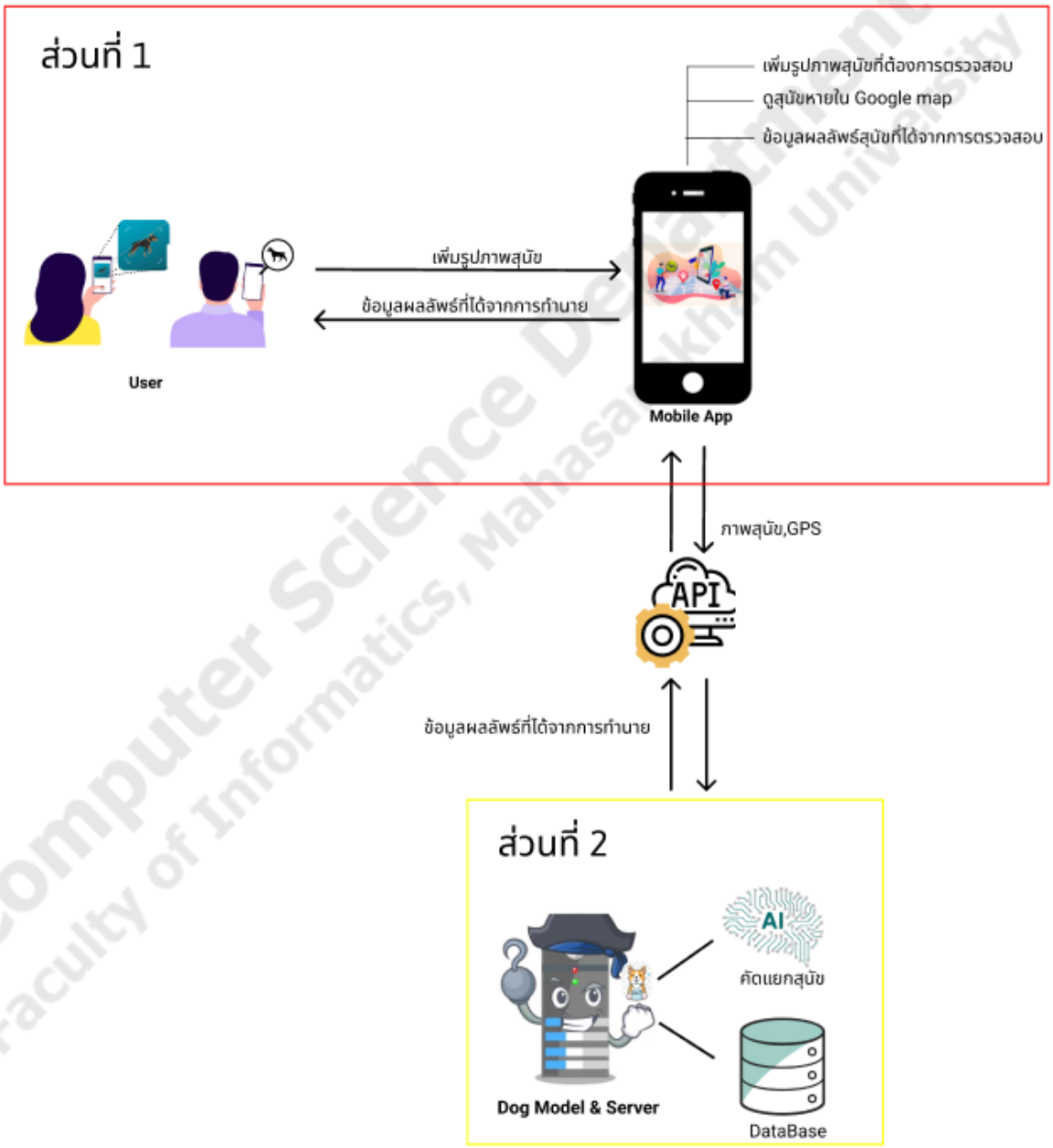
บทที่ 3

วิธีดำเนินงานวิจัย

3.1 กรอบการดำเนินงาน



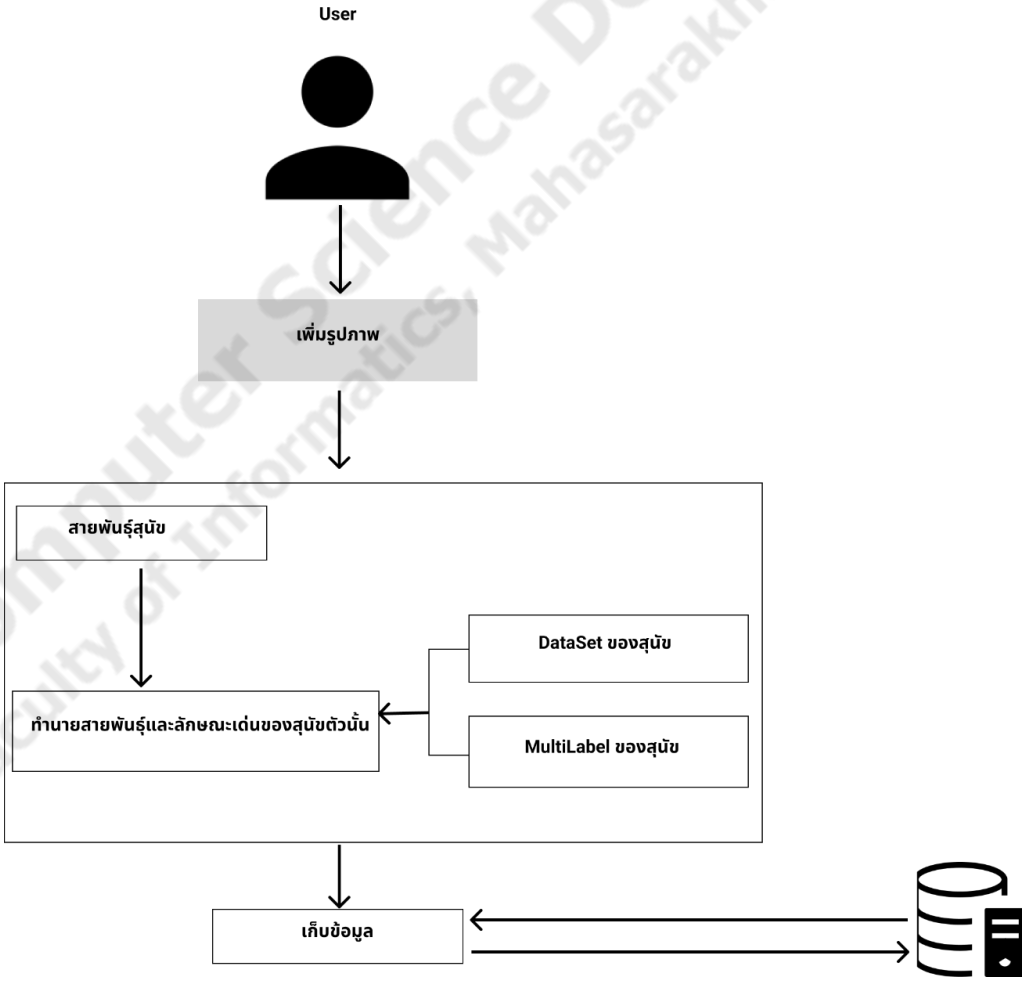
ภาพประกอบที่ 3.1 กรอบการดำเนินงานของระบบ

3.1.1 ส่วนที่ 1 ผู้ใช้

ระบบตามหาสุนัขสำหรับผู้ใช้ให้ผู้ใช้ส่งรูปถ่ายสุนัข และเก็บ GPS ระบุตำแหน่งของสุนัข

ภาพประกอบที่ 3.2 ขั้นตอนการทำงานของผู้ใช้

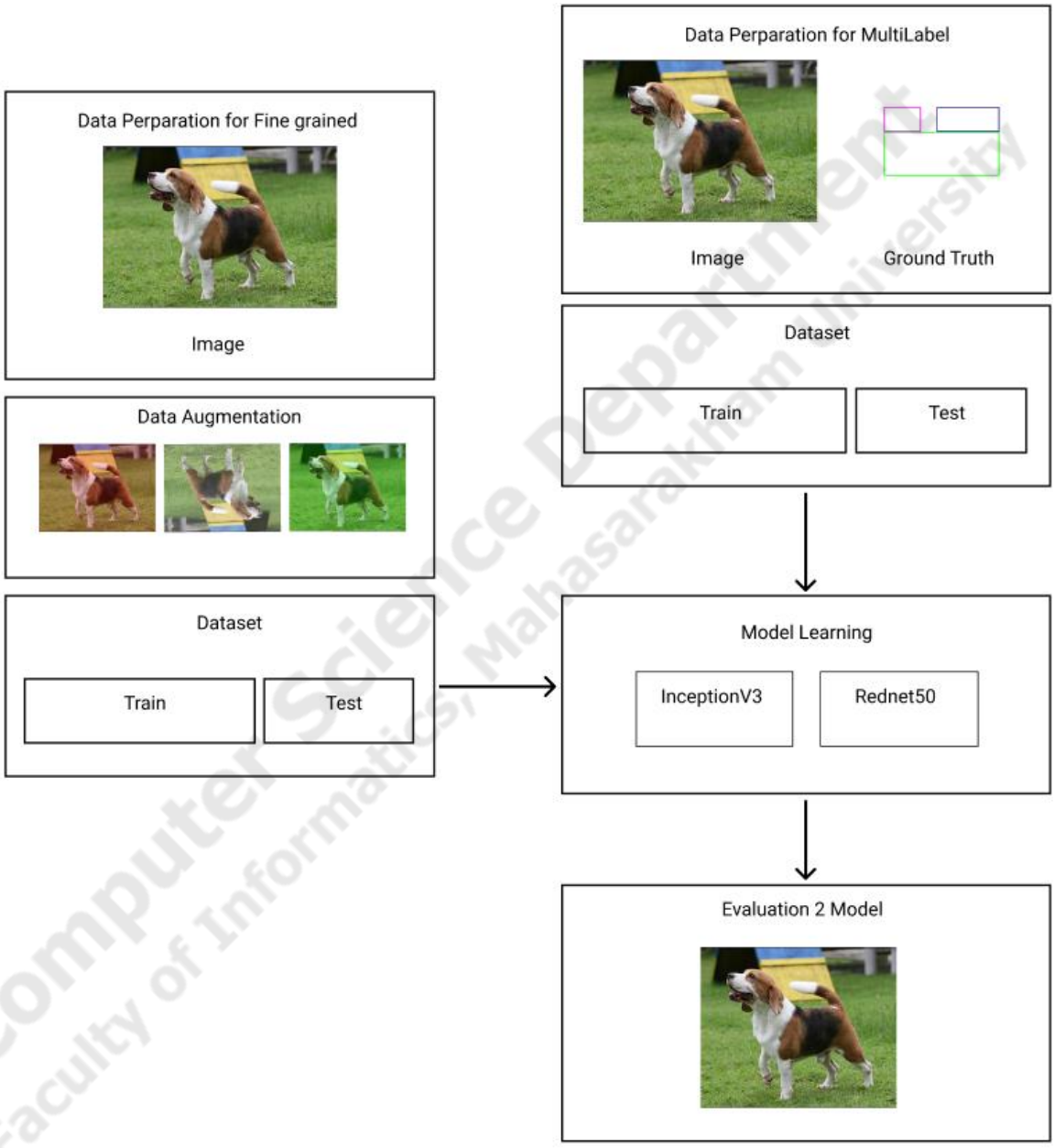
3.1.2 ส่วนที่ 2 ระบบหลังบ้าน Backend



ภาพประกอบที่ 3.3 ระบบ Backend

3.2 ขั้นตอนการดำเนินงานระบบค้นหาสุนัขด้วยการเรียนรู้เชิงลึก

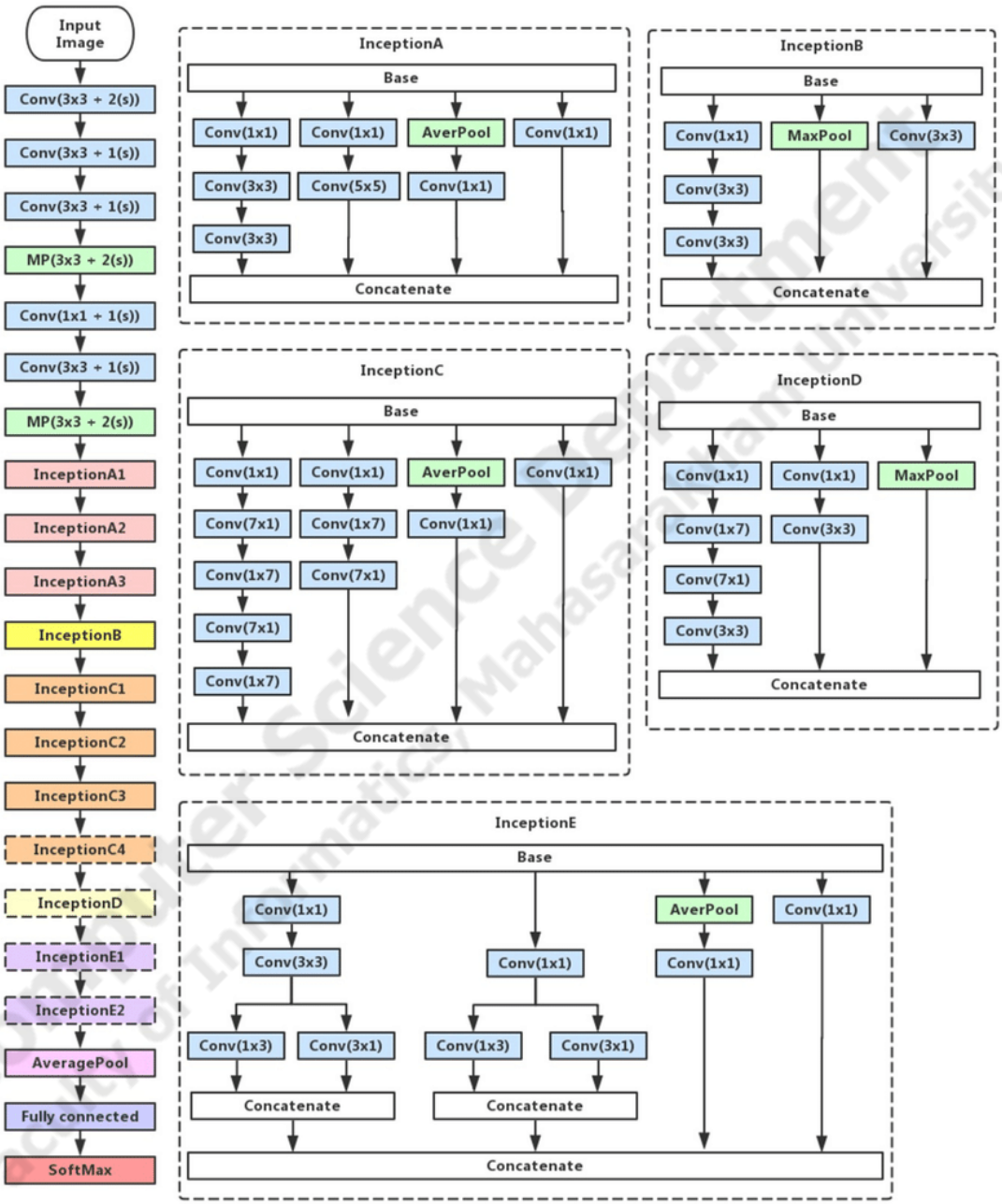
การสร้างแบบจำลองโมเดลแบบ Classification โดยใช้ CNN ในงานวิจัยนี้ใช้สถาปัตยกรรม CNN แบบ Inception V3 และ ResNet50



ภาพประกอบที่ 3.4 กรอบการดำเนินงาน

3.3 การทำงานของโมเดล

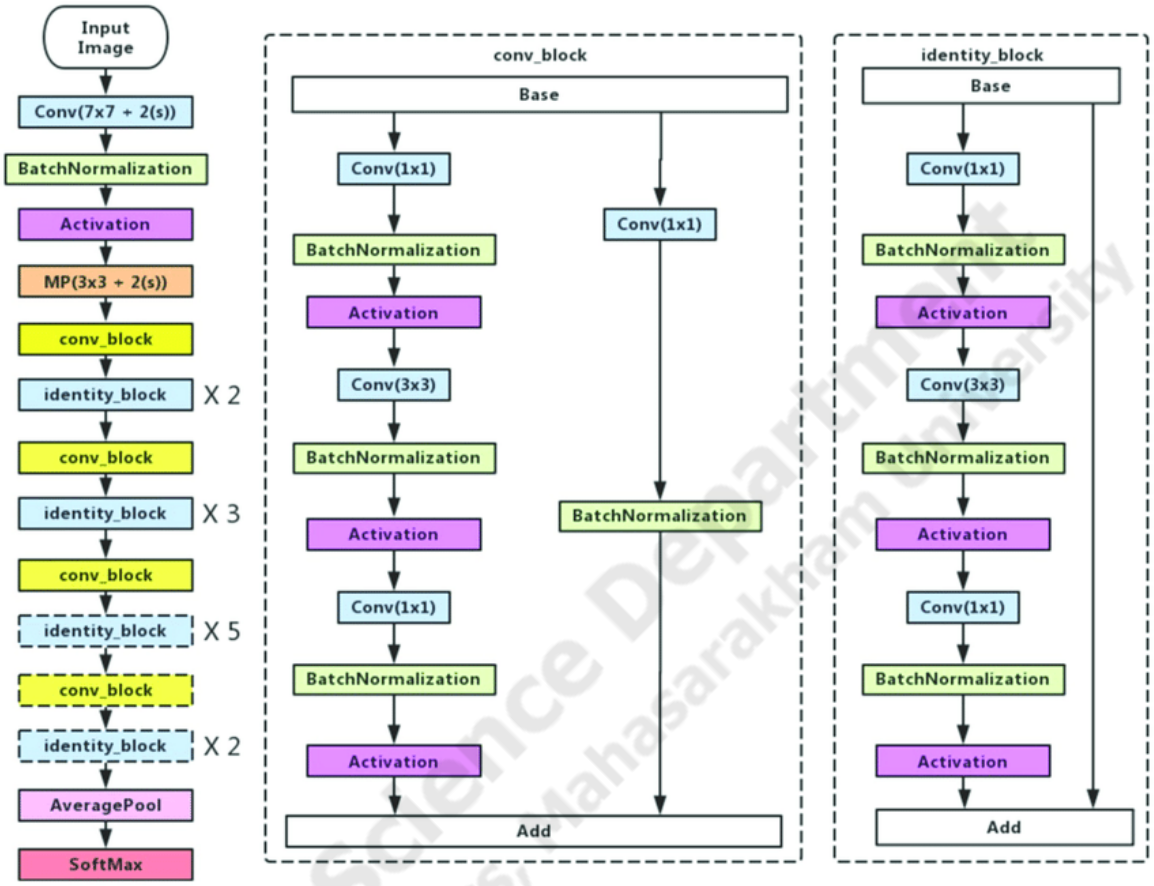
3.3.1 โครงสร้าง InceptionV3



ภาพประกอบที่ 3.5 Inception V3 Architecture [6]

จากภาพทางซ้ายเป็นสถาปัตยกรรม Inception V3 block ที่มีเส้นประเป็นโมดูลที่อาจถูกลบออกในการปรับใช้กับงานวิจัย แล้วภาพทางขวาเป็นโมดูล Inception ต่างๆ การเริ่มต้น A, C, E เป็นโมดูลการแยกตัวประกอบสามประเภท และ Inception B, D เป็นโมดูลที่ลดขนาด grid ได้อย่างมีประสิทธิภาพ

3.3.2 โครงสร้าง ResNet50



ภาพประกอบที่ 3.6 ResNet50 Architecture [6]

จากภาพทางซ้ายเป็นสถาปัตยกรรม ResNet50 block ที่มีเส้นประเป็นโมดูลที่อาจถูกลบออกในการปรับใช้กับงานวิจัย ภาพกลางเป็น Convolution block ซึ่งเปลี่ยนขนาดของ Input และภาพทางขวาเป็น Identity block ซึ่งจะไม่เปลี่ยนขนาดของ Input

3.3.3 Convolutional Neural Network (CNN) based on model

วิธีการทำงานของโมเดล Inception V3 และ ResNet50 หลักๆ มีดังต่อไปนี้

(1) ค้นหาคุณลักษณะเด่นของภาพออกมา (Convolution)

การทำงานของขั้นตอนค้นหาคุณลักษณะเด่นของภาพออกมา (Convolution) จะทำการ Sliding Windows (Filter) เพื่อค้นหาองค์ประกอบของภาพ เช่น สี หรือรูปร่าง เป็นต้น

4	8	2	2	4
4	8	4	5	3
4	3	2	5	4
5	0	6	5	7
6	6	3	7	1

ภาพขนาด 5 × 5

1	0	1
0	1	0
1	0	1

Filter ขนาด 3 × 3

ภาพประกอบที่ 3.7 ขนาดภาพนำเข้าและขนาดของ Filter

กำหนดค่าเพื่อคำนวณโดยการกำหนดค่าภาพนำเข้าขนาดเป็น 6 × 6 Filter เป็น 3 × 3 Padding เป็น 1 และ Stride เป็น 1 จากนั้นแทนค่าในสมการ Output feature map เพื่อหาขนาดภาพใหม่หลังจากการทำ Convolution ได้ดังนี้

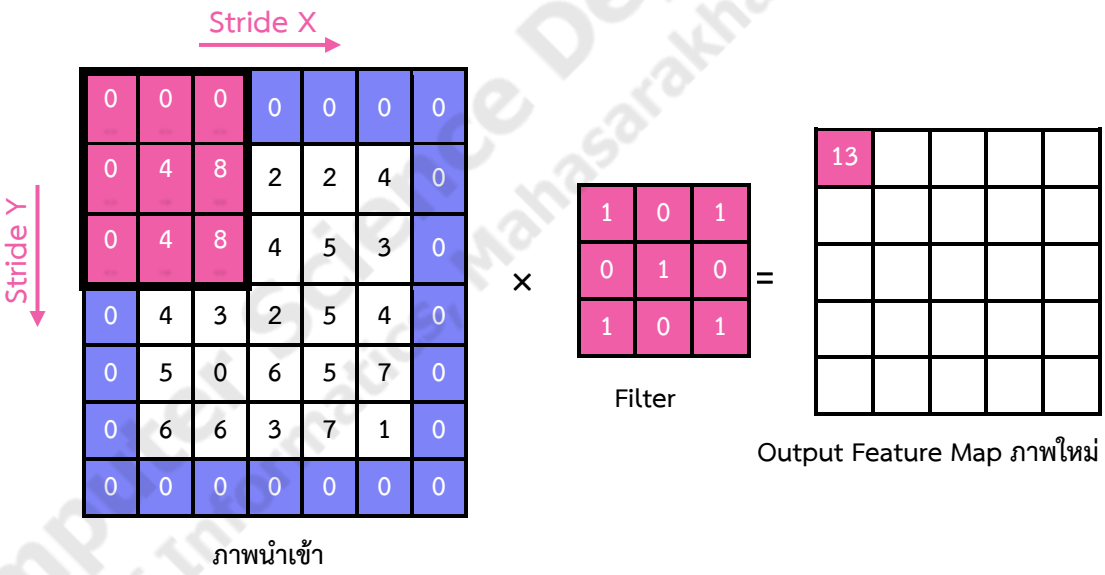
$$Output\ Feature\ Map = \frac{5 - 3 + 2(1)}{1} + 1 = 5$$

ภาพประกอบที่ 3.8 ขนาดของภาพใหม่ ขนาด 5 × 5

เนื่องจากกระบวนการ Convolution จะลดขนาดภาพให้เล็กลง ซึ่งอาจทำให้สูญเสียบางค่าข้อมูลบนขอบภาพ เมื่อเราใช้ Convolution แบบต่อเนื่องหลายชั้น จึงต้องทำการเพิ่มขนาดของขอบภาพ โดยการใช้ Padding เป็น 1 ทำให้เราต้องขยายขอบของภาพทุกด้านเท่าๆกัน เป็น 0 ทุกช่องเพื่อยังคงคุณลักษณะที่สำคัญของภาพไว้

0	0	0	0	0	0	0
0	4	8	2	2	4	0
0	4	8	4	5	3	0
0	4	3	2	5	4	0
0	5	0	6	5	7	0
0	6	6	3	7	1	0
0	0	0	0	0	0	0

ภาพประกอบที่ 3.9 ภาพนำเข้าที่ทำการเพิ่มขอบของภาพต้นฉบับ



ภาพประกอบที่ 3.10 การ Convolution

แสดงตัวอย่างการคำนวณภาพส่วนย่อยคูณกับ Filter ขนาด 3 × 3 ในรอบที่ 1 ได้ดังนี้

ตำแหน่งที่ 1 $0 \times 1 = 0$

ตำแหน่งที่ 2 $0 \times 0 = 0$

ตำแหน่งที่ 3 $1 \times 1 = 1$

ตำแหน่งที่ 4 $0 \times 0 = 0$

ตำแหน่งที่ 5 $4 \times 1 = 4$

ตำแหน่งที่ 6 $8 \times 0 = 0$

ตำแหน่งที่ 7 $0 \times 1 = 0$

ตำแหน่งที่ 8 $4 \times 0 = 0$

ตำแหน่งที่ 9 $8 \times 1 = 8$

จากนั้น นำผลลัพธ์ของทุกตำแหน่งมาบวกกัน $(0 + 0 + 1 + 0 + 4 + 0 + 0 + 0 + 8) = 13$ และเก็บผลลัพธ์ไว้ที่ Output feature map ภาพใหม่ และเลื่อนตำแหน่งไปให้ทั่วทั้งภาพผลลัพธ์ทั้งหมดจากการคำนวณได้ดังนี้

13	16	15	9	9
15	20	22	17	10
12	22	20	25	14
14	15	27	15	19
6	17	8	20	6

ภาพประกอบที่ 3.11 ผลลัพธ์ Output feature map

(2) ขั้นตอนการตรวจจับ (ReLU)

ในขั้นตอนนี้ จะทำหน้าที่รับข้อมูลที่ได้จากขั้นตอน Convolution มาแปลงให้อยู่ในรูปแบบที่ไม่เป็นเชิงเส้น (Nonlinear) โดยใช้ฟังก์ชันการกระตุ้น (Activation function) เช่น Rectified Linear Unit (ReLU) โดยผลลัพธ์ที่ได้จากการทำ Convolution ในแต่ละตำแหน่งจะผ่านการแปลงค่าด้วยฟังก์ชัน ReLU ที่เป็นการแปลงแบบไม่เป็นเชิงเส้น เพื่อความง่ายในการคำนวณและประเมินประสิทธิภาพผลลัพธ์ สามารถคำนวณด้วยสมการดังนี้

$$R(x) = \begin{cases} 0, & if\ x < 0 \\ x, & if\ x \geq 0 \end{cases}$$

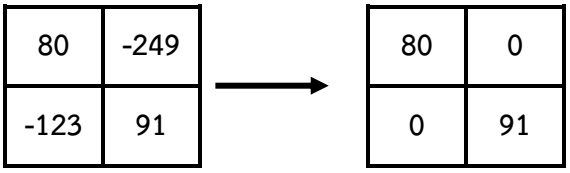
โดยที่ x คือ จุดพิกเซลของ Output feature map

ขั้นตอนการทำงานโดยเงื่อนไขดังนี้

(1) ให้ x เป็น 0 ก็ต่อเมื่อ x น้อยกว่า 0

(2) ให้ x เป็น x ก็ต่อเมื่อ x มากกว่าหรือเท่ากับ 0

การทำงาน ReLU กับภาพตัวอย่างแสดงได้ดังนี้



ภาพประกอบที่ 3.12 ตัวอย่างการทำงานของ ReLU

$$R(x_{-80}) = \max(0, -80)$$
$$R(x_{-80}) = 80$$

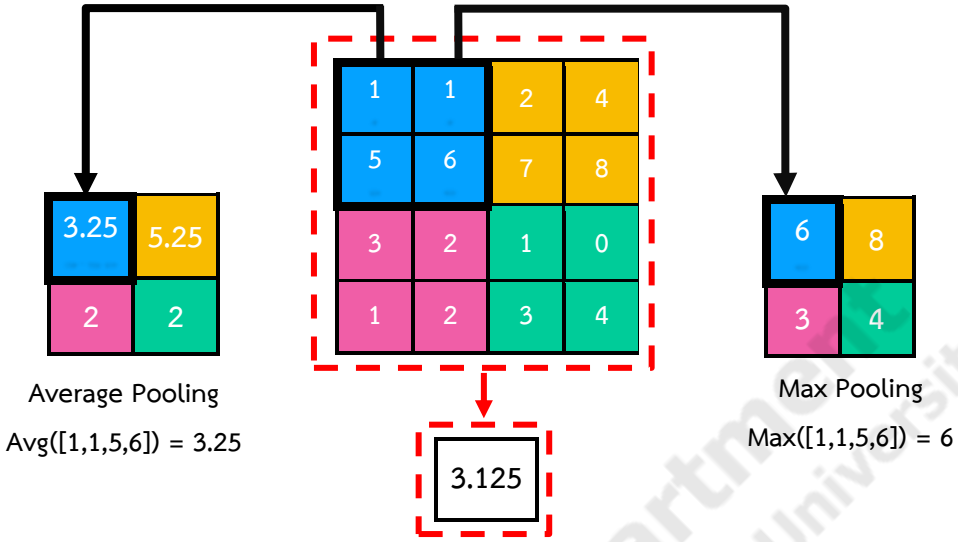
$$R(x_{-249}) = \max(0, -249)$$
$$R(x_{-249}) = 0$$

$$R(x_{-123}) = \max(0, -123)$$
$$R(x_{-123}) = 0$$

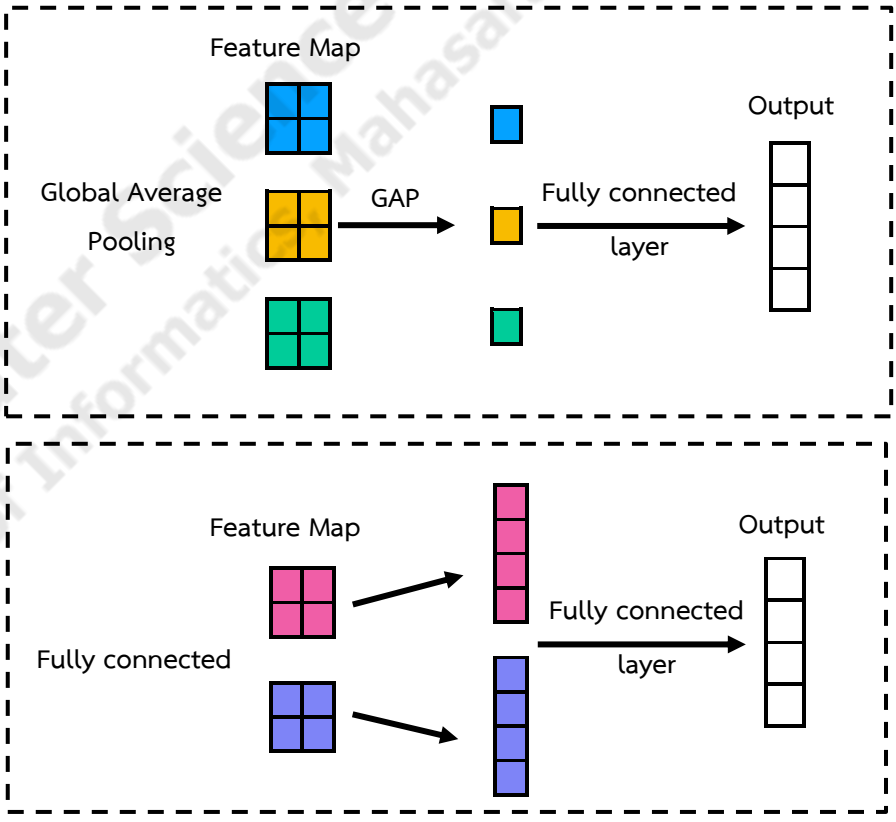
$$R(x_{91}) = \max(0,91)$$
$$R(x_{91}) = 91$$

(3) Pooling

การ Pooling เป็นการลดขนาดภาพให้เล็กลง เพื่อลดความซับซ้อนของการประมวลผลและเพิ่มความเร็วในการคำนวณในขั้นตอนต่อไป เป็นการลดมิติของ Feature map แต่ยังคงคุณลักษณะสำคัญที่จำเป็นไว้ในทุกขั้นตอนของขั้นตอนการรวม ในกระบวนการทำงานคล้ายกับ Convolution โดยขั้นตอนการทำงานจะวาง Filter ที่กำหนดและ Stride บน Feature Map เพื่อค้นหาค่าจากประเภทที่เลือกใช้ในการ Stride แต่ละครั้ง ซึ่งเราจะกำหนด Stride เป็น 2 และ Filter ขนาดเป็น 2 × 2 ซึ่งงานวิจัยเลือกใช้ Max pooling เพื่อเอาค่าที่มากที่สุด, Average pooling เพื่อเอาค่าที่เฉลี่ยได้ และการหาค่าเฉลี่ยจากทุกค่าบน Feature map ด้วย Global average pooling (GAP) เป็นการแทนที่ชั้น Fully connected ใน CNN โดยการหาค่าเฉลี่ยแต่ละ Feature map แล้วสร้าง Output feature map ใหม่ 1 รายการสำหรับแต่ละคลาสที่สอดคล้องกับคลาสทั้งหมดในชั้นสุดท้าย

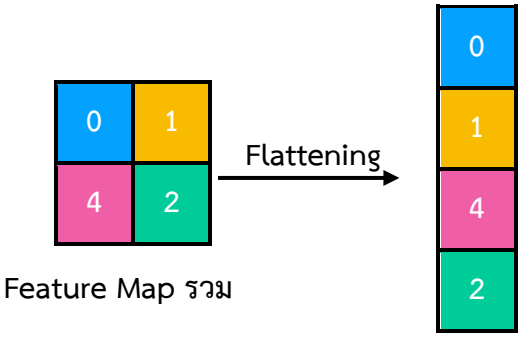


ภาพประกอบที่ 3.13 การ Pooling แบบ Average, Max และ Global average



ภาพประกอบที่ 3.14 การเปรียบเทียบ Fully connected กับ Global average pooling

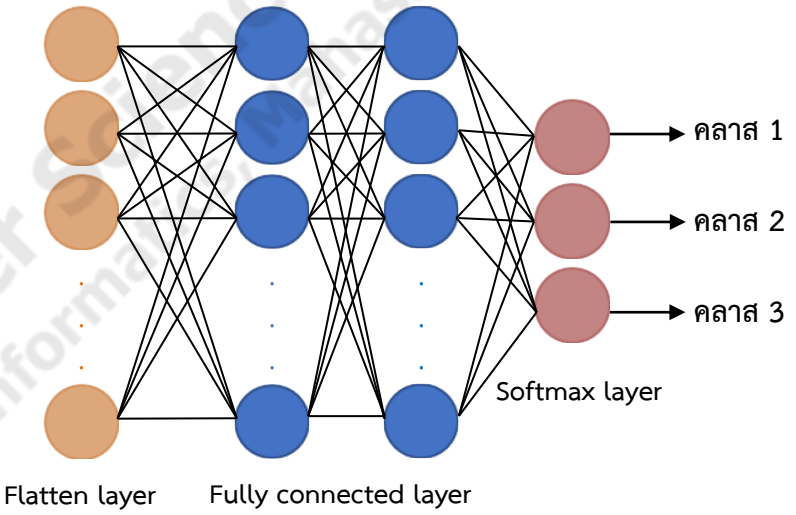
(4) แปลงภาพให้เป็นแนวตั้ง (Flatten)



ภาพประกอบที่ 3.15 ผลลัพธ์การเปลี่ยนโครงสร้างชุดข้อมูล

การ Flatten เป็นการแปลงข้อมูลผลลัพธ์ทั้งหมดจาก Feature map เป็นอาร์เรย์ 1 มิติ แบบเวกเตอร์ยาวต่อเนื่องกัน เพื่อส่งข้อมูลไปยังขั้นตอน Full connection เป็นขั้นตอนโครงข่ายประสาทเทียมที่เป็นการรับข้อมูล Input แบบเวกเตอร์แนวตั้ง

(5) Fully Connected



ภาพประกอบที่ 3.16 ภาพรวมของการ Fully connection

เมื่อผ่านขั้นตอน Full connection จนเหลือ 1000 ความเป็นไปได้ของผลลัพธ์สุดท้ายจะต้องส่งค่าที่ได้ทั้งหมดไปยังขั้นตอนต่อไป โดยผ่านกระบวนการ 2 ขั้นตอน สำหรับ Classification ก่อนแล้วจะได้คำตอบผลลัพธ์แสดงตัวอย่างด้วยชุดข้อมูลที่กำหนดให้ดังนี้

(1) Activation Function เป็นการเพิ่มความซับซ้อนในการเรียนรู้จากชุดข้อมูล มี 2 ประเภทหลักที่เลือกใช้ในงานวิจัยดังต่อไปนี้

– Softmax Function เป็น Activation Function สำหรับ Multi Class เป็นการแปลงค่าให้เป็นมาตรฐานสำหรับการแจกแจงความน่าจะเป็นตามสัดส่วนของค่า Output ทั้งหมดสำหรับคลาส

โดยที่ x_i คือ ค่าที่ Input

e^{x_i} คือ ค่าที่ได้จากการนำ e มายกกำลังด้วย x_i

แสดงวิธีการคำนวณตำแหน่งที่ 1 ดังนี้

$$S(0) = \frac{e^0}{e^0 + e^1 + e^4 + e^3}$$

$$S(0) = \frac{1}{1 + 2.718 + 54.598 + 7.389}$$

$$S(0) = \frac{1}{65.705} = 0.015$$

เมื่อคำนวณครบทุกค่าผลลัพธ์จะได้ตั้งแต่ 0 ถึง 1 ถ้าทั้งหมดรวมกันแล้วจะเท่ากับ 1 เสมอ

ตารางที่ 3.1 การคำนวณ Softmax

i	x	e^{x_i}	$S(x)$
1	0	1	0.01521954189178905714937980366791
2	1	2.718	0.04136671486188265733201430636938
3	4	54.598	0.83095654820789894224183852066053
4	3	7.389	0.11245719503842934327676736930218
รวม			1

– Sigmoid Function เป็น Activation Function สำหรับ Multi-label เป็นการหาความน่าจะเป็นเป็นผลลัพธ์ โดยผลลัพธ์ Output แต่ละค่ามีความสัมพันธ์แบบไม่ต่อเนื่องกัน เพื่อหาคำตอบที่ถูกต้องมากกว่า 1 คำตอบ

โดยที่ x คือ ค่าที่ Input

แสดงวิธีการคำนวณตำแหน่งที่ 1 ดังนี้

$$S(0) = \frac{1}{1 + e^0}$$

$$S(0) = \frac{1}{1 + 1}$$

$$S(0) = \frac{1}{2} = 0.5$$

เมื่อคำนวณครบทุกค่าผลลัพธ์แต่ละค่าจะได้ตั้งแต่ 0 ถึง 1

ตารางที่ 3.2 การคำนวณ Sigmoid

i	x	e^{-x}	$S(x)$
1	0	1	0.5
2	1	2.7182	0.26894142137
3	4	54.5981	0.01798620996
4	3	20.0855	0.04742587317

(2) Loss Function แบบ Cross Entropy เป็นการเปรียบเทียบความแตกต่างที่แน่นอนระหว่างค่าผลลัพธ์ของการแจกแจงความน่าจะเป็นโดยค่าที่ทำนาย (Prediction) และค่าจริง (Actual)

$$Actual(\boldsymbol{y_i}) = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad Prediction(\hat{\boldsymbol{y}}_i) = \begin{bmatrix}0.7\\0.3\\0.2\end{bmatrix}, \quad Class = \begin{bmatrix}Beagle\\Corgi\\Doberman\end{bmatrix}$$

แสดงการคำนวณคลาสที่ 1 ได้ดังนี้

$$CE_i = \boldsymbol{y_i} \cdot log\ \hat{\boldsymbol{y}}_i$$
$$CE_1 = \boldsymbol{y_1} \cdot log\ \hat{\boldsymbol{y}}_1$$
$$CE_1 = 1 \cdot \log(0.7)$$
$$CE_1 = 0.5145$$

เมื่อคำนวณครบทุกค่าผลลัพธ์ของแต่ละคลาสจะรวมกันเพื่อหา Cross Entropy ทั้งหมดสำหรับภาพได้ดังนี้

$$CE = -(CE_1 + CE_2 + CE_3)$$
$$CE = -(-0.5145 - 0 - 0)$$
$$CE = 0.5145$$

3.4 การรวบรวมชุดข้อมูล (Data Preparation)

ในการเก็บข้อมูลในการวิจัยนี้ ได้ทำการเก็บชุดข้อมูลสุนัข 11 สายพันธุ์จากเว็บไซต์ต่างๆ จาก Google Search, Facebook, Stanford Dogs จาก ImageNet และจากการภาพถ่ายทางมือถือ โดยจะแบ่งชุดข้อมูลสำหรับ 2 โมเดล ซึ่งมีรายละเอียดดังนี้

3.4.1 ชุดข้อมูลสำหรับ Fine-Grained Model

ชุดข้อมูลที่ใช้ในการสร้างโมเดลมีทั้งหมด 16,243 ภาพ จะแบ่งเป็นชุดข้อมูลสำหรับการสอน 12,994 ภาพ การทดสอบ 3,249 ภาพ

ภาพประกอบที่ 3.17 ตัวอย่างสุนัขแต่ละสายพันธุ์

ตารางที่ 3.3 รายละเอียดของชุดข้อมูลภาพ

ลำดับที่	สายพันธุ์สุนัข	ชุดข้อมูล	
		สอน	ทดสอบ
1	บีเกิ้ล	1,454	364
2	ชิวาว่า	1,004	251
3	โดเบอร์แมน	1,000	250
4	เฟรนช์ บลูด็อก	1,205	301
5	เยอรมัน เชเพิร์ด	1,390	348
6	โกลเด้น รีทรีฟเวอร์	1,043	261

ตารางที่ 3.3 รายละเอียดของชุดข้อมูลภาพ (ต่อ)

ลำดับที่	สายพันธุ์สุนัข	ชุดข้อมูล	
		สอน	ทดสอบ
7	เพมโพร์ก คอร์กี้	1,131	283
8	ปอมเมอเรเนียน	1,301	325
9	ไซบีเรียน ฮัสกี้	995	249
10	บางแก้ว	1,077	269
11	หลังอาน	1,394	348
รวม		12,994	3,240

3.4.2 ชุดข้อมูลสำหรับ Multi-label Model

ชุดข้อมูลที่ใช้ในการสร้างโมเดลมีทั้งหมด 13,872 ภาพ ทำการปรับแต่ละภาพเป็นขนาด 227 × 227 และนำมาทำ Ground Truth จากนั้นจะแบ่งเป็นชุดข้อมูลสำหรับการสอน 80% และการทดสอบ 20% ด้วยมือ โดยจะกำหนดให้แต่ละ Label ของ Feature มีอยู่ 2 กรณี คือ 0 หรือ 1

ตารางที่ 3.4 การกำหนดค่า Ground truth สำหรับแต่ละ Feature ที่ใช้ในโมเดล

ลำดับที่	Feature	Label	
		0	1
1	หูตก	ไม่ใช่	ใช่
2	หูตั้ง	ไม่ใช่	ใช่
3	หางยาว	ไม่ใช่	ใช่
4	หางสั้น	ไม่ใช่	ใช่
5	สีดำ	ไม่ใช่	ใช่
6	สีน้ำตาลเข้ม	ไม่ใช่	ใช่
7	สีน้ำตาลอ่อน	ไม่ใช่	ใช่
8	สีขาว	ไม่ใช่	ใช่

ตารางที่ 3.4 การกำหนดค่า Ground truth สำหรับแต่ละ Feature ที่ใช้ในโมเดล (ต่อ)

ลำดับที่	Feature	Label	
		0	1
9	ขนยาว	ไม่ใช่	ใช่
10	ขนสั้น	ไม่ใช่	ใช่
11	ปลอกคอสีดำ	ไม่ใช่	ใช่
12	ปลอกคอสีน้ำเงิน	ไม่ใช่	ใช่
13	ปลอกคอสีทอง	ไม่ใช่	ใช่
14	ปลอกคอสีเขียว	ไม่ใช่	ใช่
15	ปลอกคอสีแดง	ไม่ใช่	ใช่
16	ปลอกคอสีเงิน	ไม่ใช่	ใช่
17	ไม่มีปลอกคอ	ไม่ใช่	ใช่

ตารางที่ 3.5 ตัวอย่างข้อมูล Ground Truth สำหรับการสอน

ภาพตัวอย่าง			
			
โมเดล	**ลำดับที่**	**Feature**	**Label**
หู	1	หูตก	1
	2	หูตั้ง	0
หาง	1	หางสั้น	1
	2	หางยาว	0

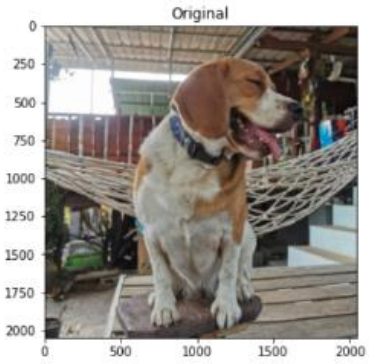
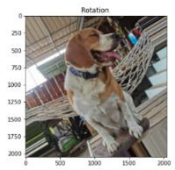
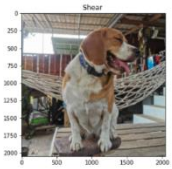
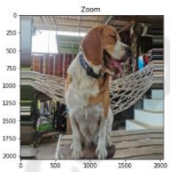
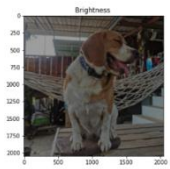
ตารางที่ 3.5 ตัวอย่างข้อมูล Ground Truth สำหรับการสอน (ต่อ)

โมเดล	ลำดับที่	Feature	Label
สีตัว	1	สีดำ	0
	2	สีน้ำตาลเข้ม	0
	3	สีน้ำตาลอ่อน	1
	4	สีขาว	1
ขน	1	ขนยาว	0
	2	ขนสั้น	1
ปลอกคอ	1	ปลอกคอสีดำ	0
	2	ปลอกคอสีน้ำเงิน	1
	3	ปลอกคอสีทอง	0
	4	ปลอกคอสีเขียว	0
	5	ปลอกคอสีแดง	0
	6	ปลอกคอสีเงิน	0
	7	ไม่มีปลอกคอ	0

3.5 Data Augmentation

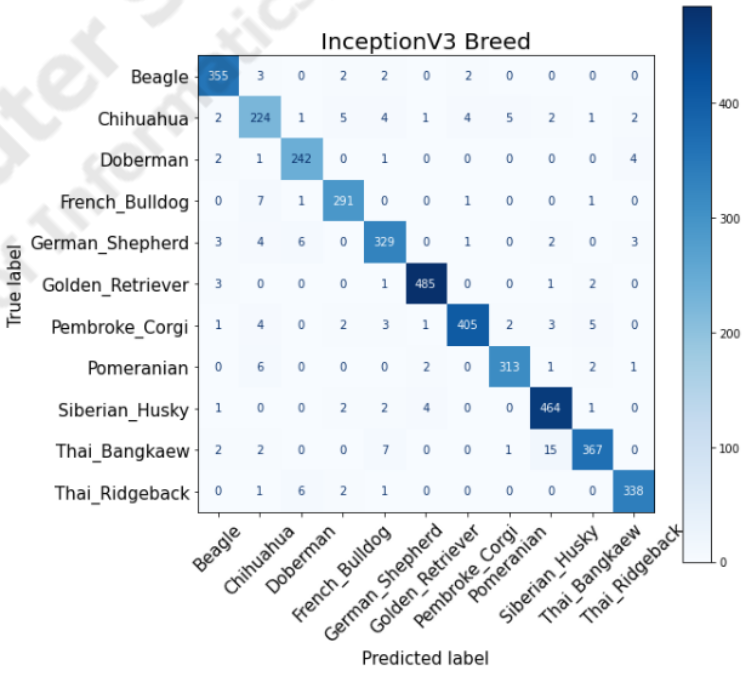
การขยายชุดข้อมูลภาพให้มีความหลากหลาย จะช่วยให้มีการสร้างข้อมูลเพิ่มจากเดิมได้มากกว่าสองเท่า และช่วยให้โมเดลได้เรียนรู้ข้อมูลหลายๆแบบ โดยการทำ Data augmentation ได้ใช้ ImageDataGenerator เพิ่มข้อมูลภาพด้วยการประมวลข้อมูลภาพล่วงหน้า (Image preprocessing) ด้วยการเสริมข้อมูลแบบทันที (Real-time) ในขณะที่โมเดลยังอยู่ในขั้นตอนการฝึก โดยมีฟังก์ชันให้เลือกในการสุ่มสร้างภาพใหม่มากมาย เช่น เพิ่มจุดรบกวน การปรับขนาดภาพ หมุน เฉือน ซูม พลิก ปรับความสว่างภาพ แทนจุดภาพที่อยู่นอกขอบเขตด้วยจุดที่ใกล้ที่สุด (nearest) ที่กำหนด เป็นต้น แต่งานวิจัยเลือกใช้วิธีตามตารางดังนี้

ตารางที่ 3.6 ตัวอย่างภาพ Data Augmentation

Image				
				
Rotation	Shear	Zoom	Horizontal Flip	Brightness
				

3.6 การวัดประสิทธิภาพ (Evaluation)

ตัวอย่างการคำนวณหา Precision, Recall และ Accuracy จาก Confusion Matrix ดังรูปต่อไปนี้



ภาพประกอบที่ 3.18 ผลลัพธ์ Confusion Matrix ของ Inception V3 Model

แสดงการคำนวณลำดับที่ 1 ดังต่อไปนี้

$$Precision_{Beagle} = \frac{355}{(355+14)} = 0.9620$$

$$Recall_{Beagle} = \frac{355}{(355+9)} = 0.9752$$

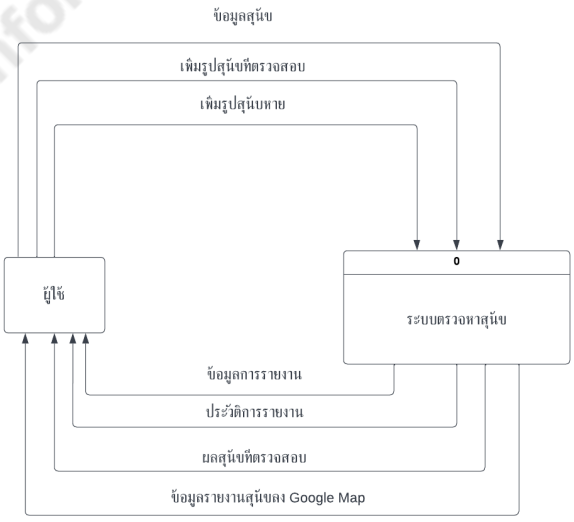
$$Accuracy_{Beagle} = \frac{355+3458}{(355+14+3458+9)} = 0.99$$

เมื่อคำนวณแต่ละผลลัพธ์ทั้งหมด จะได้ค่า Accuracy ของโมเดล 0.96

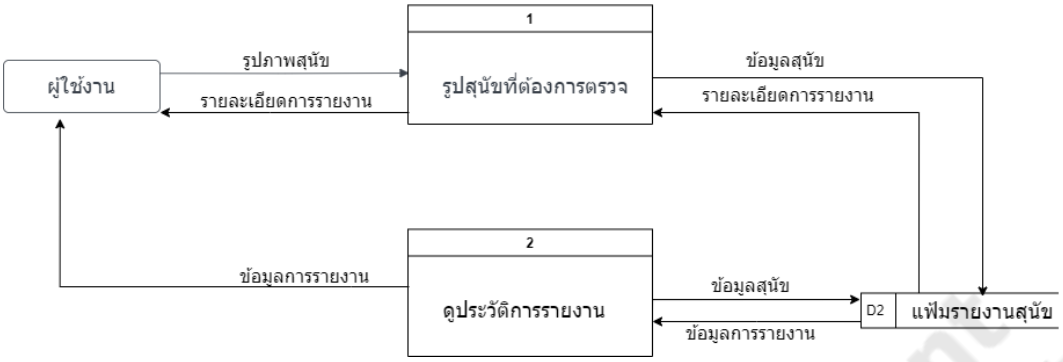
	precision	recall	f1-score	support
Beagle	0.96	0.98	0.97	364
Chihuahua	0.89	0.89	0.89	251
Doberman	0.95	0.97	0.96	250
French_Bulldog	0.96	0.97	0.96	301
German_Shepherd	0.94	0.95	0.94	348
Golden_Retriever	0.98	0.99	0.98	492
Pembroke_Corgi	0.98	0.95	0.97	426
Pomeranian	0.98	0.96	0.97	325
Siberian_Husky	0.95	0.98	0.96	474
Thai_Bangkaew	0.97	0.93	0.95	394
Thai_Ridgeback	0.97	0.97	0.97	348
accuracy			0.96	3973
macro avg	0.96	0.96	0.96	3973
weighted avg	0.96	0.96	0.96	3973

ภาพประกอบที่ 3.19 Report ผลลัพธ์การประเมินประสิทธิภาพโมเดล

3.7 การออกแบบและพัฒนาระบบ



ภาพประกอบที่ 3.20 Context diagram



ภาพประกอบที่ 3.21 แผนภาพกระแสข้อมูล ระดับที่ 1 (Data Flow Diagram Level 1)

3.7.1 External Entity Description

ตารางที่ 3.7 แสดง External Entity Description

Name	Description	Input Data Flow	Output Data Flow
ผู้ใช้งาน	ผู้ที่เข้ามาใช้งานระบบตรวจหาสุนัข	- ข้อมูลสุนัข - ข้อมูลการรายงาน	- ผลสุนัขที่ตรวจสอบ - ข้อมูลรายงานสุนัขลง Google Map - ประวัติการรายงาน - ผลการจัดการข้อมูล

3.7.2 Data Store Description

ตารางที่ 3.8 แสดง Data Store Description

ID	Name	Description	Data Structure
D1	แฟ้มรายงานสุนัข	ฐานข้อมูลสำหรับเก็บข้อมูลการเพิ่มรูปสุนัขที่ต้องการตรวจ	Id + ชื่อ + สายพันธุ์อันดับ1+ สายพันธุ์อันดับ2+ สายพันธุ์อันดับ3 + ผลทำนายอันดับที่1 + ผลทำนายอันดับที่2 + ผลทำนายอันดับที่3 + หูตก +หางยาว + สีดำ + สีน้ำตาลเข้ม + สีน้ำตาลอ่อน + สีขาว + ปลอกคอสีดำ + ปลอกคอสีน้ำเงิน + ปลอดคอสีทอง + สีปลอกเขียว

ตารางที่ 3.8 แสดง Data Store Description (ต่อ)

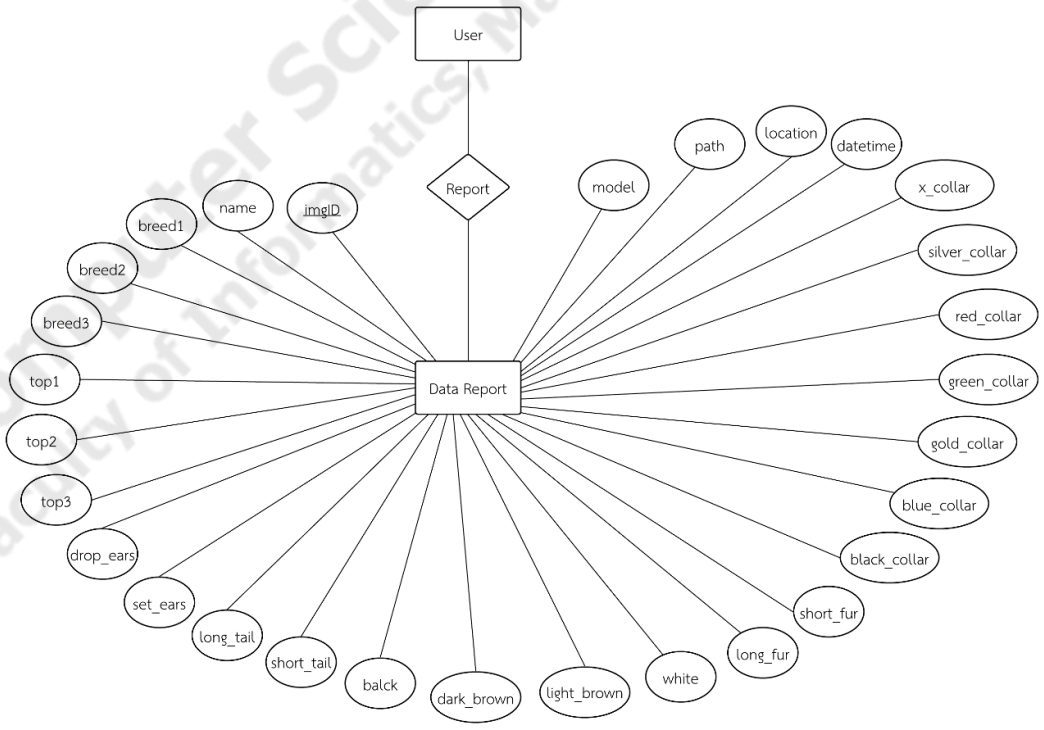
ID	Name	Description	Data Structure
			+ ปลอกคอสีแดง + ปลอกคอสีเงิน + ไม่มีปลอกคอ + วันเวลา ตำแหน่งสถานที่ + ที่อยู่ภาพ + โมเดลที่ใช้

3.7.3 Data Structure Description

ตารางที่ 3.9 แสดง Data Structure Description

Name	Description	Source	Destination	Data Structure
รูปสุนัขที่ตรวจสอบ	ไฟล์รูปภาพของสุนัขที่ผู้ใช้ส่งเข้ามา	ผู้ใช้	Process 1.0 รูปสุนัขที่ต้องการตรวจ	รูปภาพ + ชื่อไฟล์ + วันเวลา + ตำแหน่งสถานที่
			D1 แฟ้มรายงานสุนัข	

3.8 ER Diagram



ภาพประกอบที่ 3.22 ER Diagram

3.8.1 รายละเอียดตารางข้อมูล (Data table Description)

ตารางที่ 3.10 ตารางแสดงตัวอย่างข้อมูลการทำนาย (Predicts)

Attribute Name	Type	Size	Description	Key type	References
imgID	Integer	11	รหัสรูป	PK	67
name	Varchar	255	ชื่อรูป		Beagle_5119.jpg
breed1	Varchar	255	สายพันธุ์อันดับ 1		Beagle
breed2	Varchar	255	สายพันธุ์อันดับ 2		Thai_Ridgeback
breed3	Varchar	255	สายพันธุ์อันดับ 3		French_Bulldog
prob1	Float	-	ค่าทำนายอันดับ 1		0.999997
prob2	Float	-	ค่าทำนายอันดับ 2		0.00000144601
prob3	Float	-	ค่าทำนายอันดับ 3		0.0000006408
drop_ears	Float	-	หูตก		0.104492
set_ears	Float	-	หูตั้ง		0.895508
long_tail	Float	-	หางยาว		0.0781615
short_tail	Float	-	หางสั้น		0.921838
black	Float	-	ตัวสีดำ		0.0116582
dark_brown	Float	-	ตัวสีน้ำตาลเข้ม		0.0000227268
light_brown	Float	-	ตัวสีน้ำตาลอ่อน		0.992001
white	Float	-	ตัวสีขาว		0.00417269
long_fur	Float	-	ขนยาว		0.421777
short_fur	Float	-	ขนสั้น		0.578223
black_collar	Float	-	ปลอกคอสีดำ		0.00121636

ตารางที่ 3.10 ตารางแสดงตัวอย่างข้อมูลการทำนาย (Predicts) (ต่อ)

Attribute Name	Type	Size	Description	Key type	References
blue_collar	Float	-	ปลอกคอสีน้ำเงิน		0.00259532
gold_collar	Float	-	ปลอกคอสีทอง		0.000957409
green_collar	Float	-	ปลอกคอสีเขียว		0.000786348
red_collar	Float	-	ปลอกคอสีแดง		0.0061568
silver_collar	Float	-	ปลอกคอสีเงิน		0.00128032
x_collar	Float	-	ไม่มีปลอกคอ		0.987007
datetime	Datetime	-	วันเวลา		2022-06-09 09:31:44
location	Varchar	255	ตำแหน่งสถานที่		37.4219983,-122.084
Path	Varchar	255	ที่อยู่ภาพ		C:/Users/LENOVO/ProServer/Beagle_5119.jpg
model	Varchar	255	โมเดลที่ใช้		InceptionV3

3.9 อธิบายการทำงาน

3.9.1 ส่วนของการทำ Fine-grained model

3.9.1.1 การทำ Augmentation และอ่านภาพ

ทำการเสริมชุดข้อมูลแบบ real-time และสร้างชุดข้อมูลจากไฟล์รูปภาพใน directory ที่มีข้อมูลอยู่ จะ return ผลลัพธ์ แล้วจะทำการสร้างชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบ

```
SIZE = 227
target_size = (SIZE, SIZE)
batch_size = 32
class_mode = 'categorical'

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range = 40,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   brightness_range = (0.6, 1)
                                  )
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_path,
                                                    target_size=target_size ,
                                                    batch_size=batch_size,
                                                    class_mode=class_mode)
test_generator = test_datagen.flow_from_directory(test_path,
                                                  target_size=target_size,
                                                  batch_size=batch_size,
                                                  class_mode=class_mode)
```

```
Found 12994 images belonging to 11 classes.
Found 3973 images belonging to 11 classes.
```

ภาพประกอบที่ 3.23 การ Augmentation และอ่านภาพ

3.9.1.2 ฟังก์ชันการสร้างโมเดลสำหรับ Fine-grained

ประยุกต์ใช้สถาปัตยกรรมของโมเดลที่ได้รับการเรียนรู้มาแล้วจาก ImageNet ให้เริ่มเรียนรู้เริ่มต้น เพิ่มและปรับโครงสร้างลำดับชั้นโมเดลให้เข้ากับงาน และสร้างโมเดลเพื่อนำไปใช้งาน

```
SIZE = 227
INPUT_SHAPE = [SIZE, SIZE, 3]
n_classes = len(classes)

def createModel(MODEL, Preprocessor, LEARNING_RATE , input_shape=INPUT_SHAPE, output_shape=n_classes):
  base_model = MODEL(weights='imagenet', include_top=False, input_shape=input_shape)
  inputs = Input(shape=input_shape)
  x = Preprocessor(inputs)
  x = base_model(x)
  x = GlobalAveragePooling2D()(x)
  outputs = Dense(output_shape, activation='softmax')(x)
  model = Model(inputs=inputs, outputs=outputs)

  model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),
                optimizer=tf.keras.optimizers.Adam(learning_rate=LEARNING_RATE),
                metrics=["accuracy"]
  )
  return model

incep_model = createModel(InceptionV3, incep_preprocess_input, 1e-5)
incep_model.summary()
```

ภาพประกอบที่ 3.24 ฟังก์ชันการสร้างโมเดลสำหรับ Fine-grained

3.9.1.3 ฟังก์ชันการเรียนรู้โมเดล

ฝึกโมเดลให้เรียนรู้ตามชุดข้อมูลและจำนวนรอบที่กำหนดด้วยการวนซ้ำบนชุดข้อมูล รวมทั้งประเมินความสูญเสีย และตรวจสอบการทำงานของโมเดล

```
def createTrain(model, callback, train_data, test_data, EPOCHS):
  STEP_SIZE_TRAIN=train_data.n//train_data.batch_size
  STEP_SIZE_TEST=test_data.n//test_data.batch_size

  model.fit(x=train_data,
            steps_per_epoch=STEP_SIZE_TRAIN,
            validation_data=test_data,
            validation_steps=STEP_SIZE_TEST,
            epochs=EPOCHS,
            validation_freq=1,
            callbacks=[callback])
  return model

incep_train = createTrain(incep_model, incep_callback, train_generator, test_generator, 15)
```

ภาพประกอบที่ 3.25 ฟังก์ชันการเรียนรู้โมเดล

3.9.2 ส่วนของการทำ Multi-label model

3.9.2.1 อ่านชุดข้อมูล

อ่านไฟล์ชุดข้อมูลทั้งหมดสำหรับการฝึกและทดสอบโมเดล

```
MULTI_DIR ='/content/drive/MyDrive/Colab Notebooks/Test_project/_Pro1/Dog_Dataset/MultiLabel'
TRAIN_DIR = '%s/images/MultiLabelFix7.csv'%MULTI_DIR
TEST_DIR = '%s/Test' % MULTI_DIR

df = pd.read_csv(TRAIN_DIR)
def append_ext(fn):
    return fn+'.jpg'

df['id'] = df['id'].apply(append_ext)
df
```

	id	drop_ears	set_ears	long_tail	short_tail	black	dark_brown	light_brown	white	long_fur	short_fur	black_collar	blue_collar	gold_collar	green_collar	red_collar	silver_collar	x_collar
0	Beagle1.jpg	1	0	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0
1	Beagle4.jpg	1	0	1	0	1	1	0	1	0	1	0	0	0	0	0	0	1
2	Beagle5.jpg	1	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1
3	Beagle7.jpg	1	0	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0
4	Beagle8.jpg	1	0	0	1	1	1	0	1	0	1	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13859	Doberman1518.jpg	1	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	1
13860	Doberman1519.jpg	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	0
13861	Doberman1520.jpg	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1
13862	Doberman1521.jpg	1	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	1
13863	Doberman1522.jpg	0	1	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0

13864 rows × 19 columns

ภาพประกอบที่ 3.26 อ่านชุดข้อมูล

3.9.2.2 ฟังก์ชันการสร้างชุดข้อมูลเฉพาะ

สร้างชุดข้อมูลเฉพาะของลักษณะที่ต้องการจากลักษณะทั้งหมด รวมทั้งสร้างกลุ่มรายการ (classes) เพื่อสร้างชุดข้อมูลใหม่

```
3
4 def create_dfFunc(name, df=df):
5   label = ['id']
6   if name == 'ears':
7     labels = 'drop_ears', 'set_ears'
8     label.extend(labels)
9     df_label = df[label]
10  elif name == 'tail':
11    labels = ['long_tail', 'short_tail']
12    label.extend(labels)
13    df_label = df[label]
14  elif name == 'color':
15    labels = ['black', 'dark_brown', 'light_brown', 'white']
16    label.extend(labels)
17    df_label = df[label]
18  elif name == 'fur':
19    labels = ['long_fur', 'short_fur']
20    label.extend(labels)
21    df_label = df[label]
22  elif name == 'collar':
23    labels = ['black_collar', 'blue_collar', 'gold_collar', 'green_collar','red_collar', 'silver_collar', 'x_collar']
24    label.extend(labels)
25    df_label = df[label]
26
27  df_label['classes'] = df_label.apply(lambda n: df_label.columns[n==1.0].tolist(), axis = 1)
28  return labels, df_label
29
```

ภาพประกอบที่ 3.27 ฟังก์ชันการสร้างชุดข้อมูลเฉพาะ

3.9.2.3 ฟังก์ชันการสร้างชุดข้อมูลและอ่านภาพ

ทำการสร้างชุดข้อมูลจากไฟล์รูปภาพจาก dataframe ที่มีข้อมูลอยู่ จะ return ผลลัพธ์ แล้วจะทำการแบ่งชุดข้อมูลเรียนรู้ (training) และชุดข้อมูลทดสอบ (validation)

```
1 BATCH = 32
2 SIZE = 227
3 target_size = (SIZE,SIZE)
4 class_mode = "categorical"
5
6 def createGenerator(label, DIR=TEST_DIR, target_size=target_size, batch_size=BATCH, class_mode=class_mode):
7   labels, df_label = create_dfFunc(label)
8
9   datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
10
11  train_generator = datagen.flow_from_dataframe(dataframe=df_label,
12                                                directory=DIR,
13                                                x_col="id",
14                                                y_col="classes",
15                                                subset="training",
16                                                batch_size=batch_size,
17                                                shuffle=True,
18                                                class_mode=class_mode,
19                                                target_size=target_size)
20  test_generator = datagen.flow_from_dataframe(dataframe=df_label,
21                                                directory=DIR,
22                                                x_col="id",
23                                                y_col="classes",
24                                                subset="validation",
25                                                batch_size=batch_size,
26                                                shuffle=False,
27                                                class_mode=class_mode,
28                                                target_size=target_size)
29  return train_generator, test_generator, labels, df_label
30
31 label = "color"
32 train_generator, test_generator, labels, df_label = createGenerator(label)
33 output = len(test_generator.class_indices)
34
35 df_label
```

ภาพประกอบที่ 3.28 ฟังก์ชันการสร้างชุดข้อมูลและอ่านภาพ

```
Found 11092 validated image filenames belonging to 4 classes.
Found 2772 validated image filenames belonging to 4 classes.
```

	id	black	dark_brown	light_brown	white	classes
0	Beagle1.jpg	1	1	0	1	[black, dark_brown, white]
1	Beagle4.jpg	1	1	0	1	[black, dark_brown, white]
2	Beagle5.jpg	1	1	0	1	[black, dark_brown, white]
3	Beagle7.jpg	0	0	1	1	[light_brown, white]
4	Beagle8.jpg	1	1	0	1	[black, dark_brown, white]
...	...	...	...	...	...	...
13859	Doberman1518.jpg	1	0	1	0	[black, light_brown]
13860	Doberman1519.jpg	1	0	0	0	[black]
13861	Doberman1520.jpg	1	0	1	0	[black, light_brown]
13862	Doberman1521.jpg	1	0	1	0	[black, light_brown]
13863	Doberman1522.jpg	0	1	1	0	[dark_brown, light_brown]

13864 rows × 6 columns

ภาพประกอบที่ 3.29 ตัวอย่างการสร้างชุดข้อมูลของสีตัว

3.9.2.4 ฟังก์ชันการสร้างโมเดลสำหรับ Multi-label

ประยุกต์ใช้สถาปัตยกรรมของโมเดลที่ได้รับการเรียนรู้มาแล้วจาก ImageNet ให้เริ่มเรียนรู้เริ่มต้น เพิ่มและปรับโครงสร้างลำดับชั้นโมเดลให้เข้ากับงาน และสร้างโมเดลเพื่อนำไปใช้งาน รวมทั้งกำหนด activation สำหรับประเภทผลลัพธ์ที่ต้องการและการกำหนด loss

```
5
6 SIZE = 227
7 INPUT_SHAPE = [SIZE, SIZE, 3]
8
9 def createModel(MODEL, Preprocessor, activation, output, LEARNING_RATE, input_shape=INPUT_SHAPE):
10   base_model = MODEL(weights='imagenet', include_top=False, input_shape=input_shape)
11   inputs = Input(shape=input_shape)
12   x = Preprocessor(inputs)
13   x = base_model(x)
14   x = GlobalAveragePooling2D()(x)
15   outputs = Dense(output, activation=activation)(x)
16   model = Model(inputs=inputs,outputs=outputs)
17
18   if activation == 'softmax':
19     Loss = tf.keras.losses.CategoricalCrossentropy()
20   elif activation == 'sigmoid':
21     Loss = tf.keras.losses.BinaryCrossentropy()
22
23   model.compile(loss=Loss,
24                 optimizer=tf.keras.optimizers.Adam(learning_rate = LEARNING_RATE),
25                 metrics=["accuracy"]
26   )
27   return model
28
29 output = len(test_generator.class_indices)
30 incep_model =createModel(InceptionV3, incep_preprocess_input, 'sigmoid', output, 1e-5)
31 incep_model.summary()
32
```

ภาพประกอบที่ 3.30 ฟังก์ชันการสร้างโมเดลสำหรับ Multi-label

ฟังก์ชันการเรียนรู้โมเดล เพื่อฝึกโมเดลสำหรับ Fine-grained และ Multi-label สร้างและเรียกใช้งานเหมือนกัน