Introduction

งานนี้เป็นการทำ Image Classification ของขนมไทยทั้งหมด 10 ชนิด โดยใช้ ImageNet Pre-trained CNN โดยทำในลักษณะของ Fine-Tuning Transfer Learning ซึ่งจะทำการเปรียบเทียบประสิทธิภาพด้านเวลาในการ training และประสิทธิผลของความแม่นยำ ทั้งหมด 5 Models ประกอบด้วย VGG-16, ResNet152V2, MobileNet, DenseNet121 และ EfficientNetB0 และการนำ Pre-trained Model มาใช้โดยตรงเพื่อทดสอบว่าขนมแต่ละประเภทนั้นถูกทำนายว่าเป็น Class อะไรบ้างใน 1,000 class และสุดท้ายการทำ Class Activation Mapping (CAM) เพื่อตรวจสอบว่า Model ได้ตรวจจับอยู่ในบริเวณของวัตถุหรือไม่

Data

ข้อมูลรูปภาพขนมไทยทำการรวบรวมมาจากเว็บไซต์ต่างๆ และมีบางส่วนที่ถ่ายเอง โดยมีความละเอียดที่แตกต่างกัน และแบ่งออกเป็นทั้งหมด 10 ชนิด ได้แก่ กล้วยบวชชี, ขนมตาล, ขนมเบื้องฝอยทอง, ขนมฝักบัว, ข้าวเหนียวทุเรียน, ข้าวเหนียวสังขยา, บัวลอย, ปั้นขลิบ และเปียกปูนกระทิสด ดังรูปที่ 1 โดยมีรูปภาพทั้งหมด 2,000 รูป จากนั้นข้อมูลรูปภาพถูกแบ่งเป็น Train Set, Validate Set และ Test Set ในสัดส่วน 80:10:10 ตามลำดับ หลังจากที่ข้อมูลถูกแบ่งเรียบร้อยแล้ว ได้มีการคัดเลือกรูปภาพที่นามสกุลไฟล์เป็น .JFIF ออก เนื่องจากข้อมูลในลักษณะนี้ไม่สามารถนำไปใช้ได้ ทำให้เหลือรูปภาพทั้งหมดสำหรับการทดลอง 1,733 รูป และแบ่งข้อมูลรูปภาพออกเป็น Train Set 1,389 รูป Validate Set 165 รูป และ Test Set 179 รูป โดยรูปภาพในแต่ละประเภทมีจำนวน ดังตารางที่ 1

รูปที่ 1 ภาพตัวอย่างขนมไทยแต่ละประเภท

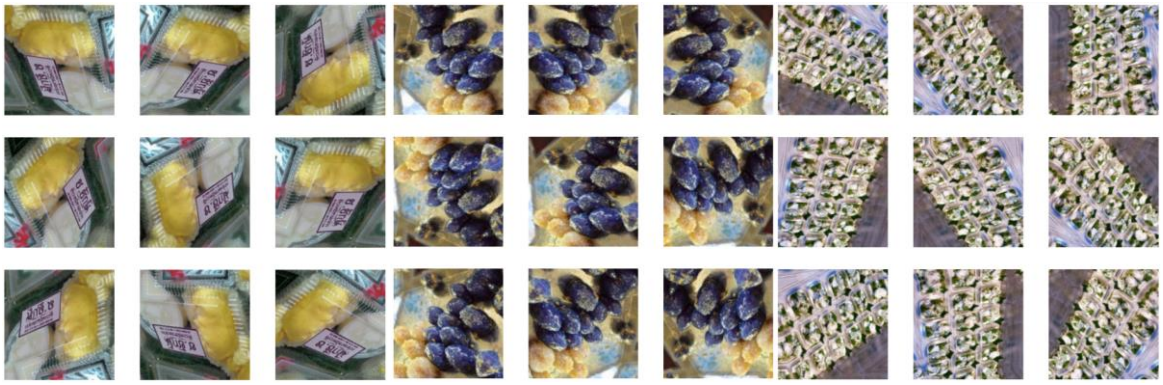
Class	Train set	Validation set	Test set	Total
Banana In Coconut Milk (กล้วยบวชชี)	146	18	18	182
Deep-Fried Rice Flour (ขนมฝักบวช)	151	13	18	182
Durian Sticky Rice (ข้าวเหนียวทุเรียน)	130	14	14	158
Pandanus Pudding in Coconut Cream (เปียกปูนกระทิสด)	152	20	20	192
Rice Balls in Sweet Coconut Milk (บัวลอย)	122	15	17	154
Sweet Sticky Rice with Thai Custard (ข้าวเหนียวสังขยา)	143	20	20	183
Tapioca Balls with Pork Filling (สาคูไส้หมู)	151	15	16	182
Thai Crispy Pancake topped with Golden Threads (ขนมเบื้องฝอยทอง)	142	20	20	182
Thai Style Fried Savory Dumplings with Fish Filling (ปั้นขลิบ)	130	15	18	163
Toddy Palm Cake (ขนมตาล)	125	15	19	159
Total	1,389	165	179	**1,737**

ตารางที่ 1 แสดงจำนวนรูปภาพของขนมไทยแต่ละประเภท

หลังจากการ Cleansing ข้อมูลรูปภาพเรียบร้อยแล้ว ก่อนการทดลองรูปภาพทั้งหมดจะถูกย่อให้มีความละเอียด 224**x**224 พิกเซล กำหนด color mode เป็น RGB และกำหนด Label Mode เป็น Categorical ผ่านคำสั่ง tf.keras.utils.image_dataset_from_directory จากนั้นมีการทำ Data Augmentation ด้วยการ Flip และ Rotation ดังรูปที่ 2 ผ่านคำสั่ง tf.keras.Sequential และมีการ Pre-process โดยใช้ Built-in Function ของ Keras ดังตารางที่ 2

Model	Pre-processing Method
VGG16	tf.keras.applications.vgg16.preprocess_input
ResNet152V2	tf.keras.applications.resnet_v2.preprocess_input
MobileNet	tf.keras.applications.mobilenet.preprocess_input
DenseNet121	tf.keras.applications.densenet.preprocess_input
EfficientNetB0	tf.keras.applications.efficientnet.preprocess_input

ตารางที่ 2 แสดง Built-in Function ของ Keras ที่ใช้สำหรับ Pre-processing ของแต่ละ Model



รูปที่ 2 ภาพตัวอย่างการของดการทำ Data Augmentation

Network Architecture

ทางกลุ่มเลือก Network ทั้งหมด 5 แบบได้แก่ VGG16, ResNet152V2, MobileNet, DenseNet121 และ EfficientNetB0 โดยใช้ Weights จากชุดข้อมูล ImageNet โดยที่ทุก Model จะทำการ Freeze ส่วนของ Convolutional layers

ในส่วนของ Classifier ทุก Model ใช้ Activation Function เป็น Softmax สำหรับการแบ่งประเภทขนมไทยทั้ง 10 อย่าง โดยมีรายละเอียดของแต่ละ Network ดังตารางที่ 3

Deep Model	# of Parameters (Base Model)	Depth (Base Model)	# of Train Parameters
VGG16	14,714,688	19	2,104,842
ResNet152V2	58,331,648	564	20,490
MobileNet	3,228,864	86	10,250
DenseNet121	7,037,504	427	10,250
EfficientNetB0	4,049,571	237	12,800

ตารางที่ 3 แสดงรายละเอียดแต่ละ Network

VGG16: Base Model มีทั้งหมด 19 Layers โดยทำการ Freeze ทั้งหมด จากนั้นในส่วน Classifier เพิ่ม GlobalAveragePooling2D 1 Layer ตามด้วย Dense 3 Layers ที่มีจำนวน Node เท่ากับ 1024, 1024, และ 512 ตามลำดับ โดยทั้ง 3 Layers นี้จะใช้ Activation เป็น ReLu และ Dropout เท่ากับ 0.2 และในส่วนของ Prediction Layer ใช้ Activation Function เป็น Softmax

ResNet152V2: Base Model มีทั้งหมด 564 Layers จากนั้นเพิ่มส่วน Classifier โดยใช้ GlobalAveragePooling2D และ Dropout เท่ากับ 0.2 และตามด้วย Prediction Layer โดยใช้ Softmax สำหรับ Activation Function

MobileNet: Base Model มีทั้งหมด 86 Layers จากนั้นเพิ่มส่วน Classifier โดยใช้ GlobalAveragePooling2D และ Dropout เท่ากับ 0.2 และตามด้วย Prediction Layer โดยใช้ Softmax สำหรับ Activation function

DenseNet121: Base Model มีทั้งหมด 427 layers จากนั้นเพิ่มส่วน Classifier โดยใช้ GlobalAveragePooling2D และ Dropout เท่ากับ 0.2 และตามด้วย Prediction Layer โดยใช้ Softmax สำหรับ Activation function

EfficientNetB0: Base Model มีทั้งหมด 237 layers จากนั้นเพิ่มส่วน Classifier โดยใช้ GlobalAveragePooling2D และ Dropout เท่ากับ 0.2 และตามด้วย Prediction Layer โดยใช้ Softmax สำหรับ Activation function

Training

ทางกลุ่มได้ทำการ Train Model ผ่าน Google Colab ทำให้ Environment และ Speed ในการ Train มีความแตกต่างกัน ดังตารางที่ 4

Model	GPU	Speed per Epoch (sec)
VGG-16	Tesla T4	13 - 15
ResNet152V2	Tesla T4	18 - 21
MobileNet	**Tesla P100**	**12 – 13**
DenseNet121	Tesla T4	14 - 16
EfficientNetB0	Tesla T4	14

ตารางที่ 4 แสดงรายละเอียดของ Environment ที่ใช้ในการ Train แต่ละ Model

แต่ละ Model ถูก Train ทั้งหมด 100 Epochs โดยกำหนด Batch size เท่ากับ 32, Learning rate เท่ากับ 0.0001 และใช้ Loss Function เป็น Categorical Cross-Entropy หลังจาก Train เรียบร้อยแล้ว ทางกลุ่มจึงเลือกใช้ weight ที่ให้ค่า Validation Accuracy สูงสุด โดยแต่ละ Model ให้ค่า Accuracy และ Loss บน Validation set ดังตารางที่ 5

Deep Model	Optimizer	Learning rate	Accuracy	Loss
VGG16	Adam	0.0001	0.9030	0.7633
ResNet152V2	Adam	0.0001	0.8788	0.4867
MobileNet	Adam	0.0001	0.8848	0.4563
DenseNet121	Adam	0.0001	**0.9152**	0.4669
EfficientNetB0	Adam	0.0001	0.9091	0.3509

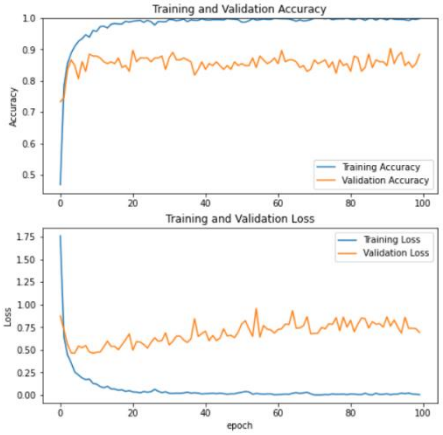
ตารางที่ 5 แสดงรายละเอียดในการ Train และ ผลลัพธ์ของ Model กับ Validation Set

จากตารางที่ 4 จะเห็นได้ว่าระหว่างการ Train Model ทุก Model ยกเว้น MobileNet ใช้ GPU Tesla T4 เหมือนกันในขณะที่ MobileNet ใช้ Tesla P100 ซึ่งเป็น GPU ที่มี Performance ต่ำสุด แต่ MobileNet ก็ยังให้ประสิทธิภาพดีที่สุด เนื่องจากใช้เวลาในการ Train น้อยที่สุด เมื่อเทียบกับ Model ตัวอื่น

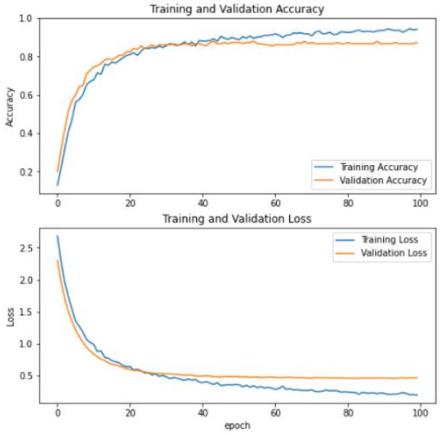
ในด้านการวัดประสิทธิผลของ Model กับ Validation Set จะเห็นได้ว่า Model ที่มีประสิทธิผลดีที่สุด คือ DenseNet121 ซึ่งมี Accuracy สูงสุด และ Loss ไม่ได้สูงมากนัก Model ที่ดีรองลงมาคือ EfficientNetB0 ที่มีค่า Accuracy น้อยกว่า DenseNet121 อยู่ 0.0061 แต่ในขณะเดียวกัน Loss ของ EfficientNetB0 ก็น้อยกว่า DenseNet121 อยู่ 0.116 จะเห็นได้ว่าถึงแม้ Accuracy ของ EfficientNetB0 จะน้อยกว่า DenseNet121 แต่ค่า Loss ก็น้อยกว่าด้วยเช่นกัน ลำดับที่3 คือ VGG16 ซึ่งมี Accuracy สูงเป็นอันดับ3 แต่ Loss ของ VGG16 นั้นก็สูงที่สุดเช่นกันเมื่อเทียบกับทั้ง 5 Models ลำดับรั้งท้ายที่มี Accuracy และ Loss ใกล้เคียงกันคือ ResNet152V2 และ MobileNet ซึ่งมีค่า Accuracy ประมาณ 0.88 และ Loss ประมาณ 0.46

เมื่อพิจารณาทั้งในแง่ของประสิทธิภาพและประสิทธิผลเราจะเห็นว่า หากต้องการความถูกต้องแม่นยำ DenseNet121 นั้นดีที่สุด แต่หากต้องการประหยัดเวลาในการ Train Model สามารถเลือกใช้ MobileNet แทนได้ แต่หากเมื่อมองในภาพรวมแล้ว EfficientNetB0 เป็น Model ที่ดีที่สุด เนื่องจาก Accuracy สูง ในขณะที่ Loss และ Speed ต่ำ

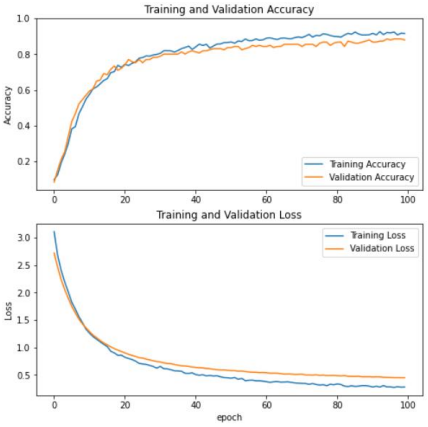
หลังจากที่ Train และ Validate Model เรียบร้อยแล้ว จึงนำค่า Accuracy และ Loss ของ Training Set และ Validation Set มา Plot กราฟเพื่อดูแนวโน้ม Overfit/Underfit ดังรูปที่ 3



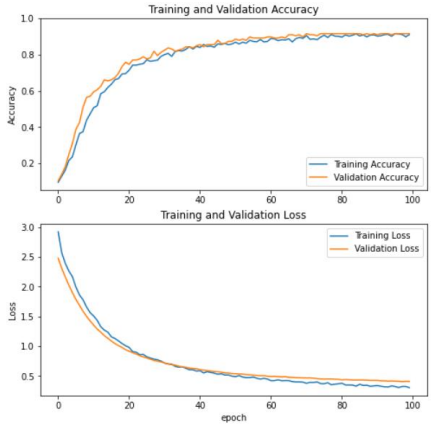
(a) VGG-16 model Accuracy 90.3%



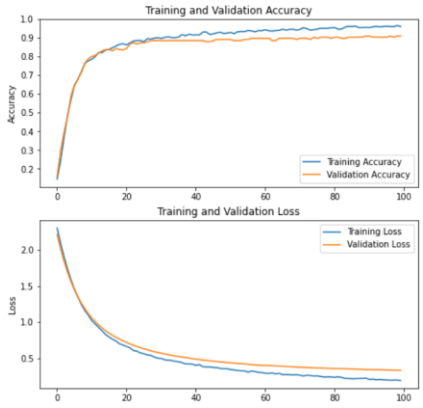
(b) ResNet152V2 model Accuracy 87.88%



(c) MobileNet model Accuracy 88.48%



(d) DenseNet121 model Accuracy 91.52%



(e) EfficientNetB0 model Accuracy 90.91%

รูปที่ 3 เปรียบเทียบค่า Accuracy และ Loss ของแต่ละ Model บน Training set และ Validation set

Results

แต่ละ Model ถูกนำมาทดสอบกับ Test set โดยใช้ค่า Weight จาก Model ที่ให้ค่า Validation Accuracy สูงสุด ซึ่งจะได้ค่า Accuracy และ Loss ดังตารางที่ 4

Deep Model	Accuracy	Loss	Speed (sec)
VGG16	0.8492	0.9800	37
ResNet152V2	0.8715	0.4635	30
MobileNet	**0.8939**	0.3680	27
DenseNet121	0.8436	0.4360	50
EfficientNetB0	0.8659	0.3518	27

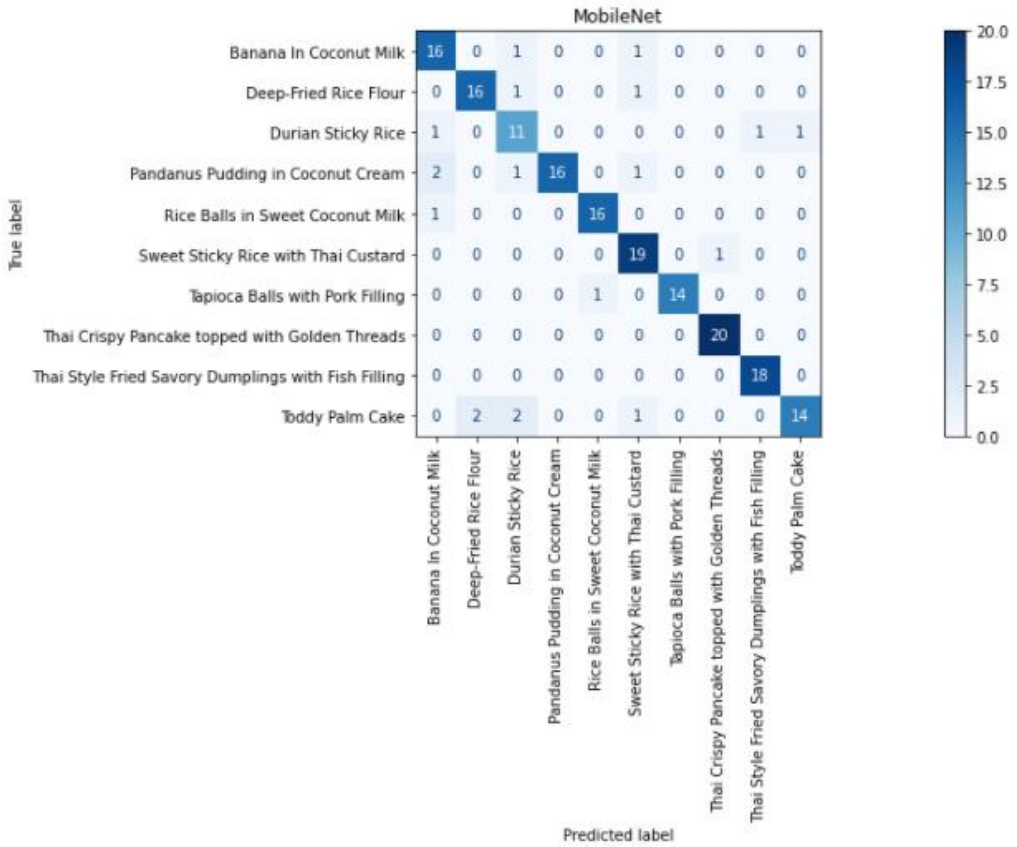
ตารางที่ 6 แสดงผลลัพธ์ของ Model กับ Test Set

จะเห็นได้ว่าตอนที่วัดประสิทธิผลของ Model กับ Validation Set นั้น DenseNet121 เป็น Model ที่ดีที่สุด ในขณะที่ MobileNet เป็น Model ที่มี Accuracy เป็นอันดับรั้งท้าย แต่เมื่อทดสอบกับ Test Set กลายเป็นว่า MobileNet นั้นเป็น Model ที่ให้ Accuracy สูงสุด และประสิทธิภาพดีที่สุดเนื่องจากใช้เวลาในการ Predict น้อยที่สุดคือ 27 วินาที

และเมื่อพิจารณาค่า Precision ซึ่งหมายถึงประสิทธิภาพที่คำนวณจากสัดส่วนระหว่างจำนวนของวัตถุที่ตรวจจับได้ถูกต้อง (TP) กับจำนวนวัตถุที่ตรวจจับได้ทั้งหมด (TP+FP) และค่า Recall หมายถึงประสิทธิภาพที่คำนวณจากสัดส่วนระหว่างจำนวนของวัตถุที่ตรวจจับได้ถูกต้อง (TP) กับจำนวนวัตถุที่ถูกต้องทั้งหมด (TP+FN) และสุดท้ายค่า F1 score หมายถึงค่าเฉลี่ยแบบ Harmonic mean ระหว่าง Precision และ Recall (2*(Recall * Precision) / (Recall + Precision)) ของ MobileNet จะได้ค่าในของขนมแต่ละประเภทแสดงดังตารางที่ 7 และ Confusion Matrix ดังรูปที่ 4

Class	Precision	Recall	F1 score
Banana In Coconut Milk (กล้วยบวชชี)	0.8000	0.8889	0.8421
Deep-Fried Rice Flour (ขนมฝักบวช)	0.8889	0.8889	0.8889
Durian Sticky Rice (ข้าวเหนียวทุเรียน)	0.6875	0.7857	0.7333
Pandanus Pudding in Coconut Cream (เปียกปูนกระทิสด)	1.0000	0.8000	0.8889
Rice Balls in Sweet Coconut Milk (บัวลอย)	0.9412	0.9412	0.9412
Sweet Sticky Rice with Thai Custard (ข้าวเหนียวสังขยา)	0.8261	0.9500	0.8837
Tapioca Balls with Pork Filling (สาคูไส้หมู)	1.0000	0.9333	0.9655
Thai Crispy Pancake topped with Golden Threads (ขนมเบื้องฝอยทอง)	0.9524	1.0000	0.9756
Thai Style Fried Savory Dumplings with Fish Filling (ปั้นขลิบ)	0.9474	1.0000	0.9730
Toddy Palm Cake (ขนมตาล)	0.9333	0.7368	0.8235

ตารางที่ 7 แสดงข้อมูล Precision, Recall และ F1 score ของ MobileNet



รูปที่ 4 แสดง Confusion Matrix ของ MobileNet

ตัวอย่างการ Predict ของ MobileNet แสดง ดังรูปที่ 5

รูปที่ 5 แสดง ตัวอย่างการ Predict Test Set ของ MobileNet

ตัวอย่างการ Predict ของ MobileNet แสดง ดังรูปที่ 6

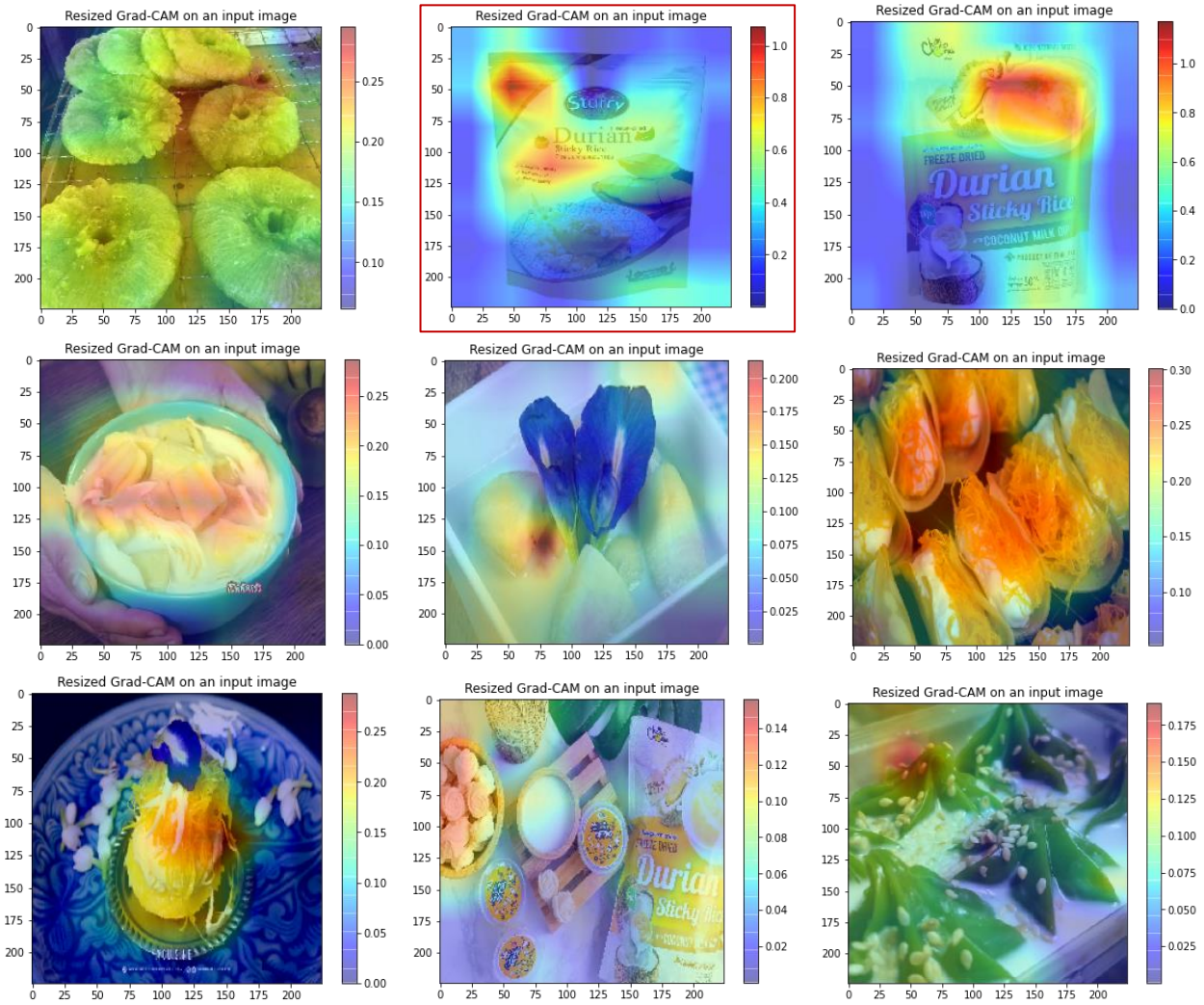
รูปที่ 6 แสดง ตัวอย่างการ Predict Test Set ของ MobileNet

เมื่อเราทดลองนำ Base Model ของ MobileNet มา Predict กับ Test Set จะได้ผลดังรูปที่ 7

รูปที่ 7 แสดง ตัวอย่างการ Predict Test Set ของ MobileNet ที่เป็น Base Model

เนื่องจาก Base Model ทางกลุ่มใช้ Weight จาก ImageNet ซึ่งมีทั้งหมด 1,000 Classes แต่ในบรรดา Class เหล่านี้ไม่มี Class ขนมไทยจึงทำให้ Base Model ทำนายผิด โดยจะพบว่าจาก 30 รูป ส่วนใหญ่จะถูก Predict เป็น shower_curtain, jellyfish, digital_clock และ honeycomb เมื่อพิจารณาจะพบว่า รูปที่มีสีสันเยอะ มีลักษณะเป็นทรงกลมเล็ก จะถูก predict เป็น shower_curtain ในขณะที่รูปที่มีลักษณะเป็นทรงกลมใหญ่ และกินพื้นที่เกือบทั้งภาพจะถูก Predict ว่าเป็น digital_clock ส่วนรูปที่ถูก Predict ว่าเป็น honeycomb จะมีลักษณะเป็นทรงกลมขนาดกลางวางอยู่ติดๆกัน และรูปที่ถูก Predict ว่าเป็น jellyfish จะมีลักษณะที่มีส่วนโค้ง และเป็นเส้น

หลังจากที่ Train, Validate และ Test Model จนได้ Model ที่ดีที่สุด นั่นคือ MobileNet กลุ่มเราจึงทำการตรวจสอบ Model ว่าเรียนรู้รูปภาพได้ตรงจุดตามที่เราต้องการหรือไม่ โดยใช้ Class Activation Mapping (CAM) ได้ผล ดังรูปที่ 7



รูปที่ 7 แสดง ตัวอย่าง Class Activation Mapping (CAM) ของ MobileNet ที่ Predict ได้ถูกต้อง

จะเห็นว่า Model สามารถเรียนรู้รูปภาพได้ตรงจุดตามที่ทางกลุ่มต้องการ แต่จะมีบางรูปภาพเท่านั้นที่ Model เลือกดูผิดจุด ดังรูปที่ 7 ที่ถูกวงด้วยกรอบสีแดง จะเห็นว่ารูดังกล่าวจัดอยู่ใน Class ข้าวเหนียวทุเรียน บริเวณที่ Model ควรจะใช้ในการเรียนรู้คือส่วนของรูปทุเรียน แต่ในรูปดังกล่าว Model เลือกดูเฉพาะบริเวณถุง แต่เนื่องจากบริเวณนั้นมีสีที่ใกล้เคียงกับรูปอื่นๆใน Class เดียวกัน ทำให้ Model Predict ได้ถูกต้อง

Discussion

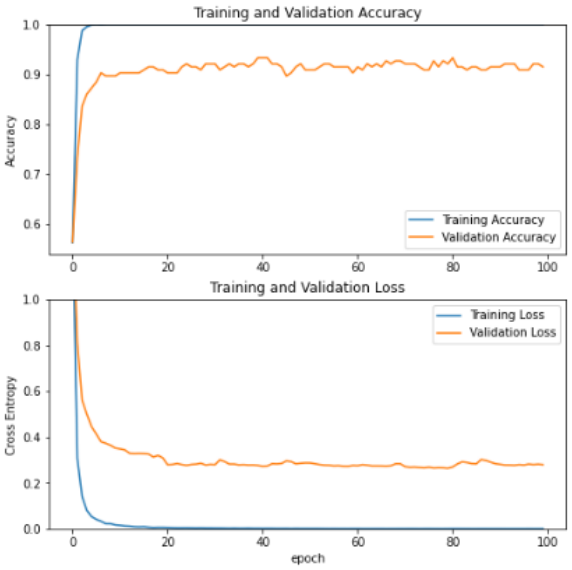
จากการทำ Fine-Tuning Transfer Learning โดยใช้ ImageNet Pre-trained Weight ของ 5 Models ประกอบด้วย VGG-16, ResNet152V2, MobileNet, DenseNet121 และ EfficientNetB0 เพื่อแยกประเภทของขนมไทยจำนวน 10 ประเภท นั้นพบว่า ในขั้นตอนการทดสอบกับข้อมูล Validation Set ประสิทธิภาพด้านเวลาในการ training ของ Model จากมากไปน้อย คือ MobileNet > EfficientNetB0 > VGG-16 > DenseNet121 > ResNet152V2 ในขณะที่ประสิทธิผลของความแม่นยำของ Model จากมากไปน้อย คือ DenseNet121 > EfficientNetB0 > VGG-16 > MobileNet > ResNet152V2

เมื่อทดสอบ Model กับ Test Set พบว่า ประสิทธิภาพด้านเวลาของ Model จากมากไปน้อย คือ คือ MobileNet > EfficientNetB0 > ResNet152V2 > VGG-16 > DenseNet121 ในขณะที่ประสิทธิผลของความแม่นยำของ Model จากมากไปน้อย คือ MobileNet > ResNet152V2 > EfficientNetB0 > VGG-16 > DenseNet121 จะเห็นได้ว่าตอนที่วัดประสิทธิผลของ Model กับ Validation Set นั้น DenseNet121 เป็น Model ที่ดีที่สุด ในขณะที่ MobileNet เป็น Model ที่มี Accuracy เป็นอันดับรั้งท้าย แต่เมื่อทดสอบกับ Test Set กลายเป็นว่า MobileNet นั้นเป็น Model ที่ให้ Accuracy สูงสุด และประสิทธิภาพดีที่สุดเนื่องจากใช้เวลาในการ Predict น้อยที่สุดคือ 27 วินาที นั่นหมายความว่า MobileNet มีความ Generalize มากกว่า DenseNet121

จากผลของการทดสอบ Model ด้วย Validation และ Test Set ทำให้สมมติฐานที่เราคิดไว้นั้นไม่เป็นไปตามคาด เนื่องจาก ทางกลุ่มมองว่า Model ที่มีประสิทธิผลดีกับ Validation Set อย่าง DenseNet121 และ EfficientNetB0 ที่มีค่า Accuracy สูงถึง 0.91 ควรจะ ให้ Accuracy กับ Test Set ได้ดีเช่นกัน แต่หลังจากที่เราทดสอบกับ Test Set Model ที่ดีที่สุดคือ MobileNet ทั้งในด้านประสิทธิภาพและประสิทธิผล ในขณะที่ทางด้านความเร็วในการ Train Model เป็นไปตามที่ทางกลุ่มประเมินไว้ คือ Model ที่มีจำนวน Layer และ Parameter น้อยจะ train ได้เร็วกว่า Model ที่มีจำนวน Layer และ Parameter มาก ซึ่งเมื่อพิจารณาตามตารางที่ 3 จะเห็นว่า MobileNet นั้นเป็น Model ขนาดเล็กเมื่อเทียบกับ Model อื่นๆ จึงมีประสิทธิภาพที่ดีกว่า

ในการทำงานครั้งนี้ ทางกลุ่มเจอปัญหาในการทำงานกับข้อมูลรูปภาพ เนื่องจากข้อมูลที่เราใช้นั้นเป็นข้อมูลจากจากเว็บไซต์ต่างๆ โดยในเริ่มแรกทางกลุ่มได้หยิบรูปภาพทั้งหมดมา Preprocess และทำ Data Augmentation จากนั้นนำข้อมูลทั้งหมดเหล่านี้ไป Train Model พบว่ามีบางรูปภาพไม่สามารถใช้ได้ เราจึงได้ค้นหาสาเหตุจนพบว่า ข้อมูลรูปภาพที่นามสกุลไฟล์ .JFIF นั้นไม่สามารถนำมาใช้ได้ จำเป็นที่จะต้องกรองออกก่อนที่จะนำข้อมูลเข้า Model

ปัญหาต่อมา คือ ในครั้งแรกที่ทางกลุ่ม Pre-process Data เราได้ใช้ Function หนึ่งของ Keras นั่นคือ tf.keras.preprocessing.image.ImageDataGenerator หลังจากนั้นเราจึงทำการ Train และ Validate Model เมื่อเสร็จสิ้นแล้ว เรา Plot กราฟเพื่อดู Accurcy และ Loss พบว่า กราฟมีลักษณะแปลก และค่อนข้างที่จะ Overfit ดังรูปที่ 6



รูปที่ 8 แสดงค่า Accuracy และ Loss ของ MobileNet

เมื่อเราหาสาเหตุจึงพบว่าคำสั่งดังกล่าวได้เลิกใช้ไปแล้ว ดังนั้นกลุ่มของจึงเปลี่ยนวิธีการ Pre-process Data ด้วยคำสั่ง tf.keras.utils.image_dataset_from_directory และทำการ Pre-Processing ต่อด้วยคำสั่งตามตารางที่ 2 ที่กล่าวไปในตอนต้น หลังจากที่เปลี่ยนวิธีในการจัดการกับรูปภาพแล้ว ทำให้กราฟ Accuracy และ Loss ดีขึ้น ดังรูปที่ 2

นอกจากปัญหาที่ทางกลุ่มได้เจอแล้ว ยังมีอีกหนึ่งสิ่งที่น่าสนใจคือ ในครั้งแรกของการทดสอบ Model เราตั้งต้นข้อมูลรูปภาพด้วย 100 รูป/Class ดังนั้นข้อมูลเริ่มแรกที่ใช้มีทั้งหมด 1,000 รูป โดยแบ่งข้อมูลเป็น Train Set, Validate Set และ Test Set ในสัดส่วน 80:10:10 ตามลำดับ หลังจาก Train Model เรียบร้อยแล้ว ทางกลุ่มต้องการเพิ่มประสิทธิภาพของ Model ให้ดีขึ้น วิธีการหนึ่งที่เราเลือกใช้คือการเพิ่มจำนวน Data โดยเราเพิ่มเข้าไปอีก 100 รูป/Class ทำให้มีจำนวนรูปทั้งหมด 2,000 รูป และมีการแบ่งข้อมูลในสัดส่วนตามเดิม ผลปรากฏว่าการเพิ่มจำนวนรูปภาพไม่ได้ทำให้ประสิทธิผลของ Model เพิ่มขึ้นอย่างมีนัยสำคัญ

ในขั้นตอนของการทำ Class Activation Mapping (CAM) พบว่า Model สามารถเรียนรู้รูปภาพได้ตรงจุดตามที่ทางกลุ่มต้องการ แต่จะมีบางรูปภาพเท่านั้นที่ Model เลือกดูผิดจุด ซึ่งเกิดจากบริเวณนั้นมีสีที่ใกล้เคียงกับรูปอื่นๆใน Class เดียวกัน ทำให้ Model Predict ได้ถูกต้อง

Conclusion

จากการเปรียบเทียบประสิทธิผลของ Model กับ Validation Set นั้น DenseNet121 เป็น Model ที่ดีที่สุด ในขณะที่ MobileNet เป็น Model ที่มี Accuracy เป็นอันดับรั้งท้าย แต่เมื่อทดสอบกับ Test Set กลายเป็นว่า MobileNet นั้นเป็น Model ที่ให้ Accuracy สูงสุด และประสิทธิภาพดีที่สุด ดังนั้นในการ Classify ขนมไทย 10 ชนิด ทางกลุ่มจึงแนะนำให้ใช้ MobileNet

References

https://www.tensorflow.org/tutorials/images/classification

https://www.tensorflow.org/tutorials/images/transfer_learning

https://keras.io/examples/vision/image_classification_from_scratch/