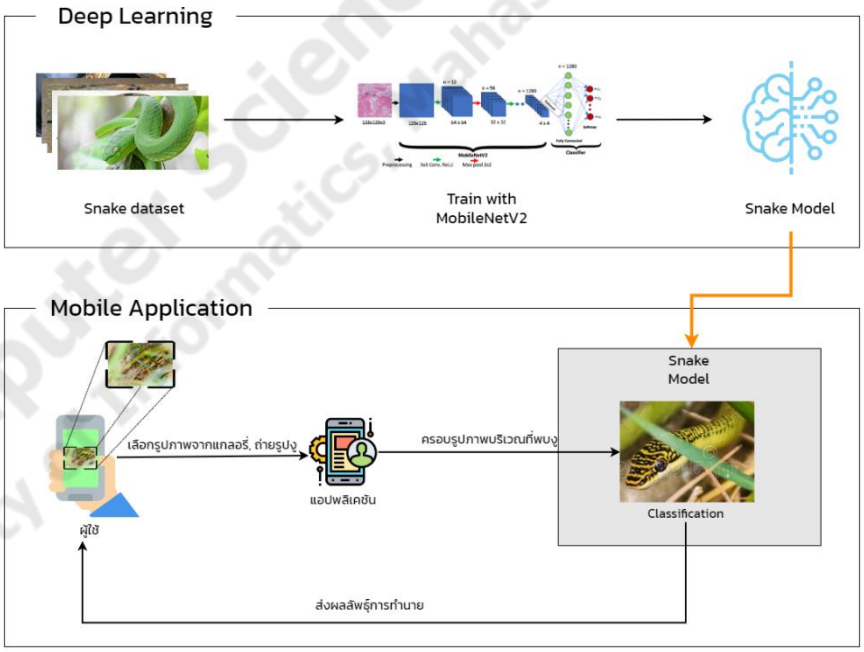
บทที่ 3

วิธีดำเนินงานวิจัย

ในบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานของโครงงานปริญญานิพนธ์ ซึ่งจะทำให้ทราบถึงการวิเคราะห์ระบบ กระบวนการประมวลผลของระบบ และขั้นตอนการทำงานของระบบเป็นอย่างไร โดยขั้นตอนในการดำเนินงานวิจัยมีรายละเอียดดังนี้

1. กรอบการดำเนินงาน
2. Dataset
3. การทำงานของ MobileNetV2
4. ขั้นตอนในการพัฒนา Mobile Application
5. ขั้นตอนในการพัฒนา Snake Model (Deep learning)

3.1 กรอบการดำเนินงาน



ภาพประกอบที่ 3.1 กรอบการดำเนินงาน

จากภาพประกอบที่ 3.1 เป็นภาพรวมในการดำเนินงานของงาน โดยจะแบ่งเป็นสองส่วนโดยส่วนแรกคือ Deep learning จะเป็นส่วนที่ใช้ในการสร้าง model ที่ใช้ในการจำแนกสายพันธุ์งู และส่วนที่สองคือ Mobile Application เป็นส่วนในการสร้าง application โดยแต่ละส่วนมีรายละเอียดดังต่อไปนี้

3.1.1 Deep Learning

จากภาพประกอบที่ 3.1 ในส่วนของ deep learning จะเป็นขั้นตอนในการสร้าง Snake model ที่ใช้ในการ classification ที่จะนำไปใช้ใน Mobile Application ก่อนที่จะสร้าง Snake Model จะต้องมี Dataset ของงูก่อน โดย dataset ที่ได้จะรวบรวมจากหลากหลายเว็บไซต์ หลังจากนั้นจะนำ dataset ที่ได้ไปใช้ในการ train model โดยใช้ pre-train model ของ MobileNetV2 architecture พอ train model ได้ตามที่ต้องการแล้วจากนั้นจะบันทึก model เป็นไฟล์ tflite เพื่อนำไปใช้ใน mobile application

3.1.2 Mobile Application

จากภาพประกอบที่ 3.1 ในส่วนของ Mobile Application โดยขั้นตอนในการใช้งาน Snake model คือผู้ใช้ต้องถ่ายรูปภาพงูหรือเลือกรูปภาพงูจากแกลเลอรี่และหลังจากนั้นผู้ใช้สามารถครอบตัดเฉพาะส่วนงูได้ หลังจากยืนยันการครอบตัดแล้ว model จะ classification และส่งผลลัพธ์ 3 อันดับแรกที่ใกล้เคียงกลับมาให้ผู้ใช้

3.2 Dataset

โครงงานได้รวบรวมข้อมูลและภาพของงูชนิดต่าง ๆ จากหลากหลายเว็บไซต์บนอินเทอร์เน็ต เช่น สถานเสาวภา สภากาชาดไทย, วิกิพีเดีย, zoothailand เป็นต้น โดยข้อมูลที่รวบรวมมาจะมีแหล่งที่พบอาการหลังได้รับพิษ ลักษณะของพิษ พฤติกรรม และลักษณะทางกายภาพของงูแต่ละชนิด

โครงงานนี้ได้ใช้ชุดข้อมูลของงู 20 สายพันธุ์(ภาพประกอบที่ 3.2)ในการสร้างโมเดล ซึ่งจะมีรูปภาพทั้งหมด 4,069 รูป ประกอบด้วยชุดข้อมูลสำหรับการ Train 3,269 รูป และ Test 800 รูป ดังตารางที่ 3.1 รูปภาพอินพุตทั้งหมดจะถูกปรับขนาดเป็น 224*224*3 รูปภาพทั้งหมดได้ทำการรวบรวมมาจากเว็บไซต์ต่างๆ เช่น iNaturalist, dreamstime, iStock

ภาพประกอบที่ 3.2 ตัวอย่างงูแต่ละสายพันธุ์

ตารางที่ 3.1 รายละเอียดชุดข้อมูลรูปภาพ

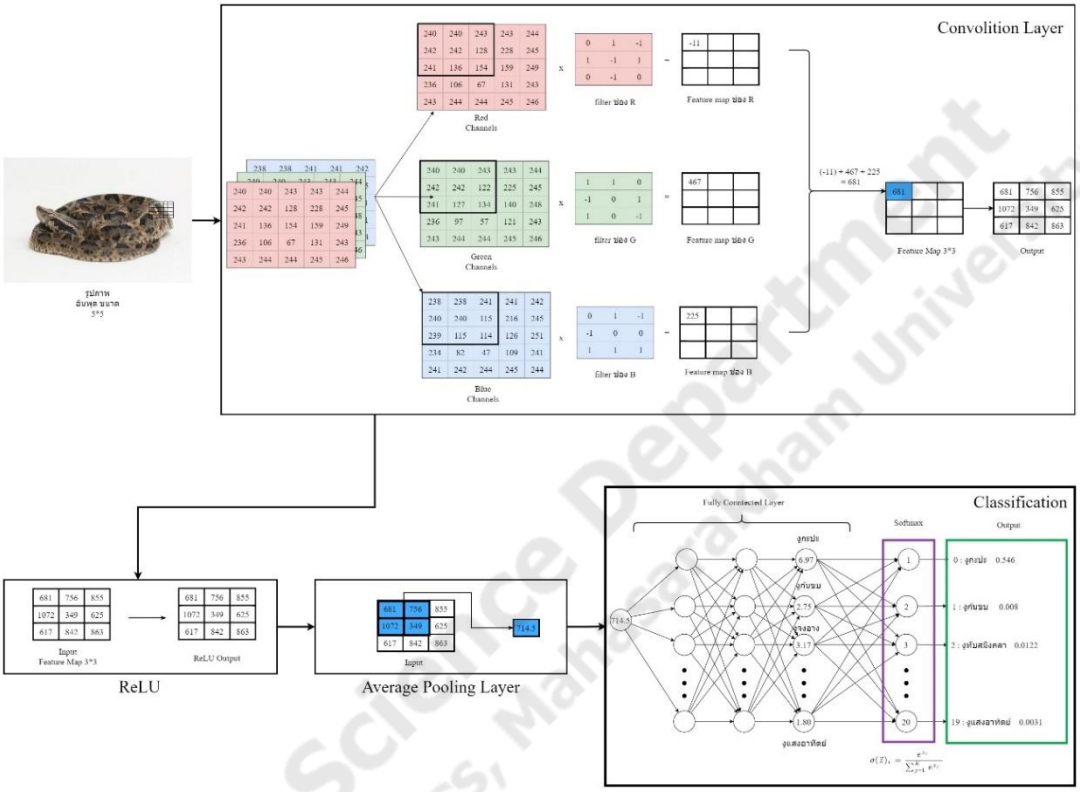
ลำดับที่	สายพันธุ์งู	สัดส่วนของชุดข้อมูล		
		Train(80%)	Validation(20%)	Test
1	งูเห่า	140	34	40
2	งูเหลือม	134	33	40
3	งูหลาม	128	32	40
4	งูแสงอาทิตย์	127	31	40
5	งูสิงหางลาย	128	32	40
6	งูสิง	135	33	40
7	งูสามเหลี่ยม	128	32	40
8	งูสมิงทะเลปากเหลือง	140	35	40
9	งูลายสาบคอแดง	128	32	40
10	งูลายสอ	136	34	40
11	งูแมวเซา	130	32	40
12	งูปี่แก้วลายแต้ม	128	32	40

ตารางที่ 3.1 รายละเอียดชุดข้อมูลรูปภาพ (ต่อ)

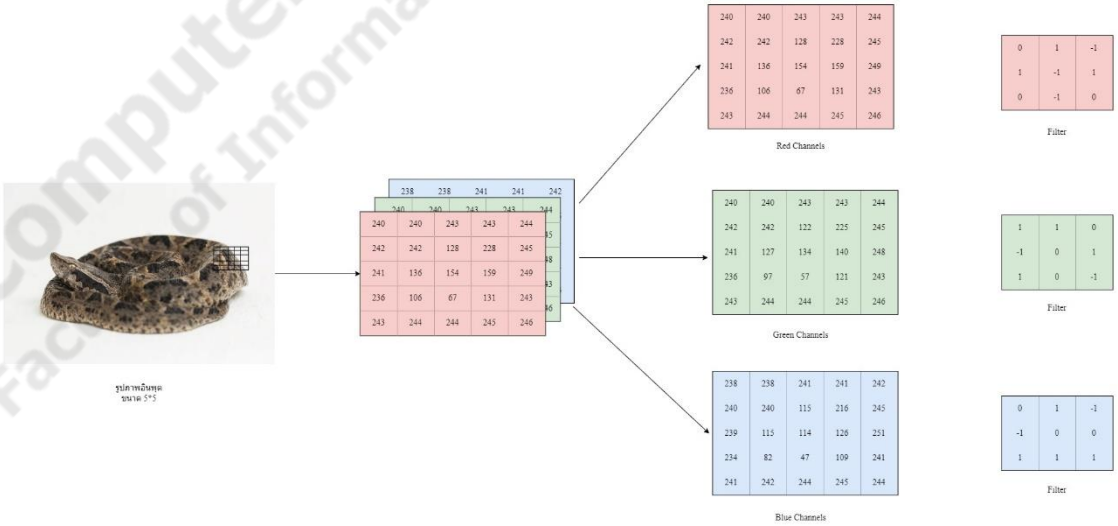
ลำดับ	สายพันธุ์งู	สัดส่วนของชุดข้อมูล		
		Train(80%)	Validation(20%)	Test
13	งูปล้องฉนวนสร้อยเหลือง	131	32	40
14	งูทางมะพร้าว	130	32	40
15	งูทับสมิงคลา	132	32	40
16	งูจงอาง	128	32	40
17	งูเขียวหางไหม้ท้องเหลือง	131	32	40
18	งูเขียวพระอินทร์	128	32	40
19	งูกะปะ	128	32	40
20	งูก้นขบ	131	32	40
รวม		3269		800

3.3 การทำงานของ MobileNetV2

3.3.1 วิธีการทำงานของ MobileNetV2



ภาพประกอบที่ 3.3 ภาพรวมวิธีการทำงานของ MobileNetV2

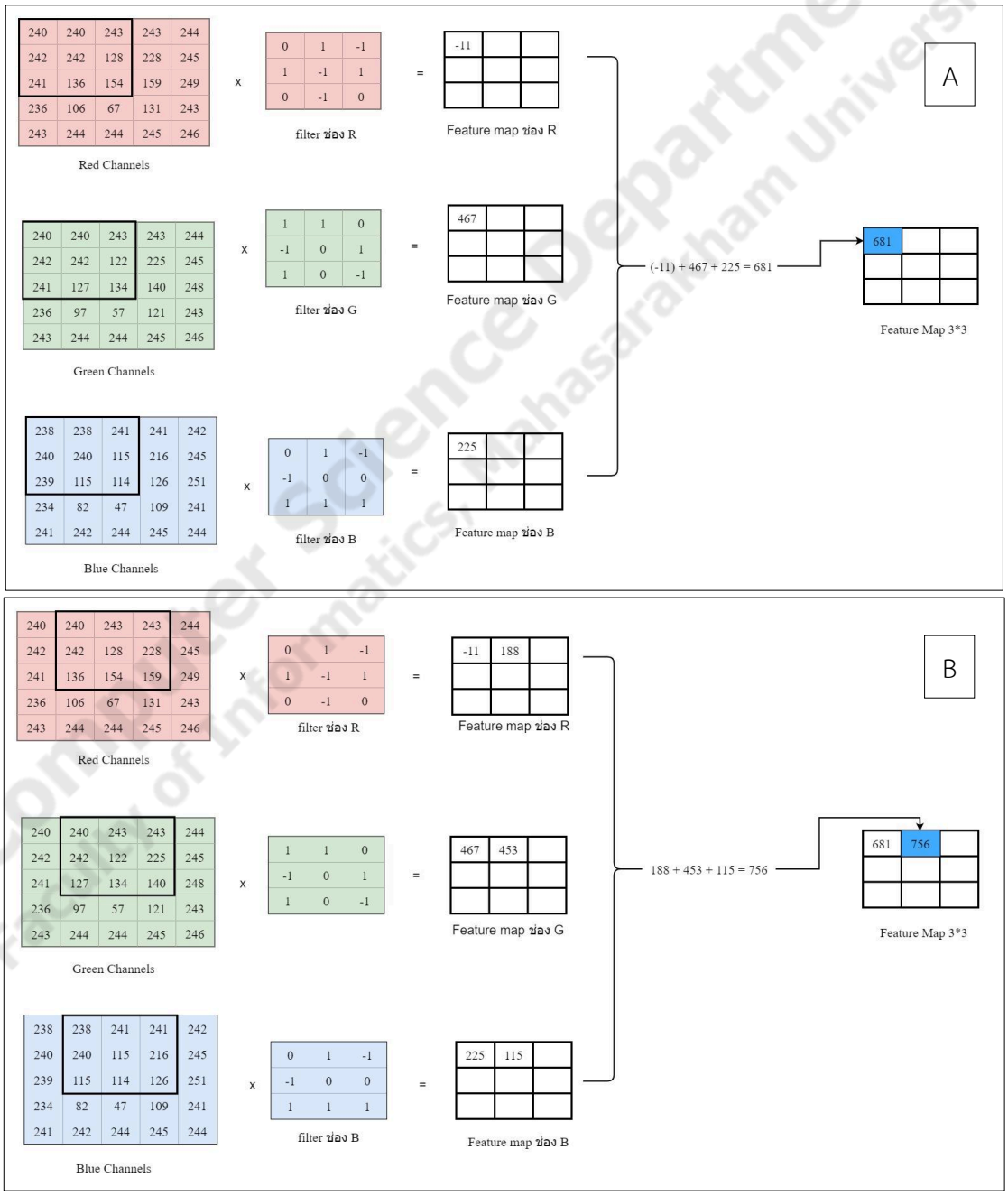


ภาพประกอบที่ 3.4 ตัวอย่างรูปภาพอินพุต

รูปภาพอินพุตมีขนาด 224 x 224 x 3 แต่ตัวอย่างที่ใช้ในการแสดงวิธีการทำงานของโมเดลจะใช้รูปภาพอินพุตขนาด 5 x 5 x 3 เพื่อให้ง่ายต่อการแสดงวิธีการคำนวณ โดยวิธีการทำงานของโมเดล MobileNetV2 หลัก ๆ มีดังต่อไปนี้

1) การทำงานของ Convolutional layer

Convolutional layer จะรับรูปภาพอินพุต โดยรูปภาพจะถูกแบ่งเป็นช่อง pixel ซึ่งในแต่ละช่องจะเก็บค่าสีเอาไว้ Convolutional layer จะนำค่าสีเหล่านี้ไปคำนวณด้วยการนำไปคูณกับ filter เพื่อนำมาค้นหาคุณลักษณะแล้วเก็บไว้ใน Feature map



ภาพประกอบที่ 3.5 การ Convolution

ขนาดของ Feature map จะถูกปรับตามขนาดของรูปภาพอินพุต, ขนาดของ filter, จำนวน padding และ stride โดยมีสมการดังนี้

$$D_R = \frac{D_1 - D_F + 2p}{s} + 1 \text{ ; flooring} \quad (5)$$

โดยที่

- D_R คือขนาดของ Feature map
- D_F คือขนาดของ Filter
- D_1 คือขนาดของรูปอินพุต
- p คือขนาดของ padding
- s คือจำนวนของ stride

กำหนดค่าเพื่อทำการคำนวณ โดยนำขนาดของรูปภาพอินพุตและขนาดของ filter มาจากภาพประกอบที่ 3.4 โดยรูปภาพอินพุตมีขนาด 5 x 5 x 3 และ filter ขนาด 3 x 3 x 3 Padding เป็น 0 และ Stride เป็น 1 จากนั้นแทนในสมการ D_R เพื่อหาขนาด feature map

$$D_R = \frac{5-3+2(0)}{1} + 1 = 3 \quad (6)$$

ดังนั้น feature map ที่ได้จะมีขนาด 3 x 3 โดยการทำ convolution กับรูปภาพอินพุตที่มีมากกว่า 1 channels จะได้ feature map ตามจำนวน channel ที่มี และจะรวม feature map เข้าด้วยกันเหลือโดยนำตำแหน่งเดียวกันแต่ละ feature map มาบวกกัน

จากภาพประกอบที่ 3.5 แสดงตัวอย่างการคำนวณส่วนย่อยของรูปภาพอินพุตคูณกับ Filter ขนาด 3 x 3 x 3 ของทั้ง 3 Channels

- ในรอบที่ 1 (A)

Channel 1 = (240x0)+(240x1)+(243x-1)+(242x1)+(242x-1)+(128x1)+(241x0)+(136x-1)+(154x0) = -11

Channel 2 = (240x1)+(240x1)+(243x0)+(242x-1)+(242x0)+(122x1)+(241x1)+(127x0)+(134x-1) = 467

Channel 3 = (238x0)+(238x1)+(241x-1)+(240x-1)+(240x0)+(115x0)+(239x1)+(115x1)+(114x1) = 225

- ในรอบที่ 2 (B)

Channel 1 = (240x0)+(243x1)+(243x-1)+(242x1)+(128x-1)+(228x1)+(136x0)+(154x-1)+(159x0) = 118

Channel 2 = (240x1)+(243x1)+(243x0)+(242x-1)+(122x0)+(255x1)+(127x1)+(134x0)+(140x-1) = 453

Channel 3 = (238x0)+(241x1)+(241x-1)+(240x-1)+ +(115x0)+(216x0)+(115x1)+

(114x1)+(126x1) = 115

จากผลลัพธ์ด้านบนทำไปเรื่อย ๆ โดยการเลื่อนตำแหน่งของ filter ไปให้ทั่วทั้งภาพผลลัพธ์ทั้งหมดจากการคำนวณได้ดังภาพประกอบที่ 3.6

681	756	855
1072	349	625
617	842	863

Feature Map 3*3

ภาพประกอบที่ 3.6 ผลลัพธ์ feature map จากการทำ Convolution

2) Activation Function ReLu[18]

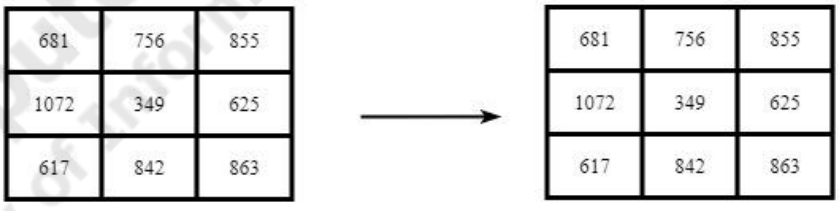
ผลลัพธ์ที่ได้จากการทำ Convolution ในแต่ละตำแหน่งจะแปลงค่าด้วยฟังก์ชัน ReLU ที่เป็นการแปลงแบบไม่เป็นเชิงเส้น เพื่อความง่ายในการคำนวณและประเมิน ประสิทธิภาพผลลัพธ์ สามารถคำนวณด้วยสมการดังนี้.

$$R(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \quad (7)$$

โดยที่

- x คือ ค่าตัวเลขใน feature map

โดยจากภาพประกอบที่ 3.6 ค่าในตารางมีค่าเป็นบวกทั้งหมดทำให้เมื่อทำ ReLu ก็ได้ผลลัพธ์เท่าเดิม



ภาพประกอบที่ 3.7 การทำ ReLu

ตัวอย่างคำนวณของ ReLu จากภาพประกอบที่ 3.7 จะได้ดังต่อไปนี้

$ReLu(x_{681}) = \max(0, 681) = 681$

$ReLu(x_{756}) = \max(0, 756) = 756$

$ReLu(x_{855}) = \max(0, 855) = 855$

$ReLu(x_{1072}) = \max(0, 1072) = 1072$

$ReLu(x_{349}) = \max(0, 349) = 349$

$ReLu(x_{625})$ = max(0, 625) = 625

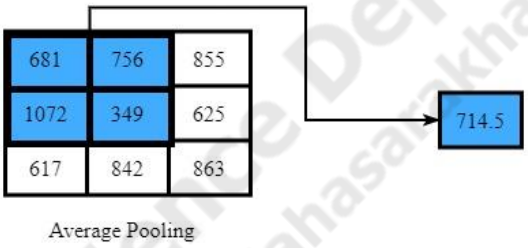
$ReLu(x_{617})$ = max(0, 617) = 617

$ReLu(x_{842})$ = max(0, 842) = 842

$ReLu(x_{863})$ = max(0, 863) = 863

3) Pooling Layer

หลังจากที่ได้ผลลัพธ์จากการทำ ReLu(ภาพประกอบที่ 3.7) จะนำผลลัพธ์ที่ได้มาทำ Pooling เพื่อลดจำนวนของพารามิเตอร์ในกรณีที่ภาพมีขนาดใหญ่เกินไป ด้วยการลดมิติลงแต่ยังคงข้อมูลสำคัญเอาไว้ โดย MobileNetV2 จะใช้ Average Pooling ในการคำนวณ และ Pooling จะมี Stride คือ 2 ทำให้การเลื่อนหาผลลัพธ์บนภาพประกอบที่ 3.7 เลื่อนได้เพียงครั้งเดียว ทำให้ขนาด pooling ที่ได้คือ 1 x 1



ภาพประกอบที่ 3.8 การทำ Pooling

ตัวอย่างการคำนวณ Average Pooling จากภาพประกอบที่ 3.8

pooling = (681+756+1072+349)/4 = 714.5

4) Fully Connected

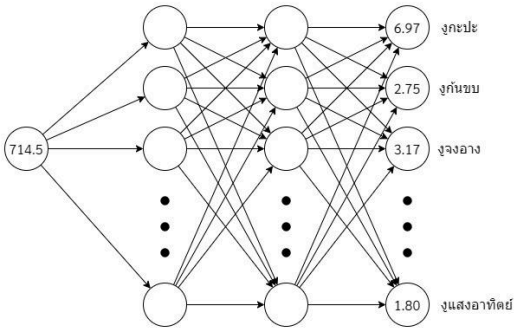
หลังจากที่ทำ Average Pooling แล้วจะแปลงข้อมูลให้อยู่ในรูปแบบ vector เรียกว่าการทำ Flatten และมองค่าข้อมูลใน vector แต่ละตัวเป็นโหนด โดยโหนด 1 โหนดจะต้องมีเส้นเชื่อมไปหาโหนดที่อยู่เลเวลถัดไปครบทุกตัว โดยค่าของโหนดถัดไปหาได้จากสมการต่อไปนี้

$$y=W_i x_i+B \quad (8)$$

โดยที่

- y คือค่าในโหนดถัดไป
- W_i คือค่าน้ำหนักในแต่ละเส้นที่เชื่อมเข้าหาโหนดถัดไป
- x_i คือค่าสีที่อยู่ใน vector หรือโหนดนั้น ๆ
- B คือ bias

เนื่องจากผลลัพธ์จากการทำ Pooling(ภาพประกอบที่ 3.8) ได้ค่าเดียวทำให้มีโหนดเริ่มต้นโหนดเดียว และโหนดสุดท้ายจะมีจำนวน 20 คลาสตามจำนวนสายพันธุ์งูที่ใช้ในการฝึกฝนโมเดล



ภาพประกอบที่ 3.9 Fully Connected

ตัวอย่างการคำนวณในชั้นก่อนถึงโหนดสุดท้ายเพื่อหาโหนดสุดท้าย

$y_{งูกะปะ}$ = (714.5 * 0.00962) + 0.0919 = 6.96539

$y_{งูก้นขบ}$ = (714.5 * 0.00374) + 0.0748 = 2.74703

$y_{งูจงอาง}$ = (714.5 * 0.00449) + (-0.0421) = 3.166

$y_{งูทับสมิงคลา}$ = (714.5 * 0.00344) + (-0.0251) = 2.4327

$y_{งูทางมะพร้าว}$ = (714.5 * 0.00421) + (-0.0139) = 2.994145

$y_{งูปล้องฉนวนสร้อยเหลือง}$ = (714.5 * 0.00174) + 0.0119 = 1.25513

$y_{งูปี่แก้วลายแต้ม}$ = (714.5 * -0.0014) + (-0.086) = -1.0863

$y_{งูลายสอ}$ = (714.5 * 0.00387) + 0.00387 = 2.803

$y_{งูลายสาบคอแดง}$ = (714.5 * 0.0089) + (-0.0039) = 6.355

$y_{งูสมิงทะเลปากเหลือง}$ = (714.5 * 0.00324) + (-0.0649) = 2.25

$y_{งูสามเหลี่ยม}$ = (714.5 * 0.0023) + 0.0189 = 1.66

$y_{งูสิง}$ = (714.5 * 0.00426) + (-0.0361) = 3.007

$y_{งูสิงหางลาย}$ = (714.5 * 0.00565) + 0.0055 = 4.0424

$y_{งูหลาม}$ = (714.5 * 0.00519) + 0.0482 = 3.7564

$y_{งูเขียวพระอินทร์}$ = (714.5 * 0.0018) + (-0.0834) = 1.2027

$y_{งูเขียวหางไหม้ท้องเหลือง}$ = (714.5 * 0.00237) + (-0.0181) = 1.675265

$y_{งูเหลือม}$ = (714.5 * -0.0022) + 0.0177 = -1.5542

$y_{งูเห่า}$ = (714.5 * 0.00365) + 0.0388 = 2.6467

$y_{งูแมวเซา}$ = (714.5 * 0.00542) + 0.0218 = 3.89439

$y_{งูแสงอาทิตย์}$ = (714.5 * 0.00245) + 0.0519 = 1.8024

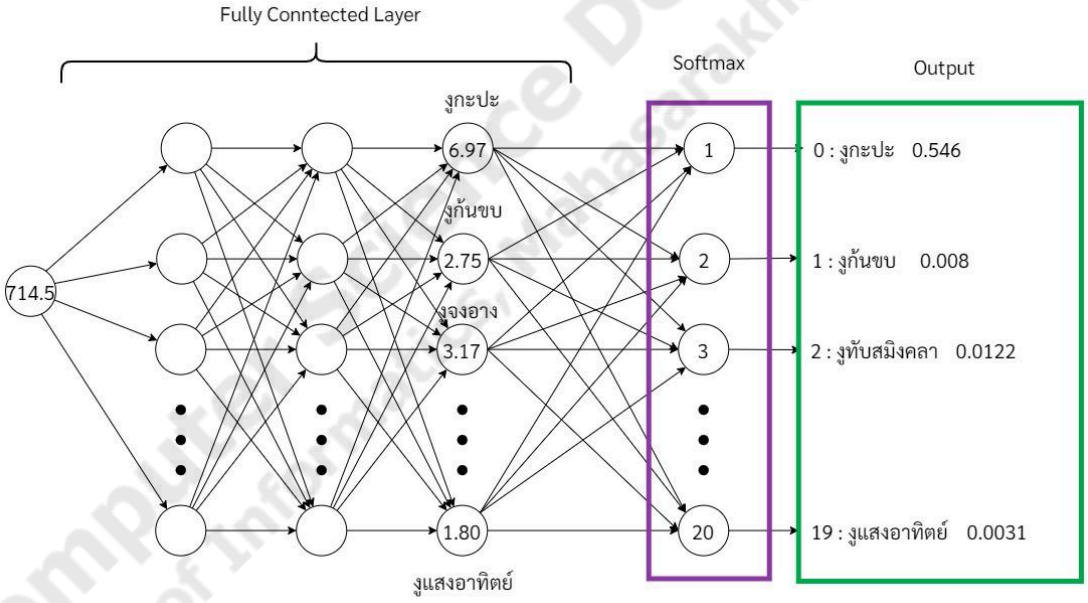
5) Softmax

Softmax function เป็นฟังก์ชันที่จะแปลงคะแนนผลลัพธ์ของชั้นสุดท้ายในแต่ละโหนด(ภาพประกอบที่ 3.9)ของโครงข่ายประสาทเทียมเป็นค่าความน่าจะเป็นตามสัดส่วนของคลาส คลาสไหนที่มีค่าความน่าจะเป็นมากที่สุดโมเดลจะเลือกคลาสนั้นเป็นคำตอบ

$$\sigma(\vec{Z})_i = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}} \quad (9)$$

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
- $\sum_{j=1}^{K} e^{Z_j}$ คือ ผลรวมของ e^{Z_i} ทุกตัว



ภาพประกอบที่ 3.10 Softmax

ตามภาพประกอบที่ 3.10 ที่โหนดสุดท้ายของ fully connected layer คือค่าของแต่ละคลาส โดยที่ค่า Z_i คือค่าของแต่ละคลาส และสามารถนำมาคำนวณหาค่ามาตรฐานเอกซ์โพเนนเชียลได้ดังต่อไปนี้ภาพประกอบที่ 3.10

$e^{Z_1} = e^{6.96539} = 1059.323$

$e^{Z_2} = e^{2.74703} = 15.596$

$e^{Z_3} = e^{3.166} = 23.7125$

$e^{Z_4} = e^{2.4327} = 11.390$

$e^{Z_5} = e^{2.994145} = 19.968$

$e^{Z_6} = e^{1.25513} = 3.5082$

$e^{Z_7} = e^{-1.0863} = 0.3374$

$e^{Z_8} = e^{2.803} = 16.507$

$e^{Z_9} = e^{6.355} = 575.446$

$e^{Z_{10}} = e^{2.25} = 9.4884$

$e^{Z_{11}} = e^{1.66} = 5.2711$

$e^{Z_{12}} = e^{3.007} = 20.2401$

$e^{Z_{13}} = e^{4.0424} = 56.964$

$e^{Z_{14}} = e^{3.7564} = 42.796$

$e^{Z_{15}} = e^{1.2027} = 3.329$

$e^{Z_{16}} = e^{1.675265} = 5.340$

$e^{Z_{17}} = e^{-1.5542} = 0.2113$

$e^{Z_{18}} = e^{2.6467} = 14.1077$

$e^{Z_{19}} = e^{3.89439} = 49.1259$

$e^{Z_{20}} = e^{1.8024} = 6.0643$

หลังจากได้ค่ามาตรฐานเอกซ์โพเนนเชียลแล้วจะนำไปหาผลรวม

$$\sum_{j=1}^{K} e^{Z_j} = e^{Z_1}+e^{Z_2}+\ldots+e^{Z_{20}} = 1059323 + 15.596 + 23.7125 + 11.390 + 19.968 + 3.5082 + 0.3374 + 16.507 + 575.446 + 9.4884 + 5.2711 + 20.2401 + 56.964 + 42.796 + 3.329 + 5.340 + 0.2113 + 14.1077 + 49.1259 + 6.0643 = 1938.7288$$

หลังจากที่ได้ค่ามาตรฐานเอกซ์โพเนนเชียลของแต่และคลาสและได้ผลรวมแล้วจะนำไปหาความน่าจะเป็นดังต่อไปนี้

$$\sigma(z)_1 = \frac{1059.323}{1938.7288} = 0.546$$

$$\sigma(z)_2 = \frac{15.596}{1938.7288} = 0.008$$

$$\sigma(z)_3 = \frac{23.7125}{1938.7288} = 0.0122$$

$$\sigma(z)_4 = \frac{11.390}{1938.7288} = 0.00587$$

$$\sigma(z)_5 = \frac{19.968}{1938.7288} = 0.01$$

$$\sigma(z)_6 = \frac{3.5082}{1938.7288} = 0.0018$$

$$\sigma(z)_7 = \frac{16.507}{1938.7288} = 0.0001$$

$$\sigma(z)_8 = \frac{16.507}{1938.7288} = 0.0085$$

$$\sigma(z)_9 = \frac{575.446}{1938.7288} = 0.2968$$

$$\sigma(z)_{10} = \frac{9.4884}{1938.7288} = 0.00489$$

$$\sigma(z)_{11} = \frac{5.2711}{1938.7288} = 0.0027$$

$$\sigma(z)_{12} = \frac{20.2401}{1938.7288} = 0.0104$$

$$\sigma(z)_{13} = \frac{56.964}{1938.7288} = 0.0293$$

$$\sigma(z)_{14} = \frac{42.796}{1938.7288} = 0.02207$$

$$\sigma(z)_{15} = \frac{3.329}{1938.7288} = 0.0017$$

$$\sigma(z)_{16} = \frac{5.304}{1938.7288} = 0.0027$$

$$\sigma(z)_{17} = \frac{0.2113}{1938.7288} = 0.0001$$

$$\sigma(z)_{18} = \frac{14.1077}{1938.7288} = 0.0072$$

$$\sigma(z)_{19} = \frac{49.1259}{1938.7288} = 0.0253$$

$$\sigma(z)_{20} = \frac{6.0643}{1938.7288} = 0.0031$$

เมื่อคำนวณเวกเตอร์ของอินพุตครบทุกค่า ผลลัพธ์จะเป็นได้ตั้งแต่ 0 จนถึง 1 ซึ่งถ้ารวมกันทั้งหมดจะได้เท่ากับ 1 เสมอ

ตารางที่ 3.2 ตารางการคำนวณ Softmax

i	Z_i	e^{Z_1}	$\sigma(z)_i$
1	6.96539	1059.323	0.546
2	2.74703	15.596	0.008
3	3.166	23.7125	0.0122
4	2.4327	11.390	0.00587
5	2.994145	19.968	0.01
6	1.25513	3.5082	0.0018
7	-1.0863	0.3374	0.0001
8	2.803	16.507	0.0085
9	6.355	575.446	0.2968
10	2.25	9.4884	0.00489
11	1.66	5.2711	0.0027
12	3.007	20.2401	0.0104
13	4.0424	56.964	0.0293
14	3.7564	42.796	0.02207
15	1.2027	3.329	0.0017
16	1.675265	5.340	0.0027
17	-1.5542	0.2113	0.0001
18	2.6467	14.1077	0.0072
19	3.89439	49.1259	0.0253
20	1.8024	6.0643	0.0031
รวม		1938.7288	1.0

ลำดับคลาสที่มีความแม่นยำสูงที่สุด 3 อันดับแรกได้ดังนี้

1.คลาส 1 : 0.546 = งูกะปะ

2.คลาส 9 : 0.2968 = งูลายสาบคอแดง

3.คลาส 13 : 0.0293 = งูสิงหางลาย

โดยภายใน Snake model มีการเรียงอันดับคลาสดังนี้

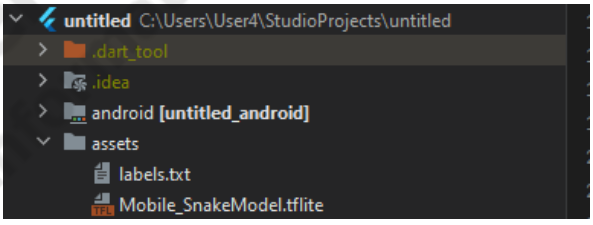
งูกะปะ : 0, งูก้นขบ : 1 งูจงอาง : 2, งูทับสมิงคลา : 3, งูทางมะพร้าว : 4, งูปล้องฉนวนสร้อยเหลือง : 5, งูปี่แก้วลายแต้ม : 6, งูลายสอ : 7, งูลายสาบคอแดง : 8 , งูสมิงทะเลปากเหลือง : 9, งูสามเหลี่ยม : 10, งูสิง : 11, งูสิงหางลาย : 12, งูหลาม : 13, งูเขียวพระอินทร์ : 14, งูเขียวหางไหม้ท้องเหลือง : 15,งูเหลือม : 16, งูเห่า : 17, งูแมวเซา : 18, งูแสงอาทิตย์ : 19

3.4 ขั้นตอนในการพัฒนา Mobile Application

ใช้โปแกรม Android studio เพื่อออกแบบแอปพลิเคชัน และแอปพลิเคชันจะใช้ในการดูข้อมูลและรูปภาพงูชนิดต่าง ๆ ได้ รวมถึงวิธีการปฐมพยาบาลเบื้องต้นเมื่อโดนงูกัด และส่วนในการจำแนกรูปภาพงูโดยใช้การถ่ายรูปภาพหรือเลือกรูปภาพจากแกลเลอรี โดยหลังจากถ่ายรูปหรือเลือกรูปจะให้ผู้ใช้ครอบตัดเฉพาะบริเวณที่มีงูเพื่อเพิ่มความแม่นยำ โดย Model ที่นำเข้ามาในโปรแกรมจะถูกฝึกฝนด้วย MobileNetV2 และบันทึก Model เป็นประเภท TensorFlow lite(.tflite) เพื่อที่จะนำเข้ามาใช้ในแอปพลิเคชันได้

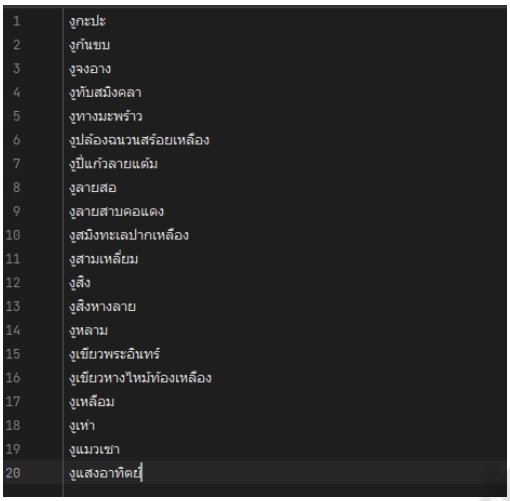
3.4.1 วิธีนำเข้า model เพื่อใช้ในแอปพลิเคชัน

โดย model ที่จะนำเข้ามาใช้ต้องเป็นไฟล์ประเภท TensorFlow lite(.tflite) และสิ่งที่ต้องนำเข้าเพื่อใช้คู่กับ model คือ labels โดยจะใช้เป็นไฟล์ประเภท text(.txt) โดยจะสร้าง โฟลเดอร์ใน Project ของ Android studio ชื่อว่า assets เก็บสองไฟล์นี้ไว้ ตามภาพประกอบที่ 3.11



ภาพประกอบที่ 3.11 การเก็บไฟล์สำหรับ model

รายละเอียดในไฟล์ labels.txt จะเป็นชื่อคลาสของงูทั้ง 20 สายพันธุ์ ดังภาพประกอบที่ 3.12



ภาพประกอบที่ 3.12 รายละเอียดไฟล์ labels.txt

เพิ่มชื่อโฟลเดอร์ - assets/ ในหัวข้อ assets ที่อยู่ใน pubspec.yaml เพื่อให้สามารถอ้างอิงถึงไฟล์ของ model และ labels ได้ ดังภาพประกอบที่ 3.13

```
assets:
  - images/
  - JSON/
  - assets/
```

ภาพประกอบที่ 3.13 การเพิ่มชื่อโฟลเดอร์ใน pubspec

หน้าที่จะดึงตัว model มาใช้งาน import library tflite ให้เรียบร้อย ดังภาพประกอบที่ 3.14

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

ภาพประกอบที่ 3.14 การ import tflite

สร้าง function สำหรับ load model เข้ามาเพื่อรอใช้งานดังภาพประกอบที่ 3.16 และให้ load model ทุกครั้งที่มาหน้านี้ ดังภาพประกอบที่ 3.15

```
@override
void initState() {
  loadModel();
  super.initState();
}
```

ภาพประกอบที่ 3.15 การ load model ทุกครั้งเมื่อใช้งาน

```
Future loadModel() async {
  Tflite.close();
  String? res;
  res = await Tflite.loadModel(
      model: "assets/Mobile_SnakeModel.tflite", labels: "assets/labels.txt");
  print("Model loading status: $res");
}
```

ภาพประกอบที่ 3.16 function สำหรับ load model

สร้าง function ที่จะนำรูปเข้าไป predict กับ model ที่ได้โหลดมาก่อนหน้านี้ โดยจะมี parameter 1 ตัวประเภท File ชื่อว่า image ดังภาพประกอบที่ 3.17 โดย function นี้จะถูกเรียกใช้ใน function การเลือกรูปภาพจากแกลเลอรีดังภาพประกอบที่ 3.18 และ function การถ่ายภาพด้วยกล้องดังภาพประกอบที่ 3.19 โดยทั้งใน function เลือกรูปและถ่ายรูปจะมีการเรียกใช้ function ที่ใช้ในการครอปตัดรูปภาพดังภาพประกอบที่ 3.20 เมื่อ function เลือกรูปหรือถ่ายรูปทำงานสำเร็จจะส่งผลลัพธ์เก็บไว้ที่ตัวแปรชื่อว่า _results เพื่อเตรียมนำไปแสดงผลลัพธ์ โดยกำหนดให้ numResults เท่ากับ 3 หมายความว่าจะได้ผลลัพธ์ที่ใกล้เคียงที่สุด 3 อันดับแรก

```
Future imageClassification(File image) async {
  var recognitions = await Tflite.runModelOnImage(
    path: image.path, // required
    imageMean: 0.0, // defaults to 117.0
    imageStd: 255.0, // defaults to 1.0
    numResults: 3, // defaults to 5
    threshold: 0.0, // defaults to 0.1
  );
  setState(() {
    _results = recognitions!;
    _image = image;
    imageSelect = true;
  });
}
```

ภาพประกอบที่ 3.17 function สำหรับการทำนายผลลัพธ์

```
Future pickImage() async {
  final ImagePicker picker = ImagePicker();
  final XFile? pickFile = await picker.pickImage(source: ImageSource.gallery);
  File image = File(pickFile!.path);
  image = await _cropImage(imageFile: image) as File;
  await imageClassification(image);
}
```

ภาพประกอบที่ 3.18 function สำหรับเลือกรูปภาพจากแกลเลอรี

```dart
Future capturePhoto() async {
  try {
    final ImagePicker picker = ImagePicker();
    final XFile? photo = await picker.pickImage(source: ImageSource.camera);
    File image = File(photo!.path);
    image = await _cropImage(imageFile: image) as File;
    await imageClassification(image);
  } catch (e) {
    print(e);
  }
}
```

ภาพประกอบที่ 3.19 function สำหรับถ่ายภาพด้วยกล้อง

```dart
Future<File?> _cropImage({required File imageFile}) async {
  CroppedFile? croppedImage =
      await ImageCropper().cropImage(sourcePath: imageFile.path, uiSettings: [
    AndroidUiSettings(
      toolbarWidgetColor: Colors.green,
      statusBarColor: Colors.green,
      lockAspectRatio: false,
      activeControlsWidgetColor: Colors.green,
    )
  ]);
  if (croppedImage == null) return null;
  return File(croppedImage.path);
}
```

ภาพประกอบที่ 3.20 function สำหรับครอบตัดรูปภาพ

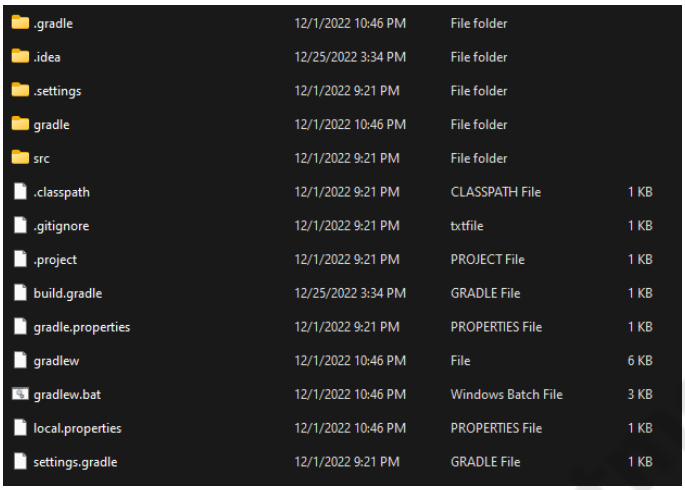
3.4.2 ในกรณีที่ tflite ใช้งานไม่ได้ ระบบแจ้งเตือนข้อผิดพลาด

```
FAILURE: Build failed with an exception.

* Where:
Build file 'C:\flutter\.pub-cache\hosted\pub.dartlang.org\tflite-1.1.2\android\build.gradle' line: 24

* What went wrong:
A problem occurred evaluating project ':tflite'.
> No signature of method: build_29ou9k6xal0pml6qvuo3exmb2.android() is applicable for argument types: (build_29ou9k6xal0pml6qvuo3exmb2$_run_closure2) values: [build_29ou9k6xal0pml6qvuo3exmb2$_run_closure2@a6b8189]

* Try:
> Run with --stacktrace option to get the stack trace.
> Run with --info or --debug option to get more log output.
> Run with --scan to get full insights.

* Get more help at https://help.gradle.org

BUILD FAILED in 15s
Exception: Gradle task assembleDebug failed with exit code 1
```

ภาพประกอบที่ 3.21 ตัวอย่างการแจ้งเตือนข้อผิดพลาด

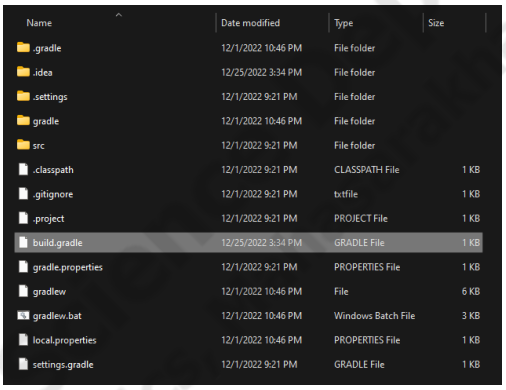
หากหลังจากการการ run โปรแกรมแล้วมีข้อผิดพลาดเกิดขึ้นดังภาพประกอบที่ 3.21 มีวิธีแก้ไขตามขั้นตอนต่อไปนี้

1) เข้าไปที่ไฟล์ของ tflite โดย path คือC:\flutter\.pubcache\hosted\pub.dartlang.org\tflite-1.1.2\android ดังภาพประกอบที่ 3.22



ภาพประกอบที่ 3.22 ไฟล์ของ tflite

2) เปิดไฟล์ build.gradle ดังภาพประกอบที่ 3.23



ภาพประกอบที่ 3.23 ไฟล์ build.gradle

3) เลื่อนลงมาดูที่ dependencies ตัวด้านล่างสุด ด้านในจะมีเนื้อหาสองบรรทัด ดังภาพประกอบที่ 3.24

```
dependencies {
    compile 'org.tensorflow:tensorflow-lite:+'
    compile 'org.tensorflow:tensorflow-lite-gpu:+'
}
```

ภาพประกอบที่ 3.24 dependencies ใน tflite

4) เปลี่ยนจากคำว่า compile เป็น implementation แล้วบันทึกเป็นการเสร็จสิ้น ดังภาพประกอบที่ 3.25

```
dependencies {
    implementation 'org.tensorflow:tensorflow-lite:+'
    implementation 'org.tensorflow:tensorflow-lite-gpu:+'
}
```

ภาพประกอบที่ 3.25 การเปลี่ยนแปลง dependencies

3.4.3 การใช้งาน API ภายนอก

1) TensorFlow Lite หรือ tflite เป็น library แบบ open source ที่ถูกพัฒนาโดย Google สำหรับการใช้ machine learning models บนอุปกรณ์มือถือที่ใช้ได้ทั้งระบบ android และ IOS สามารถใช้ tflite กับ natural language processing, computer vison หรือ machine learning model ที่สร้างจาก TensorFlow ได้ และหลังจากได้ model ที่เป็น tflite แล้ว จะต้องติดตั้ง tflite version 1.1.2 เพื่อที่จะใช้งาน model ได้

2) image_picker version 0.8.6เป็น library แบบ open source ที่มีไว้สำหรับการเลือกรูปภาพภายในเครื่องหรือการถ่ายรูปภายในเครื่อง โดยโครงงานนี้จะใช้ตัวนี้เป็นทั้งตัวเลือกรูปภาพจากแกลเลอรีและถ่ายภาพด้วย

3) camera version 0.10.0+5 เป็น library แบบ open source ใช้สำหรับควบคุมกล้อง รองรับการถ่ายภาพและวิดีโอโดยการโหลด library นี้จะทำให้การเข้าถึงกล้องของอุปกรณ์ได้อย่างถูกต้องและสามารถใช้งานได้โดยไม่ต้องเขียน permission เพราะมันดำเนินการ permission ให้ตอนที่ติดตั้ง

4) flutter_image_slider version 0.0.2 เป็น library แบบ open source ใช้สำหรับภาพสไลด์ แถบเลื่อนรูปภาพ ในโครงงานนี้ใช้ในส่วนหน้าดูข้อมูลงูที่เป็นรูปภาพ

5) google_fonts version 3.0.1 เป็น library แบบ open source ใช้สำหรับเปลี่ยน font ที่มีใน fonts.google.com

6) image_cropper version 3.0.1 เป็น library แบบ open source ใช้สำหรับ ครอบตัดรูปภาพใช้ได้ทั้ง android, IOS และ Web ที่รองรับ โครงงานนี้ใช้สำหรับการครอบรูปภาพหลังจากเลือกรูปภาพจากแกลเลอรีหรือถ่ายรูปภาพ

7) flutter_launcher_icons version 0.11.0 เป็น library แบบ open source ใช้สำหรับเปลี่ยน app’s launcher icon

3.5 ขั้นตอนในการพัฒนา Snake Model (Deep learning)

3.5.1 Data Augmentation

เทคนิค Data Augmentation เป็นเทคนิคที่ใช้เพื่อเพิ่มข้อมูลใหม่จากข้อมูลที่มีอยู่เดิม ซึ่งส่วนใหญ่จะใช้กับชุดข้อมูลที่ใช้ในการฝึกอบรม (Train) โมเดล โดยเทคนิคนี้จะช่วยแก้ไขปัญหาต่าง ๆ ของตัวโมเดล เช่น Overfitting หรือ ปัญหาการขาดแคลนชุดข้อมูล ซึ่งจะทำให้เพิ่ม robust ของโมเดล

และโมเดลมีประสิทธิภาพดีขึ้น(robustness : สำหรับ CNN หมายถึง ข้อมูลอินพุตที่ไม่เหมือนกับข้อมูลที่ใช้ในการ Train แต่การทำนายของโมเดลนั้นไม่ได้ลดประสิทธิภาพลง)

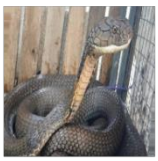
```
BATCH_SIZE = 32
IMG_SIZE = (331, 331)
train_directory = "/content/drive/MyDrive/Project/Snake Data/New Snake Photo 3"
valid_directory = "/content/drive/MyDrive/Project/Snake Data/Validation 2"
train_datagen = ImageDataGenerator( rescale=1./255,
      rotation_range = 45,
      width_shift_range = 0.2,
      height_shift_range = 0.2,
      shear_range = 0.2,
      zoom_range = 0.2,
      horizontal_flip = True,
      fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_directory,
    target_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
    class_mode = 'categorical',
    shuffle = True,
    seed = 42
)

valid_generator = train_datagen.flow_from_directory(
    valid_directory,
    target_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
    class_mode = 'categorical',
    shuffle = True,
    seed = 42
)

test_generator = ImageDataGenerator(rescale = 1.0/255.)
```

ภาพประกอบที่ 3.26 โค้ดการทำ Data Augmentation โดยใช้ ImageDataGenerator

ซึ่งเทคนิคที่ใช้สำหรับการทำ Data Augmentation มีทั้งหมดดังนี้

- rotation range : สุ่มหมุนภาพตามองศาที่กำหนด ในที่นี้จะใช้ 45 องศา
- width shift range : สุ่มเลื่อนภาพในแนวกว้าง
- height shift range : สุ่มเลื่อนภาพในแนวสูง
- shear range : เอียงภาพแบบสุ่ม
- zoom range : ซูมภาพแบบสุ่ม
- horizontal flip : สุ่มพลิกภาพในแนวนอน
- fill mode : เป็น method ที่ใช้ในการเติม pixel ที่สร้างขึ้นใหม่

ตารางที่ 3.3 ตัวอย่างผลลัพธ์ของการทำ Data Augmentation

ชื่อเทคนิค	การดำเนินการ	รูปต้นฉบับ	ผลลัพธ์
Rotation range	สุ่มหมุนภาพตามองศาที่กำหนด ในที่นี้จะใช้ 45 องศา		  
Width shift range	สุ่มเลื่อนภาพในแนวกว้าง		  
Height shift range	สุ่มเลื่อนภาพในแนวสูง		  
Shear range	เอียงภาพแบบสุ่ม		  
Zoom range	ขยายภาพแบบสุ่ม		  
Horizontal flip	สุ่มพลิกภาพในแนวนอน		  

3.5.2 เตรียมการชุดข้อมูลสำหรับการฝึกฝนโมเดล

ชุดข้อมูลที่ใช้สำหรับการฝึกฝนโมเดลได้มีการเปลี่ยนขนาดภาพให้เป็น 224 * 224 *3 และใช้เทคนิค data augmentation ดังที่กล่าวไว้ในหัวข้อ 3.3.1 ชุดข้อมูลสำหรับการ train ถูกแบ่งไว้สำหรับ validation 20% ซึ่งทั้งสองส่วนมีการกำหนด shuffle = True คือการทำให้ชุดข้อมูลจะถูกสับแบบสุ่มเพื่อหลีกเลี่ยงการเกิด overfitting ในระหว่างฝึกฝนโมเดล และ seed ซึ่งใช้สำหรับใช้การเพิ่มภาพแบบสุ่มและสับเปลี่ยนลำดับของภาพ

3.5.3 การสร้าง Model

3.5.3.1 สร้าง base model

จากภาพประกอบที่ 3.27 งานวิจัยนี้ใช้ MobilenetV2 model และกำหนดให้ imagenet เป็น weight เพื่อมาเป็น base model ในการทำ fine tuning และ freeze เลเยอร์ไว้เพื่อไม่ให้มีการอัปเดตค่า weights ในการ train และให้ include top = False เพื่อที่จะปรับโครงสร้างต่างๆให้เข้ากับเป้าหมายของงาน

```
[ ]  base_model = tf.keras.applications.MobileNetV2(
                        include_top=False,
                        weights='imagenet',
                        input_shape=(224,224,3)
                        )

     base_model.trainable = False
     base_model.summary()
```

ภาพประกอบที่ 3.27 โค้ดการตั้งค่า parameter ต่าง ๆ ของการเรียกใช้ MobileNetV2 model

3.5.3.2 ปรับแต่ง Model ในส่วน Classification

จากภาพประกอบที่ 3.28 model ภายในจะมีการปรับแต่งดังนี้

- ทำ Normalize data ด้วย Batch Normalization เพื่อปรับค่าของข้อมูลให้อยู่ในขอบเขตที่กำหนดก่อนจะส่งออกไปเป็น Input ของ layer ถัดไป

- Global Average Pooling ใช้แทนที่ fully connected layer ซึ่งจะทำให้ได้ feature ผลลัพธ์ทั้งหมดแค่หนึ่งค่าสำหรับแต่ละคลาส โดยขนาดของ feature จะอยู่ที่ 1*1*d (d is num of channel)

- Dense ใช้สำหรับเชื่อม layer ก่อนหน้าไปยัง layer ถัดไป

- Dropout คือการดรอปโหนดที่ไม่ได้รับการ train หรือ train แล้วแต่ค่าไม่เปลี่ยนแปลง การใช้ dropout จะทำให้โหนดเหล่านี้โดนดรอปทิ้งไป ซึ่งจะช่วยเพิ่ม accuracy

- ReLu เป็น activation function ที่จะทำให้เอาต์พุตมีค่าอยู่ระหว่าง 0 ถึงค่าบวก infinity ถ้าค่าที่ input เข้าไปต่ำกว่า 0 จะถูกแปลงให้เป็น 0 ทั้งหมด

- LeakyReLu เป็น ReLu ที่อนุญาตให้เอาต์พุตสามารถมีค่าเป็นลบได้ ซึ่งอาจจะทำให้มีความสมดุลมากกว่าและเรียนรู้ได้เร็วกว่า

- Dense layer อันสุดท้ายใช้เพื่อเชื่อมกับ Softmax ในการคำนวณและแสดงค่าความน่าจะเป็นของแต่ละคลาสเพื่อเลือกคำตอบที่มีความน่าจะเป็นสูงที่สุด

```
model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.BatchNormalization(renorm=True),
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(256, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(1024, activation='relu'),
        tf.keras.layers.LeakyReLU(alpha = 0.2),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(20, activation='softmax')
    ])

model.summary()
```

ภาพประกอบที่ 3.28 การปรับแต่ง model

3.5.3.3 ตั้งค่า Compile และ Optimizer

ตามภาพประกอบที่ 3.29 งานวิจัยนี้จะใช้ Optimizer เป็น Adam (Adaptive Moment Estimation) เพราะเป็นที่นิยมที่สุดและสามารถแก้ปัญหา decaying ของ gradients ทั้งยังลดปัญหาการแกว่งของพารามิเตอร์ ในส่วนของ loss function นั้นใช้เป็น categorical cross entropy ที่ช่วยในการตรวจสอบข้อผิดพลาดเล็กๆน้อยๆในระหว่างการ train

```
[ ]  model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

ภาพประกอบที่ 3.29 การตั้งค่าตัว Compile และ Optimizer

3.5.3.4 Train Model

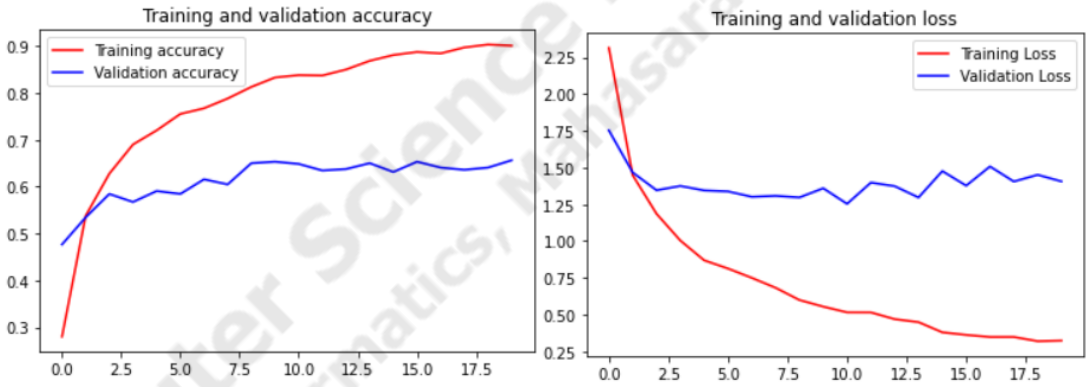
ตามภาพประกอบที่ 3.30 ในการ Train model จะใช้ 20 epochs และกำหนด steps_per_epoch โดยคำนวณจากสมการที่ใช้กันโดยทั่วไปคือ steps_per_epoch = train dataset

/ batch size และกำหนด validation_steps ก็ได้จากสมการ validation_steps = validation dataset / batch size

```
[ ]  batch_size=32
     STEP_SIZE_TRAIN = train_generator.n//train_generator.batch_size
     STEP_SIZE_VALID = valid_generator.n//valid_generator.batch_size

     # fit model
     history = model.fit(train_generator,
                         steps_per_epoch = STEP_SIZE_TRAIN,
                         validation_data = valid_generator,
                         validation_steps = STEP_SIZE_VALID,
                         epochs = 20,
                         callbacks=[early])
```

ภาพประกอบที่ 3.30 การตั้งค่าสำหรับการ train model ของ function fit

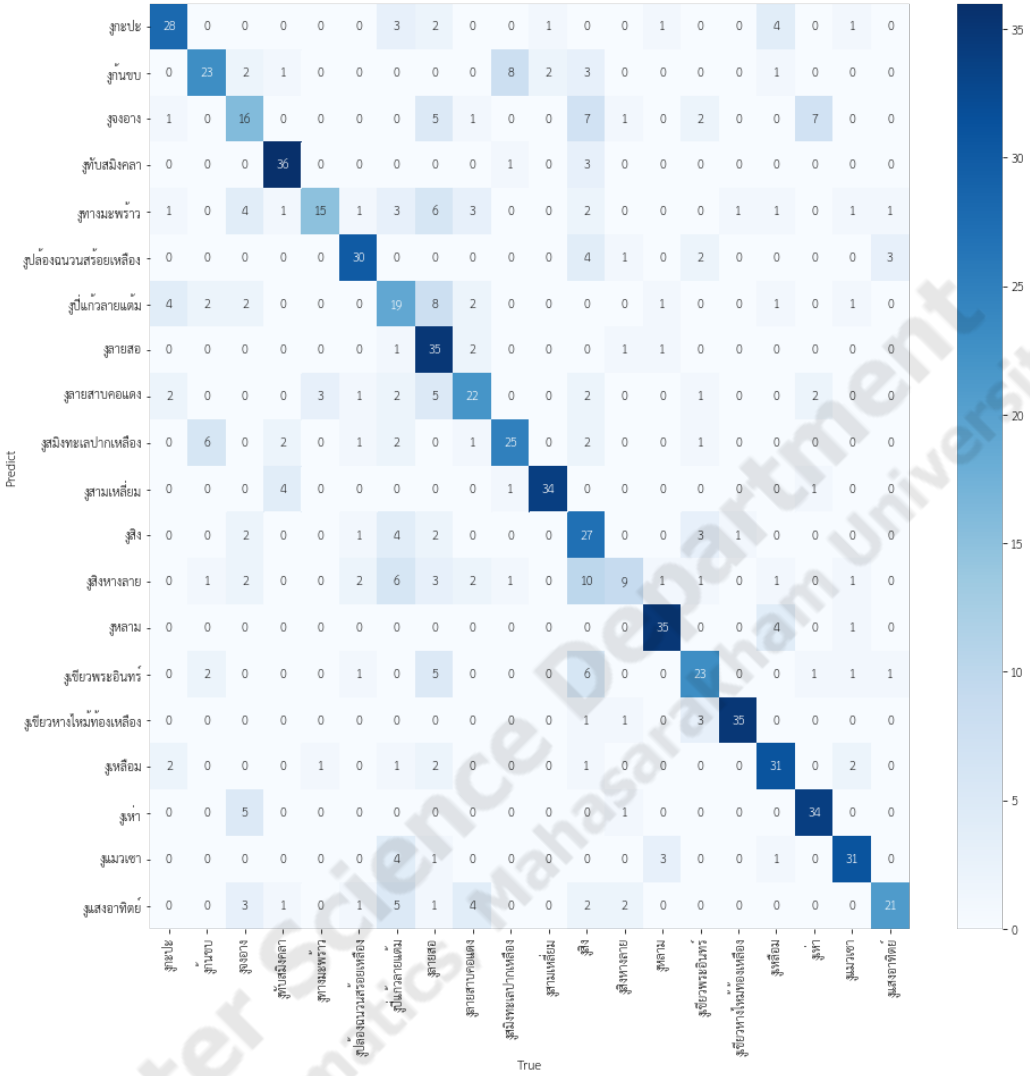
จากภาพประกอบที่ 3.31 แสดงกราฟผลลัพธ์ของ model ที่ได้จากการ train โดยเส้นสีแดง คือ train และ เส้นสีน้ำเงิน คือ validation กราฟ A แสดงค่าของ Accuracy และ กราฟ B แสดงค่า Loss



ภาพประกอบที่ 3.31 กราฟแสดงค่า accuracy และ loss ของการ train และ validation

3.5.4 การวัดประสิทธิภาพ (Evaluation)

ในขั้นตอนการวัดประสิทธิภาพจะใช้ model ทำนายสายพันธุ์งูตาม dataset ที่จัดเตรียมไว้ คือ 800 รูป แบ่งเป็นสายพันธุ์ละ 40 รูป โดยจากภาพประกอบที่ 3.32 คำตอบในแนวทแยงนั้นคือผลลัพธ์ของจำนวนที่ model ทำนายถูกต้องจากจำนวน 40 รูป



ภาพประกอบที่ 3.32 Evaluation Confusion matrix