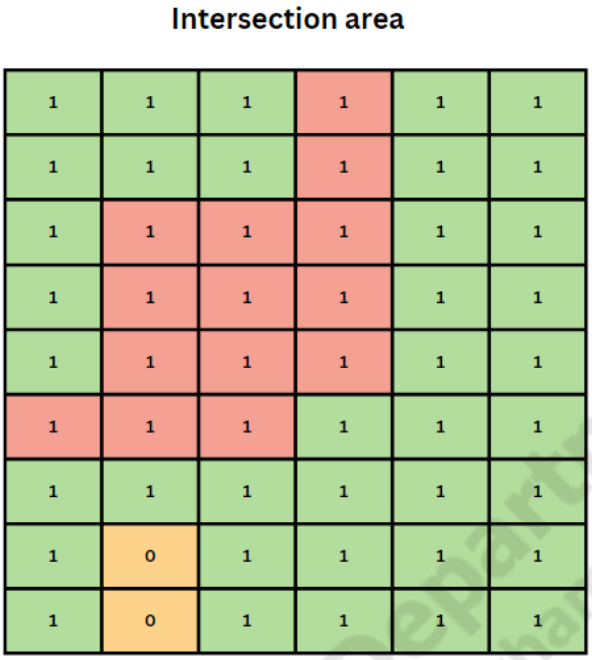
bounding box สีน้ำเงิน

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

ผลเฉลย

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	0	1	1	1	1

ภาพประกอบที่ 3.52 ตัวอย่างข้อมูลของ bounding box สีน้ำเงินและผลเฉลย



ภาพประกอบที่ 3.53 intersection_area

จากสูตร IOU คือ

$$IOU = \frac{area\_intersection}{areaUnion} \quad (1)$$

areaunio = area_box1 + area_box2 - area_intersection (2)

areaunio = 54 + 54 – 38 = 70

$IOU = \frac{38}{70} = 0.54$

ดังนั้นค่า probity ของ bounding box สีน้ำเงินเท่ากับ 0.54 ใน bounding box สีแดงใช้วิธีการเดียวกัน และคำนวณทุก grid

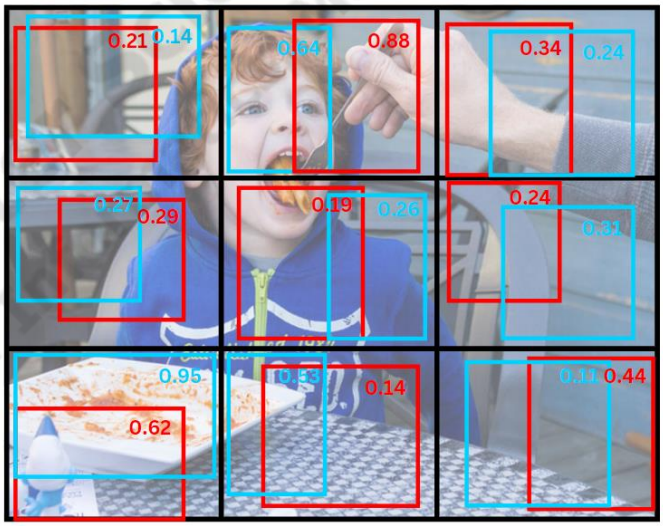
การเก็บค่า Y ของ grid ตัวอย่าง bounding box สีน้ำเงินและ bounding box สีแดง หลังจากทำ IOU จะได้ค่า probity

$$y = \begin{vmatrix} p(0.54) \\ 0.102 \\ 0.34 \\ 0.57 \\ 0.40 \\ p(0.88) \\ 0.142 \\ 0.32 \\ 0.62 \\ 0.50 \end{vmatrix}$$

ตัวอย่างการเก็บค่า Y ของ grid ที่ 7 bounding box สีน้ำเงินและ bounding box สีแดง หลังจากทำ IOU จะได้ค่า probity

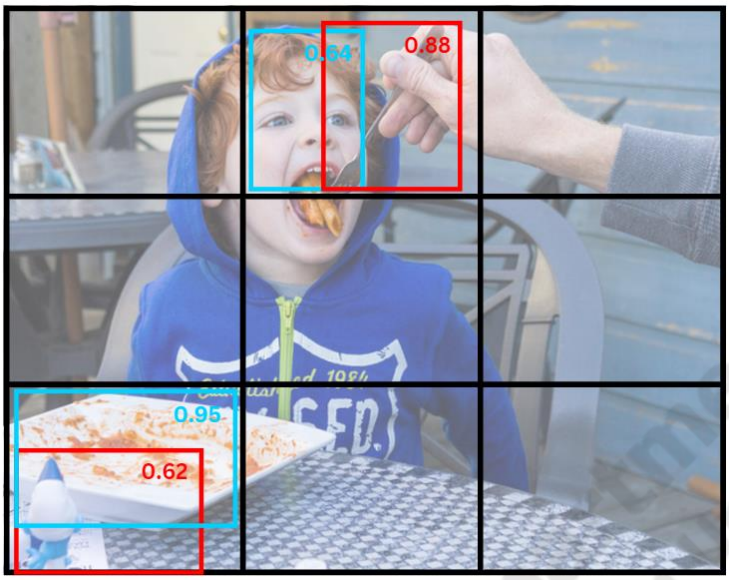
$$y = \begin{vmatrix} p(0.95) \\ 0.23 \\ 0.220 \\ 0.77 \\ 0.50 \\ p(0.62) \\ 0.12 \\ 0.201 \\ 0.121 \\ 0.50 \end{vmatrix}$$

หลังจากหาค่า probity จาก IOU ทั้งหมด แสดงดังภาพตัวอย่างที่ 3.53



ภาพประกอบที่ 3.54 ตัวอย่างหลังจากการหาค่า probity ทั้งหมด

เมื่อหาค่า Y ได้ครบแล้วเราจะกำหนดว่าถ้าค่า probity น้อยกว่า 0.6 จะไม่นำค่านั้นมาคำนวณต่อ

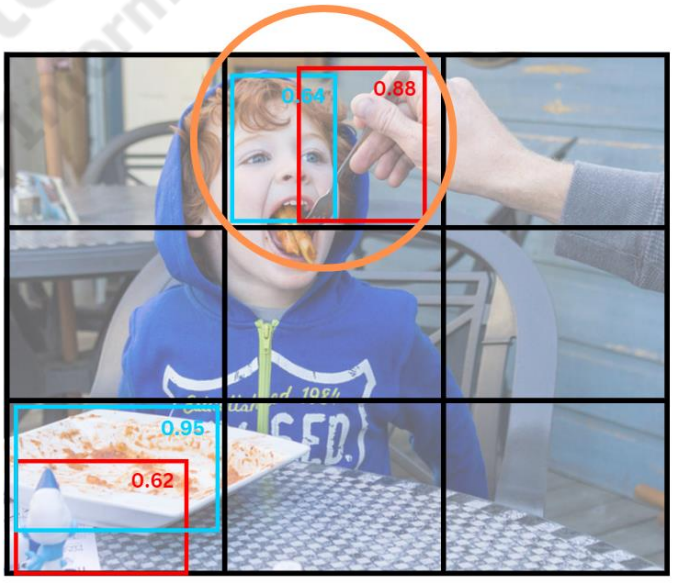


ภาพประกอบที่ 3.55 ตัดข้อมูลที่มีความน่าจะเป็นน้อยกว่าที่กำหนดออก

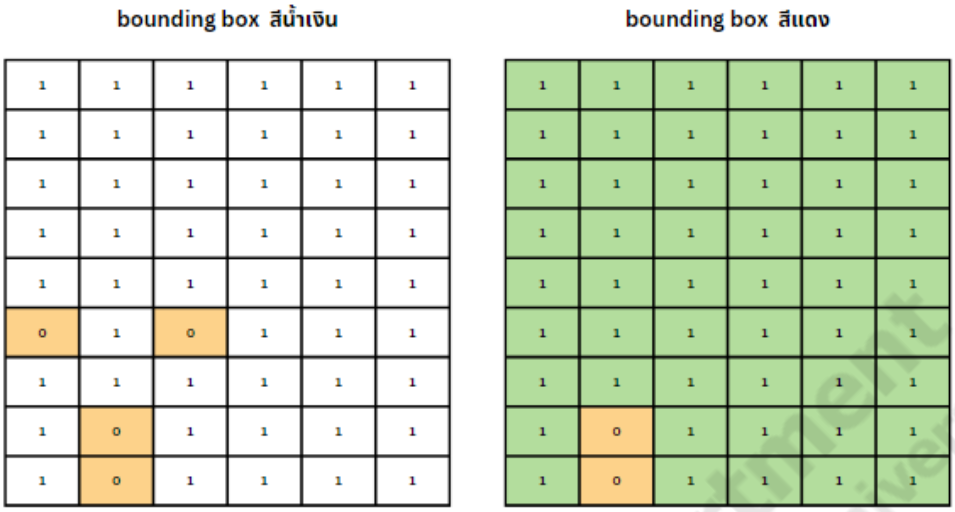
3.4.1.1 non-max suppression

โดยวิธีการนี้จะทำการคำนวณค่า IOU ซึ่งเป็นหาอัตราส่วน ของ Intersection area และ Union area ซึ่งเราจะกำหนดว่า ถ้าค่า IOU มากกว่า 0.5 เมื่อค่า IOU สูง แสดงมี bounding box กำลัง ซ้อนกันอยู่เราจะใช้วิธีการ non-max suppression ซึ่งจะสนใจ bounding box ที่มีความน่าจะเป็นสูงสุด ถ้าค่า IOU ต่ำกว่า 0.5 เราจะใช้คำตอบของทั้งสอง bounding box

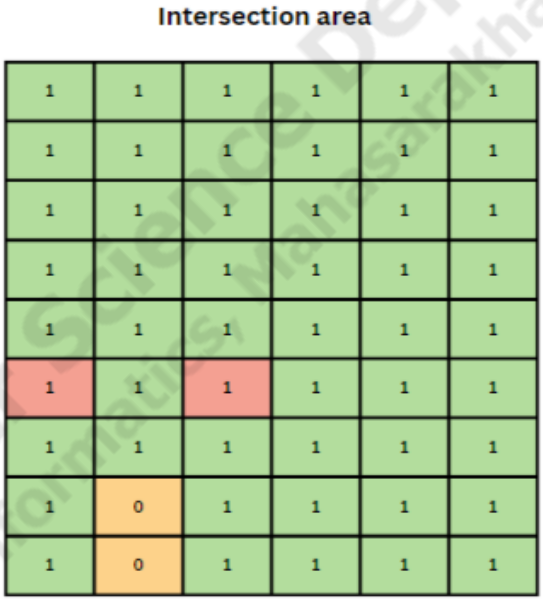
$$IOU = \frac{area\_intersection}{areaUnion}$$



ภาพประกอบที่ 3.56 ตัวอย่าง grid ที่ 2



ภาพประกอบที่ 3.57 ตัวอย่างข้อมูลของ bounding box สีน้ำเงินและ bounding box สีแดง



ภาพประกอบที่ 3.58 intersection_area ของ bounding box สีน้ำเงินและ bounding box สีแดง

จากสูตร IOU คือ

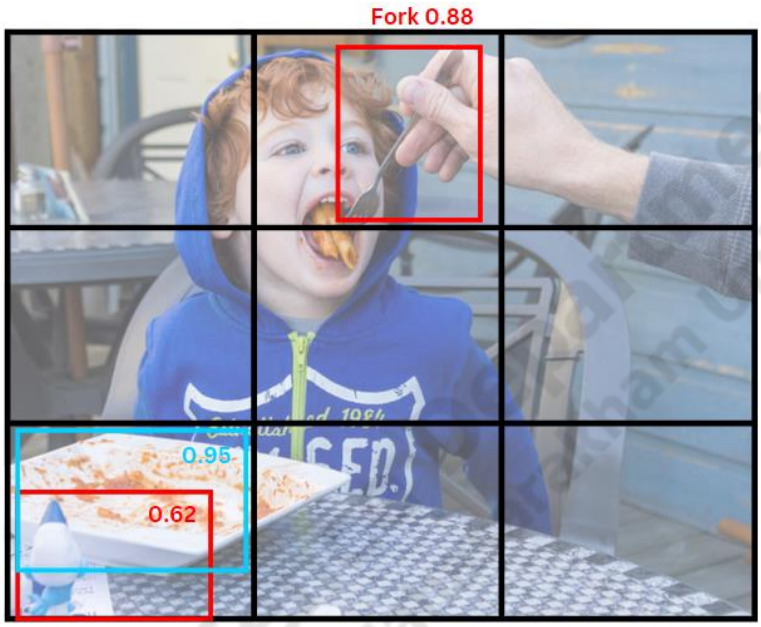
$$IOU = \frac{area\_intersection}{areaUnion} \quad (1)$$

areaunio = area_box1 + area_box2 - area_intersection (2)

areaunio = 54 + 54 − 50 = 58

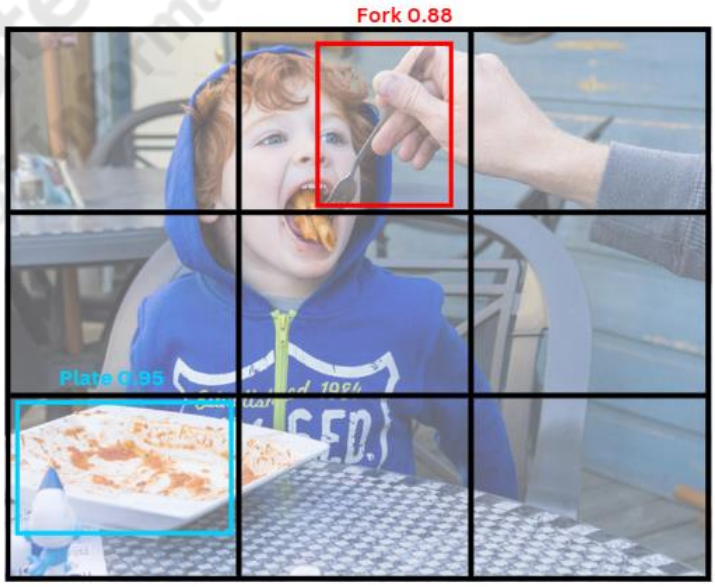
$IOU = \frac{50}{58} = 0.86$

จากการคำนวณ IOU ของ grid ตัวอย่างได้ค่า IOU เท่ากับ 0.86 ซึ่งมากกว่าที่เรากำหนดไว้ แสดงว่า bounding box สีน้ำเงินและ bounding box สีแดงกำลังครอบวัตถุเดียวกันอยู่ จึงใช้วิธีการ non-max suppression ซึ่งจะสนใจ bounding box ที่มีความน่าจะเป็นสูงสุด ใน gird ที่ 7 ก็ทำเช่นกัน จนครบทุก grid



ภาพประกอบที่ 3.59 ตัวอย่าง bounding box ที่มีความน่าจะเป็นสูงสุด

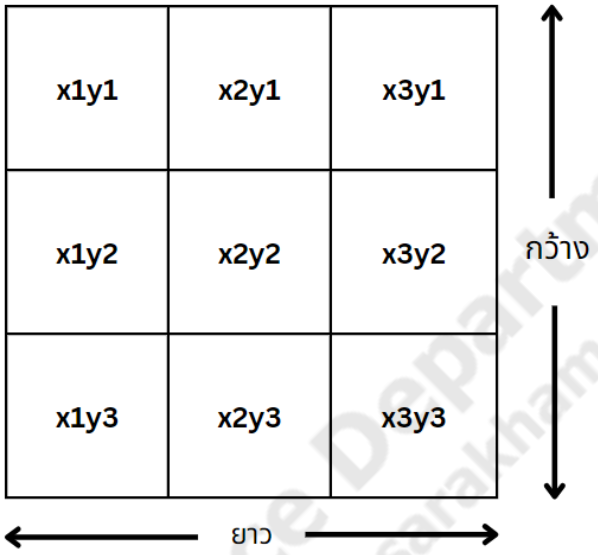
เมื่อคำนวณครบทุก gird แล้วจะได้ out put แสดงดังภาพประกอบที่ 3.60



ภาพประกอบที่ 3.60 ตัวอย่างผลลัพธ์ non-max suppression

3.5 การระบุตำแหน่งวัตถุ

ในการระบุตำแหน่งวัตถุ จะใช้จุดตัด 9 ช่องในการระบุ ซึ่งในการระบุ หลังจากที่ได้ค่า x และ y ของวัตถุ แล้วนำมาคำนวณกับความกว้าง และ ความยาว ของจอมือถือ ตามภาพประกอบที่ 3.60



ภาพประกอบที่ 3.61 การระบุตำแหน่งวัตถุ

ตำแหน่งใน จุดตัด 9 ช่อง

1. Upper Left = x1y1
2. Upper Middle = x2y1
3. Upper Right = x3y1
4. Middle Left = x1y2
5. Middle = x2y2
6. Middle Right = x3y2
7. Lower Left = x1y3
8. Lower Middle = x2y3
9. Lower Right = x3y3

xi = ความยาวของจอ / 3

yi = ความกว้างของจอ / 3

width = min(x * ratioX, ค่าความกว้างของพื้นที่หรือส่วนของหน้าจอที่กล้องสามารถแสดงผลภาพ)

height = min(y * ratioY, ค่าความสูงของพื้นที่หรือส่วนของหน้าจอที่กล้องสามารถแสดงผลภาพ)

min(a, b) นี้ใช้ในการเปรียบเทียบค่า a และ b และส่งคืนค่าที่น้อยกว่าระหว่างค่าทั้งสอง

ratioX และ ratioY คือ สัดส่วนที่ใช้ในการปรับขนาดของวัตถุให้อยู่ในขอบเขตที่กล้องสามารถแสดงผล

centerX = x + (width / 2) หาจุดตรงกลางของความกว้างของวัตถุ

centerY = y + (height / 2) หาจุดตรงกลางของความสูงของวัตถุ

localX = centerX / xi หาตำแหน่ง x ของวัตถุ

localY = centerY / yi หาตำแหน่ง y ของวัตถุ

จากนั้น นำ localX, localY มาคำนวณหาตำแหน่งจุดตรงกลางของวัตถุ

ตัวอย่าง หน้าจอมีขนาด 900 x 600 ตรวจจับวัตถุได้ตำแหน่ง x = 150 , y = 450 และวัตถุมีขนาด width = 245, height = 565

วิธีทำ xi = 600 / 3 = 200

yi = 900 / 3 = 300

centerX = 150 + (245 / 2) = 272.5

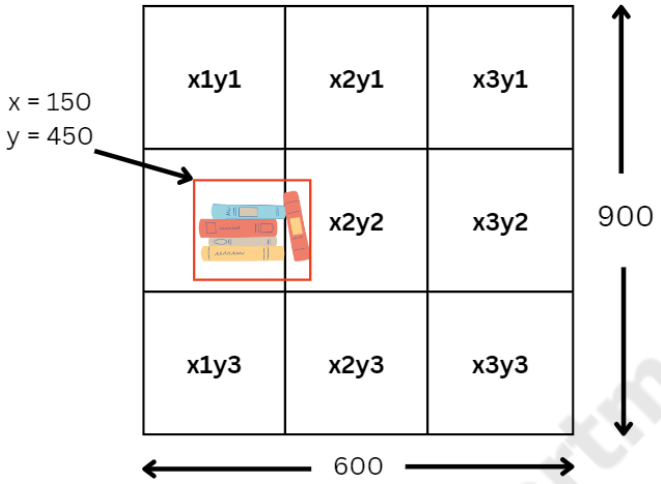
centerY = 450 + (565 / 2) = 732.5

หาตำแหน่งวัตถุ

localX = 272.5 / 200 = 1.3625 ≈ 1

localY = 732.5 / 300 = 2.4416 ≈ 2

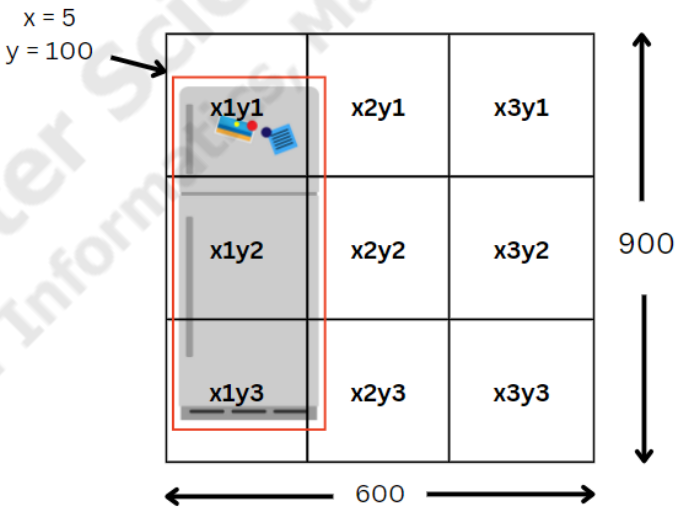
จะได้ตำแหน่งจุดตัด 9 ช่อง คือ x1y2 หรือคือ ตำแหน่ง Middle Left ตามภาพประกอบที่ 3.61



ภาพประกอบที่ 3.62 ตัวอย่างการคำนวณหาตำแหน่ง

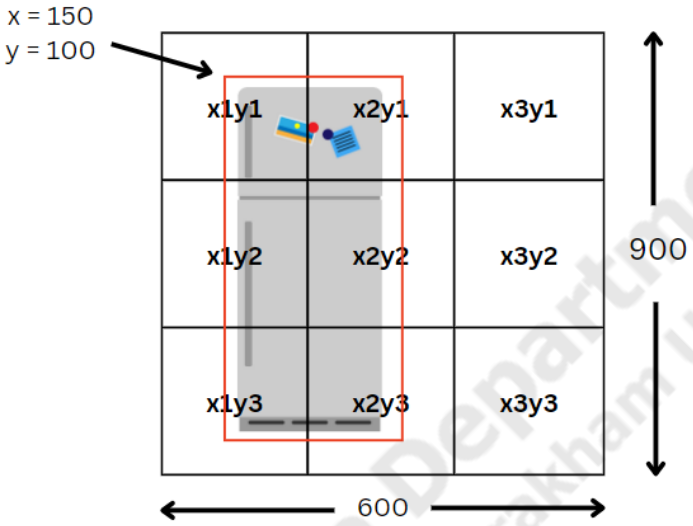
วัตถุอยู่มากกว่า 1 ช่องในแนวตั้ง

หากวัตถุอยู่ทั้ง 3 ช่อง x1y1, x1y2, x1y3 วัตถุจะระบุที่ช่อง x1y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ในช่อง x1y2 ตามภาพประกอบที่ 3.62



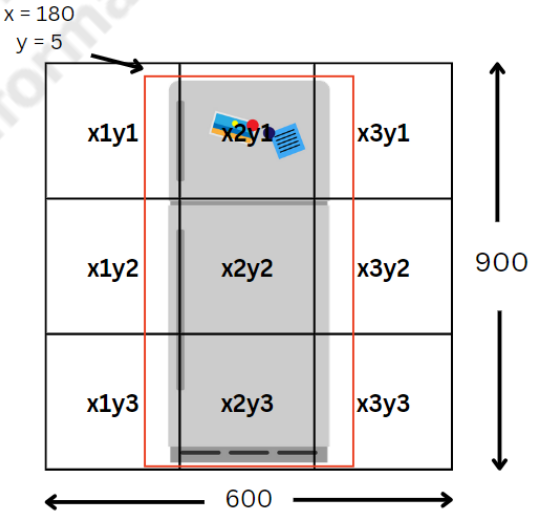
ภาพประกอบที่ 3.63 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 3 ช่อง ในแนวตั้ง

หากวัตถุอยู่ทั้ง 6 ช่อง x1y1, x2y1, x1y2, x2y2, x1y3, x2,3 วัตถุจะระบุที่ช่อง x1y2 หรือ x2y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ระหว่างช่อง x1y2 และ x2y2 ขึ้นอยู่กับว่าผู้ใช้งานขยับกล้องไปทางไหนมากว่าก็จะระบุในช่องนั้น ตามภาพประกอบที่ 3.63



ภาพประกอบที่ 3.64 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 6 ช่อง ในแนวตั้ง

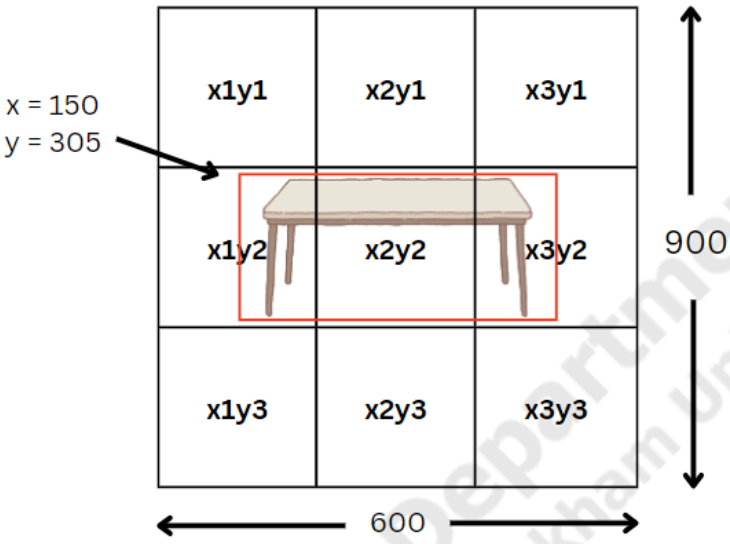
หากวัตถุอยู่ทั้ง 9 ช่อง x1y1, x2y1, x3y1, x1y2, x2y2, x3y2, x1y3, x2y3, x3y3 วัตถุจะระบุที่ช่อง x2y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ x2y2 ตามภาพประกอบที่ 3.64



ภาพประกอบที่ 3.65 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 9 ช่อง ในแนวตั้ง

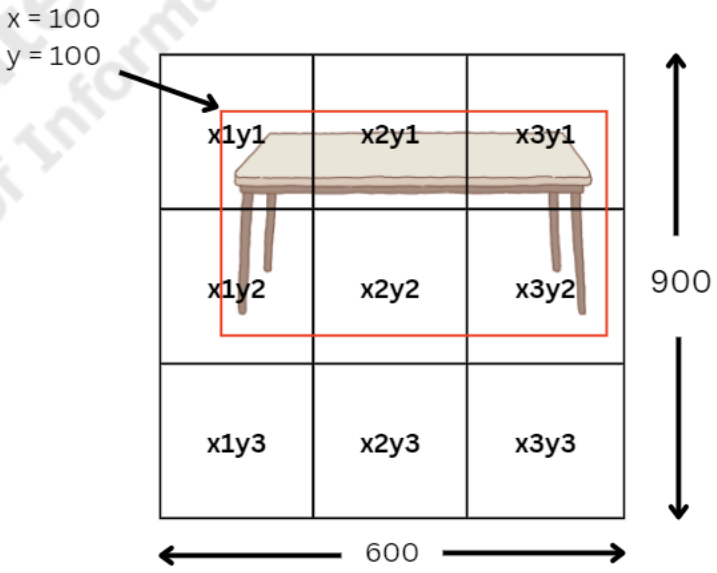
วัตถุอยู่มากกว่า 1 ช่องในแนวนอน

หากวัตถุอยู่ทั้ง 3 ช่อง x1y1, x1y2, x1y3 วัตถุจะระบุที่ช่อง x1y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ในช่อง x1y2 ตามภาพประกอบที่ 3.65



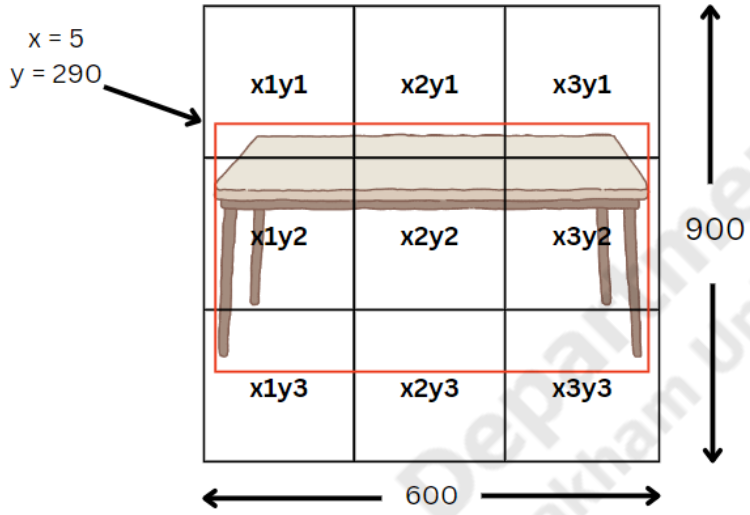
ภาพประกอบที่ 3.66 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 3 ช่อง ในแนวนอน

หากวัตถุอยู่ทั้ง 6 ช่อง x1y1, x2y1, x3y1, x1y2, x2y2, x3,y2 วัตถุจะระบุที่ช่อง x2y1 หรือ x2y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ระหว่างช่อง x2y1 และ x2y2 ขึ้นอยู่กับว่าผู้ใช้งานขยับกล้องไปทางไหนมากว่าก็จะระบุในช่องนั้น ตามภาพประกอบที่ 3.66



ภาพประกอบที่ 3.67 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 6 ช่อง ในแนวนอน

หากวัตถุอยู่ทั้ง 9 ช่อง x1y1, x2y1, x3y1, x1y2, x2y2, x3y2, x1y3, x2y3, x3y3 วัตถุจะระบุที่ช่อง x2y2 เนื่องจากจุดตรงกลางของวัตถุอยู่ x2y2 ตามภาพประกอบที่ 3.67



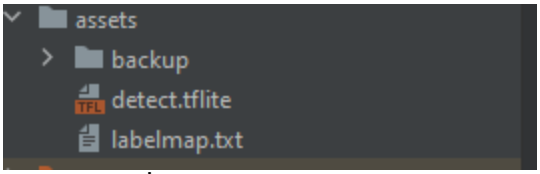
ภาพประกอบที่ 3.68 ตัวอย่างการคำนวณหาตำแหน่งหากวัตถุอยู่ทั้ง 9 ช่อง ในแนวนอน

3.6 ขั้นตอนในการพัฒนา Mobile application

ใช้โปแกรม Android studio ออกแบบแอปพลิเคชัน และแอปพลิเคชันจะใช้ในการตรวจจับสิ่งของภายในบ้านจากนั้นจะส่งข้อมูลกลับมาให้ผู้ใช้ผ่านการพูดชื่อสิ่งของที่ตรวจจับพบ รวมถึงการระบุหาสิ่งของโดยเฉพาะ จากการพูดชื่อสิ่งของ โดย model ที่นำเข้ามาในโปรแกรมจะถูก train ด้วย YOLOV5 และบันทึก model เป็นประเภท TensorFlow lite(.tflite) เพื่อที่จะนำเข้ามาใช้ในแอปพลิเคชันได้

3.6.1 นำ model มาใช้ในแอปพลิเคชัน

โดย model ที่จะนำเข้ามาต้องเป็นไฟล์ประเภท TensorFlow lite (.tflite) และสิ่งที่ต้องนำเข้าเพื่อใช้คู่กับ model คือ labelmap โดยจะใช้เป็นไฟล์ประเภท text(.txt) โดยจะสร้าง โฟลเดอร์ใน Project ของ Android studio ชื่อว่า assets เก็บสองไฟล์นี้ไว้ ตามภาพประกอบที่ 3.68



ภาพประกอบที่ 3.69 การเก็บไฟล์สำหรับ model

รายละเอียดในไฟล์ labels.txt จะเป็นชื่อคลาสของสิ่งของทั้ง 24 ชนิด ดังภาพประกอบที่ 3.70

```
1000Baht
100Baht
20Baht
500Baht
50Baht
Backpack
Book
Box
Chair
Charger cable
Earphone
Fan
Fork
Glass
Key
Knife
Mouth Mask
Outlet
Plate
Spoon
TV Remote
Table
Vase
Walking stick
```

ภาพประกอบที่ 3.70 รายละเอียดไฟล์ labelmap.txt

เพิ่มชื่อโฟลเดอร์ - assets/ ใน assets ที่อยู่ในไฟล์ pubspec.yaml เพื่อให้สามารถระบุถึงไฟล์ของ model และ labels ได้ ดังภาพประกอบที่ 3.71

```
74    assets:
75      - assets/
```

ภาพประกอบที่ 3.71 การเพิ่มชื่อโฟลเดอร์ใน pubspec

import library tflite ในไฟล์ lib/tflite/classifier.dart ซึ่งทำหน้าที่ดึงตัว model มาใช้งาน ดังภาพประกอบที่ 3.72

```
7   import 'package:tflite_flutter/tflite_flutter.dart';
```

ภาพประกอบที่ 3.72 import library tflite

ประกาศตัวแปร MODEL_FILE_NAME ที่เก็บค่าเป็นสตริง (String) และกำหนดค่าให้เป็น "detect.tflite" ซึ่งอาจเป็นชื่อของไฟล์ model และประกาศตัวแปร LABEL_FILE_NAME ที่เก็บค่าเป็นสตริง (String) และกำหนดค่าให้เป็น "labelmap.txt" ซึ่งอาจเป็นชื่อของไฟล์ label ที่นำมาใช้กับ model ในแอปพลิเคชัน ดังภาพประกอบที่ 3.73

```
20    static const String MODEL_FILE_NAME = "detect.tflite";
21    static const String LABEL_FILE_NAME = "labelmap.txt";
```

ภาพประกอบที่ 3.73 ประกาศตัวแปร MODEL_FILE_NAME และLABEL_FILE_NAME

สร้าง function สำหรับ load model เข้ามาเพื่อรอใช้งานดังภาพประกอบที่ 3.74

```
53    void loadModel({Interpreter? interpreter}) async {
54      try {
55        _interpreter = interpreter != null
56            ? interpreter
57            : await Interpreter.fromAsset(
58                MODEL_FILE_NAME,
59                options: InterpreterOptions()..threads = 4,
60              );
61
62        var outputTensors = _interpreter!.getOutputTensors();
63        _outputShapes = [];
64        _outputTypes = [];
65        outputTensors.forEach((tensor) {
66          _outputShapes!.add(tensor.shape);
67          _outputTypes!.add(tensor.type);
68        });
69      } catch (e) {
70        print("Error while creating interpreter: $e");
71      }
72    }
```

ภาพประกอบที่ 3.74 function สำหรับ load model

สร้าง function สำหรับ load label เข้ามาเพื่อรอใช้งานดังภาพประกอบที่ 3.75

```
75  void loadLabels({List<String>? labels}) async {
76    try {
77      _labels = labels != null
78          ? labels
79          : await FileUtil.loadLabels("assets/" + LABEL_FILE_NAME);
80    } catch (e) {
81      print("Error while loading labels: $e");
82    }
83  }
```

ภาพประกอบที่ 3.75 function สำหรับ load label

สร้าง function สำหรับ ปรับขนาดและประมวลผลรูปภาพที่จะนำเข้าไปใช้ในการทำนายผลด้วยโมเดล โดย INPUT_SIZE กำหนดไว้ที่ 640 ตามขนาด model ที่ train ดังภาพประกอบที่ 3.76

```
86    TensorImage getProcessedImage(TensorImage inputImage) {
87      padSize = max(inputImage.height, inputImage.width);
88      if (imageProcessor == null) {
89        imageProcessor = ImageProcessorBuilder()
90            .add(ResizeWithCropOrPadOp(padSize!, padSize!))
91            .add(ResizeOp(INPUT_SIZE, INPUT_SIZE, ResizeMethod.BILINEAR))
92            .build();
93      }
94      inputImage = imageProcessor!.process(inputImage);
95      return inputImage;
96    }
```

ภาพประกอบที่ 3.76 function ปรับขนาดและประมวลผลรูปภาพ

สร้าง function สำหรับทำนายผลจากภาพที่รับเข้ามา และคืนผลลัพธ์ที่ได้พร้อมสถิติเกี่ยวกับกระบวนการทำนาย รายละเอียดการทำงานดังนี้

บันทึกเวลาเริ่มต้นของกระบวนการทำนายผล เพื่อใช้ในการคำนวณระยะเวลาที่ใช้ในกระบวนการทำนายในภายหลังดังภาพประกอบที่ 3.77 และตรวจสอบว่า Interpreter ได้ถูกโหลดและสร้างขึ้นแล้วหรือไม่ ถ้ายังไม่ได้สร้างก็แสดงข้อความแจ้งเตือนและส่งคืน null ดังภาพประกอบที่ 3.78

```
101      var predictStartTime = DateTime.now().millisecondsSinceEpoch;
```

ภาพประกอบที่ 3.77 บันทึกเวลาเริ่มต้นของกระบวนการทำนายผล

```
103      if (_interpreter == null) {
104        print("Interpreter not initialized");
105        return null;
106      }
```

ภาพประกอบที่ 3.78 ตรวจสอบ Interpreter

แปลง image เป็น TensorImage ใช้ TensorImage.fromImage(image) ทำการเตรียมข้อมูลด้วยการเรียกใช้ฟังก์ชัน getProcessedImage เพื่อปรับขนาดและประมวลผลภาพ ดังภาพประกอบที่ 3.79

```
110      // Create TensorImage from image
111      TensorImage inputImage = TensorImage.fromImage(image);
112
113      // Pre-process TensorImage
114      inputImage = getProcessedImage(inputImage);
```

ภาพประกอบที่ 3.79 แปลง image เป็น TensorImage และ ปรับขนาดและประมวลผลภาพ

สร้าง TensorBuffer สำหรับเก็บผลลัพธ์ของ output tensors ตามรูปแบบที่กำหนดใน _outputShapes โดยตัวแปรที่ใช้เก็บผลลัพธ์ที่ได้จากการทำนายวัตถุบนภาพ เช่น พิกัดตำแหน่งของวัตถุ (locations), คลาสของวัตถุ (classes), ค่าคะแนนความมั่นใจในการทำนาย (scores), และจำนวนวัตถุที่ตรวจพบ (numLocations) ดังภาพประกอบที่ 3.80

```
119     TensorBuffer outputLocations = TensorBufferFloat(_outputShapes![0]);
120     TensorBuffer outputClasses = TensorBufferFloat(_outputShapes![1]);
121     TensorBuffer outputScores = TensorBufferFloat(_outputShapes![2]);
122     TensorBuffer numLocations = TensorBufferFloat(_outputShapes![3]);
```

ภาพประกอบที่ 3.80 สร้าง TensorBuffer สำหรับเก็บผลลัพธ์ของ output tensors

สร้างรายการ inputs และแผนที่ outputs สำหรับเรียกใช้ runForMultipleInputs เพื่อทำนายผล ดังภาพประกอบที่ 3.81 จากนั้นทำนายผลจากโมเดล TensorFlow Lite และเก็บผลลัพธ์ใน outputs ดังภาพประกอบที่ 3.82 และคำนวณจำนวนผลลัพธ์ที่จะแสดงผล และกำหนดค่า labelOffset ที่ใช้ในการค้นหาชื่อป้าย หรือ label ดังภาพประกอบที่ 3.83

```
126     List<Object> inputs = [inputImage.buffer];
127
128     // Outputs map
129     Map<int, Object> outputs = {
130       0: outputLocations.buffer,
131       1: outputClasses.buffer,
132       2: outputScores.buffer,
133       3: numLocations.buffer,
134     };
```

ภาพประกอบที่ 3.81 สร้างรายการ inputs และแผนที่ outputs

```
139     _interpreter!.runForMultipleInputs(inputs, outputs);
```

ภาพประกอบที่ 3.82 ทำนายผลจากโมเดล และเก็บผลลัพธ์

```
145     int resultsCount = min(NUM_RESULTS, numLocations.getIntValue(0));
146
147     int labelOffset = 1;
```

ภาพประกอบที่ 3.83 คำนวณจำนวนผลลัพธ์ และกำหนดค่า labelOffset

แปลงข้อมูลจาก TensorBuffer ที่ได้จากการทำนายผลเป็น List ของพื้นที่ของวัตถุ (Rect) ที่

ตรวจพบในภาพ เพื่อที่จะนำข้อมูลเหล่านี้ไปใช้ในขั้นตอนต่อไปเช่นการวาดกรอบสี่เหลี่ยมรอบวัตถุที่ตรวจพบบนภาพหรือแสดงผลผลลัพธ์ให้ผู้ใช้รับรู้ ดังภาพประกอบที่ 3.84

```
List<Rect> locations = BoundingBoxUtils.convert(
  tensor: outputLocations,
  valueIndex: [1, 0, 3, 2],
  boundingBoxAxis: 2,
  boundingBoxType: BoundingBoxType.BOUNDARIES,
  coordinateType: CoordinateType.RATIO,
  height: INPUT_SIZE,
  width: INPUT_SIZE,
);
```

ภาพประกอบที่ 3.84 แปลง TensorBuffer เป็น List

วนลูปเพื่อค้นหาผลลัพธ์ที่ผ่านเงื่อนไขค่าคะแนน (score) มากกว่าค่า THRESHOLD(THRESHOLD เซทค่าไว้ที่ 0.5) และสร้างอ็อบเจ็กต์ Recognition จากผลลัพธ์นั้นแล้วเพิ่มลงในรายการ recognitions ซึ่งจะเก็บวัตถุที่ถูกทำนายและผ่านเกณฑ์ค่าคะแนนที่กำหนดไว้ ดังภาพประกอบที่ 3.85

```
List<Recognition> recognitions = [];

for (int i = 0; i < resultsCount; i++) {
  // Prediction score
  var score = outputScores.getDoubleValue(i);

  // Label string
  var labelIndex = outputClasses.getIntValue(i) + labelOffset;
  var label = _labels!.elementAt(labelIndex);

  if (score > THRESHOLD) {
    // inverse of rect
    // [locations] corresponds to the image size 300 X 300
    // inverseTransformRect transforms it our [inputImage]
    Rect transformedRect = imageProcessor!
        .inverseTransformRect(locations[i], image.height, image.width);

    recognitions.add(
      Recognition(i, label, score, transformedRect),
    );
  }
}
```

ภาพประกอบที่ 3.85 ค้นหาผลลัพธ์ และเพิ่มลงในรายการ recognitions

3.6.2 Text to speech

สร้าง class TextToSpeech ที่มีหน้าที่เรียกใช้งานตัวแปรและ method จาก Flutter TTS (Text-to-Speech) เพื่อให้แอปพลิเคชันสามารถแสดงข้อความออกเป็นเสียงพูดได้ โดย method initTTS สำหรับกำหนดภาษาที่ใช้ในการออกเสียง โดยกำหนดให้ใช้ภาษาอังกฤษสหรัฐ “en-US” และ method speak สำหรับเริ่มการออกเสียงข้อความที่ส่งเข้ามาในรูปแบบของข้อความ ดังภาพประกอบที่ 3.86

```dart
import 'package:flutter_tts/flutter_tts.dart';

class TextToSpeech {
  static FlutterTts tts = FlutterTts();

  static initTTS() {
    tts.setLanguage('en-US');
  }

  static speak(String text) async{

    await tts.awaitSpeakCompletion(true);


    tts.speak(text);
  }
}
```

ภาพประกอบที่ 3.86 class TextToSpeech

3.6.3 Speech to text

Speech to text จะนำมาใช้กับหน้าโหมดการค้นหา(Find) ในการรับเสียงพูดมาแสดงภายใน mobile application

สร้าง Map ที่เก็บคำศัพท์ (words) และข้อความ (text) ของสิ่งของใช้ภายในบ้าน เพื่อใช้ในการแปลงคำศัพท์ที่ถูกตรวจจับจากการพูดเป็นข้อความที่ใช้ในการออกเสียงในส่วน text to speech ดังภาพประกอบที่ 3.87

```
Map<String, String> wordMap = {
  "person": "Person",
  "bicycle": "Bicycle",
  "car": "Car",
  "motorcycle": "Motorcycle",
  "airplane": "Airplane",
  "bus": "Bus",
  "train": "Train",
  "truck": "Truck",
  "boat": "Boat",
  "traffic light": "Traffic Light",
  "fire hydrant": "Fire Hydrant",
  "stop sign": "Stop Sign",
  "parking meter": "Parking Meter",
  "bench": "Bench",
  "bird": "Bird",
  "cat": "Cat",
  "dog": "Dog",
  "horse": "Horse",
  "sheep": "Sheep",
  "cow": "Cow",
  "elephant": "Elephant",
  "bear": "Bear",
  "zebra": "Zebra",
  "giraffe": "Giraffe",
  "backpack": "Backpack",
  "umbrella": "Umbrella",
  "handbag": "Handbag",
  "tie": "Tie",
  "suitcase": "Suitcase",
  "frisbee": "Frisbee",
  "skis": "Skis",
  "snowboard": "Snowboard",
  "sports ball": "Sports Ball",
  "kite": "Kite",
  "baseball bat": "Baseball Bat",
  "baseball glove": "Baseball Glove",
  "skateboard": "Skateboard",
  "surfboard": "Surfboard",
```

ภาพประกอบที่ 3.87 สร้าง Map เก็บคำศัพท์ในการทำ text-to-speech

สร้าง instance ของ SpeechToText และตั้งค่าเริ่มต้นบนปุ่ม speech to text เป็น text "Speech" และ ตัว listening = false คือตัวแปรที่บอกว่ากำลังฟังเสียงหรือไม่ DTText ไว้เก็บชื่อ label สิ่งของที่ตรวจจับได้และจะนำไป text to speech และ oldDT คือชื่อ label ที่ได้มาจาก DTText ไว้เช็คว่าเป็น object เก่าหรือไม่ ดังภาพประกอบที่ 3.88

```
31    SpeechToText speechToText = SpeechToText();
32    var text = "Speech";
33    var listening = false;
34    var DTText;
35    var oldDT;
```

ภาพประกอบที่ 3.88 สร้าง instance ของ SpeechToText และตัวแปร

สร้าง State ใน StatefulWidget และเรียกใช้งาน TextToSpeech เพื่อพูดโหมด "FIND" เพื่อให้กูใช้งานรับรู้ ดังภาพประกอบที่ 3.89

```
186   void statsCallback(Stats stats) {
187     setState(() {
188       this.stats = stats;
189     });
190   }
```

ภาพประกอบที่ 3.89 สร้าง State ใน StatefulWidget และเรียกใช้งาน TextToSpeech

สร้างปุ่มที่เรียกใช้งานการจับเสียงพูด Speech to Text และเรียกใช้งานเมธอด initialize เพื่อเตรียมการทำงาน กำหนดค่า text ใหม่หากตัวตรวจจับเจอวัตถุบนปุ่มก็จะกำหนดบนปุ่มว่า “Speech” เปลี่ยนค่า listening เป็น true เพื่อบอกว่ากำลังฟังเสียง text = result.recognizedWords ใช้กำหนดค่าข้อความจากเสียงที่ถูกตรวจจับ จากนั้นนำไปแยกคำออกมาเป็น List ดังภาพประกอบที่ 3.90

```
child: GestureDetector(
  onTapDown: (details) async {
    if (!listening) {
      var available = await speechToText.initialize();
      if (available) {

        text = "Speech";//set text default
        setState(() {
          listening = true;
          speechToText.listen(
            onResult: (result) {
              setState(() {
                text = result.recognizedWords;
                List<String> words = text.split(" ");
                print(words);
```

ภาพประกอบที่ 3.90 สร้างปุ่มที่เรียกใช้งานการจับเสียงพูด Speech to Text

ตรวจสอบคำที่ถูกจับเป็นเสียง ว่ามีคำใดในรายการคำที่ตรงกับค่าที่กำหนดใน wordMap หรือไม่ ถ้ามีคำที่ตรงกันจะทำการกำหนดค่าของ text เป็นคำที่จับได้ และสุดท้ายจะเรียกใช้งาน TextToSpeech.speak(text) เพื่อให้พูดข้อความที่เป็นคำที่ถูกจับมา ดังภาพประกอบที่ 3.91

```
if (text != null) {
  for (String word in words) {
    if (wordMap.containsKey(word)) {
      text = wordMap[word]!;
      TextToSpeech.speak(text);
      break;
    }
  }
}
```

ภาพประกอบที่ 3.91 ตรวจสอบคำที่ถูกจับเป็นเสียง

เมื่อปล่อยกดปุ่ม กำหนดค่า listening เป็น false เพื่อหยุดการฟังเสียง และหยุดการทำงาน Speech to Text ดังภาพประกอบที่ 3.92

```
onTapUp: (details) {
  setState(() {
    listening = false;
  });
  speechToText.stop();
},
```

ภาพประกอบที่ 3.92 หยุดการรับเสียง และหยุดการทำงาน Speech to Text

เปรียบเทียบข้อมูลระบุวัตถุใหม่กับข้อมูลระบุวัตถุเดิมที่ตรวจพบเมื่อครั้งก่อน และถ้าวัตถุระบุวัตถุใหม่ไม่เหมือนกับวัตถุระบุวัตถุเดิมและไม่เป็นค่าว่าง ก็จะทำการอัพเดทค่าข้อมูลระบุวัตถุเดิมและส่งข้อความเสียงที่อ่านออกมาจากคำวัตถุใหม่ที่ตรวจจับได้ให้กับผู้ใช้ ดังภาพประกอบที่ 3.93

```
for (var recognition in results) {
  DTText = recognition.label ;
  if (DTText != oldDT && DTText != null) {
    TextToSpeech.speak(recognition.label);
    oldDT = DTText;
  }
}
}
```

ภาพประกอบที่ 3.93 เปรียบเทียบข้อมูลวัตถุใหม่ กับข้อมูลวัตถุเดิม

3.6.4 การคำนวณหาตำแหน่งวัตถุ

สร้างคลาส Recognition ที่ใช้เก็บข้อมูลผลลัพธ์จากการรู้จำวัตถุจากโมเดล มีตัวแปรส่วนตัว (_id, _label, _score, _location) เพื่อเก็บข้อมูลเกี่ยวกับผลลัพธ์การรู้จำวัตถุ ดังภาพประกอบที่ 3.94 Method renderLocation หน้าที่คำนวณและคืนค่า Rect ที่แทนพิกัดของกล่องคำนวณที่จะถูกแสดงบนหน้าจอโดยคำนึงถึงอัตราส่วนของการแสดงผลและขนาดตัวอย่างภาพที่ถูกแสดงบนหน้าจอ ดังภาพประกอบที่ 3.95 และ toString ถูกโอเวอร์ไรด์ในคลาส Recognition มีการทำงานเพื่อสร้างและคืนค่าสตริง (String) ที่เป็นข้อความที่แสดงข้อมูลของอ็อบเจ็กต์ ดังภาพประกอบที่ 3.96

```
class Recognition {

  int _id;

  String _label;

  double _score;

  Rect? _location;

  Recognition(this._id, this._label, this._score, [this._location]);

  int get id => _id;

  String get label => _label;

  double get score => _score;

  Rect get location => _location!;
```

ภาพประกอบที่ 3.94 ประกาศตัวแปรภายในคลาส

1. _id: คือตัวแปรเก็บลำดับของผลลัพธ์
2. _label: คือตัวแปรเก็บป้ายชื่อของวัตถุที่ตรวจพบ
3. _score: คือตัวแปรเก็บความมั่นใจในการตรวจพบวัตถุ (ค่าระหว่าง 0.0 ถึง 1.0)
4. _location: คือตัวแปรเก็บข้อมูลพิกัดของกล่องคำนวณที่มีวัตถุอยู่
5. Recognition(this._id, this._label, this._score, [this._location]); คือคอนสตรักเตอร์ของคลาส Recognition ที่ใช้สำหรับกำหนดค่าให้กับ _id, _label, _score, และ _location โดยมี _location เป็นอาร์กิวเมนต์ทางเลือก.
6. int get id => _id, String get label => _label, double get score => _score, Rect get location => _location!: เป็นเมธอดเก็บค่า (getter) ที่ใช้สำหรับเข้าถึงค่า _id, _label, _score, และ _location ตามลำดับ

```
Rect get renderLocation {

  double ratioX = CameraViewSingleton.ratio;
  double ratioY = ratioX;

  double transLeft = max(0.1, location.left * ratioX);
  double transTop = max(0.1, location.top * ratioY);
  double transWidth = min(
      location.width * ratioX, CameraViewSingleton.actualPreviewSize.width);
  double transHeight = min(
      location.height * ratioY, CameraViewSingleton.actualPreviewSize.height);

  Rect transformedRect =
      Rect.fromLTWH(transLeft, transTop, transWidth, transHeight);
  return transformedRect;
}
```

ภาพประกอบที่ 3.95 Method renderLocation

1. double ratioX = CameraViewSingleton.ratio นี้คือการกำหนดค่าตัวแปร ratioX เท่ากับค่า ratio ที่ถูกเรียกจาก CameraViewSingleton ซึ่งอาจเป็นอัตราส่วนระหว่างความกว้างของหน้าจอแอปพลิเคชันและความกว้างของภาพต้นฉบับ

2. double ratioY = ratioX นี่คือการกำหนดค่า ratioY เท่ากับ ratioX ซึ่งแสดงถึงความสัมพันธ์ของอัตราส่วนระหว่างความกว้างและความสูงของภาพที่ถูกแสดง

3. double transLeft = max(0.1, location.left * ratioX) คำนวณค่า transLeft โดยการนำค่า location.left (พิกัดซ้ายของกล่องคำนวณ) คูณกับ ratioX และใช้ max เพื่อรับประกันว่าค่าไม่ต่ำกว่า 0.1 ซึ่งอาจจะเป็นการปรับแต่งค่าตำแหน่งในกรณีที่พิกัดน้อยเกินไป

4. double transTop = max(0.1, location.top * ratioY) คำนวณค่า transTop โดยการนำค่า location.top (พิกัดบนของกล่องคำนวณ) คูณกับ ratioY และใช้ max เพื่อรับประกันว่าค่าไม่ต่ำกว่า 0.1

5. double transWidth = min(location.width * ratioX, CameraViewSingleton.actualPreviewSize.width) คำนวณค่า transWidth โดยการนำค่า location.width (ความกว้างของกล่องคำนวณ) คูณกับ ratioX และใช้ min เพื่อรับประกันว่าค่าไม่เกินความกว้างของภาพตัวอย่างที่ถูกแสดงบนหน้าจอ

6. double transHeight = min(location.height * ratioY, CameraViewSingleton.actualPreviewSize.height) คำนวณค่า transHeight โดยการนำค่า location.height (ความสูงของกล่องคำนวณ) คูณกับ ratioY และใช้ min เพื่อรับประกันว่าค่าไม่เกินความสูงของภาพตัวอย่างที่ถูกแสดงบนหน้าจอ

7. Rect transformedRect = Rect.fromLTWH(transLeft, transTop, transWidth, transHeight) สร้าง Rect ใหม่โดยใช้ค่า transLeft, transTop, transWidth, และ transHeight ที่คำนวณมาเพื่อสร้างพิกัดของกล่องคำนวณที่จะถูกแสดงบนหน้าจอ

8. return transformedRect คืนค่า transformedRect ที่เป็น Rect ที่ถูกคำนวณแล้วเพื่อแสดงกล่องคำนวณในรูปแบบที่ถูกแปลงเหมาะสมสำหรับการแสดงบนหน้าจอ

```
  @override
  String toString() {
    return 'Recognition(id: $id, label: $label, score: $score, location: $location)';

  }
}
```

ภาพประกอบที่ 3.96 override Method

toString ถูกโอเวอร์ไรด์ในคลาส Recognition มีการทำงานเพื่อสร้างและคืนค่าสตริง (String) ที่เป็นข้อความที่แสดงข้อมูลของอ็อบเจ็กต์ Recognition ในรูปแบบที่ถูกกำหนด ในคำสั่งนี้เมธอด toString() สร้างสตริงที่มีรูปแบบเป็นข้อความ โดยมีข้อมูลจากตัวแปร _id, _label, _score, และ _location ของอ็อบเจ็กต์ Recognition ที่ถูกเรียกใช. การใช้ $id, $label, $score, และ $location คือการแทนค่าของตัวแปรดังกล่าวในสตริง ดังนั้น เมื่อเรียกใช้ toString() บนอ็อบเจ็กต์ Recognition จะคืนสตริงที่แสดงข้อมูลของอ็บเจ็กต์นั้นในรูปแบบเช่น "Recognition(id: 1, label: 'object', score: 0.85, location: Rect.fromLTRB(10.0, 20.0, 30.0, 40.0))"

ประกาศตัวแปร checkText ใช้เมธอด toLowerCase() เพื่อแปลง speakText ที่ได้จากการรับข้อมูลทางเสียงจากผู้ใช้งาน ให้เป็นตัวพิมพ์เล็กทั้งหมดและเก็บผลลัพธ์ลงในตัวแปร checkText และประกาศตัวแปร recognitionLabel มีค่าเป็นอ็อบเจ็กต์ recognition ซึ่งมีคุณสมบัติ _label เป็นข้อความที่เราต้องการที่จะตรวจสอบหรือเปรียบเทียบ เราใช้เมธอด toLowerCase() เพื่อแปลง _label ใน recognition เป็นตัวพิมพ์เล็กทั้งหมดและเก็บผลลัพธ์ลงในตัวแปร recognitionLabel เพื่อที่จะได้นำมาเช็คกับ checkText ดังภาพประกอบที่ 3.97

```
String checkText = speakText.toLowerCase();
String recognitionLabel = recognition.label.toLowerCase();
```

ภาพประกอบที่ 3.97 ประกาศตัวแปร checkText และ recognitionLabel

กำหนดและคำนวณตัวแปรต่างๆ ที่ใช้ในการคำนวณระบุตำแหน่ง ดังภาพประกอบที่ 3.98

```
double Cx = x + transWidth / 2 ;
double Cy = y + transHeight / 2  ;

double xi = CameraViewSingleton.actualPreviewSize.width/3;
double yi = CameraViewSingleton.actualPreviewSize.height/3;

double localX = Cx / xi;
double localY = Cy / yi;

int X = localX.ceil();
int Y = localY.ceil();

X = X.clamp(1, 3);
Y = Y.clamp(1, 3);
```

ภาพประกอบที่ 3.98 กำหนดค่าต่างๆที่ใช้ในการคำนวณระบุตำแหน่ง

1. Cx คือ ค่าจุดตรงกลางของ x หาได้โดยค่า x+ ค่าความกว้างของวัตถุ(transWidth)/ 2
2. Cy คือ ค่าจุดตรงกลางของ y หาได้โดยค่า y+ ค่าความสูงของวัตถุ(transHetght) / 2
3. xi คือ ค่าความกว้างในแต่ละช่องของจุดตัด 9 ช่อง หาได้โดย ค่าส่วนความกว้างของหน้าจอกล้อง(CameraViewSingleton.actualPreviewSize.width) / 3
4. yi คือ ค่าความสูงในแต่ละช่องของจุดตัด 9 ช่อง หาได้โดย ค่าส่วนความสูงของหน้าจอกล้อง(CameraViewSingleton.actualPreviewSize.height) / 3
5. localX คือ ค่าตำแหน่ง X ในจุดตัด 9 ช่อง หาได้โดย Cx / xi
6. localY คือ ค่าตำแหน่ง Y ในจุดตัด 9 ช่อง หาได้โดย Cy / yi
7. X คือ ค่า localX ที่ทำการคำนวณที่ถูกแปลงเป็นจำนวนเต็ม โดยใช้ ceil() เพื่อปัดขึ้น
8. Y คือ ค่า localY ที่ทำการคำนวณที่ถูกแปลงเป็นจำนวนเต็ม โดยใช้ ceil() เพื่อปัดขึ้น

จากที่ได้ค่า X, Y นำค่าทั้งสองมากำหนดให้อยู่ในช่วงระหว่าง 1 ถึง 3 โดยใช้ clamp() ซึ่งคือการรักษาค่าให้อยู่ในขอบเขตที่กำหนด เพื่อป้องกันค่าที่อาจเกินขอบเขต จากนั้นนำมาเช็คในเงื่อนไขเพื่อหาตำแหน่ง ดังภาพประกอบที่ 3.99

```
if (checkText != null && recognitionLabel != null) {

  if (recognitionLabel.toLowerCase().contains(checkText.toLowerCase())) {

    if (X == 1 && Y == 1) {
      TextToSpeech.speak("${recognition.label} is Upper Left");
    }
    else if (X == 2 && Y == 1) {
      TextToSpeech.speak("${recognition.label} is Upper Middle");
    }
    else if (X == 3 && Y == 1) {
      TextToSpeech.speak("${recognition.label} is Upper Right");
    }
    else if (X == 1 && Y == 2) {
      TextToSpeech.speak("${recognition.label} is Middle Left");
    }
    else if (X == 2 && Y == 2) {
      TextToSpeech.speak("${recognition.label} is Middle");
    }
    else if (X == 3 && Y == 2) {
      TextToSpeech.speak("${recognition.label} is Middle Right");
    }
    else if (X == 1 && Y == 3) {
      TextToSpeech.speak("${recognition.label} is Lower Left");
    }
    else if (X == 2 && Y == 3) {
      TextToSpeech.speak("${recognition.label} is Lower Middle");
    }
    else if (X == 3 && Y == 3) {
      TextToSpeech.speak("${recognition.label} is Lower Right");
    }
  }
}
```

ภาพประกอบที่ 3.99 นำ checkText และ recognitionLabel มาตรวจสอบ

โดยเช็คว่า checkText และ recognitionLabel ไม่เป็นค่า null ก่อนที่จะดำเนินการ จากนั้นตรวจสอบว่า recognitionLabel มีข้อความที่อยู่ใน checkText หากใช่ ก็จะดำเนินการเสียงของข้อความ จากนั้น ตรวจสอบค่า X และ Y หากตรงตามที่กำหนด จะพูดระบุตำแหน่งนั้นๆ

3.6.5 หน้าหลัก Application

แอปพลิเคชันมีทั้งหมด 2 โหมด โหมด Explore และ โหมด Find โดยใช้ PageView เพื่อเปลี่ยนหน้าแสดงผลได้ และมีแถบเมนูด้านล่าง (BottomNavigationBar) เพื่อเปิดเลือกการแสดงผลระหว่าง "EXPLORE" และ "FIND" โดยการทำงานเหล่านี้อยู่ภายในไฟล์ main ดังภาพประกอบที่ 3.100 และ ภาพประกอบที่ 3.101

```dart
class _MyAppState extends State<MyApp> {
  PageController _pageController = PageController();
  int _selectedIndex = 0;

  final List<Widget> _views = [
    HomeView(),
    FineView(),
    // NewSpeech(),
  ];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
      _pageController.animateToPage(
        index,
        duration: Duration(milliseconds: 500), // Set the duration for the animation
        curve: Curves.easeInOut, // Set the animation curve
      );
    });
  }

  @override
  void dispose() {
    _pageController.dispose(); // Dispose of the PageController
    super.dispose();
  }
```

ภาพประกอบที่ 3.100 โค้ดภายในหน้าหลัก Application

1. PageController _pageController = PageController() สร้าง PageController เพื่อควบคุมการเปลี่ยนหน้าของ PageView

2. int _selectedIndex = 0 กำหนดตัวแปร _selectedIndex เพื่อเก็บดัชนีของหน้าที่ถูกเลือกใน BottomNavigationBar โดยกำหนดค่าเริ่มต้นให้เป็น 0 (หน้าแรก)

3. final List<Widget> _views = [...] สร้างรายการของ widgets ที่จะแสดงผลใน PageView โดยให้ _views เป็นรายการของ HomeView() และ FineView()

4. _onItemTapped(int index) { ... } เมื่อผู้ใช้คลิกที่แถบเมนูด้านล่าง BottomNavigationBar จะทำการเรียกเมธอด _onItemTapped โดยกำหนด index ของหน้าที่ถูกคลิก เมธอดนี้จะทำการอัปเดตค่า _selectedIndex และใช้ animateToPage เพื่อทำการเปลี่ยนหน้าของ PageView ไปที่หน้าที่ถูกเลือก

5. animateToPage ใช้ในการสไลด์หน้าจอไปยังหน้าที่เลือก โดยสามารถกำหนดระยะเวลา (duration) ในการสไลด์ และ curve ในการแสดงผลอนิเมชัน (ในตัวอย่างนี้ใช้ Curves.easeInOut) ซึ่งทำให้การสไลด์หน้าจอดูนุ่มนวลและมีความเรียบร้อย.

6. dispose() { ... }: เมื่อ MyApp ถูกทำลาย (dispose) จะทำการลบ PageController โดยเรียก dispose เพื่อป้องกันการรักษาทรัพยากรที่ไม่จำเป็น

```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    debugShowCheckedModeBanner: false,
    title: 'Object Detection TFLite',
    theme: ThemeData(
      primarySwatch: Colors.blue,
      visualDensity: VisualDensity.adaptivePlatformDensity,
    ),  // ThemeData
    home: Scaffold(
      body: PageView(
        controller: _pageController,
        children: _views,
        onPageChanged: (index) {
          setState(() {
            _selectedIndex = index;
          });
        },
      ),  // PageView
      bottomNavigationBar: BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'EXPLORE',
          ),  // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: 'FINE',
          ),  // BottomNavigationBarItem
        ],  // <BottomNavigationBarItem>[]
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
      ),  // BottomNavigationBar
    ),  // Scaffold
  );  // MaterialApp
}
}
```

ภาพประกอบที่ 3.101 โค้ดภายในหน้าหลัก Application

7. Widget build(BuildContext context) { ... } เป็นเมธอดที่แสดงถึงการสร้างและคืนค่า Widget ที่จะแสดงผลบนหน้าแอปพลิเคชัน โดยมีพารามิเตอร์ BuildContext context ที่ใช้ในการสร้าง Widget และใช้ในตัวแปร context เมื่อจำเป็น

8. return MaterialApp(...) ในบรรทัดนี้เริ่มต้นการสร้างแอปพลิเคชันด้วย MaterialApp ซึ่งเป็น Widget หลักของแอปพลิเคชัน

9. debugShowCheckedModeBanner: false กำหนดเป็น false เพื่อปิดการแสดงแบนเนอร์สีแดง (Debug Banner)

10. title: 'Object Detection TFLite' กำหนด title ของแอปพลิเคชันให้เป็น "Object Detection TFLite" ซึ่งจะแสดงที่แถบหัวของแอปพลิเคชัน

11. theme: ThemeData(...) กำหนดธีม (theme) ของแอปพลิเคชัน ในที่นี้ใช้ primarySwatch ในการกำหนดสีหลักของแอปพลิเคชันเป็นสีน้ำเงิน (blue) และ visualDensity เป็น VisualDensity.adaptivePlatformDensity เพื่อให้แอปพลิเคชันปรับความหนาและระยะห่างของข้อความและ Widget ต่าง ๆ ให้เหมาะกับแพลตฟอร์มที่ใช้อยู่

12. home: Scaffold(...) กำหนดหน้าแรกของแอปพลิเคชันให้เป็น Scaffold ซึ่งเป็นหน้าจอหลักที่มีส่วนประกอบต่าง ๆ ภายใน

13. body: PageView(...) กำหนด PageView ให้เป็นส่วนของ body ของ Scaffold ซึ่งจะแสดงผลหน้าจอแต่ละหน้าของแอปพลิเคชัน โดยใช้ _pageController เป็นคอนโทรลเลอร์ของ PageView และ children ในการระบุ Widget ที่จะแสดงผลบนแต่ละหน้า

14. onPageChanged: (index) { ... } กำหนด callback ที่จะทำงานเมื่อหน้าใน PageView ถูกเปลี่ยน ในกรณีนี้จะทำการอัปเดตค่า _selectedIndex เพื่อให้เก็บค่าดัชนีของหน้าที่ถูกเลือก

15. bottomNavigationBar: BottomNavigationBar(...): กำหนด BottomNavigationBar เป็นส่วนล่างของ Scaffold ซึ่งเป็นเมนูด้านล่างที่ให้ผู้ใช้เลือกหน้า. items ใช้ในการระบุรายการแท็บในเมนูด้านล่าง โดยในที่นี้มี "EXPLORE" และ "FIND" ซึ่งมีไอคอนแสดงผลด้วย icon และ label เป็นข้อความ

16. currentIndex: _selectedIndex,: กำหนด currentIndex ให้เป็นค่าของ _selectedIndex ซึ่งเป็นดัชนีของหน้าที่ถูกเลือกใน BottomNavigationBar

17. onTap: _onItemTapped,: กำหนด callback ที่จะทำงานเมื่อผู้ใช้คลิกที่แท็บใน BottomNavigationBar โดยเรียกใช้ _onItemTapped ซึ่งจะทำการอัปเดตหน้าแสดงผลใน PageView ให้เป็นหน้าที่ถูกเลือก

3.6.6 การใช้งาน API ภายนอก

1. camera: ^0.9.4+5: แพ็คเกจสำหรับการเข้าถึงกล้องถ่ายรูปบนอุปกรณ์ ช่วยให้คุณสามารถเข้าถึงและควบคุมกล้องได้ง่าย

2. cupertino_icons: ^1.0.2: แพ็คเกจสำหรับไอคอนสไตล์ iOS Cupertino. ใช้สำหรับการแสดงผลไอคอนในแอปพลิเคชัน

3. tflite_flutter: ^0.9.0: แพ็คเกจสำหรับการใช้ TensorFlow Lite ใน Flutter สำหรับการตรวจจับวัตถุและการทำนาย

4. tflite_flutter_helper: ^0.3.1: ช่วยในการใช้ TensorFlow Lite ใน Flutter โดยมีเครื่องมือและคลาสที่มีอยู่เพื่อสนับสนุนการใช้งาน TensorFlow Lite

5. image: ^3.0.2: แพ็คเกจสำหรับการจัดการและประมวลผลภาพ รวมถึงการเข้าถึงและการจัดการภาพแบบพื้นฐาน

6. image: ^3.0.2: แพ็คเกจสำหรับการจัดการและประมวลผลภาพ รวมถึงการเข้าถึงและการจัดการภาพแบบพื้นฐาน

7. flutter_easyloading: ^3.0.5: แพ็คเกจสำหรับแสดงหน้าต่างการโหลดหรือการดำเนินการอื่น ๆ อย่างง่ายในแอปพลิเคชัน

8. avatar_glow: ^2.0.2: แพ็คเกจสำหรับการสร้างเอฟเฟกต์การกระพริบสว่างๆ รอบวัตถุหรือตัวอักษร

9. speech_to_text: ^6.1.1: แพ็คเกจสำหรับใช้งานการแปลงเสียงเป็นข้อความ (Speech to Text) โดยให้คุณสามารถรับเสียงและแปลงเป็นข้อความได้

10. flutter_tts: ^3.6.3: แพ็คเกจสำหรับใช้งาน Text-to-Speech (TTS) ซึ่งทำให้แอปพลิเคชันของคุณสามารถอ่านข้อความออกเสียงได้

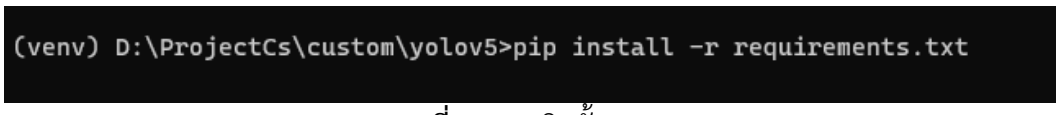
3.7 ขั้นตอนในการพัฒนา Household Model

3.7.1 การ Train model

ทำการ clone yolov5 ดังภาพประกอบที่ 3.102 จากนั้นสร้าง Virtual Environment ไว้สำหรับลง package หรือ library ดังภาพประกอบที่ 3.103 และทำการติดตั้ง requirements ที่ไว้สำหรับใช้งานกับ yolov5 ดังภาพประกอบที่ 3.104

ภาพประกอบที่ 3.102 clone yolov5

ภาพประกอบที่ 3.103 สร้าง Virtual Environment



ภาพประกอบที่ 3.104 ติดตั้ง requirements

สร้าง data.yaml ใช้เพื่อกำหนดพารามิเตอร์และการตั้งค่าต่างๆ ที่เกี่ยวข้องกับชุดข้อมูล ไฟล์นี้มีความสำคัญสำหรับการกำหนดค่ากระบวนการฝึกเทรน ดังภาพประกอบที่ 3.105 ภายในไฟล์ data.yaml ประกอบไปด้วย

- train ชี้ไปยังตำแหน่งรูปภาพที่ใช้ในการฝึกโมเดล
- val ชี้ไปยังตำแหน่งรูปภาพที่ใช้ในการตรวจสอบกับโมเดล
- test ชี้ไปยังตำแหน่งรูปภาพที่ใช้ในการทดสอบกับโมเดล
- nc คือจำนวนคลาสในชุดข้อมูล
- names คือรายการชื่อคลาส โดยที่แต่ละองค์ประกอบสอดคล้องกับคลาสในชุดข้อมูล

```
train: images/train
val: images/val
test: images/test


nc: 24
names: ['1000Baht', '100Baht', '20Baht', '500Baht', '50Baht', 'Backpack', 'Book', 'Box', 'Chair', 'Charger
cable', 'Earphone', 'Fan', 'Fork', 'Glass', 'Key', 'Knife', 'Mouth Mask', 'Outlet', 'Plate', 'Spoon', 'TV
Remote', 'Table', 'Vase', 'Walking stick']
```

ภาพประกอบที่ 3.105 กำหนดพารามิเตอร์และการตั้งค่าชุดข้อมูล

ในส่วนการ train เราจะใช้ script จาก train.py ภายในไฟล์ yolov5 ที่ clone ดังภาพประกอบที่ 3.106

โดยมีคำสั่งดังนี้:

python train.py --img 640 --batch 4 --workers 1 --epochs 10 --data data.yaml --weights yolov5m.pt

- img ระบุขนาดภาพอินพุตสำหรับการฝึก หมายความว่ารูปภาพจะถูกปรับขนาดให้มีความกว้างและความสูง 640 พิกเซลระหว่างการฝึก

- batch เป็นการตั้งค่าจำนวนข้อมูลตัวอย่าง (instances) ที่ถูกนำเข้าในโมเดลในแต่ละครั้งในกระบวนการฝึกคือ 4 ข้อมูล นี่เป็นจำนวนข้อมูลออบเจ็กต์ที่อยู่ในแต่ละรูปภาพ โมเดลใช้ batch size เพื่อกำหนดจำนวนข้อมูลที่จะถูกนำเข้าและคำนวณค่า gradient ในแต่ละรอบการฝึก โดยทั่วไปแล้ว batch size จะเป็นจำนวนข้อมูลที่มีขนาดเท่ากันในแต่ละรอบการฝึก 4 ข้อมูลที่ถูกใช้ในแต่ละครั้งอาจมาจาก 4 รูปภาพที่แตกต่างกันหรือจากรูปภาพเดียวกัน

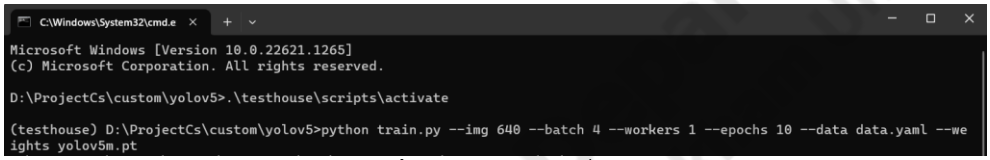
- workers หมายถึงกำหนดให้ใช้ threads เพียงเส้นทางเดียวในกระบวนการโหลดข้อมูลและฝึกโมเดล นี่คือสิ่งที่เรียกว่า "single-threaded" หรือการทำงานบนเส้นทางเดียว ซึ่งหมายความว่ากระบวนการโหลดข้อมูลและการฝึกโมเดลจะทำงานแบบลำดับหนึ่งคำสั่งต่อหนึ่งคำสั่ง และไม่ได้ใช้การ

ประมวลผลแบบพรีเซส (parallel processing) เพื่อเร่งความเร็ว จำนวน threads อาจช่วยเพิ่มประสิทธิภาพในการโหลดข้อมูลและการฝึกโมเดล และช่วยให้กระบวนการฝึกเสร็จเร็วขึ้น

- epochs จำนวนการฝึกโมเดล โมเดลจะถูกฝึกด้วยชุดข้อมูลเดิม 10 รอบซ้ำ โดยในแต่ละรอบการฝึก โมเดลจะได้รับชุดข้อมูลที่สุ่มมาจากชุดข้อมูลต้นฉบับ dataset แต่อาจมีการจับคู่กับข้อมูลซ้ำกันก็ได้ ขึ้นอยู่กับวิธีการสุ่มข้อมูลและการกำหนดค่าในโปรแกรมฝึก

- data ระบุเส้นทางไปยังไฟล์การกำหนดค่า data.yaml ไฟล์ data.yaml มีข้อมูลเกี่ยวกับชุดข้อมูล เช่น เส้นทางไปยังข้อมูลการฝึกอบรมและการตรวจสอบ จำนวนคลาส และชื่อคลาส

- weights ไฟล์น้ำหนักเริ่มต้นของโมเดล (pre-trained weights) ใช้ในกระบวนการฝึก ในที่นี้คือ "yolov5m.pt".



ภาพประกอบที่ 3.106 script ที่ใช้ในการ train