การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน

MODEL OPTIMIZATION FOR CORN LEAF DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

ปิยะณัฐ เทพสืบ

บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ

2567

การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน

ปิยะณัฐ เทพสืบ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการข้อมูล
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2567

MODEL OPTIMIZATION FOR CORN LEAF DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

PIYANAT THEPSUEB

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of MASTER OF SCIENCE

(Data Science)

Faculty of Science, Srinakharinwirot University

2024

ปริญญานิพนธ์

เรื่อง

การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่าย

ประสาทเทียมแบบคอนโวลูชัน

ของ

ปิยะณัฐ เทพสืบ

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการข้อมูล

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

.....

(รองศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญากุล)

คณบดีบัณฑิตวิทยาลัย

.....

คณะกรรมการสอบปากเปล่าปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร.บรรพตรี คมขำ)

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ศิริสรรพ เหล่าหะเกียรติ)

ชื่อเรื่อง	การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน
ผู้วิจัย	ปิยะณัฐ เทพสืบ
ปริญญา	วิทยาศาสตรมหาบัณฑิต
ปีการศึกษา	2567
อาจารย์ที่ปรึกษา	อาจารย์ ดร.บรรพตรี คมขำ

งานวิจัยนี้นำเสนอการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคใบข้าวโพดจากภาพถ่าย และปรับปรุงแบบจำลองให้เหมาะสมที่สุดกับการใช้งานบนอุปกรณ์พกพา โดยใช้ชุดข้อมูลภาพถ่ายใบข้าวโพดจากชุดข้อมูล PlantVillage โดยแบ่งข้อมูลออกเป็นชุดฝึกฝนต่อชุดทดสอบ ในอัตราส่วน 80 : 20 แบบจำลองที่ใช้ในการทดลอง ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile และใช้เทคนิค Knowledge Distillation เพื่อพัฒนาแบบจำลอง 2 แบบ ได้แก่ STD Conv Student Model และ DW Conv Student Model ซึ่งมีพารามิเตอร์น้อยกว่า 1 ล้านพารามิเตอร์ แต่ยังคงความแม่นยำสูง จากนั้นใช้เทคนิค Post-training Quantization ได้แก่ Float16 Quantization และ Dynamic Range Quantization เพื่อลดขนาดแบบจำลองแต่ยังคงรักษาประสิทธิภาพไว้ โดยสามารถลดขนาดของแบบจำลองได้มากถึง 50 % และ 75% ตามลำดับ และเมื่อทดสอบแบบจำลองด้วยชุดข้อมูลทดสอบพบ ว่า MobileNetV2 ที่ทำ Dynamic Range Quantization ให้ความแม่นยำสูงถึง 99.22% และ MobileNetV2 ที่ทำ Float16 Quantization มีความเร็วในการประมวลผลเร็วขึ้นถึงประมาณ 6 เท่า และเมื่อทดสอบแบบจำลองบนโทรศัพท์มือถือ พบว่า และ DenseNet121 ที่ทำ Float16 Quantization ให้ความแม่นยำสูงถึง 99.35% สุดท้ายเมื่อประเมินด้วย Weighted Decision Matrix โดยพิจารณา ได้แก่ ความแม่นยำ ขนาดของแบบจำลอง และความเร็วในการประมวลผล พบว่า STD Conv Student Model ที่ทำ Float16 Quantization มีคะแนนรวมสูงสุด 92.074 คะแนน ด้วยขนาดเพียง 1.5 MB และมีความแม่นยำ 98.18% จากผลการวิจัย แสดงให้เห็นว่าการใช้เทคนิค Knowledge Distillation ร่วมกับ Post-training Quantization สามารถพัฒนาแบบจำลองที่มีขนาดเล็กแต่มีประสิทธิภาพในการจำแนกโรคจากใบข้าวโพดได้ดีสามารถใช้งานบนโทรศัพท์มือถือได้อย่างมีประสิทธิภาพ

คำสำคัญ: โครงข่ายประสาทเทียมแบบคอนโวลูชัน, การจำแนกโรคใบข้าวโพด, การแปลงควอนไทซ์, การกลั่นความรู้, การลดขนาดแบบจำลอง

Title	MODEL OPTIMIZATION FOR CORN LEAF DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK
Author	PIYANAT THEPSUEB
Degree	MASTER OF SCIENCE
Academic Year	2024
Advisor	Banphatree Khomkham, Ph.D.

This research proposes the development of convolutional neural network CNN models for corn leaf diseases classification from images and aims to optimize the model for deployment on mobile devices using corn leaf images from the PlantVillage dataset, which divided into training and testing sets in an 80:20 ratio. The pre-trained models include VGG16, DenseNet121, MobileNetV2 and NASNet Mobile. Furthermore, Knowledge Distillation was applied to develop two tiny models, STD Conv Student and DW Conv Student Model, both with fewer than one million parameters but maintaining high accuracy. Post-training quantization techniques, including Float16 Quantization and Dynamic Range Quantization, were applied to reduce model size while preserving performance. These techniques successfully reduced model sizes by up to 50% and 75%, respectively. Experimental results show that the MobileNetV2 with Dynamic Range Quantization achieved a test accuracy of 99.22%, while the Float16 Quantized MobileNetV2 achieved approximately six times faster inference speed. When tested on a mobile phone evaluation, the Float16 Quantized DenseNet121 achieved an accuracy of 99.35%. Finally, an evaluation using a Weighted Decision Matrix, considering accuracy, model size, and processing speed, revealed that the STD Conv Student with Float16 Quantization scored the highest overall, 92.074 points, a compact size of only 1.5 MB and an accuracy of 98.18%. These results indicate that using Knowledge Distillation with Post-training Quantization can produce small but efficient models for corn leaf disease classification that can be effectively deployed on mobile devices.

Keywords: Convolutional Neural Network, Corn Leaf Disease Classification, Post-training Quantization, Knowledge Distillation, Model Compression

กิตติกรรมประกาศ

งานวิจัยชิ้นนี้เสร็จสมบูรณ์ได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์ และความเอาใจใส่เป็นอย่างดียิ่งจาก อาจารย์ ดร.บรรพตรี คมขำ อาจารย์ประจำภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ผู้ซึ่งเป็นอาจารย์ที่ปรึกษาในงานวิจัยชิ้นนี้ ที่ได้เสียสละเวลาอันมีค่ามาให้ความรู้ ข้อเสนอแนะ และแก้ไขข้อบกพร่องต่าง ๆ ในการทำวิจัย จนกระทั่งงานวิจัยนี้ได้สำเร็จลุล่วงตามวัตถุประสงค์ ขอกราบขอบพระคุณอย่างสูงมา ณ ที่นี้

ขอขอบคุณคณาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ทุกท่านที่ได้ประสิทธิ์ประสาทองค์ความรู้ที่ใช้ในงานวิจัยครั้งนี้ ตลอดจนได้ถ่ายทอดประสบการณ์ และให้คำชี้แนะแนวทางในการศึกษาวิจัยสำหรับงานวิจัยในครั้งนี้

ขอขอบคุณบัณฑิตวิทยาลัย และคณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ที่ได้สนับสนุนทุนการศึกษาสำหรับดำเนินการวิจัย และสนับสนุนค่าใช้จ่ายในการไปนำเสนอผลงานการวิจัยในการประชุมวิชาการระดับนานาชาติ เพื่อนำองค์ความรู้ที่ได้จากการวิจัยไปเผยแพร่ต่อสาธารณชน

สุดท้ายนี้ข้าพเจ้าหวังเป็นอย่างยิ่งว่างานวิจัยนี้จะเป็นประโยชน์กับผู้ที่เกี่ยวข้อง และหากมีข้อผิดพลาดประการใด ข้าพเจ้าขอน้อมรับความผิดพลาด และขออภัยมา ณ ที่นี้

ปิยะณัฐ เทพสืบ

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง

บทคัดย่อภาษาอังกฤษ ..... จ

กิตติกรรมประกาศ ..... ฉ

สารบัญ ..... ช

สารบัญตาราง ..... ฐ

สารบัญรูปภาพ ..... ฒ

บทที่ 1 บทนำ 1

1.1 ภูมิหลัง 1

1.2 ความมุ่งหมายของงานวิจัย 3

1.3 ความสำคัญของงานวิจัย 3

1.4 ขอบเขตงานวิจัย 3

1.4.1 ประชากรที่ใช้ในการวิจัย 3

1.4.2 กลุ่มตัวอย่างที่ใช้ในการวิจัย 3

1.4.3 ตัวแปรที่ใช้ศึกษา 4

1.5 กรอบแนวความคิดในงานวิจัย 4

1.6 สมมติฐานในการวิจัย 4

บทที่ 2 ทฤษฎีที่เกี่ยวข้องและทบทวนวรรณกรรม 5

2.1 การใช้การเรียนรู้ของเครื่องในการจำแนกโรคพืช 5

2.1.1 พัฒนาการของการใช้การเรียนรู้ของเครื่องในการจำแนกโรคพืช 5

2.1.2 ข้อจำกัดและความท้าทายของการใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ในภาคสนาม 6
2.2 โครงข่ายประสาทเทียมแบบคอนโวลูชัน 7
2.2.1 ความรู้พื้นฐานของโครงข่ายประสาทเทียมแบบคอนโวลูชัน 7
2.2.2 โครงสร้างพื้นฐานของโครงข่ายประสาทเทียมแบบคอนโวลูชัน 9
2.2.3 การทำงานของชั้น Convolution Layers 11
2.2.3.1 หลักการของการคอนโวลูชัน (Convolution Principle) 11
2.2.3.2 ตัวกรอง และการสกัดคุณลักษณะ 18
2.2.3.3 การใช้ฟังก์ชันกระตุ้น (Activation Functions) 20
2.2.4 การทำงานของชั้นพูลลิ่ง 22
2.2.4.1 วิธีการลดขนาดข้อมูล 22
2.2.4.2 ประเภทของการพูลลิ่ง (Max Pooling และ Average Pooling) 23
2.2.5 การฝึกฝนและการปรับปรุงค่าพารามิเตอร์ของโครงข่ายประสาทเทียมแบบคอนโวลูชัน 25
2.2.5.1 กระบวนการเรียนรู้แบบย้อนกลับ (Backpropagation) 25
2.2.5.2 การปรับค่าพารามิเตอร์และการหาค่าเหมาะสมที่สุด (Optimization) 26
2.2.6 เทคนิคการปรับปรุงประสิทธิภาพของแบบจำลองแบบโครงข่ายประสาทเทียมแบบคอนโวลูชัน 27
2.2.6.1 การเพิ่มข้อมูล (Data Augmentation) 27
2.2.6.2 การใช้ Dropout เพื่อลดการ Overfitting 28
2.2.6.3 การปรับปรุงอัตราการเรียนรู้ (Learning Rate Adjustment) 29
2.3 สถาปัตยกรรมแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการใช้งานเพื่อจำแนกภาพ 30
2.3.1 VGG16 31

2.3.2 DenseNet121 35
2.3.3 MobileNet V2 39
2.3.4 NASNet Mobile 44
2.4 เทคนิคการทำ Model Optimization สำหรับการลดขนาดแบบจำลอง 47
2.4.1 Model Pruning 49
2.4.1.1 การตัดแต่งแบบ Structured Pruning 50
2.4.1.2 การตัดแต่งแบบ Unstructured Pruning 51
2.4.2 Parameters Quantization 53
2.4.2.1 Quantization-Aware Training (QAT) 54
2.4.2.2 Post-Training Quantization (PTQ) 54
2.4.3 Low-Rank Decomposition / Low-Rank Factorization 60
2.4.4 Knowledge Distillation 62
2.4.4.1 Respond-based Knowledge 64
2.4.4.2 Feature-based Knowledge 65
2.4.4.3 Relation-based Knowledge 66
2.4.5 Lightweight Model Design / Lightweight Architectures 68
2.4.5.1 Depthwise Separable Convolution 68
2.4.5.2 Fire Module 70
2.4.5.3 Group Convolution 72
2.5 งานวิจัยที่เกี่ยวข้อง 74
2.5.1 การใช้การเรียนรู้ของเครื่องและแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ในการจำแนกโรคพืช 74
2.5.2 การพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันขนาดเล็กสำหรับ จำแนกโรคพืช 75

2.5.3 การปรับขนาดของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันโดยใช้เทคนิคการทำ Quantization 77
2.5.4 การพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยใช้เทคนิค Knowledge Distillation สำหรับการจำแนกโรคพืช 78
2.6 บทสรุปจากการศึกษาทฤษฎีที่เกี่ยวข้องและทบทวนวรรณกรรม 79
บทที่ 3 กระบวนการ และวิธีการดำเนินการวิจัย 81
3.1 การกำหนดประชากรและกลุ่มตัวอย่าง 82
3.1.1 ชุดข้อมูลใบข้าวโพดของชุดข้อมูล PlantVillage 82
3.1.2 การแบ่งชุดข้อมูลฝึกฝน และการแบ่งชุดข้อมูลทดสอบ 84
3.2 การฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการจำแนกโรคใบข้าวโพด 86
3.2.1 TensorFlow Framework 86
3.2.2 การทำ On-the-Fly Data Augmentation โดยใช้ ImageDataGenerator 88
3.2.3 แบบจำลอง Pre-trained ที่ใช้การศึกษาวิจัย 90
3.2.3.1 แบบจำลอง VGG16 91
3.2.3.2 แบบจำลอง DenseNet121 93
3.2.3.3 แบบจำลอง MobileNetV2 94
3.2.3.4 แบบจำลอง NASNet Mobile 96
3.2.4 การป้องกันการเกิด Overfitting ด้วย การปรับแต่งค่า Learning Rate และใช้ Early Stopping 97
3.2.4.1 การปรับแต่งค่า Learning Rate ด้วยฟังก์ชัน ReduceLROnPlateau 97
3.2.4.2 การใช้ Early Stopping 98
3.3 การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพด 98
3.3.1 การใช้แบบจำลองน้ำหนักเบา (Lightweight Model) 99

3.3.2 การทำ Knowledge Distillation 99
3.3.3 การทำ Quantization แบบ Post-training Quantization 102
3.4 การประเมินผล และเปรียบเทียบประสิทธิภาพแบบจำลองหลังจากผ่านกระบวนการปรับปรุงแบบจำลอง 105
3.4.1 การเปรียบเทียบความเหมาะสมแบบจำลอง 105
3.4.2 การใช้เมทริกซ์ตัดสินใจแบบถ่วงน้ำหนัก (Weighted Decision Matrix) 106
3.4.2.1 การกำหนดคะแนนในเกณฑ์ความแม่นยำ 111
3.4.2.2 การกำหนดคะแนนในเกณฑ์ขนาดแบบจำลอง 112
3.4.2.3 การกำหนดคะแนนในเกณฑ์เวลาประมวลผล 113
3.5 บทสรุปของกระบวนการ และวิธีการดำเนินการวิจัย 113
บทที่ 4 การทดลอง และผลลัพธ์ของการวิจัย 115
4.1 การทดลอง 115
4.1.1 การฝึกฝนแบบจำลอง Pre-trained Model 115
4.1.2 การฝึกฝนแบบจำลองสำหรับใช้เป็น Teacher Model 117
4.1.3 ออกแบบแบบจำลองสำหรับใช้เป็น Student Model 118
4.2 การปรับปรุงแบบจำลองให้เหมาะสมที่สุด 125
4.2.1 การทำ Quantization กับแบบจำลอง Pre-trained Model 125
4.2.2 การทำ Knowledge Distillation 126
4.3 ผลการทดลอง 127
4.3.1 ผลการฝึกผลแบบจำลอง 127
4.3.2 ผลการทดสอบแบบจำลองกับชุดข้อมูลทดสอบ 135
4.4 การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือ 136
4.5 การเลือกแบบจำลองที่เหมาะสมที่สุดโดยใช้ Weighted Decision Matrix 149

4.6 บทสรุปของการทดลองและผลการทดลอง 150
บทที่ 5 สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ 153
5.1 สรุปผลการวิจัย 153
5.1.1 การพัฒนาและฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน 153
5.1.2 การปรับปรุงแบบจำลองให้เหมาะสมที่สุดด้วยการทำ Quantization 156
5.1.3 การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือ 157
5.1.4 การเลือกแบบจำลองที่เหมาะสมที่สุดโดยใช้ Weighted Decision Matrix 158
5.2 อภิปรายผลการวิจัย 159
5.2.1 รูปแบบความผิดพลาดในการจำแนกโรค 159
5.2.2 ประสิทธิภาพของแบบจำลอง Pre-trained 159
5.2.3 ประสิทธิภาพของแบบจำลองที่พัฒนาด้วย Knowledge Distillation 160
5.2.4 ผลของการทำ Quantization ต่อความแม่นยำในการจำแนก 160
5.2.5 ข้อสังเกตเกี่ยวกับชุดข้อมูลที่ใช้ในการวิจัย 161
5.2.6 เปรียบเทียบกับงานวิจัยที่เกี่ยวข้อง 163
5.2.7 การเลือกใช้เทคนิคการลดขนาดของแบบจำลองให้เหมาะสมที่สุด 164
5.3 ข้อเสนอแนะ 166
5.3.1 การเพิ่มความหลากหลายของชุดข้อมูล 166
5.3.2 การปรับปรุงแบบจำลองให้เหมาะสมกับโรคของใบข้าวโพดที่เกิดในประเทศไทย 166
บรรณานุกรม 167

สารบัญตาราง

หน้า

ตาราง 1 แสดงรูปแบบของฟังก์ชันกระตุ้น 21

ตาราง 2 แสดงชั้นประเภทของการดำเนินการคอนโวลูชัน และจำนวนพารามิเตอร์ในแต่ละชั้นของ VGG16 34

ตาราง 3 โครงสร้างและจำนวนพารามิเตอร์ของ DenseNet121 39

ตาราง 4 แสดงจำนวนพารามิเตอร์ในแต่ละชั้นของแบบจำลองโครงข่ายประสาทเทียม MobileNetV2 43

ตาราง 5 แสดงรายละเอียดการดำเนินการในแต่ละชั้นของแบบจำลอง NASNet Mobile 46

ตาราง 6 แสดงความแตกต่างของการทำ Model Optimization และ Model Compression ... 48

ตาราง 7 แสดงจำนวนภาพของใบข้าวโพดในแต่ละโรค 83

ตาราง 8 แสดงจำนวนของตัวอย่างภาพในแต่ละหมวดหมู่ 84

ตาราง 9 แสดง Class Weight ของตัวอย่างแต่ละ Class ในชุดข้อมูลฝึกฝน 86

ตาราง 10 แสดงการทำ Data Augmentation ในระหว่างการฝึกฝนแบบจำลอง 90

ตาราง 11 ตารางแสดงระดับความสำคัญระหว่างเกณฑ์ 107

ตาราง 12 การกำหนดระดับความสำคัญของเกณฑ์แต่ละคู่ 108

ตาราง 13 แสดงค่าการคำนวณการหาค่าถ่วงน้ำหนักของแต่ละตัวเกณฑ์ 109

ตาราง 14 ระดับการให้คะแนนตามเกณฑ์ความแม่นยำจากการเรียงลำดับแบบจำลอง 111

ตาราง 15 ระดับการให้คะแนนตามเกณฑ์ขนาดแบบจำลอง (Model Size) 112

ตาราง 16 ระดับการให้คะแนนตามเกณฑ์เวลาประมวลผล (Inferencing Time) 113

ตาราง 17 แสดงการปรับค่าพารามิเตอร์ในการฝึกฝนแบบจำลองแบบ Pre-trained 116

ตาราง 18 แสดงการปรับค่าพารามิเตอร์ในการฝึกฝนแบบจำลองเพื่อใช้เป็น Teacher Model. 117

ตาราง 19 แสดงผลการทดสอบแบบจำลองเพื่อใช้เป็น Teacher Model กับชุดข้อมูลทดสอบ . 117

ตาราง 20 แสดงจำนวนพารามิเตอร์ในแต่ละชั้นของแบบจำลอง Standard Convolution Student Model 120

ตาราง 21 แสดงรายละเอียดจำนวนของพารามิเตอร์ในแต่ละชั้นของแบบจำลอง Depthwise Separable Convolution Student Model 124

ตาราง 22 แสดงผลการฝึกฝนแบบจำลอง Pre-trained Model 127

ตาราง 23 แสดงผลการฝึกฝนแบบจำลองโดยใช้เทคนิคการทำ Knowledge Distillation..... 132

ตาราง 24 แสดงผลการประเมินแบบจำลองกับชุดข้อมูลทดสอบ 137

ตาราง 25 แสดงคุณลักษณะของโทรศัพท์มือถือที่ใช้ทดสอบการประมวลผล 145

ตาราง 26 แสดงผลการทดสอบแบบจำลองกับขุดข้อมูลทดสอบบนอุปกรณ์โทรศัพท์มือถือ..... 146

ตาราง 27 แสดงผลการวิเคราะห์ประสิทธิภาพในการทำนายผลของแบบจำลองด้วยการใช้ Weighted Decision Matrix 151

สารบัญรูปภาพ

หน้า

ภาพประกอบ 1 แสดงแนวโน้มการเพิ่มขึ้นของผลผลิตข้าวโพดของโลกรายปี ตั้งแต่ ค.ศ. 2014 - 2023 1

ภาพประกอบ 2 แสดงแบบจำลองทางคณิตศาสตร์ภายใน Neuron ของโครงข่ายประสาทเทียม.. 7

ภาพประกอบ 3 แสดงโครงสร้างพื้นฐานของโครงข่ายประสาทเทียม 8

ภาพประกอบ 4 แสดงสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน 9

ภาพประกอบ 5 แสดงการดำเนินการคอนโวลูชัน ระหว่างนำเข้ากับตัวกรองเพื่อสร้าง Feature Map ในแต่ละชั้นของโครงข่ายประสาทเทียมแบบคอนโวลูชัน..... 12

ภาพประกอบ 6 แสดงการดำเนินการคอนโวลูชัน ระหว่างข้อมูลนำเข้ากับตัวกรอง 13

ภาพประกอบ 7 แสดงขนาดของผลลัพธ์ที่แตกต่างกัน เมื่อมีค่า stride ที่ต่างกัน 15

ภาพประกอบ 8 แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบ VGG16..... 32

ภาพประกอบ 9 แสดงจำนวน Receptive Field ของชั้นคอนโวลูชันที่ใช้ตัวกรองขนาด 3x3 จำนวน 2 ชั้น เพื่อให้ได้ Receptive Field ขนาด 5x5..... 33

ภาพประกอบ 10 แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบ VGG16..... 36

ภาพประกอบ 11 แสดงโครงสร้าง และการดำเนินการภายใน Dense Block..... 37

ภาพประกอบ 12 แสดงโครงสร้างและส่วนประกอบภายในแบบจำลองโครงข่ายประสาทเทียม แบบ MobileNetV2 40

ภาพประกอบ 13 แสดงการดำเนินการ Depthwise Convolution และ Pointwise convolution 41

ภาพประกอบ 14 แสดงการโครงสร้างสถาปัตยกรรม Inverted Residual Block..... 42

ภาพประกอบ 15 แสดงโครงสร้างภายใน Convolution Blocks ของแบบจำลอง MobileNetV243

ภาพประกอบ 16 แสดงโครงสร้างและการดำเนินการทางคณิตศาสตร์ภายใน Normal Cell และ Reduction Cell 45

ภาพประกอบ 17 แสดงการตัดแต่งแบบจำลองแบบ Structure Pruning..... 51
ภาพประกอบ 18 แสดงการตัดแต่งแบบจำลอง โดยการตัดค่าน้ำหนัก (Weight Pruning) 52
ภาพประกอบ 19 แสดงขั้นตอนการดำเนินการทำ Quantization แบบ Quantization-Aware Training และ Post-Training Quantization..... 55
ภาพประกอบ 20 แสดงโครงสร้างของตัวแปรชนิด FP32 FP16 และ INT8 56
ภาพประกอบ 21 แสดงช่วงของจำนวนจริงของการแปลงจาก FP32 ไปเป็นรูปแบบ FP16 56
ภาพประกอบ 22 แสดงช่วงของจำนวนจริงของการแปลงจาก FP32 ไปเป็นรูปแบบ INT8..... 58
ภาพประกอบ 23 แสดงการ Mapping ค่ามากที่สุดและน้อยที่สุดของ FP32 กับค่า INT8..... 59
ภาพประกอบ 24 แสดงการแปลงค่า FP32 ไปเป็นค่า INT8..... 59
ภาพประกอบ 25 แสดงการแปลงค่าจาก INT8 กลับไปเป็นค่า FP32..... 59
ภาพประกอบ 26 แสดงการตัวอย่างการทำ Matrix Decomposition..... 61
ภาพประกอบ 27 แสดงแนวความคิดในการพัฒนาแบบจำลองโดยวิธี Knowledge Distillation 62
ภาพประกอบ 28 แสดงแหล่งความรู้ (Knowledge) ที่ใช้ในการฝึกฝน Student Model ของแต่ละรูปแบบในการใช้เทคนิค Knowledge Distillation..... 63
ภาพประกอบ 29 แสดงการทำงานของ Response-based Distillation 64
ภาพประกอบ 30 แสดงการทำงานของ Feature-based Knowledge..... 66
ภาพประกอบ 31 แสดงการทำงานของ Relation-based Knowledge..... 67
ภาพประกอบ 32 แสดงการเปรียบเทียบการดำเนินการทางคณิตศาสตร์ของการคอนโวลูชันแบบปกติ และ Depthwise Separable Convolution..... 68
ภาพประกอบ 33 แสดงโครงสร้างของ Fire Module ภายในแบบจำลองแบบ SqueezeNet 70
ภาพประกอบ 34 แสดงการดำเนินการของ Fire Module กับข้อมูลรับเข้า..... 71
ภาพประกอบ 35 แสดงโครงสร้างของแบบจำลอง SqueezeNet แต่ละแบบ 72
ภาพประกอบ 36 แสดงภาพการเปรียบเทียบการคอนโวลูชันแบบปกติ กับ Group Convolution 73

ภาพประกอบ 37 แสดงการใช้งาน Group Convolution ใน Builder Block ของ แบบจำลอง ShuffleNet 74

ภาพประกอบ 38 แสดงกรอบแนวความคิดในการดำเนินการศึกษาวิจัย 81

ภาพประกอบ 39 แสดงตัวอย่างภาพใบข้าวโพดที่ในชุดข้อมูลที่นำมาใช้ในการทดลอง 83

ภาพประกอบ 40 แสดงการทำ On-the-fly Data Augmentation 89

ภาพประกอบ 41 แสดงภาพประกอบการเปรียบเทียบจำนวนพารามิเตอร์ระหว่างการใช้ Fully Connected และการใช้ Global Average Pooling 92

ภาพประกอบ 42 แสดงโครงสร้างของแบบจำลอง VGG16 หลังจากการปรับปรุงโครงสร้าง 92

ภาพประกอบ 43 แสดงโครงสร้างของแบบจำลอง DenseNet121 หลังจากการปรับปรุงโครงสร้าง 94

ภาพประกอบ 44 แสดงโครงสร้างของแบบจำลอง MovileNetV2 หลังจากการปรับปรุงโครงสร้าง 95

ภาพประกอบ 45 แสดงโครงสร้างของแบบจำลอง NASNet Mobile หลังจากการปรับปรุงโครงสร้าง 97

ภาพประกอบ 46 แสดงขั้นตอนการทำ Knowledge Distillation เพื่อพัฒนาแบบจำลองขนาดเล็กในการวิจัยครั้งนี้ 100

ภาพประกอบ 47 แสดงขั้นตอนการพัฒนาแบบจำลองและลดขนาดแบบจำลอง Pre-trained โดยการทำ Post-Training Quantization 102

ภาพประกอบ 48 แสดงขั้นตอนการพัฒนาแบบจำลองด้วยวิธีการทำ Knowledge Distillation และลดขนาดแบบจำลองโดยการทำ Post-Training Quantization 103

ภาพประกอบ 49 แสดงการรักษาสมดุลของการลดขนาดของแบบจำลองใน 3 ด้าน ได้แก่ ความแม่นยำ ขนาดของแบบจำลอง และ เวลาในการประมวลผลของแบบจำลอง 106

ภาพประกอบ 50 กราฟแสดงความแม่นยำในการทำนายผลของแบบจำลองที่ได้จากการศึกษางานวิจัยที่เกี่ยวข้องที่ผ่านมา 111

ภาพประกอบ 51กราฟแสดงขนาดของแบบจำลองจากการศึกษางานวิจัยที่เกี่ยวข้องที่ผ่านมา 112

ภาพประกอบ 52 แสดงโครงสร้างของ Standard Convolution Student Model 119

ภาพประกอบ 53 แสดงโครงสร้างแบบจำลอง Depthwise Separable Convolution Student Model 122

ภาพประกอบ 54 แสดงขั้นตอนการทำ Quantization กับแบบจำลอง Pre-trained Model 125

ภาพประกอบ 55 แสดง Training Accuracy ของแบบจำลอง VGG16 128

ภาพประกอบ 56 แสดง Training Loss ของแบบจำลอง VGG16..... 129

ภาพประกอบ 57 แสดง Training Accuracy ของแบบจำลอง DenseNet121..... 129

ภาพประกอบ 58 แสดง Training Loss ของแบบจำลอง DenseNet121 130

ภาพประกอบ 59 แสดง Training Accuracy ของแบบจำลอง MobileNetV2 130

ภาพประกอบ 60 แสดง Training Loss ของแบบจำลอง MobileNetV2..... 131

ภาพประกอบ 61 แสดง Training Accuracy ของแบบจำลอง NASNet Mobile..... 131

ภาพประกอบ 62 แสดง Training Loss ของแบบจำลอง NASNet Mobile 132

ภาพประกอบ 63 แสดง Training Accuracy ของแบบจำลอง STD Conv Student 133

ภาพประกอบ 64 แสดง Training Loss ของแบบจำลอง STD Conv Student..... 133

ภาพประกอบ 65 แสดง Training Accuracy ของแบบจำลอง DW Conv Student 134

ภาพประกอบ 66 แสดง Training Loss ของแบบจำลอง DW Conv Student..... 134

ภาพประกอบ 67 แสดง Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลองโดยใช้ชุดข้อมูลทดสอบ 144

ภาพประกอบ 68 แสดง Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง โดยใช้ชุดข้อมูลทดสอบบนอุปกรณ์โทรศัพท์มือถือ 149

ภาพประกอบ 69 แสดงการเปรียบเทียบความแม่นยำ (Accuracy) ในการทำนายผลของแบบจำลองบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime กับชุดข้อมูลทดสอบ 155

ภาพประกอบ 70 แสดงการเปรียบเทียบเวลาในการทำนายผลของแบบจำลองบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime กับชุดข้อมูลทดสอบ..... 155

ภาพประกอบ 71 แสดงการเปรียบเทียบขนาดของแบบจำลองเมื่อผ่านการปรับปรุงปรับปรุงแบบจำลองด้วยวิธี Post-training Quantization 156

ภาพประกอบ 72 แสดงการเปรียบเทียบความแม่นยำ (Accuracy) ในการทำนายผลของแบบจำลองบนอุปกรณ์โทรศัพท์มือถือกับชุดข้อมูลทดสอบ..... 157

ภาพประกอบ 73 แสดงการเปรียบเทียบเวลาที่ใช้ในการทำนายผลของแบบจำลองบนอุปกรณ์โทรศัพท์มือถือ 158

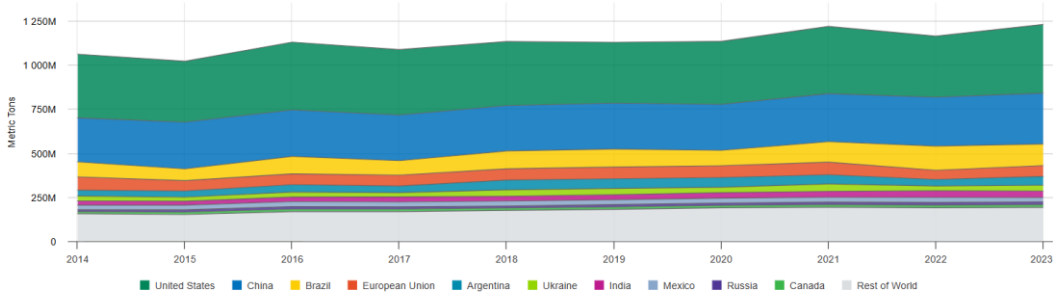
ภาพประกอบ 74 แสดงภาพหมวดหมู่จริง Northern Leaf Blight แต่ทุกแบบจำลองทำนายเป็นหมวดหมู่ Cercospora 162

ภาพประกอบ 75 แสดงภาพหมวดหมู่จริง Cercospora แต่ทุกแบบจำลองทำนายเป็น หมวดหมู่ Northern Leaf Blight 162

บทที่ 1
บทนำ

1.1 ภูมิหลัง

เกษตรกรรมยังคงเป็นรากฐานสำคัญของความมั่นคงทางอาหารและเศรษฐกิจของหลายประเทศทั่วโลก โดยเฉพาะการเพาะปลูกข้าวโพดที่ถือเป็นพืชเศรษฐกิจที่มีความสำคัญอย่างมาก ซึ่งเป็นทั้งพืชอาหารหลักเพื่อการบริโภคของมนุษย์ และยังถูกใช้เป็นวัตถุดิบสำคัญในอุตสาหกรรมอาหารสัตว์ และข้อมูลจาก USDA (USDA, 2024) พบว่าจำนวนผลผลิตของข้าวโพดมีแนวโน้มที่จะเพิ่มสูงขึ้นในแต่ละปี ดังแสดงในภาพประกอบ 1 จึงนับได้ว่าข้าวโพดมีบทบาทสำคัญในการรักษาสมดุลของระบบอาหารโลก การระบาดของโรคพืชที่เกิดกับใบข้าวโพด (Erenstein et al., 2022) เช่น โรคใบไหม้แผลใหญ่ (Northern Corn Leaf Blight) โรคราสนิม (Rust) และโรคใบจุด งถือเป็นภัยคุกคามสำคัญต่อความมั่นคงทางอาหารและรายได้ของเกษตรกร โดยเฉพาะในบริบทของการเปลี่ยนแปลงสภาพแวดล้อมทางธรรมชาติที่ทำให้รูปแบบการระบาดของโรคมีความซับซ้อนและคาดเดายากยิ่งขึ้น การตรวจจับและวินิจฉัยโรคในระยะเริ่มต้นจึงมีความสำคัญอย่างยิ่งต่อการควบคุมการแพร่ระบาดของโรค และลดความเสียหายของผลผลิตข้าวโพดในบริเวณกว้างของเกษตรกร (Kilaru & MurthyRaju, 2022)



ภาพประกอบ 1 แสดงแนวโน้มการเพิ่มขึ้นของผลผลิตข้าวโพดของโลกรายปี ตั้งแต่ ค.ศ. 2014 - 2023

ที่มา: USDA Retrieved September 28, 2024, from https://fas.usda.gov/data/production/commodity/0440000

การพัฒนาอย่างก้าวกระโดดของเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence : AI) และการเรียนรู้ของเครื่อง (Machine Learning) โดยเฉพาะอย่างยิ่งการเรียนรู้เชิงลึก โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks : CNN) ได้เปิดมิติใหม่ในการแก้ไขปัญหาที่ซับซ้อนในภาคการเกษตร ซึ่งเป็นเทคนิคการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงในการวิเคราะห์ภาพและจำแนกภาพ ให้เห็นถึงศักยภาพอันโดดเด่นในการตรวจจับ วินิจฉัย และจำแนกโรคพืชจากภาพถ่าย โดยมีความแม่นยำสูงและสามารถทำงานได้อย่างรวดเร็ว การนำแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน มาประยุกต์ใช้ในการตรวจจับโรคใบข้าวโพด จึงเป็นแนวทางที่มีความน่าสนใจในการปฏิวัติวิธีการจัดการโรคพืชแบบดั้งเดิม

อย่างไรก็ตามการนำแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันไปใช้งานจริงในภาคสนามยังคงเป็นความท้าทายที่สำคัญ ถึงแม้ว่าแบบจำลองแบบดังกล่าวจะประสิทธิภาพสูง แต่ก็มักจะมีสถาปัตยกรรมที่มีความความซับซ้อน และแบบจำลองมีขนาดใหญ่ มีจำนวนตัวแปรจำนวนมาก (Mishra et al., 2020) ทำให้ต้องการทรัพยากรในการประมวลผลและหน่วยความจำสูง ซึ่งเป็นข้อจำกัดสำคัญสำหรับการใช้งานบนอุปกรณ์พกพา หรือในพื้นที่ชนบทที่มีข้อจำกัดด้านโครงสร้างพื้นฐานทางการสื่อสารที่ไม่สามารถเชื่อมต่อกับเครือข่ายอินเตอร์เน็ตได้ตลอดเวลา นอกจากนี้ การใช้พลังงานที่สูงของแบบจำลองขนาดใหญ่ยังเป็นอุปสรรคต่อการใช้งานในระยะยาวของอุปกรณ์พกพา หากต้องนำไปใช้ในพื้นที่เพาะปลูกที่อาจมีข้อจำกัดด้านแหล่งพลังงานไฟฟ้าให้กับอุปกรณ์ในการประมวลผล

ด้วยเหตุนี้ การพัฒนาเทคนิคการปรับปรุงแบบจำลองให้เหมาะสมที่สุด (Model Optimization) จึงมีความสำคัญอย่างยิ่งในการทำให้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สามารถนำไปใช้งานได้จริงในภาคสนามกับอุปกรณ์พกพา โดยเทคนิคเหล่านี้มุ่งเน้นการลดขนาดและความซับซ้อนของแบบจำลอง ในขณะที่ยังคงรักษาประสิทธิภาพและความแม่นยำในการทำนายไว้ให้ได้มากที่สุด เทคนิคการปรับปรุงแบบจำลองที่ได้รับความนิยม ได้แก่ การตัดแต่งแบบจำลอง (Model Pruning) การบีบอัดแบบจำลอง (Model Compression) การกลั่นความรู้ (Knowledge Distillation) การแปลงควอนไทซ์ (Quantization) และการออกแบบแบบจำลองแบบเบา (Lightweight Model Design) ซึ่งแต่ละเทคนิคมีจุดเด่นและข้อจำกัดที่แตกต่างกันไป

การศึกษาและประยุกต์ใช้เทคนิคการปรับปรุงแบบจำลองที่ได้กล่าวมาข้างต้น กับแบบจำลอง โครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการตรวจจับโรคใบข้าวโพด จึงเป็นประเด็นวิจัยที่มีความสำคัญและท้าทาย โดยมีเป้าหมายในการพัฒนาแบบจำลองที่มีประสิทธิภาพสูง ใช้ทรัพยากรในการประมวลผลน้อย และสามารถนำไปใช้งานได้จริงในสภาพแวดล้อมการ

เพาะปลูกที่หลากหลาย เพื่อให้เทคโนโลยีนี้สามารถนำไปใช้ได้อย่างมีประสิทธิภาพในบริบทของการเกษตรในประเทศกำลังพัฒนา ซึ่งจะช่วยยกระดับการควบคุมจัดการโรคพืช เพิ่มผลผลิต และสร้างความยั่งยืนในภาคเกษตรกรรม

1.2 ความมุ่งหมายของงานวิจัย

ในการวิจัยครั้งนี้ผู้วิจัยได้ตั้งความมุ่งหมายไว้ดังนี้

1. เพื่อศึกษาเทคนิคการปรับปรุงแบบจำลอง ได้แก่ การกลั่นความรู้ การแปลงควอนไทซ์ และการใช้แบบจำลองแบบเบา

2. เพื่อพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่มีขนาดเล็ก และมีประสิทธิภาพในการจำแนกโรคใบข้าวโพด

3. เพื่อวิเคราะห์ และเปรียบเทียบผลกระทบของการปรับปรุงแบบจำลองต่อความแม่นยำ และประสิทธิภาพการทำงาน

1.3 ความสำคัญของงานวิจัย

งานวิจัยนี้มีความสำคัญในการพัฒนารูปแบบเทคนิคในการปรับปรุงแบบจำลองให้มีความเหมาะสมที่สุดการสำหรับการจำแนกโรคใบข้าวโพดที่มีประสิทธิภาพสูงและเหมาะสมสำหรับนำไปใช้งานได้ในสภาพแวดล้อมจริง ซึ่งจะช่วยให้เกษตรกรสามารถตรวจสอบ และจัดการโรคพืชที่ตรวจพบได้อย่างรวดเร็วและแม่นยำ ช่วยลดความเสียหายของผลผลิต ลดการใช้สารเคมี และเพิ่มประสิทธิภาพในการทำการเกษตรกรรม นอกจากนี้ ยังเป็นการพัฒนาองค์ความรู้ด้านการปรับปรุงแบบจำลองให้มีความเหมาะสมที่สุดเพื่อให้แบบจำลองให้มีขนาดเล็กลง และเหมาะสมสำหรับการนำไปใช้งานได้อย่างมีประสิทธิภาพบนอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร

1.4 ขอบเขตงานวิจัย

1.4.1 ประชากรที่ใช้ในการวิจัย

ในการวิจัยครั้งนี้ใช้ภาพถ่ายโรคใบข้าวโพด จาก ชุดข้อมูลเปิด Plant Village (Hughes & Salathé, 2015) โดยใช้ชุดข้อมูลเฉพาะใบข้าวโพดที่ไม่ได้ผ่านการการทำ Augmentation

1.4.2 กลุ่มตัวอย่างที่ใช้ในการวิจัย

ภาพถ่ายใบข้าวโพด จำนวน 4 กลุ่ม (Class) ได้แก่ เช่น โรคใบไหม้แผลใหญ่ (Northern Corn Leaf Blight) โรคราสนิม (Rust) โรคใบจุด (Gray Leaf Spot) และใบปกติ (Healthy) รวมทั้งสิ้น 3,852 ภาพ

1.4.3 ตัวแปรที่ใช้ศึกษา

1. ตัวแปรอิสระ ได้แก่ เทคนิคการปรับปรุงแบบจำลอง ได้แก่ การกลั่นความรู้ การแปลงควอนไทซ์ การออกแบบแบบจำลองแบบเบา และการใช้เทคนิคแบบผสมผสาน

2. ตัวแปรตาม ได้แก่ ขนาดของแบบจำลอง ความแม่นยำของแบบจำลอง เวลาที่ใช้ในการประมวลผล

1.5 กรอบแนวความคิดในงานวิจัย

งานวิจัยนี้จะเริ่มจากการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่มีประสิทธิภาพสูงในการจำแนกโรคใบข้าวโพด จากนั้นจะทำการศึกษาและประยุกต์ใช้เทคนิคการปรับปรุงแบบจำลองหลายวิธี ได้แก่ การตัดแต่งแบบจำลอง การกลั่นความรู้ การแปลงควอนไทซ์ และแบบผสมผสาน เพื่อลดขนาดแบบจำลองและเพิ่มประสิทธิภาพการทำงาน โดยจะทำการเปรียบเทียบประสิทธิภาพของแต่ละเทคนิค เพื่อหาแนวทางที่เหมาะสมที่สุดในการพัฒนาระบบตรวจจับโรคใบข้าวโพดที่มีประสิทธิภาพ และเหมาะสมสำหรับการนำไปใช้งานได้บนอุปกรณ์ที่มีทรัพยากรจำกัด

1.6 สมมติฐานในการวิจัย

1. การใช้เทคนิคการปรับปรุงแบบจำลอง โดยการแปลงควอนไทซ์ (Quantization) แบบ Dynamic Range Quantization จะช่วยลดขนาดของแบบจำลองได้ และมีผลกระทบต่อความแม่นยำในการจำแนกโรคเพียงเล็กน้อย

2. การใช้เทคนิคการปรับปรุงแบบจำลอง แบบผสมผสาน (Combined Techniques) จะช่วยลดขนาดของแบบจำลองได้มากกว่าวิธีแบบเดี่ยว และมีผลกระทบต่อความแม่นยำในการจำแนกโรคเพียงเล็กน้อย

บทที่ 2
ทฤษฎีที่เกี่ยวข้องและทบทวนวรรณกรรม

ในการวิจัยครั้งนี้ ผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง และได้นำเสนอตามหัวข้อต่อไปนี้

1. การใช้การเรียนรู้ของเครื่องในการจำแนกโรคพืช
2. โครงข่ายประสาทเทียมแบบคอนโวลูชัน
3. สถาปัตยกรรมแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการใช้งานเพื่อจำแนกภาพ
4. เทคนิคการ Model Optimization สำหรับการลดขนาดแบบจำลอง
5. งานวิจัยที่เกี่ยวข้องกับการใช้โครงข่ายประสาทเทียมแบบคอนโวลูชันในการตรวจจับโรคใบข้าวโพด

2.1 การใช้การเรียนรู้ของเครื่องในการจำแนกโรคพืช

การตรวจจำแนกโรคพืชเป็นหนึ่งในกระบวนการสำคัญที่ส่งผลต่อการควบคุมคุณภาพและรักษาปริมาณของผลผลิตทางการเกษตร ซึ่งการตรวจจำแนกโรคพืชได้อย่างรวดเร็วและแม่นยำจะช่วยลดผลกระทบต่อความเสียหายที่อาจจะเกิดขึ้นกับผลผลิตทางการเกษตร และป้องกันการแพร่กระจายของโรคพืชไม่ให้ระบาดไปสู่บริเวณกว้าง การตรวจจับโรคพืชด้วยวิธีการดั้งเดิมคือการใช้มนุษย์เป็นผู้ตรวจ ซึ่งจำเป็นต้องพึ่งพาการตรวจสอบจากผู้เชี่ยวชาญในการวินิจฉัยโรคพืชเหล่านั้น มักต้องใช้เวลาและทรัพยากรจำนวนมากในการดำเนินการ และบางครั้งวิธีการดังกล่าวอาจจะเกิดข้อผิดพลาดจากการตรวจสอบโรคพืช

ปัญญาประดิษฐ์ได้กลายมาเป็นเครื่องมือที่มีศักยภาพในการปรับปรุงกระบวนการนี้ โดยเฉพาะในแง่ของการประมวลผลข้อมูลจำนวนมากจากภาพถ่ายของพืช ซึ่งปัญญาประดิษฐ์สามารถจำแนกลักษณะทางกายภาพของใบพืชได้ เช่น การเปลี่ยนแปลงของสี ใบที่เป็นจุด หรือการเปลี่ยนรูปแบบของเนื้อเยื่อพืช ทำให้สามารถระบุโรคได้อย่างแม่นยำและรวดเร็ว นอกจากนี้ปัญญาประดิษฐ์ยังสามารถช่วยลดต้นทุนแรงงานและลดความผิดพลาดที่เกิดจากมนุษย์ ซึ่งช่วยให้เกษตรกรสามารถตรวจวินิจฉัยโรคพืชได้อย่างแม่นยำ และสามารถจัดการกับโรคพืชได้อย่างมีประสิทธิภาพ

2.1.1 พัฒนาการของการใช้การเรียนรู้ของเครื่องในการจำแนกโรคพืช

การใช้คอมพิวเตอร์ช่วยตรวจหาโรคในพืชได้มีการพัฒนามาอย่างต่อเนื่องตลอดหลายปีที่ผ่านมา ตั้งแต่การใช้การประมวลผลภาพ (Image Processing) และการใช้การมองเห็นของ

เครื่อง (Computer Vision) ในการประมวลผลลักษณะของภาพส่วนประกอบของพืช เช่น สี รูปร่าง และลักษณะผิวของใบพืช ถ้าเห็นอะไรผิดปกติ ก็จะทำนายผลว่าพืชอาจจะเป็นโรค วิธีนี้ใช้ได้ผลบ้าง แต่ยังไม่ค่อยแม่นยำนัก เพราะบางครั้งไม่สามารถจำแนกโรคของพืชได้อย่างชัดเจน

ต่อมามีการใช้การเรียนรู้ของเครื่อง (Machine Learning) เพื่อฝึกฝนให้คอมพิวเตอร์สามารถจำแนกโรคพืชบางอย่างจากภาพถ่ายได้ โดยใช้เทคนิคการเรียนรู้ของเครื่อง แบบดั้งเดิม เช่น K-nearest neighbors (KNN) Support Vector Machines (SVMs) และ Random Forests เป็นต้น ในการวิเคราะห์ลักษณะเฉพาะของใบพืชที่ถูกสกัดออกมาด้วยมือ (Hand-Crafted Features) เพื่อระบุโรค โดยจะใช้คอมพิวเตอร์สร้างแบบจำลอง (Model) เพื่อเรียนรู้ลักษณะของโรคจากตัวอย่างจำนวนมาก ๆ เช่น ดูภาพใบพืชที่เป็นโรคหลาย ๆ ใบ แล้วให้แบบจำลองที่ผ่านการเรียนรู้แล้วทำนายผลการจำแนกโรคจากภาพใบพืชที่พบใหม่ แม้ว่าวิธีการเหล่านี้จะให้ผลลัพธ์ที่น่าพอใจในระดับหนึ่ง แต่ก็ยังมีข้อจำกัดในด้านความแม่นยำและความสามารถในการใช้งานกับสภาพแวดล้อมที่ต้องการความซับซ้อนของแบบจำลอง (Yao et al., 2023)

จุดเปลี่ยนสำคัญเกิดขึ้นเมื่อมีการนำการเรียนรู้เชิงลึก (Deep Learning) โดยเฉพาะอย่างยิ่งโครงข่ายประสาทเทียมแบบคอนโวลูชันมาใช้ในการวิเคราะห์ภาพใบพืชเพื่อจำแนกโรคพืช ซึ่งสามารถเรียนรู้ลักษณะเฉพาะของโรคพืชได้โดยอัตโนมัติจากภาพ โดยไม่จำเป็นต้องอาศัยการสกัดลักษณะด้วยมืออีกต่อไป นอกจากนี้ ยังสามารถจัดการกับความซับซ้อนของข้อมูลภาพได้ดีกว่า ส่งผลให้มีความแม่นยำในการตรวจจับโรคสูงขึ้นอย่างมีนัยสำคัญ

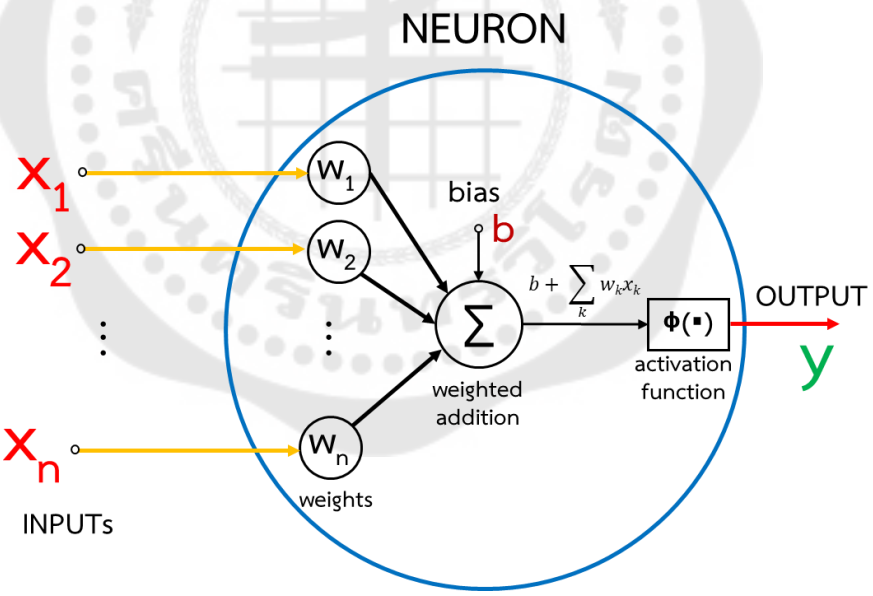
2.1.2 ข้อจำกัดและความท้าทายของการใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ในภาคสนาม

แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ได้กลายเป็นเครื่องมือที่ทรงพลังในการประมวลผลภาพและการรู้จำรูปแบบเกี่ยวข้องกับการเกษตร โดยเฉพาะการตรวจจับโรคพืช ในช่วงที่ผ่านมา แต่อย่างไรก็ตามการนำแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ไปใช้งานจริงในภาคสนาม ซึ่งจะต้องมีการนำเอาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันไปประยุกต์ใช้กับอุปกรณ์พกพา หรือการพัฒนาเป็นโปรแกรมประยุกต์ (Applications) สำหรับอุปกรณ์มือถือ โดยยังคงมีข้อจำกัดและความท้าทายที่สำคัญหลายประการ เนื่องจากโครงสร้างของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน มีความซับซ้อนมากและมีจำนวนของตัวแปรหรือพารามิเตอร์จำนวนมาก จึงต้องการทรัพยากรสูงในการประมวลผลและต้องการหน่วยความจำนวนนาก ทำให้ต้องการอุปกรณ์ประมวลผลที่มีสมรรถนะและมีประสิทธิภาพ ซึ่งอุปกรณ์เหล่านี้มักมีราคาแพงและใช้พลังงานในการประมวลผลมาก ทำให้เป็นอุปสรรคหากจะต้องนำอุปกรณ์พกพาไปใช้ในพื้นที่ที่มีข้อจำกัดด้านแหล่งพลังงาน (Zaniolo et al., 2023)

การทำงานของโครงข่ายประสาทเทียมแบบคอนโวลูชัน ยังมีข้อจำกัดในด้านการใช้งานแบบ Real-Time ซึ่งเป็นความท้าทายสำคัญในภาคสนาม ที่อยู่ห่างไกลหรือมีข้อจำกัดในการเชื่อมต่ออินเตอร์เน็ตหรือไม่มีสัญญาณอินเทอร์เน็ตเลย การประมวลผลภาพแบบ Real-Time ที่ต้องการการตอบสนองอย่างรวดเร็ว โดยไม่ต้องมีการเชื่อมต่อมายัง Server หรือ Cloud ซึ่งแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ที่มีความซับซ้อนสูงอาจไม่สามารถประมวลผลได้เร็วพอ หรืออาจจะไม่สามารถประมวลได้ ฉะนั้นการนำแบบจำลองไปใช้ในพื้นที่ห่างไกลจึงจำเป็นต้องพิจารณาการปรับแต่งเพื่อให้สามารถใช้งานได้แบบออฟไลน์ เช่น การลดขนาดแบบจำลอง หรือการใช้เทคนิคการลดความซับซ้อนของแบบจำลอง เช่น การแปลงควอนไทซ์ (Quantization) เพื่อการลดความละเอียดของพารามิเตอร์ของแบบจำลอง หรือ การตัดแต่งแบบจำลอง (Model Pruning) เพื่อลดความซับซ้อนของแบบจำลอง เป็นต้น

2.2 โครงข่ายประสาทเทียมแบบคอนโวลูชัน

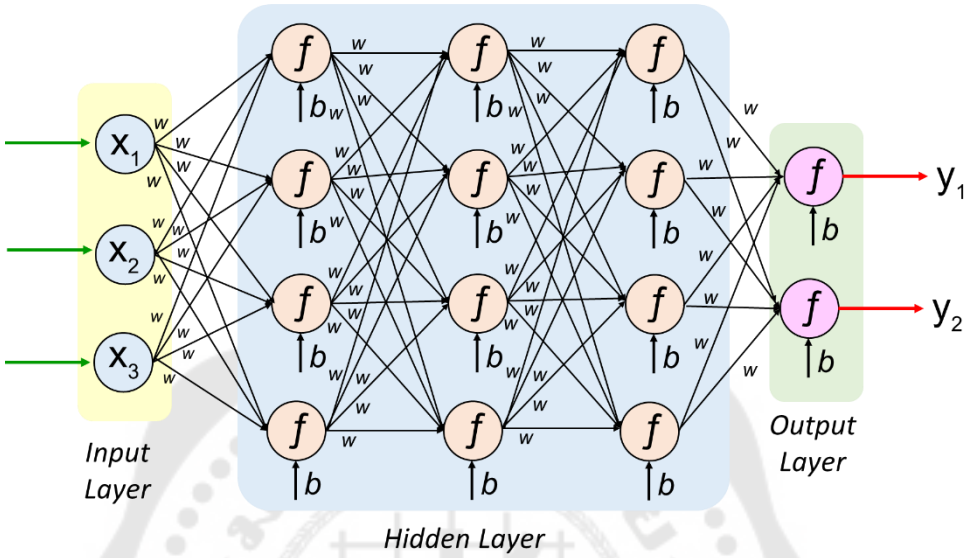
2.2.1 ความรู้พื้นฐานของโครงข่ายประสาทเทียมแบบคอนโวลูชัน



ภาพประกอบ 2 แสดงแบบจำลองทางคณิตศาสตร์ภายใน Neuron ของโครงข่ายประสาทเทียม

โครงข่ายประสาทเทียม (Neural Networks) เป็นแบบจำลองการคำนวณที่ได้รับแรงบันดาลใจจากระบบประสาทในสมองของมนุษย์ ซึ่งแนวคิดพื้นฐานของโครงข่ายประสาทเทียม คือ การจำลองการทำงานของเซลล์ประสาท (Neurons) และการเชื่อมต่อระหว่างเซลล์ (Synapses) ออกมาเป็นรูปแบบของแบบจำลองทางคณิตศาสตร์ โครงข่ายประสาทเทียมประกอบด้วยหน่วย

ประมวลผลจำนวนมากที่เชื่อมต่อกัน แต่ละหน่วยเรียกว่า "โหนด" (Node) หรือ "ยูนิต" (Unit) หรือ นิวรอน (Neuron) ซึ่งทำหน้าที่คล้ายกับเซลล์ประสาทในสมอง โดยรับข้อมูลเข้า ประมวลผล และส่งผลลัพธ์ออกไปยังโหนดอื่นที่เชื่อมต่อกันเป็นชั้น ๆ (Layers) ดังภาพประกอบ 2



ภาพประกอบ 3 แสดงโครงสร้างพื้นฐานของโครงข่ายประสาทเทียม

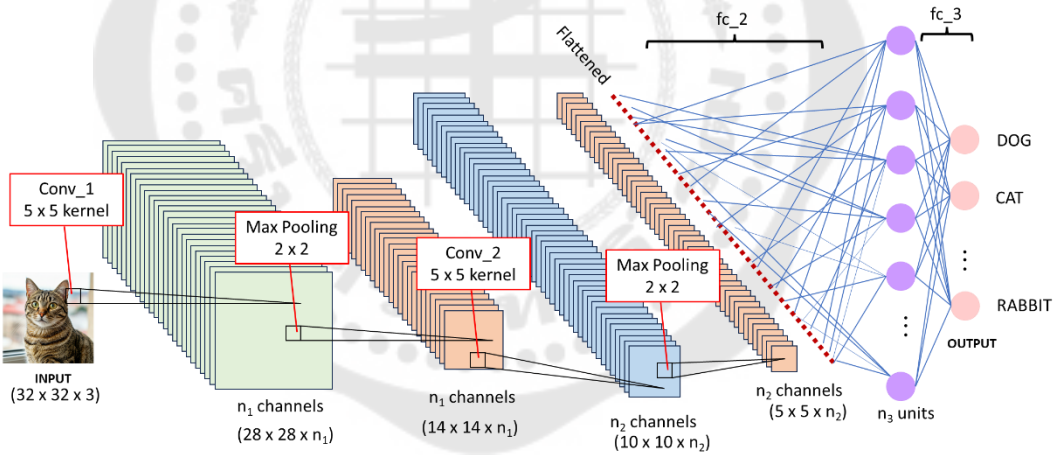
โครงสร้างพื้นฐานของโครงข่ายประสาทเทียมประกอบด้วยชั้นข้อมูลเข้า (Input Layer) ชั้นซ่อน (Hidden Layers) จำนวนตั้งแต่หนึ่งชั้นขึ้นไป และชั้นผลลัพธ์ (Output Layer) ดังภาพประกอบ 3 การเชื่อมต่อระหว่างโหนดในแต่ละชั้นมีค่าน้ำหนัก (Weights) ซึ่งจะถูกปรับค่าในระหว่างกระบวนการเรียนรู้ (Training) ความสามารถในการเรียนรู้นี้ทำให้แบบจำลองโครงข่ายประสาทเทียมสามารถแก้ปัญหาที่ซับซ้อนและปรับตัวกับข้อมูลใหม่ที่ไม่เคยพบมาก่อนได้ โดยไม่จำเป็นต้องเขียนโปรแกรมแบบชัดเจน (Explicit Programming) สำหรับแต่ละสถานการณ์ที่อาจจะต้องพบเจอข้อมูลรับเข้า (Input) ที่แปลกใหม่

กระบวนการเรียนรู้ของแบบจำลองโครงข่ายประสาทเทียมเกิดขึ้นผ่านการให้แบบจำลองเรียนรู้จากข้อมูลชุดเรียนรู้ (Training Set) แล้วปรับค่าน้ำหนักของการเชื่อมต่อระหว่างโหนด โดยใช้อัลกอริทึมการเรียนรู้ เช่น การแพร่ย้อนกลับ (Backpropagation) ซึ่งคำนวณความผิดพลาด (Loss) ระหว่างผลลัพธ์ที่ได้จากการทำนายผลของแบบจำลอง กับค่าจริง (True Label) แล้วปรับค่าน้ำหนักเพื่อลดความผิดพลาดนี้ให้ได้มากที่สุด การเรียนรู้แบบนี้ทำให้โครงข่ายประสาทเทียมสามารถสร้างแบบจำลองที่ซับซ้อนจากข้อมูลตัวอย่างจำนวนมาก และนำไปใช้ในการทำนายหรือจำแนกข้อมูลใหม่ได้อย่างมีประสิทธิภาพ

ในการประมวลผลภาพ โครงข่ายประสาทเทียมแบบคอนโวลูชันถือเป็นเทคโนโลยีที่มีบทบาทสำคัญอย่างยิ่ง ซึ่งเป็นโครงข่ายประสาทเทียมประเภทหนึ่งที่ได้รับการออกแบบมาโดยเฉพาะเพื่อประมวลผลข้อมูลที่มีโครงสร้างที่แน่นอน เช่น ภาพดิจิตอล ซึ่งประกอบด้วยขนาดของภาพและจำนวนช่อง (Channel) โดยมีแนวคิดพื้นฐานมาจากการจำลองการมองเห็นของดวงตาจากการทำงานของเซลล์ประสาทในคอร์เท็กซ์ ความสามารถอันโดดเด่นของโครงข่ายประสาทเทียมแบบคอนโวลูชัน คือ การสกัดคุณลักษณะ (Feature Extraction) จากภาพได้อย่างมีประสิทธิภาพ โดยไม่จำเป็นต้องอาศัยการสกัดหรือการหาคุณลักษณะที่อยู่ภายในภาพด้วยมือ (Hand-Crafted Features) เหมือนกับวิธีการประมวลผลภาพแบบดั้งเดิม

2.2.2 โครงสร้างพื้นฐานของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

สถาปัตยกรรมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน ประกอบด้วยชั้นพิเศษเชื่อมต่อกันหลายชั้น ได้แก่ ชั้น Convolutional Layers ชั้น Pooling Layers และชั้น Fully Connected Layers ดังภาพประกอบ 4 ซึ่งแต่ละชั้นมีบทบาทเฉพาะในการสกัดคุณลักษณะและประมวลผลข้อมูลแตกต่างกันไป (Nelson, 2018)



ภาพประกอบ 4 แสดงสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

2.2.2.1 ชั้น Convolutional Layers เป็นองค์ประกอบสำคัญที่สุดของโครงข่ายประสาทเทียมแบบคอนโวลูชันทำหน้าที่สกัดคุณลักษณะจากภาพนำเข้า กระบวนการนี้เกิดขึ้นโดยการใช้ตัวกรอง (Filters) หรือเคอร์เนล (Kernels) ซึ่งเป็นเมทริกซ์ขนาดเล็ก (เช่น 3x3 หรือ 5x5) ที่เลื่อนไปบนภาพ ในแต่ละตำแหน่ง ตัวกรองจะทำการคำนวณผลคูณแบบ Dot Product ระหว่างค่าของตัวกรองกับค่าของจุดภาพ (Pixel) ในพื้นที่ที่ซ้อนทับกัน ผลลัพธ์ที่ได้จะถูกส่งผ่านฟังก์ชันกระตุ้น (Activation Function) เช่น ReLU (Rectified Linear Unit) เพื่อสร้าง Feature Map สำหรับ

ตัวกรองนั้นๆ ชั้นคอนโวลูชั่นหนึ่งชั้นมักประกอบด้วยตัวกรองหลายตัวที่ทำงานพร้อมกัน ทำให้สามารถสกัดคุณลักษณะที่หลากหลายได้ เช่น เส้นแนวตั้ง แนวนอน หรือเฉียง ขอบ และรูปแบบพื้นผิวต่างๆ

ความสำคัญของชั้น Convolutional Layers ในการตรวจจับโรคจากใบข้าวโพดอยู่ที่ความสามารถในการเรียนรู้และสกัดคุณลักษณะที่สำคัญของโรคโดยอัตโนมัติ เช่น รูปแบบของจุดหรือรอยด่างบนใบ การเปลี่ยนแปลงของสีใบ หรือลักษณะการเสื่อมสภาพของเนื้อเยื่อ โดยไม่จำเป็นต้องกำหนดคุณลักษณะเหล่านี้ด้วยมือ นอกจากนี้ การใช้ตัวกรองเดียวกันกับทุกส่วนของภาพ (Parameter Sharing) ทำให้โครงข่ายประสาทเทียมแบบคอนโวลูชันสามารถตรวจจับคุณลักษณะเดียวกันในตำแหน่งต่างๆ ของภาพได้ ซึ่งเป็นประโยชน์อย่างมากในการตรวจจับโรคที่อาจปรากฏในตำแหน่งใดก็ได้บนใบข้าวโพด

2.2.2.2 ชั้น Pooling Layers ทำหน้าที่ลดขนาดของ Feature Map ที่ได้จากชั้น Convolution Layers โดยการสรุปข้อมูลในพื้นที่ย่อยของแผนที่ วิธีการพูลลิ่งที่นิยมใช้มากที่สุดคือการเลือกค่าสูงสุด (Max pooling) ซึ่งเลือกค่าสูงสุดจากแต่ละพื้นที่ย่อย (เช่น หน้าต่างขนาด 2x2) ของ Feature Map วิธีการนี้ช่วยลดขนาดของข้อมูลลงครึ่งหนึ่งในแต่ละมิติ นอกจาก การเลือกค่าสูงสุดแล้วยังมีวิธีอื่นๆ เช่น การเลือกค่าเฉลี่ย (Average Pooling) ที่ใช้ค่าเฉลี่ยแทนค่าสูงสุด หรือ Global Pooling ที่สรุปข้อมูลจากทั้ง Feature Map ให้เหลือเพียงค่าเดียว

ชั้นพูลลิ่งมีประโยชน์หลายประการในการตรวจจับโรคจากใบข้าวโพด ประการแรก การลดขนาดของข้อมูลช่วยลดจำนวนพารามิเตอร์ในแบบจำลอง ทำให้การคำนวณเร็วขึ้นและลดความเสี่ยงของการเกิด Overfitting ประการที่สอง การเลือกค่าสูงสุดหรือค่าเฉลี่ยช่วยให้แบบจำลองมีความทนทานต่อการเปลี่ยนแปลงเล็กน้อยในตำแหน่งของคุณลักษณะ (Spatial Invariance) ซึ่งเป็นประโยชน์ในการตรวจจับโรคที่อาจมีลักษณะแตกต่างกันเล็กน้อยในแต่ละใบ ประการที่สาม การลดขนาดของข้อมูลช่วยขยาย Receptive Field ของโหนดในชั้นถัดไป ทำให้สามารถมองเห็นบริบทที่กว้างขึ้นของภาพ ซึ่งอาจช่วยในการระบุโรคที่มีผลกระทบต่อใบในวงกว้าง

2.2.2.3 ชั้น Fully Connected Layers ทำหน้าที่รวมคุณลักษณะที่สกัดได้จากชั้นก่อนหน้าเพื่อทำการจำแนกประเภทหรือทำนายผลลัพธ์สุดท้าย ในชั้นนี้ โหนดทุกตัวจะเชื่อมต่อกับโหนดทั้งหมดในชั้นก่อนหน้า ทำให้สามารถรวมข้อมูลจากทุกส่วนของภาพเพื่อตัดสินใจ โดยทั่วไป CNN จะมีชั้นเชื่อมต่อแบบเต็มหนึ่งหรือสองชั้นก่อนที่จะส่งข้อมูลไปยังชั้นผลลัพธ์สุดท้าย ซึ่งใช้ฟังก์ชันกระตุ้นที่เหมาะสมกับงาน เช่น SoftMax สำหรับการจำแนกประเภทหลายคลาส หรือ sigmoid สำหรับการจำแนกประเภทแบบ Binary

ในบริบทของการตรวจจับโรคจากใบข้าวโพด ชั้น Fully Connected Layers ทำหน้าที่สำคัญในการรวมข้อมูลจากคุณลักษณะต่างๆ ที่สกัดได้จากชั้น Convolutional Layers และชั้น Pooling Layers เพื่อตัดสินใจว่าใบนั้นเป็นโรคหรือไม่ และหากเป็นโรค เป็นโรคชนิดใด การเชื่อมต่อแบบ Fully Connected ช่วยให้แบบจำลองสามารถเรียนรู้ความสัมพันธ์ที่ซับซ้อนระหว่างคุณลักษณะต่างๆ เช่น การปรากฏร่วมกันของลักษณะอาการหลายอย่างที่บ่งชี้ถึงโรคเฉพาะ หรือการพิจารณาความรุนแรงของอาการร่วมกับขนาดของพื้นที่ที่ได้รับผลกระทบ

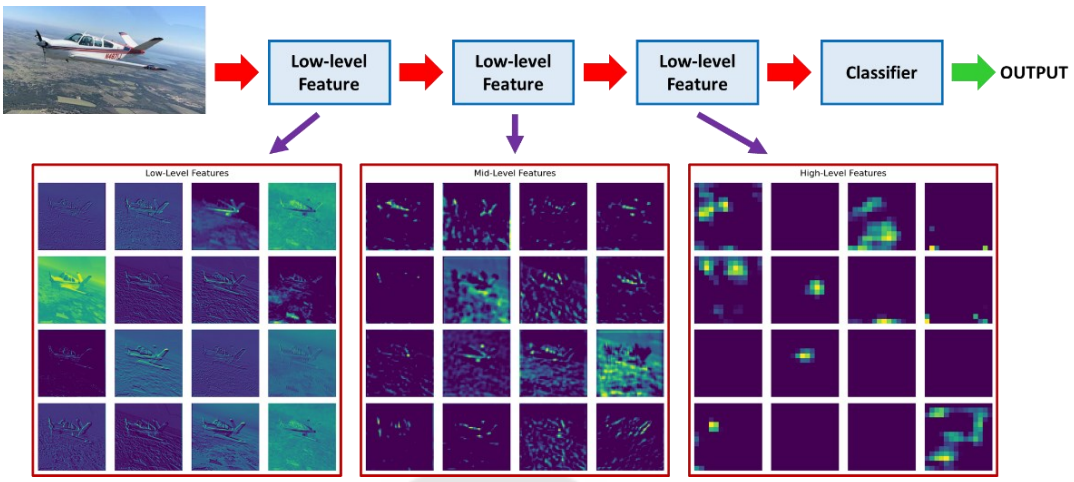
2.2.3 การทำงานของชั้น Convolution Layers

ชั้น Convolution Layers เป็นองค์ประกอบสำคัญของโครงข่ายประสาทเทียมแบบคอนโวลูชัน ซึ่งมีบทบาทสำคัญในการประมวลผลภาพและการรู้จำรูปแบบ การทำงานของชั้น Convolution Layers อาศัยหลักการสำคัญหลายประการที่ทำให้โครงข่ายประสาทเทียมแบบคอนโวลูชันมีประสิทธิภาพสูงในการวิเคราะห์ข้อมูลภาพ

2.2.3.1 หลักการของการคอนโวลูชัน (Convolution Principle)

การดำเนินการแบบคอนโวลูชัน เป็นหัวใจสำคัญของโครงข่ายประสาทเทียมแบบคอนโวลูชันซึ่งเป็นดำเนินการทางคณิตศาสตร์ที่มีรากฐานมาจากทฤษฎีสัญญาณและระบบ ในบริบทของโครงข่ายประสาทเทียมแบบคอนโวลูชัน การดำเนินการคอนโวลูชันถูกนำมาประยุกต์ใช้เพื่อการประมวลผลข้อมูลที่มีโครงสร้าง โดยเฉพาะอย่างยิ่งในการวิเคราะห์ภาพดิจิตอล

หลักการพื้นฐานของการคอนโวลูชัน คือ การใช้ตัวกรอง หรือเคอร์เนลขนาดเล็กเลื่อนไปบนข้อมูลภาพเพื่อสกัดคุณลักษณะสำคัญ เพื่อสร้าง Feature Map ที่แสดงถึงการปรากฏของรูปแบบหรือโครงสร้างเฉพาะในข้อมูลภาพรับเข้าในกระบวนการคอนโวลูชัน ตัวกรองจะเลื่อนไปทีละตำแหน่งบนข้อมูลภาพ โดยในแต่ละตำแหน่ง จะทำการคูณแบบจุดต่อจุด (Element-Wise Multiplication) แบบ Dot Product ระหว่างค่าในตัวกรองกับค่าในพื้นที่ที่ทับซ้อนกันของข้อมูลภาพ จากนั้นนำผลคูณทั้งหมดมารวมกันเพื่อสร้างค่าเอาต์พุตสำหรับตำแหน่งนั้นๆ ผลลัพธ์ที่ได้จากการเลื่อนตัวกรองไปทั่วทั้งภาพจะเรียกว่า Feature Map ดังภาพประกอบ 5



ภาพประกอบ 5 แสดงการดำเนินการคอนโวลูชัน ระหว่างนำเข้ากับตัวกรองเพื่อสร้าง Feature Map ในแต่ละชั้นของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

ความพิเศษของการคอนโวลูชัน คือ ความสามารถในการรักษาความสัมพันธ์เชิงพื้นที่ของข้อมูลภาพ กล่าวคือ ตัวกรองสามารถเรียนรู้ที่จะตรวจจับรูปแบบหรือคุณลักษณะโดยไม่ขึ้นกับตำแหน่งที่ปรากฏในภาพ คุณสมบัตินี้เรียกว่า Translation Invariance ซึ่งเป็นประโยชน์อย่างมากในการวิเคราะห์ภาพ เนื่องจากคุณลักษณะสำคัญ เช่น ขอบของวัตถุหรือลวดลายเฉพาะ อาจปรากฏในตำแหน่งที่แตกต่างกันในแต่ละภาพ

นอกจากนี้ การใช้ตัวกรองขนาดเล็กในการคอนโวลูชัน ยังช่วยลดจำนวนพารามิเตอร์ที่ต้องเรียนรู้เมื่อเทียบกับการเชื่อมต่อแบบ Fully Connected ในโครงข่ายประสาทเทียมแบบดั้งเดิม การลดจำนวนพารามิเตอร์นี้มีประโยชน์หลายประการ ได้แก่ การลดความซับซ้อนของแบบจำลอง ซึ่งช่วยลดความเสี่ยงของการเกิด Overfitting การลดทรัพยากรที่ใช้คำนวณ และการเพิ่มประสิทธิภาพในการเรียนรู้ โดยเฉพาะอย่างยิ่งเมื่อต้องทำงานกับข้อมูลภาพขนาดใหญ่

ในบริบทของการตรวจจับโรคจากใบข้าวโพด การคอนโวลูชัน มีบทบาทสำคัญในการสกัดคุณลักษณะที่เกี่ยวข้องกับอาการของโรค ตัวอย่างเช่น ตัวกรองอาจเรียนรู้ที่จะตรวจจับรูปแบบของจุดสีบนใบ ลักษณะการเปลี่ยนสีของเนื้อเยื่อใบ หรือรูปร่างของบริเวณที่ถูกทำลายโดยโรค คุณลักษณะเหล่านี้จะถูกสกัดออกมาโดยชั้นคอนโวลูชั่นในระดับต่างๆ โดยชั้นแรกๆ อาจตรวจจับคุณลักษณะพื้นฐานเช่นเส้นขอบหรือการเปลี่ยนแปลงของสี ในขณะที่ชั้นที่ลึกขึ้นจะสามารถตรวจจับรูปแบบที่ซับซ้อนมากขึ้น เช่น รูปร่างเฉพาะของรอยโรคหรือลักษณะการกระจายตัวของอาการบนใบ

นิยามทางคณิตศาสตร์ของการคอนโวลูชัน

ในบริบทของการประมวลผลภาพ การคอนโวลูชันจะแตกต่างกับการคอน จะถูกปรับให้อยู่ในรูปแบบของการคำนวณแบบไม่ต่อเนื่อง (Discrete) การทำงานในชั้นคอนโวลูชั่นในโครงข่ายประสาทเทียมแบบคอนโวลูชัน คือ การคูณระหว่างตัวกรองกับข้อมูลรับเข้าแบบเลื่อนตำแหน่งไปตามภาพ ตามสมการ (2)

$$Z_{ij}^{k} = \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} X_{i+m,j+n} \cdot W_{m,n}^{k} + b^{k} \qquad (2)$$

โดยที่

Z คือ ผลลัพธ์ของการคอนโวลูชัน

i, j คือ ตำแหน่งของจุดภาพในภาพ

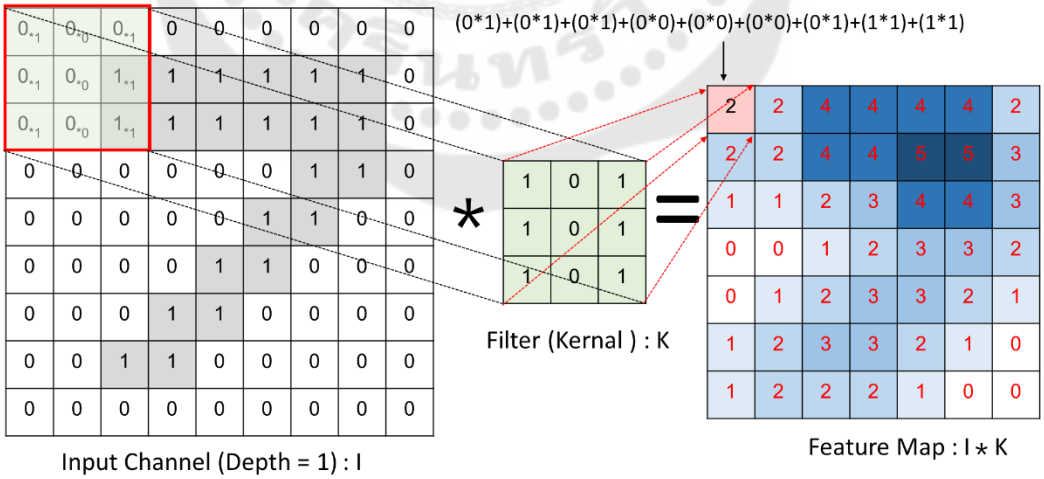
k คือ ดัชนีของตัวกรองที่ใช้

X คือ Input Feature Map

X คือ เมทริกซ์ของตัวกรอง หรือค่าน้ำหนัก (Weight) ของการคอนโวลูชัน

b คือ ค่า Bias

m, n คือ ขนาดของตัวกรอง



ภาพประกอบ 6 แสดงการดำเนินการคอนโวลูชัน ระหว่างข้อมูลนำเข้ากับตัวกรอง

กลไกการทำงานของการคอนโวลูชันในการประมวลผลภาพ ทำงานโดยการเลื่อนตัวกรองขนาดเล็กไปบนภาพนำเข้า ดังภาพประกอบ 6 โดยมีขั้นตอน ดังนี้

1) กำหนดขนาดของตัวกรอง (เช่น 3x3, 5x5)
2) เริ่มต้นที่มุมบนซ้ายของภาพ
3) ทำการคูณแบบจุดต่อจุด (Element-Wise Multiplication) ระหว่างค่าในตัวกรองกับค่าในพื้นที่ที่ทับซ้อนของภาพ
4) รวมผลคูณทั้งหมดเพื่อได้ค่าเอาต์พุตสำหรับตำแหน่งนั้น
5) เลื่อนตัวกรองไปทางขวาหรือลงตามระยะที่กำหนด (Stride)
6) ทำซ้ำขั้นตอนที่ 3 - 5 จนกว่าจะครอบคลุมทั้งภาพ

ผลลัพธ์ที่ได้จากกระบวนการนี้เรียกว่า “Feature Map” ซึ่งแสดงถึงการกระจายตัวของคุณลักษณะที่ตัวกรองนั้นๆ ตรวจจับได้ในภาพ

ระยะก้าว (Stride)

Stride ในบริบทของโครงข่ายประสาทเทียมแบบคอนโวลูชัน หมายถึง ระยะการเลื่อนของตัวกรอง หรือเคอร์เนล ในแต่ละครั้งที่ดำเนินการคอนโวลูชัน บนข้อมูลรับเข้า Stride เป็นหนึ่งในไฮเปอร์พารามิเตอร์ (Hyperparameter) ที่สำคัญในการออกแบบสถาปัตยกรรมของโครงข่ายประสาทเทียมแบบคอนโวลูชัน ที่ส่งผลต่อกระบวนการคำนวณในชั้น Convolution Layers และมีบทบาทสำคัญในการควบคุมขนาดของ Feature Map ที่ได้จากการทำคอนโวลูชัน ซึ่งเป็นการคำนวณที่เกิดจากการนำตัวกรองมาทำการเลื่อนไปบนข้อมูลรับเข้าตามจุดต่างๆ ในรูปภาพเพื่อดึงลักษณะเด่น (Feature) ออกมา

เนื่องจากมีผลโดยตรงต่อขนาดของ Feature Map ที่ได้จากการคอนโวลูชัน และมีผลต่อการเรียนรู้คุณลักษณะของข้อมูล

โดยทั่วไป Stride จะถูกกำหนดเป็นค่าจำนวนเต็มบวก ซึ่งจะให้ผลลัพธ์ที่แตกต่างกัน

1) Stride = 1 ตัวกรองจะเลื่อนทีละ 1 จุดภาพ ทำให้ได้ Feature Map ที่มีขนาดใกล้เคียงกับนำเข้าเดิม (ขึ้นอยู่กับการใช้ padding)
2) Stride = 2 ตัวกรองจะเลื่อนทีละ 2 จุดภาพ ทำให้ได้ Feature Map ที่มีขนาดเล็กลงประมาณครึ่งหนึ่งของนำเข้าเดิม
3) Stride > 2 จะทำให้ได้ Feature Map ที่มีขนาดเล็กลงมากขึ้นตามลำดับ

การคำนวณขนาดของผลลัพธ์ Feature Map ที่ได้จากการกำหนดค่า Stride ในการดำเนินการคอนโวลูชัน สามารถทำได้โดยใช้สูตรทางคณิตศาสตร์ ดังสมการ (3)

$$0 = \frac{(I - K + 2P)}{S} + 1 \qquad (3)$$

โดยที่

O คือ ขนาดของ Feature Map ที่ได้ (Output size)

I คือ ขนาดของนำเข้า (Input size)

K คือ ขนาดของตัวกรอง (Kernel size)

P คือ ขนาดของ Padding

S คือ ค่า Stride

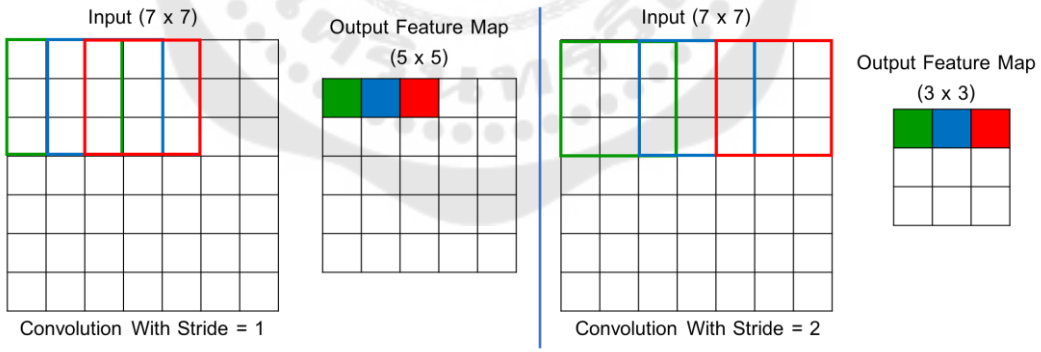
ตัวอย่างการคำนวณ

สมมติว่า มีนำเข้าขนาด 32x32 pixel ใช้ตัวกรองขนาด 3x3 ไม่มี Padding (P=0) และ Stride = 2 แทนค่าตัวแปรลงในสมการ (3) จะได้

$$O = \frac{(32 - 3 + 2(0))}{2} + 1 = 15.5$$

ในกรณีนี้ เราจะได้ขนาดของผลลัพธ์ Feature Map ขนาด 15 x 15 pixel (ปัดเศษทศนิยมลงเป็นจำนวนเต็ม)

จากสมการ (3) จะเห็นได้ว่าการเพิ่มค่า stride จะลดขนาดของ Feature Map เนื่องจากจำนวนครั้งที่ฟิลเตอร์ถูกเลื่อนในภาพจะน้อยลง ส่งผลให้จำนวนจุดที่เกิดการคำนวณลดลง ดังนั้นการใช้ค่า stride ขนาดใหญ่จะทำให้ภาพถูกบีบอัดในระดับที่สูงขึ้น ซึ่งเป็นการลดขนาดของ Feature Map แต่ก็อาจส่งผลต่อการสูญเสียข้อมูลเชิงลึกในลักษณะที่สำคัญของภาพไปด้วย



ภาพประกอบ 7 แสดงขนาดของผลลัพธ์ที่แตกต่างกัน เมื่อมีค่า stride ที่ต่างกัน

จากภาพประกอบ 7 นำเข้ามีขนาด 7x7 และตัวกรองมีขนาด 3x3 เมื่อดำเนินการคอนโวลูชัน ด้วย Stride = 1 จะได้ผลลัพธ์ขนาด 5x5 แต่ถ้าหากดำเนินการคอนโวลูชัน ด้วย Stride = 2 จะได้ผลลัพธ์ขนาด 3x3

ผลกระทบของ Stride ต่อการทำงานของ CNN

1) Stride และการลดขนาดข้อมูล

เมื่อ stride มีค่ามากขึ้น Feature Map จะมีขนาดเล็กลง ส่งผลให้จำนวนพารามิเตอร์ที่ต้องคำนวณในขั้นตอนถัดไปลดลง ทำให้แบบจำลองทำงานได้เร็วขึ้นและใช้ทรัพยากรน้อยลง อย่างไรก็ตาม การลดขนาดของ Feature Map มากเกินไปอาจทำให้สูญเสียรายละเอียดสำคัญของข้อมูลที่ส่งผลต่อประสิทธิภาพในการจำแนกข้อมูลของแบบจำลอง

2) Stride กับการควบคุมการบีบอัดข้อมูล

ค่า stride ช่วยควบคุมระดับการบีบอัดข้อมูล ตัวอย่างเช่น ถ้า Stride = 2 จะเทียบเท่ากับการลดขนาดของภาพลงครึ่งหนึ่ง ซึ่งสามารถใช้แทนการทำ Max Pooling ได้ในบางกรณี

3) Stride และการสูญเสียข้อมูล

การใช้ stride ที่มากเกินไปอาจทำให้ภาพสูญเสียข้อมูลที่มีความสำคัญไป เพราะในขณะที่ stride ทำให้ภาพมีขนาดเล็กลง จะทำให้การคำนวณบางจุดถูกข้ามไป เช่น การใช้ Stride = 2 ทำให้ภาพถูกขยับไปทีละ 2 จุดภาพ ซึ่งทำให้แบบจำลองไม่ได้ประมวลผลบางจุดภาพในภาพเลย

การเสริมขอบ (Padding)

Padding เป็นกระบวนการเพิ่มขอบเขตของข้อมูลรับเข้าด้วยค่าที่กำหนด (มักเป็นค่า 0) ก่อนที่จะทำการคอนโวลูชั่น ซึ่งหากไม่มีการเสริมขอบของข้อมูลรับเข้าก่อนที่จะทำการคอนโวลูชั่น จะทำให้ข้อมูลจุดภาพบางส่วนที่อยู่บริเวณขอบของนำเข้าสูญหายไป ไม่ได้ผ่านการทำคอนโวลูชั่น การใช้ Padding ช่วยให้สามารถควบคุมขนาดของ Feature Map ให้เท่ากับหรือใกล้เคียงกับขนาดของนำเข้า และการใช้ Padding ช่วยให้สามารถสร้างโครงข่ายที่มีความลึกมากขึ้นได้ โดยไม่ทำให้ขนาดของ Feature Map ลดลงอย่างรวดเร็วจนเกินไป ซึ่ง Padding เป็นเทคนิคที่สำคัญในการออกแบบและปรับแต่ง CNN เนื่องจากมีผลโดยตรงต่อขนาดของ Feature Map ที่ได้จากการคอนโวลูชัน และยังช่วยรักษาข้อมูลที่อยู่บริเวณขอบของนำเข้า

ประเภทของ Padding

1) Valid Padding (No Padding) ไม่มีการเพิ่ม Padding ทำให้ขนาดของ Feature Map ลดลงหลังจากการคอนโวลูชัน
2) Same Padding เพิ่ม Padding เพื่อให้ขนาดของ Feature Map เท่ากับขนาดของนำเข้า
3) Full Padding เพิ่ม Padding มากพอที่จะทำให้ตัวกรองสัมผัสกับทุกจุดภาพของนำเข้าอย่างน้อยหนึ่งครั้ง ทำให้ขนาดของ Feature Map ใหญ่กว่านำเข้า

4) Reflection Padding ใช้ค่าจากจุดภาพภายในภาพมาเป็น Padding แทนการใช้ค่าคงที่

5) Replication Padding ทำซ้ำค่าจุดภาพที่อยู่ขอบนอกสุดเพื่อเป็น Padding

ในการคำนวณหาจำนวนของการเสริมขอบ (Padding : P) สามารถใช้สมการ (3) ในการคำนวณได้ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1 การคำนวณ Same Padding

สมมติว่าเรามี Input ขนาด 64x64 และใช้ตัวกรองขนาด 5x5 โดยต้องการให้ขนาดของ Feature Map เท่ากับขนาดของ Input (Same Padding) และใช้ Stride เท่ากับ 1

แทนค่าใน สมการ (3) ได้

$$O = [(I - K - 2P) / S] + 1$$

$$64 = [(64 - 5 + 2P) / 1] + 1$$

$$63 = (59 + 2P) / 1$$

$$63 = 59 + 2P$$

$$4 = 2P$$

$$P = 2$$

ดังนั้น เราต้องใช้ Padding ขนาด 2 จุดภาพรอบนำเข้าเพื่อให้ได้ Same Padding

ตัวอย่างที่ 2 การคำนวณ Padding เมื่อใช้ Stride มากกว่า 1

สมมติว่าเรามี Input ขนาด 64x64 และใช้ตัวกรองขนาด 3x3 โดยต้องการให้ขนาดของ Feature Map เป็นครึ่งหนึ่งของ Input (O = I / 2 = 64 / 2 = 32) และใช้ Stride เท่ากับ 2

แทนค่าใน สมการ (3) ได้

$$O = [(I - K - 2P) / S] + 1$$

$$32 = [(64 - 3 + 2P) / 2] + 1$$

$$31 = (61 + 2P) / 1$$

$$62 = 61 + 2P$$

$$1 = 2P$$

$$P = 0.5$$

ทั้งนี้ ในทางปฏิบัติ เราไม่สามารถใช้ Padding เป็นเศษส่วนได้ ดังนั้นเราอาจต้องปรับขนาดของ Feature Map เล็กน้อย ดังนี้

- ถ้าใช้ P = 0 จะได้ O = [(64 - 3 + 0) / 2] + 1 = 31.5 ~ 31

- ถ้าใช้ P = 1 จะได้ O = [(64 - 3 + 2) / 2] + 1 = 32.5 ~ 32

ในกรณีนี้ การใช้ P = 1 จะให้ผลลัพธ์ที่ใกล้เคียงกับที่ต้องการมากที่สุด

ผลกระทบของ Padding ต่อโครงข่ายประสาทเทียมแบบคอนโวลูชัน

1) รักษาขนาดของภาพ (Preserving Spatial Dimensions)

เมื่อใช้ Same Padding ขนาดของ Feature Map จะเท่ากับขนาดของนำเข้า ซึ่งช่วยให้สามารถรักษาข้อมูลเชิงพื้นที่ที่สำคัญในภาพต้นฉบับได้ และช่วยให้ CNN สามารถเรียนรู้ลักษณะที่อยู่บริเวณขอบภาพได้อย่างมีประสิทธิภาพ

2) ป้องกันการสูญเสียข้อมูลบริเวณขอบ

การใช้ Valid Padding (ไม่มี Padding) อาจทำให้ขอบของภาพไม่ถูกนำมาคำนวณอย่างครบถ้วน ซึ่งอาจนำไปสู่การสูญเสียข้อมูลที่สำคัญบริเวณขอบ ในทางกลับกัน การใช้ Same Padding จะช่วยรักษาข้อมูลในบริเวณขอบ ทำให้ฟิลเตอร์สามารถประมวลผลข้อมูลในทุกส่วนของภาพได้

3) การควบคุมการลดขนาด

การไม่ใช้ Padding จะส่งผลให้ Feature Map ลดขนาดลงทุกครั้งที่ฟิลเตอร์เลื่อนผ่าน แต่การเพิ่ม Padding จะช่วยรักษาหรือควบคุมขนาดของ Feature Map ไม่ให้ลดลงอย่างรวดเร็วเกินไป นี่เป็นปัจจัยสำคัญในการออกแบบสถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชันโดยเฉพาะในงานที่ต้องการให้แบบจำลองรักษาขนาดของข้อมูลเพื่อใช้ในการประมวลผลขั้นต่อไป เช่น ในกรณีการตรวจจับวัตถุหรือการจำแนกรูปภาพในขอบเขตที่กว้าง

4) ผลต่อจำนวนพารามิเตอร์

การใช้ Padding มีผลต่อการคำนวณและจำนวนพารามิเตอร์ในโครงข่ายประสาทเทียมแบบคอนโวลูชันเนื่องจากการใช้ Same Padding อาจทำให้การคำนวณเกิดขึ้นมากขึ้นตามขนาดของ Feature Map ที่ไม่ได้ลดลง ทำให้จำนวนพารามิเตอร์มีมากขึ้น ส่งผลให้แบบจำลองใช้ทรัพยากรในการคำนวณมากขึ้น

2.2.3.2 ตัวกรอง และการสกัดคุณลักษณะ

ตัวกรองหรือเคอร์เนลเป็นเมทริกซ์ หรือเทนเซอร์ของค่าน้ำหนัก (Weight Matrices หรือ Weight Tensors) ที่ใช้ในการคอนโวลูชันกับข้อมูลรับเข้า ตัวกรองเหล่านี้ทำหน้าที่ในการสกัดคุณลักษณะจากข้อมูลรับเข้า ซึ่งเป็นกระบวนการสำคัญที่ทำให้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สามารถเรียนรู้และจดจำรูปแบบหรือคุณลักษณะเฉพาะในข้อมูลภาพ ตัวกรองแต่ละตัวกรองจะเรียนรู้ที่จะตรวจจับรูปแบบหรือคุณลักษณะเฉพาะ เช่น เส้นขอบ พื้นผิว หรือรูปทรงพื้นฐานต่างๆ เป็นต้น ในชั้นคอนโวลูชั่นแรกๆ ตัวกรองมักจะเรียนรู้ที่จะตรวจจับคุณลักษณะ

พื้นฐานอย่างเส้นตรง เส้นโค้ง หรือจุด ในขณะที่ชั้นที่ลึกลงไปจะสามารถตรวจจับคุณลักษณะที่ซับซ้อนมากขึ้น เช่น รูปร่างของวัตถุหรือส่วนประกอบของใบหน้า หรือสิ่งที่ต้องการตรวจจับ

ตัวกรองโดยทั่วไปจะมีขนาดเล็กกว่าข้อมูลรับเข้า เช่น 3x3 5x5 หรือ 7x7 จุดภาพ สำหรับข้อมูล 2 มิติ ในกรณีของข้อมูลภาพสี (เช่น RGB) ตัวกรองจะมี 3 มิติ โดยมิติที่ 3 คือ ความลึก (Depth) ของตัวกรอง ซึ่งมีขนาดเท่ากับจำนวนช่องสัญญาณ (Channels) ของข้อมูลรับเข้า ในทางคณิตศาสตร์ เราสามารถแทนตัวกรองด้วยเทนเซอร์ $W \in R^{h \times w \times d}$ โดยที่ h, w และ d คือ ความสูง ความกว้าง และความลึกของตัวกรองตามลำดับ

ในการใช้งานกับแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน อาจจะมีมากกว่า 1 ตัวกรอง ดังนั้น ตัวกรอง 3 มิติ ของข้อมูลภาพแบบ RGB ที่มีมากกว่า 1 ตัวกรอง จะอยู่ในรูปของเทนเซอร์ (Tensor) 4 มิติของค่าน้ำหนัก (4D Weight Tensor) ที่ใช้ในการคอนโวลูชันกับข้อมูลรับเข้าภาพสี RGB ตัวกรองนี้สามารถตรวจจับรูปแบบและคุณลักษณะที่เกิดจากความสัมพันธ์ระหว่างช่องสีทั้งสามในข้อมูลภาพ โดยสามารถนิยาม ตัวกรอง 3 มิติสำหรับภาพ RGB ได้ว่า $W \in R^{h \times w \times d \times k}$ โดยที่ k คือ จำนวนของตัวกรอง เช่น ตัวกรองขนาด 3x3 ของข้อมูลภาพแบบ RGB จำนวน 64 ตัวจะมีขนาดจริงเป็น 3x3x3x64 เป็นต้น

ตัวกรองจะถูกนำไปดำเนินการกับข้อมูลรับเข้าแบบคอนโวลูชัน ซึ่งสามารถนิยามทางคณิตศาสตร์ได้ ดังสมการ (4)

$$S_k(i,j) = \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} \sum_{c=0}^{c-1} I_{(i+m,j+n,c)} \cdot W_{k(m,n,d)} \qquad (4)$$

โดยที่

$S_k(i,j)$ คือ ค่าของ Feature Map ที่ตำแหน่ง (i,j) จากตัวกรองที่ k

$I_{(i+m,j+n,c)}$ คือ ค่าจุดภาพของนำเข้าในช่องสีที่ c

$W_{k(m,n,d)}$ คือ ค่าของตัวกรองที่ k

m, n คือ ขนาดของตัวกรอง

เมื่อ Feature Maps ที่เป็นผลลัพธ์จากการคอนโวลูชันของตัวกรองแต่ละตัว จะมีขนาดเป็น $HxWxk$ โดยที่

H คือ ความสูงของ Feature Map

W คือ ความกว้างของ Feature Map

k คือ จำนวนตัวกรองที่ใช้ในชั้นนั้น

ในกระบวนการฝึกฝนแบบจำลอง ค่าน้ำหนักของตัวกรองจะถูกปรับ โดยอัลกอริทึม เช่น การเรียนรู้แบบย้อนกลับ (Backpropagation) ซึ่งเป็นกระบวนการคำนวณหาความคลาดเคลื่อน (Error) ระหว่างผลลัพธ์ที่ได้จากการทำนายผล (Predicted Value) กับค่าจริง (Ground Truth)

ข้อมูลที่ได้จะเป็นการรวมกันของหลายๆ Feature Map ที่เป็นผลลัพธ์จากการดำเนินการคอนโวลูชัน ของข้อมูลรับเข้ากับตัวกรองแต่ละตัว ซึ่งแสดงถึงการกระจายตัวของคุณลักษณะที่ตัวกรองตัวนั้นตรวจจับได้ โดยที่ Feature Map จากชั้นหนึ่งจะถูกใช้เป็นนำเข้าสำหรับชั้นถัดไป ทำให้เกิดการเรียนรู้คุณลักษณะแบบลำดับชั้น (Hierarchical Feature Learning)

Feature Map จากการทำคอนโวลูชันแล้ว ข้อมูลนี้จะถูกส่งผ่านฟังก์ชันกระตุ้น เพื่อเพิ่มความไม่เชิงเส้น (Non-Linearity) ให้กับระบบ การเพิ่มความไม่เชิงเส้นนี้สำคัญมากในการทำให้โครงข่ายสามารถเรียนรู้และสกัดคุณลักษณะที่ซับซ้อนมากขึ้นได้

2.2.3.3 **การใช้ฟังก์ชันกระตุ้น** (Activation Functions)

ฟังก์ชันกระตุ้นถูกนำมาใช้หลังจากการคอนโวลูชันและการสกัดคุณลักษณะจากนำเข้าทำหน้าที่แปลงผลลัพธ์จากการคอนโวลูชัน ของแต่ละโหนดให้อยู่ในช่วงค่าที่เหมาะสมในการเรียนรู้ เช่น การแปลงผลลัพธ์จากค่าลบให้เป็นค่าบวก หรือการจำกัดขอบเขตของค่าผลลัพธ์ให้อยู่ในช่วงหนึ่ง เช่น ปรับค่า Output ให้อยู่ในช่วง 0 ถึง 1 หรือ ปรับค่า Output ให้มีค่าเป็นบวก (ค่ามากกว่า 0) เป็นต้น ซึ่งมีลักษณะที่แบบจำลองสามารถเรียนรู้ได้อย่างมีประสิทธิภาพมากขึ้น ฟังก์ชันกระตุ้นแต่ละชนิดมีคุณสมบัติและความเหมาะสมที่แตกต่างกันในการใช้งานในชั้นของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

ในการแปลงผลรวมถ่วงน้ำหนัก (Weighted Sum) จากการคอนโวลูชันของ Input ให้เป็น Output ของนิวรอน กระบวนการนี้เป็นหัวใจสำคัญที่ทำให้โครงข่ายประสาทเทียมสามารถเรียนรู้และแทนความสัมพันธ์ที่ซับซ้อนในข้อมูลได้ ดังแสดงในแบบจำลอง ในภาพประกอบ 2 ซึ่งประกอบด้วย 2 ขั้นตอน

1) การคำนวณผลรวมถ่วงน้ำหนักจากการคอนโวลูชัน

การคำนวณผลรวมถ่วงน้ำหนักสำหรับข้อมูลภาพนำเข้าแบบภาพสี (RGB) สามารถเขียนเป็นสมการทางคณิตศาสตร์ได้ตามสมการ (5)

$$Z_{(i,\,j,\,k)} = \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} \sum_{c=0}^{c-1} I_{(i+m,j+n,c)} \cdot W_{(m,n,d,k)} + b_k \qquad (5)$$

โดยที่

$Z_{(i,\,j,\,k)}$ คือ ผลรวมถ่วงน้ำหนักของการคอนโวลูชันสำหรับข้อมูลรับเข้าตำแหน่ง (i, j, k) จากตัวกรองที่ k

$I_{(i+m,j+n,c)}$ คือ ค่าจุดภาพของนำเข้าในช่องสีที่ c

$W_{(m,n,d,k)}$ คือ ค่าของตัวกรองที่ k

b_k คือ ค่าความเอนเอียง (bias)

m, n คือ ขนาดของตัวกรอง

2) การใช้ฟังก์ชันกระตุ้นกับผลลัพธ์จากการคำนวณผลรวมถ่วงน้ำหนัก

หลังจากได้ผลรวมถ่วงน้ำหนัก $Z_{(i,\,j,\,k)}$ แล้ว จะนำไปผ่านฟังก์ชันกระตุ้นในสมการ (6)

$$y_{(i,j,k)} = f\left(Z_{(i,j,k)}\right) \tag{6}$$

โดยที่

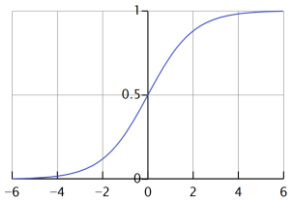
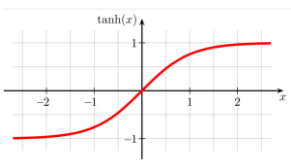
$Z_{(i,\,j,\,k)}$ คือ ผลรวมถ่วงน้ำหนักของการคอนโวลูชันสำหรับข้อมูลรับเข้าตำแหน่ง (i, j, k) จากตัวกรองที่ k

$f()$ คือ ฟังก์ชันกระตุ้น

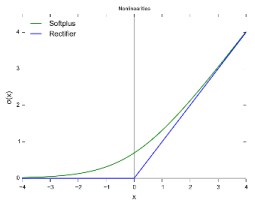
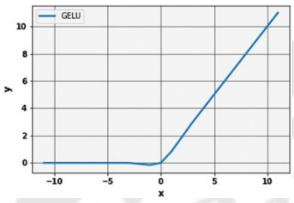
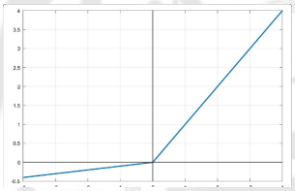
$Z_{(i,j,k)}$ คือ ค่าเอาต์พุตของนิวรอน

ฟังก์ชันกระตุ้นหลัก ๆ ที่นิยมใช้ในโครงข่ายประสาทเทียมแบบคอนโวลูชัน แสดงในตาราง 1

ตาราง 1 แสดงรูปแบบของฟังก์ชันกระตุ้น

Name	Plot	Function, f	Derivative of g, g'	Range
Logistic (Sigmoid)		f = 1 / (1 + e^(-x))	f' = f * (1 - f(x))	(0, 1)
Hyperbolic Tangent (Tanh)		f = (e^x - e^(-x)) / (e^x + e^(-x))	f' = 1 - f^2	(-1, 1)

ตาราง 1 (ต่อ)

Name	Plot	Function, f	Derivative of g, g'	Range
Rectified Linear Unit (ReLU)		f = max(0, x)	f' = 1 if x > 0, else 0	[0, infinity)
Gaussian Error Linear Unit (GELU)		f = x * P(X ≤ x), where P is the cumulative distribution function of a Gaussian distribution.	Non-trivial, involves Gaussian distribution.	[0, x]
Leaky Rectified Linear Unit (Leaky ReLU)		f = x if x > 0, else αx (α small constant, e.g., 0.01)	f' = 1 if x > 0, else α	(-∞,∞)

2.2.4 การทำงานของชั้นพูลลิ่ง

ชั้น Pooling Layers เป็นองค์ประกอบสำคัญในโครงสร้างของโครงข่ายประสาทเทียมแบบคอนโวลูชัน ซึ่งทำหน้าที่หลักในการลดขนาดของข้อมูลที่ผ่านการประมวลผลจากชั้นคอนโวลูชัน โดยมีวัตถุประสงค์เพื่อลดจำนวนพารามิเตอร์และการคำนวณในโครงข่าย ซึ่งช่วยป้องกันปัญหา Overfitting และเพิ่มประสิทธิภาพในการสกัดคุณลักษณะที่สำคัญของข้อมูล กระบวนการพูลลิ่งทำงานโดยการใช้ฟิลเตอร์หรือหน้าต่างขนาดกำหนด (เช่น 2x2 หรือ 3x3) เลื่อนไปบนข้อมูลรับเข้า โดยมีค่า Stride ที่กำหนดไว้ ซึ่งควบคุมระยะการเลื่อนของฟิลเตอร์ในแต่ละครั้ง

2.2.4.1 วิธีการลดขนาดข้อมูล

การลดขนาดข้อมูล Feature Map ในชั้น Pooling Layers เป็นกระบวนการสำคัญในโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ช่วยลดขนาดข้อมูลเชิงพื้นที่ (Spatial Dimensions) ของภาพ ซึ่งทำให้การคำนวณเร็วขึ้นและลดความซับซ้อนของโครงข่ายไปพร้อมกันโดยยังคงคุณสมบัติที่สำคัญของข้อมูลไว้มิติของข้อมูลและจำนวนพารามิเตอร์ในโครงข่าย โดยยังคงรักษาข้อมูลที่สำคัญไว้ วิธีการลดขนาดข้อมูลในชั้นพูลลิ่งนี้อาศัยหลักการของการสุ่มตัวอย่างแบบลดทอน

(Down Sampling) ซึ่งเป็นเทคนิคที่ช่วยลดความละเอียดเชิงพื้นที่ของข้อมูลโดยการเลือกค่าที่เป็นตัวแทนจากกลุ่มของข้อมูลในพื้นที่ที่กำหนด

การลดขนาดข้อมูลในชั้นพูลลิ่งเริ่มต้นด้วยการกำหนดขนาดของหน้าต่างพูลลิ่ง (Pooling Window) ซึ่งโดยทั่วไปมักมีขนาด 2x2 หรือ 3x3 และค่าระยะก้าว (Stride) ซึ่งกำหนดระยะการเลื่อนของหน้าต่างพูลลิ่งในแต่ละครั้ง หน้าต่างพูลลิ่งนี้จะเลื่อนไปบนข้อมูลรับเข้า (ซึ่งอาจเป็นข้อมูลจุดภาพ หรืออาจจะเป็น Feature Map) เช่น Stride = 2 จะเลื่อนหน้าต่างพูลลิ่งทีละ 2 จุดข้อมูลบน Feature Map ต่อครั้ง ทำให้ผลลัพธ์ที่ได้จากกระบวนการพูลลิ่งจะมีขนาดเล็กลงอย่างมากเมื่อเทียบกับข้อมูลรับเข้า โดยยังคงรักษาลักษณะสำคัญของภาพ เช่น ขอบ (Edges) และรูปร่าง (Shapes) ไว้ โดยในแต่ละตำแหน่ง จะมีการคำนวณค่าเอาต์พุตตามฟังก์ชันพูลลิ่งที่กำหนด เช่น Max Pooling หรือ Average Pooling

การลดขนาดข้อมูลผ่านชั้นพูลลิ่งสามารถอธิบายได้ด้วยสมการ (7)

$$\text{Output\_size} = \lfloor(\text{Input\_size} - \text{Filter\_size}) / S\rfloor + 1 \qquad (7)$$

โดยที่

Input_size คือ ขนาดของข้อมูลรับเข้า

Filter_size คือ ขนาดของหน้าต่างพูลลิ่ง

S คือ ระยะการเลื่อนของหน้าต่างพูลลิ่งในแต่ละครั้ง

$\lfloor\ \rfloor$ คือ ฟังก์ชันปัดเศษลง (Floor Function)

ตัวอย่าง เช่น หากมีข้อมูลรับเข้าขนาด 128x128 จุดข้อมูล ใช้หน้าต่างพูลลิ่งขนาด 3x3 และ Stride เท่ากับ 2 สามารถหาขนาดของผลลัพธ์จากสมการ (7) จะได้ว่า

$$\text{Output\_size} = \lfloor(128 - 3) / 2\rfloor + 1 = 63$$

ดังนั้น ผลลัพธ์จะมีขนาด 63x63 ซึ่งเป็นการลดขนาดข้อมูลลงประมาณ 75.7%

2.2.4.2 ประเภทของการพูลลิ่ง (Max Pooling และ Average Pooling)

ประเภทของการพูลลิ่งที่ใช้กันบ่อยในโครงข่ายประสาทเทียมแบบคอนโวลูชันมี 2 ประเภทหลัก ได้แก่ Max Pooling และ Average Pooling ซึ่งแต่ละประเภทมีหลักการทำงาน คุณสมบัติ และการประยุกต์ใช้ที่แตกต่างกัน

1) การเลือกค่าสูงสุด (Max Pooling)

Max Pooling เป็นการเลือกค่าสูงสุดจากพื้นที่ย่อยที่ที่ถูกครอบคลุมโดยหน้าต่างพูลลิ่ง ตัวอย่างเช่น ในกรณีของฟิลเตอร์ขนาด 2x2 ฟิลเตอร์จะเลือกค่าเฉพาะตำแหน่งที่มีค่าสูงสุดใน

พื้นที่ 2x2 นั้นครอบอยู่ ซึ่งวิธีนี้เหมาะสมอย่างยิ่งในการรักษาลักษณะเชิงลักษณ์ที่สำคัญ (Features) ของภาพ เช่น การรักษาค่าของจุดที่มีความโดดเด่นที่สุด ซึ่งมีประโยชน์ในการตรวจจับรูปร่าง ขอบ หรือลักษณะที่สำคัญในภาพ ทำให้สามารถลดขนาดของข้อมูลได้อย่างมีประสิทธิภาพ โดยยังคงคุณลักษณะที่สำคัญของภาพไว้ นอกจากนี้ Max Pooling ยังช่วยลดความไวต่อการเปลี่ยนแปลงตำแหน่งเล็กน้อยของคุณลักษณะในข้อมูลรับเข้า (Translation Invariance) และลดข้อมูลสัญญาณรบกวน (Noise) ที่อาจไม่จำเป็นลง ทำให้ข้อมูลที่ถูกส่งต่อไปยังชั้นถัดไปมีแต่ส่วนสำคัญที่โดดเด่น ซึ่ง Max Pooling สามารถอธิบายได้ด้วยสมการ (8)

$$\text{Output}(i,j) = \max\{\text{Input}(m,n) \mid m \in [i \times s, i \times s + f), n \in [j \times s, j \times s + f)\} \quad (8)$$

โดยที่

(i,j) คือ ตำแหน่งในเอาต์พุต

s คือ ค่า Stride (ระยะการเลื่อนของหน้าต่างพูลลิ่ง)

f คือ ขนาดของหน้าต่างพูลลิ่ง

$\text{Input}(m,n)$ คือ ค่าของข้อมูลรับเข้าที่ตำแหน่ง (m,n)

2) การเลือกค่าเฉลี่ย (Average Pooling)

Average Pooling จะเลือกค่าเฉลี่ยของค่าทั้งหมดในพื้นที่ย่อยพื้นที่ย่อยที่ที่ถูกครอบคลุมโดยหน้าต่างพูลลิ่ง วิธีการนี้จะช่วยรักษาข้อมูลเชิงพื้นที่ทั้งหมดโดยไม่เลือกเฉพาะค่าเดียวที่โดดเด่น ซึ่งทำให้ผลลัพธ์ของภาพที่ผ่านชั้นนี้มีความเรียบเนียนมากกว่า วิธีนี้ช่วยในการรักษาข้อมูลทั่วไปของพื้นที่นั้นๆ ทั้งข้อมูลพื้นหลังและความละเอียดของพื้นผิว และมีประสิทธิภาพในการช่วยลดผลกระทบของค่าผิดปกติ (Outliers) ในข้อมูล และสามารถเก็บรักษาความแตกต่างที่ละเอียดอ่อนได้มากขึ้น แม้ว่า Average Pooling อาจทำให้คุณลักษณะที่โดดเด่นบางอย่างถูกทำให้จางลงเมื่อเฉลี่ยกับค่ารอบข้าง

การหาค่า Average Pooling หาได้จากสมการ (9)

$$\text{Output}(i,j) = \frac{1}{f^2} \text{x} \sum\{Input(m,n) \mid m \in [i \times s, i \times s + f), n \in [j \times s, j \times s + f)\} \quad (9)$$

โดยที่

(i,j) คือ ตำแหน่งในเอาต์พุต

s คือ ค่า Stride (ระยะการเลื่อนของหน้าต่างพูลลิ่ง)

f คือ ขนาดของหน้าต่างพูลลิ่ง

$Input(m,n)$ คือ ค่าของข้อมูลรับเข้าที่ตำแหน่ง (m,n)

การเลือกใช้ Max Pooling หรือ Average Pooling ขึ้นอยู่กับลักษณะของงานและประเภทของข้อมูล โดยทั่วไป Max Pooling มักจะให้ผลลัพธ์ที่ดีกว่าในงานการรู้จำภาพ (Image Recognition) และการตรวจจับวัตถุ (Object Detection) เนื่องจากสามารถสกัดคุณลักษณะที่สำคัญได้ดี ในขณะที่ Average Pooling อาจเหมาะสมกับงานที่ต้องการรักษาข้อมูลพื้นหลังหรือต้องการความละเอียดของพื้นผิว เช่น การแบ่งส่วนภาพ (Image Segmentation) หรือการสร้างภาพ (Image Generation)

2.2.5 การฝึกฝนและการปรับปรุงค่าพารามิเตอร์ของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

การฝึกฝน (Training) และการปรับปรุงค่าพารามิเตอร์ (Parameters Adjustment) ของโครงข่ายประสาทเทียมแบบคอนโวลูชันเป็นกระบวนการสำคัญที่ทำให้แบบจำลองสามารถเรียนรู้จากข้อมูลและปรับปรุงประสิทธิภาพได้อย่างต่อเนื่อง โดยมีองค์ประกอบหลักสองส่วน คือ อัลกอริทึมการเรียนรู้แบบย้อนกลับ (Backpropagation) และการปรับค่าพารามิเตอร์และการหาค่าเหมาะสมที่สุด (Optimization)

2.2.5.1 กระบวนการเรียนรู้แบบย้อนกลับ (Backpropagation)

กระบวนการเรียนรู้แบบย้อนกลับ (Backpropagation) เป็นกลไกสำคัญในการฝึกฝนโครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยเฉพาะในการปรับปรุงพารามิเตอร์ของโครงข่าย เช่น น้ำหนัก และค่าความเอนเอียง (Biases) ซึ่งส่งผลต่อประสิทธิภาพในการทำนาย หรือจำแนกข้อมูลที่ซับซ้อนได้อย่างแม่นยำ อัลกอริทึมนี้ใช้แนวคิดทางคณิตศาสตร์ในการคำนวณค่าความผิดพลาดที่เกิดขึ้นจากการทำนายแล้วถ่ายทอดค่าความผิดพลาดนั้นย้อนกลับไปยังชั้นต่าง ๆ ในโครงข่ายเพื่อให้เกิดการปรับค่าพารามิเตอร์ในลักษณะที่ทำให้ค่าความผิดพลาดลดลงในรอบถัดไปของการเรียนรู้ กระบวนการนี้ช่วยให้โครงข่ายสามารถเรียนรู้จากข้อมูลได้อย่างมีประสิทธิภาพมากขึ้นเมื่อผ่านการฝึกฝนวนซ้ำหลายรอบ (Epochs)

กระบวนการของ Backpropagation ประกอบด้วย 2 ขั้นตอนหลัก

1) การแพร่กระจายไปข้างหน้า (Forward Propagation)

ในขั้นตอนนี้ ข้อมูลรับเข้าจะถูกส่งผ่านแต่ละชั้นของโครงข่ายจนไปถึงชั้นเอาต์พุต โดยในแต่ละชั้นจะมีการคำนวณค่าตามฟังก์ชันการทำงานของชั้นนั้นๆ จนได้ค่าผลลัพธ์ออกมา ผลลัพธ์ที่ได้ (Predicted value) จะถูกเปรียบเทียบกับค่าที่ถูกต้องหรือค่า Ground Truth โดยใช้ฟังก์ชันความสูญเสีย (Loss Function) เช่น Mean Squared Error (MSE) หรือ Categorical Cross-Entropy เพื่อคำนวณความผิดพลาด (Error) หรือค่าความสูญเสีย สำหรับโครงข่ายระบบประสาท

เทียมแบบคอนโวลูชัน สำหรับการจำแนกภาพ (Image Classification) ฟังก์ชันความสูญเสียที่นิยมใช้คือ Categorical Cross-Entropy ซึ่งในทางคณิตศาสตร์แสดงเป็นดังสมการ (10)

$$L = \sum_{i=1}^{C} y_i \log(\hat{y}_i) \quad (10)$$

โดยที่

L คือ ค่าฟังก์ชันความสูญเสีย

C คือ จำนวนคลาส

y_i คือ ค่าที่ถูกต้องหรือค่าจริง (Ground Truth)

$\hat{y}_i$ คือ ค่าผลลัพธ์ที่ได้ (Predicted Value)

2) การแพร่กระจายย้อนกลับ (Backward Propagation)

หลังจากได้ค่าเอาต์พุตและคำนวณค่าความสูญเสียแล้ว จะมีการส่งค่าความสูญเสียย้อนกลับผ่านแบบจำลองโครงข่าย เพื่อคำนวณคำนวณอนุพันธ์ หรือเกรเดียนต์ (Gradient) ของฟังก์ชันความสูญเสียเทียบกับค่าพารามิเตอร์ (ค่า Weights และค่า Biases) แต่ละตัวในโครงข่าย ซึ่งเป็นสามารถคำนวณได้โดยใช้กฎลูกโซ่ (Chain Rule) ในการหาอนุพันธ์หลายตัวแปร โดยอนุพันธ์ของฟังก์ชันความสูญเสียต่อพารามิเตอร์ สามารถหาได้ตามสมการ (11)

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}} \quad (11)$$

โดยที่

L คือ ค่าฟังก์ชันความสูญเสีย

z_j คือ นำเข้าของโหนด j

w_{ij} คือ น้ำหนักที่เชื่อมโยงระหว่างโหนด i และ j

อนุพันธ์นี้แสดงถึงการแพร่กระจายของค่าความผิดพลาดจากชั้นเอาต์พุตย้อนกลับไปยังชั้นถัดไป ซึ่งในแต่ละชั้น โครงข่ายจะคำนวณการปรับค่าพารามิเตอร์ตามทิศทางและขนาดของค่าความผิดพลาด เพื่อให้ค่าความผิดพลาดลดลงในการฝึกฝนในรอบถัดไป

2.2.5.2 การปรับค่าพารามิเตอร์และการหาค่าเหมาะสมที่สุด (Optimization)

หลังจากคำนวณค่าความผิดพลาดย้อนกลับได้แล้ว ขั้นตอนต่อไป คือ การปรับปรุงค่าพารามิเตอร์ ของโครงข่าย โดยมีเป้าหมายเพื่อลดค่าความสูญเสียและเพิ่มประสิทธิภาพในการทำนายผลของแบบจำลองให้แม่นยำที่สุด วิธีการหาค่าเหมาะสมที่สุดที่นิยมใช้กับการฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน มีหลายวิธี เช่น ได้แก่ Gradient Descent และ

รูปแบบที่พัฒนาต่อยอด เช่น Stochastic Gradient Descent (SGD) Adam และ RMSprop ซึ่งแต่ละวิธีมีหลักการและข้อดีข้อเสียที่แตกต่างกัน วิธีการหาค่าที่นิยมใช้งานมีดังต่อไปนี้

1) Stochastic Gradient Descent (SGD)

2) Stochastic Gradient Descent with momentum (SGD with momentum)

3) Adaptive Gradient Algorithm (AdaGrad)

4) RMSprop

5) Adaptive Moment Estimation

2.2.6 เทคนิคการปรับปรุงประสิทธิภาพของแบบจำลองแบบโครงข่ายประสาทเทียมแบบคอนโวลูชัน

การปรับปรุงประสิทธิภาพของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันในงานประมวลผลภาพและการจำแนกประเภทภาพ เช่น การตรวจจับโรคจากพืช เป็นเรื่องสำคัญที่ช่วยให้แบบจำลองสามารถเรียนรู้จากข้อมูลได้ดียิ่งขึ้น และลดปัญหาเช่นการ Overfitting ซึ่งทำให้แบบจำลองทำงานได้ไม่ดีเมื่อเจอกับข้อมูลใหม่ ในการปรับปรุงประสิทธิภาพของแบบจำลองมีเทคนิคหลายวิธีที่นิยมใช้ ได้แก่ การเพิ่มข้อมูล (Data Augmentation) การใช้ Dropout เพื่อลดการ Overfitting และการปรับปรุงอัตราการเรียนรู้ (Learning Rate Adjustment) ซึ่งแต่ละเทคนิคมีแนวคิดและการนำไปประยุกต์ที่แตกต่างกัน และสามารถอธิบายได้ดังนี้

2.2.6.1 การเพิ่มข้อมูล (Data Augmentation)

การเพิ่มข้อมูล (Data Augmentation) เป็นเทคนิคที่ใช้ในการขยายชุดข้อมูลฝึก (Training Set) ให้มีขนาดใหญ่ขึ้นโดยสร้างข้อมูลใหม่จากข้อมูลที่มีอยู่ ด้วยการทำการเปลี่ยนแปลงรูปแบบเล็กน้อย เช่น การหมุน (Rotation) การพลิก (Flipping) การปรับขนาด (Scaling) การเลื่อน (Translation) การตัดภาพ (Cropping) การปรับความสว่างและความคมชัด (Brightness and Contrast Adjustment) และการเพิ่มสัญญาณรบกวน (Noise Injection) แต่ละวิธีการมีผลต่อการเรียนรู้ของแบบจำลองที่แตกต่างกัน เช่น การหมุนและการพลิกช่วยให้แบบจำลองเรียนรู้ลักษณะที่ไม่ขึ้นกับทิศทาง (Orientation-Invariant Features) ในขณะที่การปรับความสว่างช่วยให้แบบจำลองทนทานต่อสภาพแสงที่แตกต่างกัน เทคนิคนี้ช่วยเพิ่มความหลากหลายให้กับชุดข้อมูลฝึกฝน ที่ใช้ในการฝึกฝนแบบจำลอง ซึ่งช่วยป้องกันปัญหา Overfitting ได้อย่างมีประสิทธิภาพ

การเพิ่มข้อมูลสามารถทำได้ทั้งแบบออฟไลน์ (Offline) และแบบออนไลน์ (Online) หรือเรียกอีกอย่างหนึ่งว่า “On-the-fly Data Augmentation” การเพิ่มข้อมูลแบบออฟไลน์จะสร้างข้อมูลใหม่และบันทึกไว้ก่อนการฝึกฝนแบบจำลอง ในขณะที่การเพิ่มข้อมูลแบบออนไลน์จะสร้าง

ข้อมูลใหม่ในระหว่างกระบวนการฝึกฝนแบบจำลอง โดยไม่ได้มีการบันทึกภาพที่เพิ่มขึ้นไปยังพื้นที่จัดเก็บข้อมูล ซึ่งช่วยประหยัดพื้นที่จัดเก็บข้อมูลและเพิ่มความหลากหลายของข้อมูลในแต่ละรอบการฝึกฝน (Epoch) การเลือกใช้วิธีใดขึ้นอยู่กับปัจจัยต่างๆ เช่น ขนาดของชุดข้อมูล ทรัพยากรการประมวลผลที่มี และความต้องการในการทำซ้ำการทดลอง

หลักการทางคณิตศาสตร์ที่สนับสนุนการเพิ่มข้อมูลคือการเพิ่มโอกาสให้แบบจำลองได้สัมผัสกับข้อมูลที่หลากหลายมากขึ้น ทำให้แบบจำลองสามารถเรียนรู้คุณสมบัติที่สำคัญของภาพได้ดียิ่งขึ้น และไม่จดจำลักษณะเฉพาะบางอย่างจากข้อมูลเดิมที่อาจเป็นสัญญาณรบกวน (Noise) การเพิ่มข้อมูลสามารถมองได้ว่าเป็นการขยายขอบเขตของฟังก์ชันการสูญเสียให้ครอบคลุมถึงข้อมูลที่แบบจำลองยังไม่เคยเห็นมาก่อน ทำให้การทำงานของแบบจำลองมีประสิทธิภาพในการทำนายกับข้อมูลใหม่ที่หลากหลายยิ่งขึ้น

2.2.6.2 การใช้ Dropout เพื่อลดการ Overfitting

Dropout คือ เทคนิคการปิด (Drop) นิวรอนในชั้นของโครงข่ายประสาทเทียมบางส่วนในระหว่างการฝึกฝนแบบจำลอง ซึ่งหมายความว่าในแต่ละรอบการฝึกฝน แบบจำลองจะทำการสุ่มเลือกนิวรอนบางตัวเพื่อไม่ใช้ในกระบวนการคำนวณการถ่ายทอดข้อมูลและการปรับค่าพารามิเตอร์การทำเช่นนี้ช่วยให้แบบจำลองไม่ต้องพึ่งพานิวรอนเพียงตัวใดตัวหนึ่งในโครงข่ายหรือใช้นิวรอนในการประมวลผลน้อยตัวจนเกินไป ทำให้แบบจำลองมีความทนทานต่อข้อมูลที่หลากหลาย และสามารถเรียนรู้จากการรวมข้อมูลของนิวรอนหลายตัวแทนที่จะจดจำนิวรอนตัวใดตัวหนึ่งเป็นพิเศษ

ในการใช้งาน Dropout อัลกอริทึมจะสุ่มปิดการทำงานของนิวรอนในแต่ละชั้นของโครงข่ายด้วยความน่าจะเป็น p ซึ่งเรียกว่า "Dropout Rate" เช่น ถ้าค่าของ p = 0.5 หมายความว่า 50% ของนิวรอนในชั้นนั้นจะถูกปิดระหว่างการฝึกฝน ทำให้ลดความซับซ้อนของแบบจำลองในการพยายามสร้างรูปแบบที่ซับซ้อนเกินไปจากชุดข้อมูลการฝึกฝน

กระบวนการ Dropout สามารถอธิบายผ่านทางขั้นตอนต่อไปนี้

1) การสุ่มปิดหน่วยประสาท

ในแต่ละรอบการฝึก (Forward Pass) หน่วยประสาทบางตัวจะถูกปิดตามความน่าจะเป็น p หน่วยประสาทที่ถูกสุ่มปิดจะไม่มีส่วนในการคำนวณผลลัพธ์ในรอบการฝึกนั้น โดยแบบจำลองจะสุ่มเลือกว่า Neurons ไหนที่ควรปิด โดยกำหนดมาสก์แบบสุ่ม (Random Binary Mask) ที่จะถูกนำไปคูณกับผลลัพธ์ของ neurons ในแต่ละชั้น

ให้ z_i เป็นค่าผลลัพธ์จากหน่วยประสาทลำดับที่ i ก่อนใช้ Dropout และ $\tilde{z}_i$ เป็นผลลัพธ์หลังใช้ Dropout ดังสมการ (12)

$$\tilde{z}_i = r_i z_i \quad (12)$$

โดยที่

r_i คือ ตัวแปรสุ่มที่มีค่า 1 ด้วยความน่าจะเป็น 1 – p และมีค่า 0 ด้วยความน่าจะเป็น p

2) การทำงานในช่วงทดสอบ

เมื่อสิ้นสุดการฝึก (Training Phase) Dropout จะไม่ถูกใช้ในช่วงทดสอบ (Inference Phase) แต่จะทำการปรับค่าของน้ำหนักให้อยู่ในระดับเฉลี่ย (Expected Value) เพื่อชดเชยการที่ Dropout ถูกปิดในการทดสอบ นั่นคือ ค่าของ neurons ที่ถูกปิดในช่วงการฝึกจะถูกถัวเฉลี่ยในช่วงทดสอบ โดยค่าผลลัพธ์ z_i จะถูกคูณด้วย 1 – p เพื่อให้ผลลัพธ์ในช่วงทดสอบมีขนาดที่ใกล้เคียงกับช่วงการฝึก ในการทดสอบ แบบจำลองจะปรับค่าผลลัพธ์จากหน่วยประสาทในแต่ละชั้นให้เทียบเท่ากับการที่ไม่มี Dropout

2.2.6.3 การปรับปรุงอัตราการเรียนรู้ (Learning Rate Adjustment)

อัตราการเรียนรู้ (Learning Rate: η) เป็นตัวแปรสำคัญที่กำหนดความเร็วในการปรับค่าพารามิเตอร์ของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ในระหว่างการฝึกฝน ถ้าอัตราการเรียนรู้สูงเกินไป แบบจำลองจะไม่สามารถหาจุดต่ำสุดของฟังก์ชันการสูญเสียได้อย่างแม่นยำ และทำให้ค่าพารามิเตอร์ถูกปรับเปลี่ยนอย่างรุนแรงจนเกิดการแกว่งไปมา ในทางกลับกัน หากอัตราการเรียนรู้ต่ำเกินไป การฝึกแบบจำลองจะช้ามากและอาจติดอยู่ใน Local Minima ของฟังก์ชันการสูญเสีย ทำให้แบบจำลองไม่สามารถหาค่าพารามิเตอร์ที่ดีที่สุดได้

อัตราการเรียนรู้ถูกใช้ในขั้นตอนของการปรับปรุงค่าพารามิเตอร์ในแบบจำลองระหว่างการฝึก โดยใช้หลักการของวิธีการถดถอยแบบ Gradient Descent ซึ่งเป็นอัลกอริทึมที่ทำงานโดยการคำนวณความชันของฟังก์ชันการสูญเสีย เพื่อนำไปปรับปรุงค่าถ่วงน้ำหนักในทิศทางที่ลดค่าฟังก์ชันการสูญเสียให้น้อยที่สุด ตามสมการ (13)

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \quad (13)$$

โดยที่

θ_{t+1} คือ พารามิเตอร์ของแบบจำลองที่เวลา $t+1$

θ_t คือ พารามิเตอร์ของแบบจำลองที่เวลา t

η คือ Learning Rate

$\nabla L(\theta_t)$ คือ Gradient ของ Loss Function

เทคนิคการปรับปรุงอัตราการเรียนรู้ที่นิยมใช้คือการเริ่มต้นด้ วยค่าอัตราการเรียนรู้ที่สูง และลดค่าอัตราการเรียนรู้เมื่อการฝึกดำเนินไปเรื่อย ๆ ซึ่งสามารถทำได้หลายวิธี เช่น การลดอัตราการเรียนรู้อย่างช้าๆ ทุกครั้งที่ค่าการสูญเสียไม่ลดลง หรือการใช้ Learning Rate Schedules ที่กำหนดการลดอัตราการเรียนรู้ตามจำนวน Epoch ที่ฝึก เช่น Step Decay หรือ Exponential Decay

เทคนิคการปรับอัตราการเรียนรู้

ในทางปฏิบัติ มีการใช้เทคนิคการปรับอัตราการเรียนรู้เพื่อช่วยให้การฝึกแบบจำลองทำได้อย่างมีประสิทธิภาพและมีเสถียรภาพมากขึ้น โดยเทคนิคที่นิยมใช้ ได้แก่

1) Fixed Learning Rate

อัตราการเรียนรู้ถูกกำหนดค่าไว้คงที่ตลอดการฝึกฝน ในกรณีนี้ ต้องเลือกค่าอัตราการเรียนรู้ที่เหมาะสมเพื่อให้กระบวนการฝึกสามารถหาค่าถ่วงน้ำหนักที่ดีที่สุดได้อย่างมีประสิทธิภาพ การกำหนดค่า η ในระดับที่พอเหมาะสำคัญมาก เพราะการตั้งค่าให้สูงหรือต่ำเกินไปจะส่งผลต่อประสิทธิภาพของแบบจำลอง

2) Learning Rate Decay

อัตราการเรียนรู้จะถูกลดลงเมื่อเวลาผ่านไป หรือหลังจากการฝึกไปถึงจุดที่ค่า loss ลดลงช้า การใช้ Learning Rate Decay ช่วยให้แบบจำลองสามารถเรียนรู้ได้เร็วในช่วงแรกของการฝึกและเรียนรู้แบบละเอียดมากขึ้นเมื่อเข้าใกล้ค่าถ่วงน้ำหนักที่เหมาะสมที่สุด

3) Adaptive Learning Rate

ค่าอัตราการเรียนรู้ถูกปรับให้เข้ากับสถานการณ์ของการฝึกในแต่ละช่วงเวลาของการเรียนรู้ แบบจำลองจะใช้ค่า learning rate ที่แตกต่างกันตามการเปลี่ยนแปลงของ gradient ซึ่งอัลกอริทึมที่ใช้ในการปรับค่า Learning rate แบบอัตโนมัติมีหลายแบบ เช่น AdaGrad RMSprop และ Adam

2.3 สถาปัตยกรรมแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการใช้งานเพื่อจำแนกภาพ

สถาปัตยกรรมแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันได้รับการพัฒนาอย่างต่อเนื่องและเป็นที่ยอมรับว่าเป็นเครื่องมือที่มีประสิทธิภาพสูงในการใช้งานเพื่อจำแนกภาพ

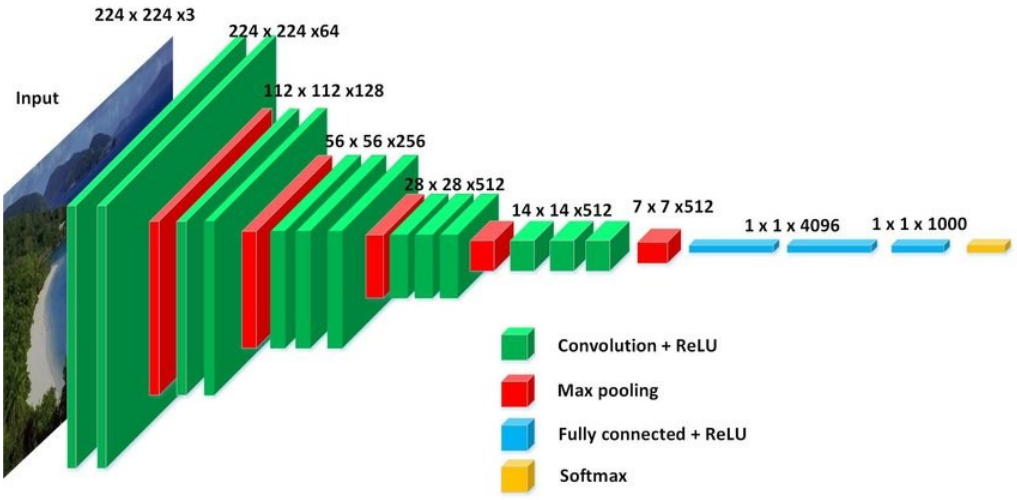
โดยเฉพาะอย่างยิ่งเมื่อเกี่ยวข้องกับการประมวลผลข้อมูลภาพ ซึ่งแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สามารถดึงคุณลักษณะซับซ้อนจากภาพได้อย่างมีประสิทธิภาพผ่านกระบวนการคอนโวลูชัน โดยแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันจะมีหลายชั้นที่ช่วยในสกัด Feature ต่าง ๆ ที่จำเป็นต่อการจำแนกประเภทภาพ

ปัจจุบันสถาปัตยกรรมที่ทันสมัยของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันมักเน้นการเพิ่มประสิทธิภาพในหลายด้าน เช่น การลดขนาดของแบบจำลองเพื่อให้สามารถใช้งานได้ในอุปกรณ์ที่มีข้อจำกัดทรัพยากร และการปรับปรุงประสิทธิภาพในการจำแนกภาพที่มีความซับซ้อนสูง นอกจากนี้ยังมีการใช้เทคนิคต่าง ๆ เช่น Transfer Learning Data Augmentation และการใช้ Pretrained Models เพื่อเพิ่มความสามารถในการประมวลผลข้อมูลที่มีจำนวนจำกัด สถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชัน ที่น่าสนใจที่ใช้ในการศึกษาวิจัยครั้งนี้มีดังต่อไปนี้

2.3.1 VGG16

VGG16 (Simonyan & Zisserman, 2014) เป็นสถาปัตยกรรมโครงข่ายประสาทเทียมเชิงลึกที่พัฒนาโดยกลุ่มวิจัย Visual Geometry Group (VGG) ที่มหาวิทยาลัยออกซฟอร์ดในปี 2014 แบบจำลองนี้ได้รับการออกแบบมาเพื่อใช้ในการจำแนกรูปภาพและการจดจำวัตถุ โดยเฉพาะในการแข่งขัน ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ซึ่งเป็นการแข่งขันที่มีชื่อเสียงในด้านการจำแนกรูปภาพ VGG16 สามารถบรรลุความแม่นยำสูงในการจำแนกรูปภาพ โดยมีความแม่นยำสูงถึง 92.7 % ในการทดสอบกับชุดข้อมูล ImageNet VGG16 สามารถนำมาใช้เป็นแบบจำลองพื้นฐานในการเรียนรู้เชิงลึก โดยการนำแบบจำลองที่ผ่านการฝึกอบรมจากชุดข้อมูลใหญ่ เช่น ImageNet มาใช้ในการฝึกอบรมกับข้อมูลใหม่ ซึ่งช่วยลดเวลาในการฝึกอบรมและเพิ่มประสิทธิภาพของแบบจำลอง

แบบจำลองโครงข่ายประสาทเทียม VGG16 ได้รับความนิยมเพราะความเรียบง่ายของสถาปัตยกรรมที่มุ่งเน้นการใช้ฟิลเตอร์ขนาดเล็ก (3x3) และการเพิ่มความลึกของเครือข่ายเพื่อจับคุณลักษณะที่ซับซ้อนของภาพ โดยมีความลึกทั้งหมด 16 ชั้น ประกอบด้วย ชั้น Convolutional Layers จำนวน 13 ชั้น และชั้น Fully Connected Layers จำนวน 3 ชั้น ดังภาพประกอบ 8



ภาพประกอบ 8 แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบ VGG16

ที่มา: Liu, F., Wang, Y., Wang, F. C., Zhang, Y. Z., & Lin, J. (2019). Intelligent and secure content-based image retrieval for mobile users. IEEE Access, 7, 119209–119222. CC – BY.

VGG16 ประกอบด้วยชั้นต่าง ๆ ดังนี้

1) ชั้น Convolutional layers มีทั้งหมด 13 ชั้น แต่ละชั้นใช้ตัวกรองขนาด 3x3 และ stride เท่ากับ 1 และใช้ฟังก์ชันกระตุ้น ReLU

2) ชั้น Max Pooling มี 5 ชั้น ใช้หน้าต่างขนาด 2x2 และ stride เท่ากับ 2

3) ชั้น Fully Connected Layers มี 3 ชั้น โดย 2 ชั้นแรกมี 4096 โหนด และชั้นสุดท้ายมี 1000 โหนด (ตามจำนวนคลาสใน ImageNet)

4) ชั้น Softmax ชั้นสุดท้ายสำหรับการจำแนกประเภท

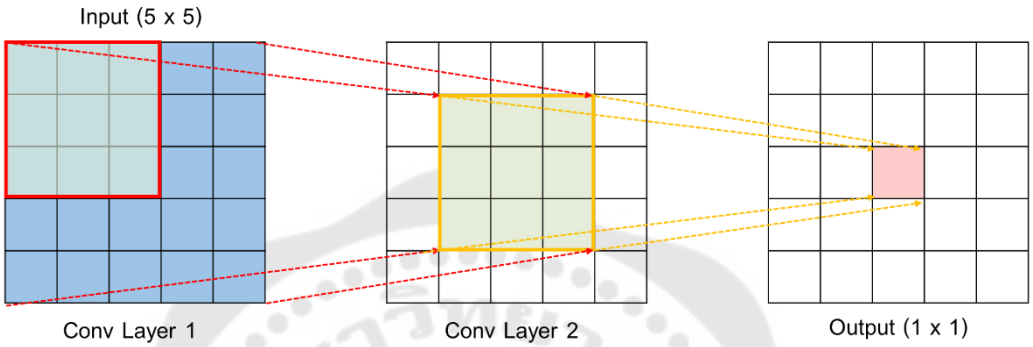
แบบจำลองโครงข่ายประสาทเทียม VGG16 จะเห็นได้ว่า มีการออกแบบให้ชั้นคอนโวลูชันมีการดำเนินการโดยใช้ตัวกรองขนาดเล็กที่มีขนาด 3x3 ติดต่อกันมากกว่า 1 ชั้น ซึ่งผลลัพธ์ที่ได้จากการดำเนินการดังกล่าวจะทำให้ได้ขนาดของ Receptive Field ที่มีขนาดใหญ่ แต่ใช้จำนวนจำนวนพารามิเตอร์ที่น้อยกว่า เปรียบเทียบกับจำนวน Receptive Field ที่เท่ากัน เมื่อใช้ตัวกรองเพียงชั้นเดียว โดยสามารถคำนวณขนาดของ Receptive Field จากจำนวนของตัวกรองและจำนวนชั้น จากสมการ (14)

$$\text{Receptive Field Size} = \mathrm{L} \cdot (\mathrm{k} - 1) + 1 \qquad (14)$$

โดยที่

L คือ จำนวนชั้น

k คือ ขนาดของตัวกรอง



ภาพประกอบ 9 แสดงจำนวน Receptive Field ของชั้นคอนโวลูชันที่ใช้ตัวกรองขนาด 3x3 จำนวน 2 ชั้น เพื่อให้ได้ Receptive Field ขนาด 5x5

ตัวอย่าง หากต้องการ Receptive Field ขนาด 5x5 จะสามารถใช้ตัวกรองขนาด 3x3 จำนวน 2 ชั้นได้ ดังภาพประกอบ 9 และสามารถคำนวณได้จากสมการ (14)

$$\text{Receptive Field Size} = 2 \cdot (3 - 1) + 1 = 5$$

เมื่อพิจารณาเปรียบเทียบจำนวนพารามิเตอร์ของตัวกรองที่จะต้องใช้ในการดำเนินการคอนโวลูชันเมื่อต้องการ Receptive Field ขนาด 5x5 หากใช้ตัวกรองขนาด 5x5 จะใช้พารามิเตอร์จำนวน 25 (5 x 5) พารามิเตอร์ แต่หากใช้ตัวกรองขนาด 3x3 จำนวน 2 ชั้น ก็จะได้ผลลัพธ์ของ Receptive Field ขนาด 5x5 เท่ากัน แต่จะใช้จำนวนของพารามิเตอร์เพียง 18 (3 x 3 x 2) พารามิเตอร์เท่านั้น

หรือ เมื่อต้องการ Receptive Field ขนาด 7x7 หากใช้ตัวกรองขนาด 7x7 จะใช้พารามิเตอร์จำนวน 49 (7 x 7) พารามิเตอร์ แต่หากใช้ตัวกรองขนาด 3x3 จำนวน 3 ชั้น จะสามารถได้ผลลัพธ์ของ Receptive Field ขนาด 5x5 เท่ากัน แต่จะใช้จำนวนของพารามิเตอร์ที่ลดลงเหลือเพียง 27 (3 x 3 x 3) พารามิเตอร์เท่านั้น ลดลงถึง 45%

VGG16 เป็นโครงข่ายประสาทเทียมขนาดใหญ่ที่มีพารามิเตอร์จำนวนมาก การเข้าใจโครงสร้างสถาปัตยกรรม การดำเนินการ (Operation) ในแต่ละชั้น และจำนวนพารามิเตอร์ในแต่

ละชั้น จะช่วยให้เห็นภาพรวมของความซับซ้อนของแบบจำลองและสามารถปรับแต่งแบบจำลองได้อย่างมีประสิทธิภาพ โครงสร้างแต่ละชั้นของ VGG16 แสดงดังตาราง 2

ตาราง 2 แสดงชั้นประเภทของการดำเนินการคอนโวลูชัน และจำนวนพารามิเตอร์ในแต่ละชั้นของ VGG16

Layer	Operations	Output Shape	Parameters	Layer Multiples
Input	Input	(224, 224, 3)	0	-
Conv1_1	Conv2D	(224, 224, 64)	1,792	(3 * 3 * 3 + 1) * 64
Conv1_2	Conv2D	(224, 224, 64)	36,928	(3 * 3 * 64 + 1) * 64
Pool1	MaxPooling2D	(112, 112, 64)	0	-
Conv2_1	Conv2D	(112, 112, 128)	73,856	(3 * 3 * 64 + 1) * 128
Conv2_2	Conv2D	(112, 112, 128)	147,584	(3 * 3 * 128 + 1) * 128
Pool2	MaxPooling2D	(56, 56, 128)	0	-
Conv3_1	Conv2D	(56, 56, 256)	295,168	(3 * 3 * 128 + 1) * 256
Conv3_2	Conv2D	(56, 56, 256)	590,080	(3 * 3 * 256 + 1) * 256
Conv3_3	Conv2D	(56, 56, 256)	590,080	(3 * 3 * 256 + 1) * 256
Pool3	MaxPooling2D	(28, 28, 256)	0	-
Conv4_1	Conv2D	(28, 28, 512)	1,180,160	(3 * 3 * 256 + 1) * 512
Conv4_2	Conv2D	(28, 28, 512)	2,359,808	(3 * 3 * 512 + 1) * 512
Conv4_3	Conv2D	(28, 28, 512)	2,359,808	(3 * 3 * 512 + 1) * 512
Pool4	MaxPooling2D	(14, 14, 512)	0	-
Conv5_1	Conv2D	(14, 14, 512)	2,359,808	(3 * 3 * 512 + 1) * 512
Conv5_2	Conv2D	(14, 14, 512)	2,359,808	(3 * 3 * 512 + 1) * 512
Conv5_3	Conv2D	(14, 14, 512)	2,359,808	(3 * 3 * 512 + 1) * 512
Pool5	MaxPooling2D	(7, 7, 512)	0	-
FC6	Dense	(4096)	102,764,544	7 * 7 * 512 * 4096 + 4096
FC7	Dense	(4096)	16,781,312	4096 * 4096 + 4096
FC8	Dense	(1000)	4,097,000	4096 * 1000 + 1000
Softmax	Activation	(1000)	0	-

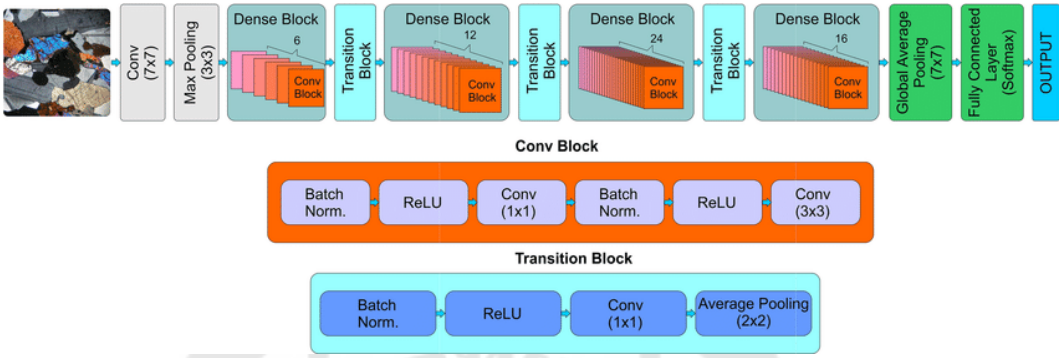
จากตาราง 2 แสดงให้เห็นว่า แบบจำลองโครงข่ายประสาทเทียม VGG16 มีข้อจำกัดในด้านของจำนวนพารามิเตอร์จำนวนมาก ซึ่งมีจำนวนพารามิเตอร์ทั้งสิ้น 138,357,544 พารามิเตอร์ เมื่อพารามิเตอร์มีขนาด 32 บิต (4 ไบต์) จะทำให้ขนาดของแบบจำลองมีขนาดใหญ่ถึงประมาณ 528 เมกะไบต์ (MB) (เมื่อจัดเก็บข้อมูลพารามิเตอร์ ในรูปแบบ HDF5) โดยพารามิเตอร์ในชั้นคอนโวลูชัน มีจำนวน 14,714,688 พารามิเตอร์ และมีจำนวนพารามิเตอร์ในชั้น Fully Connected Layer จำนวน 123,642,856 พารามิเตอร์ จะเห็นได้ว่าจำนวนพารามิเตอร์ส่วนใหญ่ (89.36 %) อยู่ในชั้น Fully Connected Layer ซึ่งถ้าหากว่าสามารถปรับปรุงโครงสร้างของชั้น Fully Connected Layer ของแบบจำลอง VGG16 ให้มีความซับซ้อนน้อยลงก็จะสามารถทำให้มีขนาดของแบบจำลองมีขนาดเล็กลงได้

แม้ว่าแบบจำลองโครงข่ายประสาทเทียม VGG16 จะมีข้อดีหลายประการ แต่ก็มีข้อจำกัดที่ต้องพิจารณาเช่นกัน ขนาดของแบบจำลอง VGG16 มีจำนวนพารามิเตอร์ที่สูง ซึ่งทำให้แบบจำลองมีขนาดใหญ่มีจำนวนพารามิเตอร์ที่มาก ทำให้ต้องการทรัพยากร และหน่วยความจำมากในการประมวลผลสูง ทำให้ฝึกฝนแบบจำลอง VGG16 ต้องใช้เวลานาน และเนื่องจากจำนวนพารามิเตอร์ที่สูง อาจทำให้แบบจำลองเกิด Overfitting ได้ง่าย หากไม่มีการใช้เทคนิคในการควบคุม

2.3.2 DenseNet121

DenseNet121 (Huang et al., 2016) หรือ Densely Connected Convolutional Networks เป็นหนึ่งในสถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชัน ที่มีสถาปัตยกรรมพิเศษที่เรียกว่า Dense Convolutional Network (DenseNet) ซึ่งถูกนำเสนอโดย Gao Huang และคณะในปี 2017 แนวคิดหลักของ DenseNet คือ การเชื่อมโยงระหว่างชั้นทั้งหมดภายในบล็อกเดียวกัน เรียกว่า “Dense Block” ให้ข้อมูลถูกส่งต่อกันอย่างต่อเนื่องจากแต่ละชั้นไปยังทุกชั้นที่อยู่ถัดไป ซึ่งแตกต่างจากโครงข่าย CNN แบบดั้งเดิมที่มักจะเชื่อมต่อเฉพาะระหว่างชั้นที่อยู่ติดกันเท่านั้น โดยทุกชั้นจะรับข้อมูลจากทุกชั้นก่อนหน้าเป็นเป็นข้อมูลรับเข้าและส่งต่อเป็นข้อมูลส่งออกไปยังบล็อกถัดไป การออกแบบโครงสร้างในลักษณะนี้จะช่วยให้สามารถพัฒนาให้โครงสร้างของแบบจำลองมีความลึก หรือมีจำนวนชั้นจำนวนมาก เพื่อให้สามารถสกัดลักษณะสำคัญของรูปภาพได้ โดยที่เมื่อทำ Backpropagation ในการหา Gradient Descent แล้ว จะไม่เกิดการสูญหายของ Gradient (Gradient Vanishing)

สถาปัตยกรรมของ DenseNet 121 มีการจัดเรียงชั้นการทำงานที่แตกต่างจากแบบจำลอง CNN ทั่วไป ดังแสดงในภาพประกอบ 10 โดยเริ่มต้นด้วยชั้น convolution ขนาด 7x7 ที่มี stride เท่ากับ 2 ตามด้วยชั้น Max pooling ขนาด 3x3 ที่มี stride เท่ากับ 2 เช่นกัน



ภาพประกอบ 10 แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบ VGG16

ที่มา: Polat, Ali & Polat, Ozlem & Ekici, Taner. (2021). Automatic classification of volcanic rocks from thin section images using transfer learning networks. Neural Computing and Applications. Copyright © 2021, The Author(s), under exclusive license to Springer-Verlag London Ltd., part of Springer Nature

หลังจากนั้นจะเป็นส่วนของ Dense Block ซึ่งเป็นหัวใจสำคัญของ DenseNet โดย DenseNet121 มีทั้งหมด 4 Dense Block แต่ละบล็อกประกอบด้วยชุดของชั้นคอนโวลูชันที่เชื่อมต่อกันอย่างหนาแน่น การเชื่อมต่อระหว่าง Dense Block จะเชื่อมต่อด้วย Transition Layer

Transition Layer ทำหน้าที่ลดขนาดของ Feature Map ด้วยการคอนโวลูชันด้วยตัวกรองขนาด 1x1 และมีการ Pooling แบบ Average ด้วยตัวกรอง 2x2

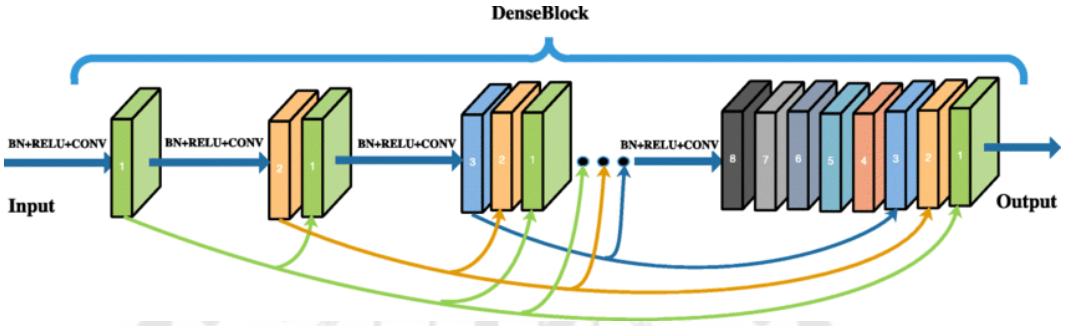
Global Average Pooling ทำหน้าที่รวมข้อมูลจากแต่ละช่องสัญญาณให้เหลือเพียงค่าเฉลี่ย

Fully Connected Layer ชั้นสุดท้ายที่ใช้ในการจำแนกข้อมูลออกเป็นหมวดหมู่ต่างๆ

โครงสร้าง และส่วนประกอบของสถาปัตยกรรมที่สำคัญของแบบจำลองโครงข่ายประสาทเทียม DenseNet121 มีดังนี้

1) Dense Block

โครงสร้างภายใน Dense Block ประกอบด้วย Convolution Block ซึ่งเป็นอนุกรมของการดำเนินการ ดังนี้ Batch Normalization ReLU 1x1 Convolutional ตามด้วย Batch Normalization ReLU 3x3 Convolutional ซึ่งข้อมูลรับเข้าจะถูกเชื่อมต่อ (Concatenate) กับผลลัพธ์ของการดำเนินการ เพื่อเป็นข้อมูลรับเข้าให้กับการดำเนินการของ Convolution Block ถัดไป ดังภาพประกอบ 11 ซึ่งการที่ข้อมูลรับเข้าของ แต่ละ Convolution Block ถูกเชื่อมต่อกับผลลัพธ์ของการดำเนินการทั้งหมดภายใน Dense Block จะเรียกว่า “Dense Connectivity”



ภาพประกอบ 11 แสดงโครงสร้าง และการดำเนินการภายใน Dense Block

ที่มา: Zhong, Yue & Liu, Lizhuang & Zhao, Dan & Li, Hongyang. (2020). A generative adversarial network for image denoising. Multimedia Tools and Applications. Copyright © 2019, Springer Science Business Media, LLC, part of Springer Nature

Growth Rate ใน Dense Block เป็นแนวคิดที่สำคัญในสถาปัตยกรรม DenseNet ซึ่งใช้ในการกำหนดจำนวน Feature Map ที่เพิ่มขึ้นในแต่ละชั้นของ Dense Block จำนวน Feature Map ที่เพิ่มขึ้นในแต่ละชั้นจะถูกกำหนดตามค่า Growth Rate

Feature Map ใน Dense Block ที่เกิดจากการดำเนินการในชั้น Convolution Block หนึ่งจะถูกส่งต่อไปยังชั้นถัดไป โดย Feature Map ใหม่จะถูกสร้างขึ้นในแต่ละชั้นตามค่า Growth Rate ที่กำหนด ยกตัวอย่างเช่น หากค่า Growth Rate เป็น 32 หมายความว่า ในแต่ละชั้น Convolution Block จะมี Feature Map ใหม่ 32 Feature Map ถูกเพิ่มเข้าไปในข้อมูล Feature Map ที่มีอยู่ดังนั้นจำนวน Feature Map จะเพิ่มขึ้นเรื่อย ๆ เมื่อผ่านไปแต่ละชั้น Convolution Block

ถ้าขนาดของ Feature Map เริ่มต้นของ Dense Block คือ k_0 และ Dense Block ประกอบด้วย l ชั้น ค่า Growth Rate คือ k จะทำให้ขนาดของ Feature Map หลังจากผ่านชั้นที่ l เป็นดังสมการ (15)

$$\text{Number of Feature Map} = k_0 + (k \cdot l) \quad (15)$$

ซึ่งหมายความว่าเมื่อเพิ่มจำนวนชั้น ขนาดของ Feature Map จะเพิ่มขึ้นอย่างเป็นสัดส่วนตรงตามค่า Growth Rate

ตัวอย่างการคำนวณ Growth Rate

สมมุติว่า Dense Block มี Feature Map ที่เข้ามาเป็น 64 Feature Map และมี Growth Rate เท่ากับ 32 และมีทั้งหมด 4 ชั้น จำนวน Feature Map ที่จะเกิดขึ้นในแต่ละชั้นจะคำนวณได้ตามสมการ (15)

จำนวน Feature Map = 64 + (32 x 4) = 192 Feature Map

สามารถอธิบายได้ ดังนี้

ชั้นที่ 1: Feature Map = 64 (Feature Map) + 32 (Growth Rate) = 96

ชั้นที่ 2: Feature Map = 96 (จากชั้นที่ 1) + 32 = 128

ชั้นที่ 3: Feature Map = 128 + 32 = 160

ชั้นที่ 4: Feature Map = 160 + 32 = 192

2) Transition Layer

Transition Layer เป็นส่วนสำคัญในสถาปัตยกรรมของ DenseNet ซึ่งใช้ในการเชื่อมต่อระหว่าง Dense Block ต่างๆ และช่วยลดขนาดของ Feature Map ที่เพิ่มขึ้นในระหว่างการประมวลผล ปรับขนาดข้อมูลให้มีความเหมาะสมสำหรับการประมวลผลของ Dense Block ถัดไป

Transition Layer ประกอบด้วยชั้น Batch Normalization 1x1 Convolutional และ 2x2 Average Pooling ดังภาพประกอบ 10 ซึ่งจะช่วยลดขนาดของข้อมูล (Feature Map) และจำนวนช่องสัญญาณ ทำให้ข้อมูลมีขนาดเล็กลงแต่ยังคงคุณลักษณะของภาพที่สำคัญไว้ ลดความซับซ้อนของข้อมูลและลดจำนวนพารามิเตอร์ที่ต้องคำนวณของแบบจำลองลง

จำนวนพารามิเตอร์ทั้งหมดใน DenseNet121 น้อยกว่าสถาปัตยกรรมอื่น ๆ ที่มีความลึกเทียบเท่ากัน เนื่องจากการใช้ Dense Connectivity ที่ช่วยลดการซ้ำซ้อนของ Featureและการใช้ Transition Layer เพื่อลดขนาดของข้อมูลลง ซึ่ง DenseNet121 มีจำนวนพารามิเตอร์ทั้งสิ้น 7,978,856 พารามิเตอร์ รายละเอียดและโครงสร้างตามตาราง 3

ตาราง 3 โครงสร้างและจำนวนพารามิเตอร์ของ DenseNet121

Layer	Output Size	Layer Information	Parameters
Input	224 x 224	RGB Image	
Convolution	112 x 112	7x7 conv, 64 filters, stride 2	9,408
Max Pooling	56 x 56	3x3 max pool, stride 2	
Dense Block 1	56 x 56	[1x1 conv, 3x3 conv] x 6	217,088
Transition Layer 1	56 x 56	1x1 conv	33,024
	28 x 28	2x2 average pool, stride 2	
Dense Block 2	28 x 28	[1x1 conv, 3x3 conv] x 12	721,984
Transition Layer 2	28 x 28	1x1 conv	131,840
	14 x 14	2x2 average pool, stride 2	
Dense Block 3	14 x 14	[1x1 conv, 3x3 conv] x 24	2,889,728
Transition Layer 3	14 x 14	1x1 conv	526,848
	7 x 7	2x2 average pool, stride 2	
Dense Block 4	7 x 7	[1x1 conv, 3x3 conv] x 16	3,855,360
Classification Layer	1 x 1	7x7 global average pool	1,024,000
	1000	1000D fully connected, softmax	

2.3.3 MobileNet V2

MobileNetV2 (Sandler et al., 2018) เป็นสถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชั่นที่ได้รับการพัฒนาต่อยอดจาก MobileNetV1 ที่ใช้หลักการคอนโวลูชันที่แตกต่างจากการคอนโวลูชันแบบปกติ (Standard Convolution) โดยใช้วิธีการคอนโวลูชันแบบ Depthwise Separable Convolution ที่ลดการคำนวณและการจำนวนพารามิเตอร์ในโครงข่ายลง โดยเน้นที่ประสิทธิภาพในการประมวลผลบนอุปกรณ์ที่มีทรัพยากรจำกัด เช่น โทรศัพท์มือถือและอุปกรณ์ IoT แบบจำลองนี้ถูกออกแบบมาให้มีขนาดเล็ก และใช้ทรัพยากรในการประมวลผลน้อย ประหยัดพลังงาน แต่ยังคงความแม่นยำสูงในงานการจำแนกประเภทภาพ และงานอื่น ๆ ที่เกี่ยวข้อง โดยแบบจำลองนี้ถูกสร้างขึ้นมาเพื่อตอบโจทย์ความต้องการของงานที่ต้องการความแม่นยำสูงใน

สภาพแวดล้อมที่มีข้อจำกัดด้านการคำนวณ นอกจากนี้โครงสร้างของ MobileNetV2 มีการปรับปรุงมาจาก MobileNetV1 โดยใช้เทคนิคที่เรียกว่า Inverted Residuals และ Linear Bottlenecks เพื่อเพิ่มประสิทธิภาพในการสกัดคุณลักษณะของภาพ ซึ่งสามารถลดจำนวนพารามิเตอร์และปริมาณการคำนวณโดยที่ยังคงความแม่นยำของการจำแนกภาพ

การจัดเรียงชั้นในโครงสร้างของ MobileNetV2 ดังแสดงในภาพประกอบ 12 จะเป็นไปตามลำดับดังนี้

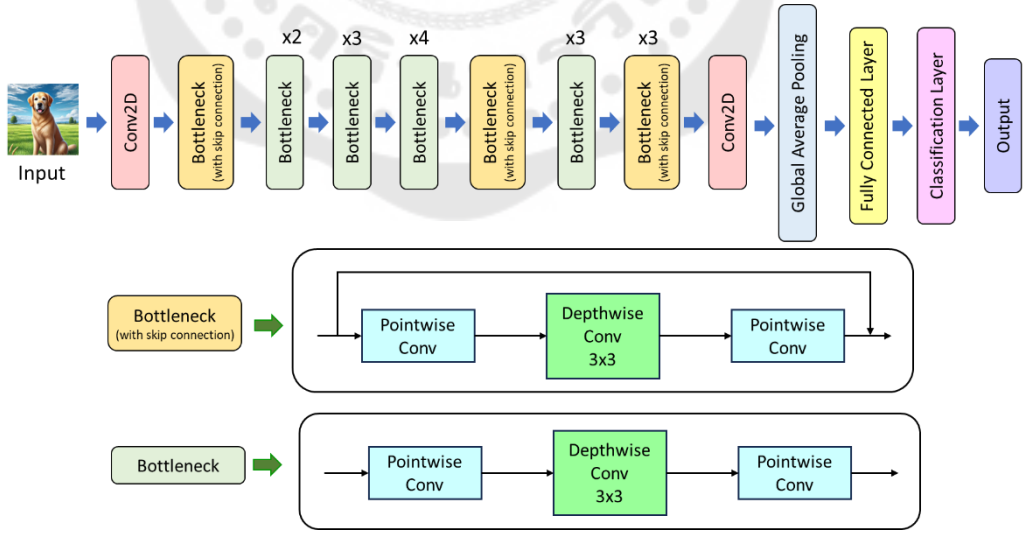
1) Initial Convolution Layer ชั้นแรกที่ใช้ Convolutional ธรรมดาเพื่อลดขนาดของข้อมูลลง

2) Bottleneck Residual Layers ประกอบด้วยชั้น Depthwise Separable Convolution, Pointwise Convolution และ Linear Bottlenecks ซึ่งจะลดจำนวนช่องสัญญาณลงก่อนที่จะเพิ่มขึ้นอีกครั้งในชั้นถัดไป

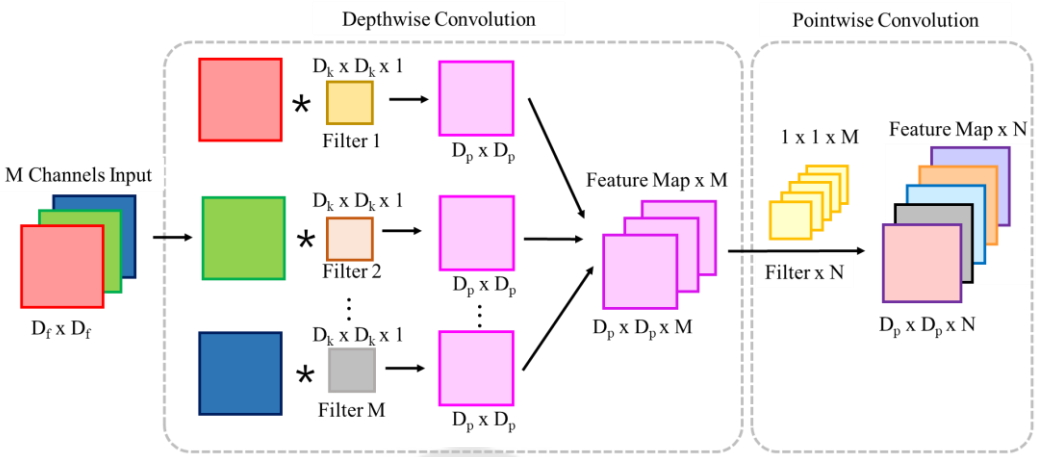
3) Final Convolution Layer ชั้นสุดท้ายที่ใช้ Convolutional ธรรมดาเพื่อเพิ่มจำนวนช่องสัญญาณให้เหมาะสมกับจำนวนหมวดหมู่ที่ต้องการจำแนก

4) Global Average Pooling ชั้นที่ใช้ในการสรุปข้อมูลจากแต่ละช่องสัญญาณให้เหลือเพียงค่าเฉลี่ย

5) Fully Connected Layer ชั้นสุดท้ายที่ใช้ในการจำแนกข้อมูลออกเป็นหมวดหมู่ต่างๆ โดยใช้ฟังก์ชัน SoftMax



ภาพประกอบ 12 แสดงโครงสร้างและส่วนประกอบภายในแบบจำลองโครงข่ายประสาทเทียมแบบ MobileNetV2



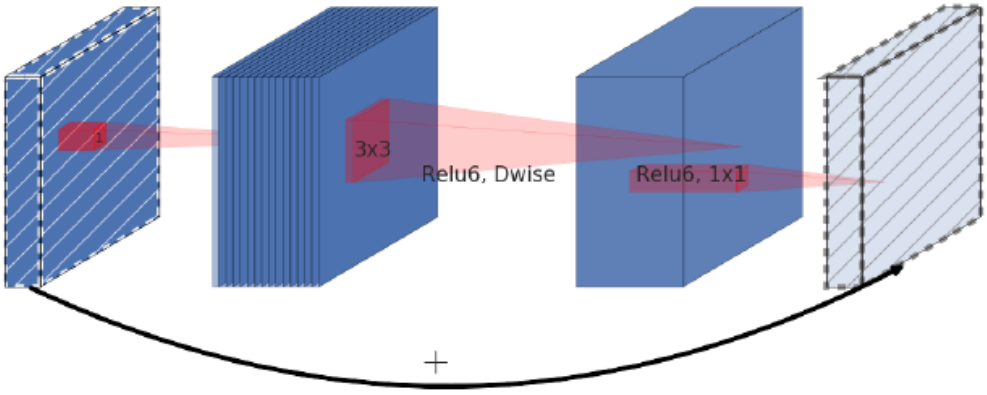
ภาพประกอบ 13 แสดงการดำเนินการ Depthwise Convolution และ Pointwise convolution

ภาพประกอบ 13 แสดงการดำเนินการคอนโวลูชันแบบ Depthwise Separable Convolutions โดยข้อมูลรับเข้ามีขนาด 3 ช่อง และขนาดของตัวกรอง Depthwise Convolution มีขนาด Dk x Dk x 1 จำนวน 3 ตัวกรอง (เท่ากับจำนวนช่องของข้อมูลรับเข้า) ทำให้ได้ขนาดของผลลัพธ์ของ Depthwise Convolution มีขนาด Dp x Dp x M จากนั้นนำไปดำเนินการ Pointwise Convolution โดยตัวกรองมีขนาด 1 x 1 x 1 x 3 (มีขนาดความลึกของตัวกรองเท่ากับขนาดความลึกของผลลัพธ์ของ Depthwise Convolution) และมีจำนวนของตัวกรอง N ตัวกรอง จะได้ผลลัพธ์สุดท้ายของการดำการแบบ Depthwise Separable Convolutions ที่มีขนาด Dp x Dp x N

MobileNetV2 ยังใช้การออกแบบโครงสร้างโดยใช้เทคนิค Inverted Residuals และ Linear Bottlenecks เพื่อช่วยลดการคำนวณแต่ยังคงประสิทธิภาพในการสกัดคุณลักษณะจากภาพ ซึ่งมีรายละเอียด ดังนี้

1) Inverted Residuals แนวคิดนี้ต่างจาก Residual Block แบบปกติ ที่ข้อมูลจะถูกส่งผ่านการแปลงเชิงลึกหลายชั้น โดยยังคงรักษาข้อมูลบางส่วนเอาไว้และส่งผ่านเส้นทางตรง (Skip Connection) เพื่อป้องกันปัญหาการสูญเสียข้อมูลเมื่อแบบจำลองมีความลึกมากเกินไป Inverted Residuals ใน MobileNetV2 กลับแนวคิดนี้ โดยแทนที่จะทำงานกับข้อมูลที่มีมิติมากก่อนแล้วค่อยลดขนาด มันจะขยายมิติของข้อมูลก่อน ด้วยการทำคอนโวลูชันด้วยตัวกรองขนาด 1x1 จำนวนมากก่อน เพื่อขยายจุดข้อมูลจากมิติต่ำให้มีมิติที่สูงขึ้น จากนั้นจะทำคอนโวลูชันด้วยตัวกรองขนาดเล็ก (3x3 Depthwise Convolution) และค่อยลดขนาดมิติของข้อมูลในขั้นสุดท้ายด้วยการทำคอนโวลูชันด้วยตัวกรองขนาด 1x1 อีกครั้ง สิ่งนี้ทำให้เกิดการประหยัดพารามิเตอร์มากขึ้น เพราะไม่ต้องคำนวณ convolutions แบบเต็มรูปแบบ Inverted Residual Blocks จะเชื่อมต่อ

ข้อมูลจากจุดเริ่มต้นของบล็อกไปยังจุดสิ้นสุด โดยผ่านเส้นทางตรง แต่จะเชื่อมต่อกับข้อมูลที่มีมิติต่ำในส่วน Bottleneck ซึ่งทำให้มีประสิทธิภาพมากยิ่งขึ้น ดังแสดงในภาพประกอบ 14

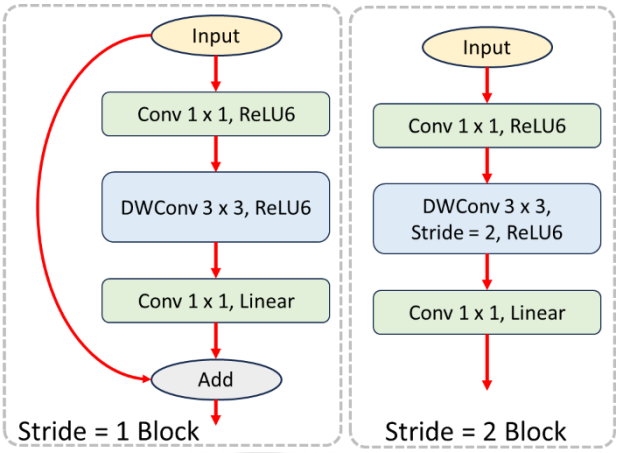


ภาพประกอบ 14 แสดงการโครงสร้างสถาปัตยกรรม Inverted Residual Block

ที่มา: Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (n.d.). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Copyright © 2018, IEEE.

2) Linear Bottleneck ในสถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชันส่วนใหญ่ จะใช้ฟังก์ชัน ReLU (Rectified Linear Unit) เป็นฟังก์ชันกระตุ้น เพื่อเพิ่มความไม่เชิงเส้น ในแบบจำลองโดยฟังก์ชันนี้จะทำงานโดยการตัดค่าลบทั้งหมดเป็น 0 และรักษาค่าบวกเอาไว้ ซึ่งช่วยให้การคำนวณในแบบจำลองทำได้เร็วขึ้น แต่เมื่อใช้ ReLU บนข้อมูลที่มีมิติเล็ก (ข้อมูลที่ผ่าน bottlenecks) การตัดค่าลบเป็น 0 อาจทำให้ข้อมูลสำคัญสูญหายได้ ทำให้ข้อมูลหายไปและส่งผลกระทบต่อความแม่นยำของแบบจำลอง เพื่อลดปัญหานี้ MobileNetV2 ได้นำเสนอการใช้ “Linear Bottlenecks” ซึ่งไม่ใช้ฟังก์ชันกระตุ้น ReLU แต่ใช้ Linear Function แทน เมื่อข้อมูลอยู่ในช่วงที่มีมิติต่ำ (bottleneck) เพื่อช่วยรักษาความเป็นเชิงเส้นของข้อมูล ทำให้ข้อมูลเชิงลึกที่อาจถูก ReLU กำจัดทิ้งสามารถถูกส่งต่อไปยังชั้นถัดไปได้โดยไม่เสียหาย

ภายใน Convolution Blocks ของแบบจำลองโครงข่ายประสาทเทียม MobileNetV2 จะมีการใช้งานโครงสร้าง Linear Bottlenecks 2 รูปแบบ คือ แบบที่กำหนดให้มีการใช้ Stride = 1 ร่วมกับการใช้ Skip Connection และ อีกแบบหนึ่งที่กำหนดให้มีการใช้ Stride = 2 โดยไม่มีการใช้ Skip Connection ดังแสดงในภาพประกอบ 15



ภาพประกอบ 15 แสดงโครงสร้างภายใน Convolution Blocks ของแบบจำลอง MobileNetV2

โครงสร้างโดยรวมของสถาปัตยกรรมแบบจำลองโครงข่ายประสาทเทียม MobileNetV2 เริ่มต้นด้วยการคอนโวลูชันข้อมูลรับเข้า ด้วยตัวกรองขนาด 3x 3 จำนวน 32 ตัวกรอง จากนั้นตามด้วยการใช้งาน Residual Bottleneck อีก 19 ชั้น ดังภาพประกอบ 12 และจะมีจำนวนพารามิเตอร์ทั้งหมด จำนวน 3,504,872 พารามิเตอร์ ซึ่งจะมีขนาดของแบบจำลอง 16 MB โดยมีรายละเอียดของจำนวนพารามิเตอร์ในแต่ละชั้นตามตาราง 4

ตาราง 4 แสดงจำนวนพารามิเตอร์ในแต่ละชั้นของแบบจำลองโครงข่ายประสาทเทียม MobileNetV2

Layer	Input Size	Output Size	Kernel Size	Stride	Repeat	Parameters
Conv2D	224x224x3	112x112x32	3x3	2	1	864
Bottleneck (t=1)	112x112x32	112x112x16	3x3	1	1	5,344
Bottleneck (t=6)	112x112x16	56x56x24	3x3	2	2	21,008
Bottleneck (t=6)	56x56x24	28x28x32	3x3	2	3	27,744
Bottleneck (t=6)	28x28x32	14x14x64	3x3	2	4	104,448
Bottleneck (t=6)	14x14x64	14x14x96	3x3	1	3	144,960
Bottleneck (t=6)	14x14x96	7x7x160	3x3	2	3	578,816
Conv2D	7x7x160	7x7x320	1x1	1	1	51,200
Avg Pooling	7x7x320	1x1x1280	7x7	-	-	1,280
Fully Connected	1x1x1280	1x1x1000	1x1	-	-	1,281,000

2.3.4 NASNet Mobile

แบบจำลอง NASNet Mobile (Zoph et al., 2018) เป็นแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ถูกพัฒนาโดยบริษัท Google ในปี 2018 สำหรับการใช้ในงาน Computer Vision โดยเฉพาะงานการจดจำภาพและการตรวจจับวัตถุ โดยมีความแม่นยำสูงถึง 74.9 % ในการทดสอบกับชุดข้อมูล ImageNet ซึ่งมีคุณสมบัติที่สำคัญ คือ การประมวลผลที่รวดเร็วและมีขนาดแบบจำลองที่เล็กเพื่อใช้ในอุปกรณ์ที่มีข้อจำกัดในด้านทรัพยากรจำกัดและพลังงาน เช่น โทรศัพท์มือถือ หรืออุปกรณ์พกพา

NASNet Mobile เป็นหนึ่งในแบบจำลองที่ถูกพัฒนาโดยใช้แนวคิด Neural Architecture Search (NAS) ซึ่งเป็นการใช้ปัญญาประดิษฐ์มาค้นหาและออกแบบสถาปัตยกรรมของโครงข่ายประสาทเทียม อย่างอัตโนมัติ โดยแบบจำลอง NASNet ที่ถูกพัฒนาโดยทีม Google Brain ซึ่งได้สร้างแบบจำลองมา 2 รูปแบบ คือ NASNet Large สำหรับใช้งานกับคอมพิวเตอร์ทั่วไป และ NasNet Mobile สำหรับใช้งานกับโทรศัพท์มือถือ และอุปกรณ์พกพา

สถาปัตยกรรมของ NASNet Mobile ใช้แนวทางของการค้นหาสถาปัตยกรรมแบบเชิงอัตโนมัติ โดยใช้ Reinforcement Learning ในการค้นหาหน่วยย่อยของเครือข่ายหรือที่เรียกว่า "เซลล์" (cells) เซลล์เหล่านี้ถูกค้นหาและออกแบบโดยอัลกอริทึม RL ที่เรียนรู้วิธีสร้างโครงสร้างที่มีประสิทธิภาพสูงสุดโดยอัตโนมัติ หลังจากนั้นเซลล์ที่ค้นพบนี้จะถูกซ้อนทับกันหลายชั้น (Stacked cells) เพื่อสร้างแบบจำลองที่สมบูรณ์ ซึ่งเป็นสถาปัตยกรรมที่คล้ายกับแบบจำลอง ResNet และ Inception แต่ถูกปรับปรุงให้มีประสิทธิภาพสูงขึ้นผ่านกระบวนการค้นหาอัตโนมัติ

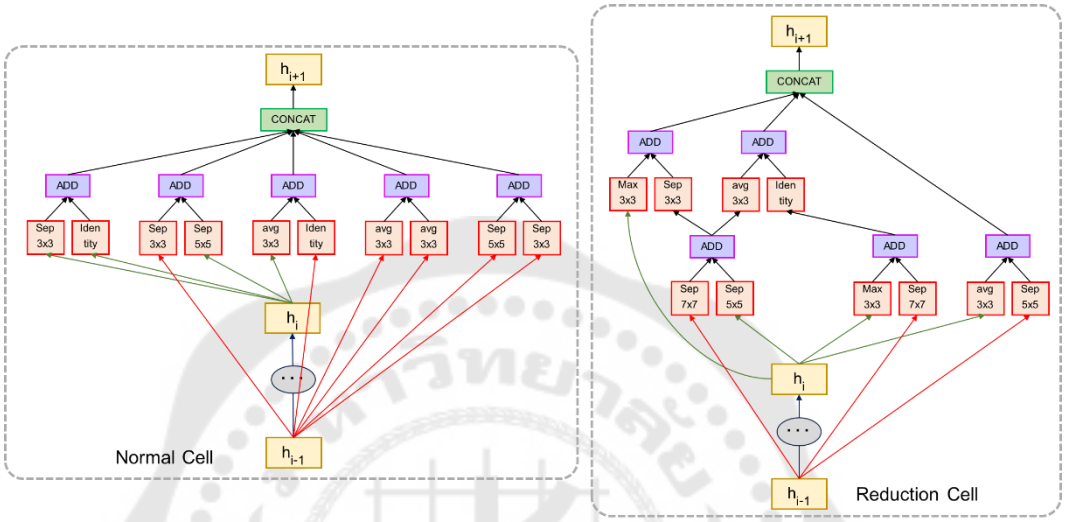
NASNet Mobile ประกอบด้วย 2 ประเภทเซลล์หลัก ได้แก่

1) Normal Cell เซลล์นี้จะรักษาขนาดของพื้นที่ของข้อมูล (Spatial Dimension) คือ ความกว้างและความสูงของภาพ ให้คงเดิม แต่จะเพิ่มจำนวน Feature Map ของภาพให้มีมิติของข้อมูลที่เพิ่มขึ้น

2) Reduction Cell เซลล์นี้จะลดขนาดของพื้นที่ของข้อมูลของข้อมูลลงครึ่งหนึ่ง เพื่อทำให้เครือข่ายมีการปรับปรุงการเรียนรู้ของคุณลักษณะของข้อมูลที่ซับซ้อนขึ้นโดยที่ไม่เพิ่มจำนวนการคำนวณมากเกินไป

เซลล์ทั้งสองรูปแบบจะถูกจัดเรียงสลับกันในลักษณะที่เรียกว่า "Cell-Based Architecture" โดยมีการเรียงซ้อนของ Normal Cell จำนวนหนึ่งเพื่อสกัดคุณลักษณะของภาพแล้วตามด้วย Reduction Cell หนึ่งครั้ง เพื่อลดขนาดของข้อมูล ซึ่งจะจัดเรียงเซลล์แบบนี้ไปเรื่อย ๆ

ในแต่ละ cell ประกอบด้วยโหนดหลาย ๆ โหนด โดยแต่ละโหนดจะรับข้อมูลรับเข้าจากโหนดก่อนหน้าสองโหนด และทำการประมวลผลด้วยการดำเนินการต่างๆ เช่น Convolution Pooling หรือ Identity Mapping แล้วรวมผลลัพธ์เข้าด้วยกัน เพื่อส่งข้อมูลไปยังชั้นถัดไป โครงสร้างลักษณะของ Normal Cell และ Reduction Cell แสดงดังภาพประกอบ 16



ภาพประกอบ 16 แสดงโครงสร้างและการดำเนินการทางคณิตศาสตร์ภายใน Normal Cell และ Reduction Cell

โครงสร้างของ NASNet Mobile มีโครงสร้างของการดำเนินการที่เรียงซ้อนเป็นชั้นอย่างซับซ้อน สามารถแยกออกเป็นกลุ่มของเซลล์ได้

1) Stem Layer

เป็นชั้นที่มีหน้าที่แปลงข้อมูลจากภาพรับเข้าให้เป็น Feature Map ขนาดเล็กลง พร้อมกับการดึงคุณลักษณะ (Feature) เริ่มต้นจากภาพ โดยการดำเนินการคอนโวลูชั่นด้วยตัวกรองขนาด 3x3 จำนวน 32 ฟิลเตอร์ เพื่อแปลงภาพรับเข้าจากขนาด 224x224x3 ไปเป็นข้อมูลขนาด 112x112x32 และมีการใช้ Batch Normalization ช่วยปรับสมดุลของค่าผลลัพธ์ (Output) เพื่อให้การเรียนรู้เป็นไปอย่างราบรื่นและลดการเกิดปัญหาการ Gradient Vanishing รวมทั้งมีการใช้ Max Pooling Layer เพื่อลดขนาดของข้อมูลแต่คงจำนวน Feature Map ไว้

2) Normal Cell

Normal Cell เป็นโครงสร้างของกลุ่มการดำเนินการคอนโวลูชันที่ถูกออกแบบมาเพื่อเพิ่มความลึกของ Feature Map โดยไม่เปลี่ยนแปลงขนาดเชิงพื้นที่ของ Feature Map เดิม ภายใน Normal Cell ใช้การดำเนินการคอนโวลูชัน แบบ Separable Convolutions ซึ่งแยกการคอนโวลูชัน

ออกเป็น 2 ขั้นตอน คือ Depthwise Convolution และ Pointwise Convolution เพื่อลดจำนวนพารามิเตอร์และการคำนวณในแบบจำลอง NASNet Mobile รวมถึงการใช้ฟังก์ชันกระตุ้น (Activation) ReLU เพื่อทำให้แบบจำลองสามารถเรียนรู้การเชื่อมโยงที่ไม่เป็นเชิงเส้น

3) Reduction Cell

Reduction Cell ถูกออกแบบมาเพื่อลดขนาดเชิงพื้นที่ของ Feature Map ลงครึ่งหนึ่งและเพิ่มจำนวน Feature Map ขึ้นในเวลาเดียวกัน โดยใช้ Separable Convolutions และมีการใช้ Max Pooling และ Average Pooling เพื่อปรับขนาดของ Feature Map

4) Final Layers

หลังจากผ่านการลดขนาดและดึงคุณลักษณะของข้อมูลจากภาพโดยใช้ Normal Cell และ Reduction Cell ข้อมูลจะถูกดำเนินการในใช้สุดท้ายโดยใช้ Global Average Pooling Layer เพื่อทำการรวมคุณลักษณะทั้งหมดโดยคำนวณค่าเฉลี่ยของแต่ละช่องใน Feature Map ทำให้ได้ขนาดของเวกเตอร์ จากนั้นจะเป็นการใช้งาน Fully Connected Layer เพื่อเปลี่ยนข้อมูลเป็นเวกเตอร์แบบ Fully Connected เพื่อนำไปใช้ในการจำแนกประเภทด้วย Softmax Layer เพื่อใช้คำนวณค่าความน่าจะเป็นสำหรับการจำแนกประเภท

โครงสร้างสถาปัตยกรรมและการดำเนินการของ NASNet Mobile แสดงรายละเอียดในแต่ละชั้นดังตาราง 5

ตาราง 5 แสดงรายละเอียดการดำเนินการในแต่ละชั้นของแบบจำลอง NASNet Mobile

Layer	Output Size	Description	Parameters
Input	224 x 224 x 3	RGB image input	0
Stem	112 x 112 x 32	3x3 conv, stride 2	896
4x Normal Cells	56 x 56 x 44	44 filters per cell	542,080
Reduction Cell 1	28 x 28 x 88	Stride 2	135,520
4x Normal Cells	28 x 28 x 88	88 filters per cell	2,168,320
Reduction Cell 2	14 x 14 x 176	Stride 2	542,080
4x Normal Cells	14 x 14 x 176	176 filters per cell	8,673,280
Global Average Pooling	1 x 1 x 1056	-	0
Fully Connected	1000	1000 class outputs	1,057,000
*Batch Normalization	-	ใช้ในทุก conv layer	207,540

แบบจำลอง NASNet Mobile จะมีจำนวนพารามิเตอร์ประมาณ 5,326,716 พารามิเตอร์ และมีขนาดของแบบจำลองประมาณ 20.43 MB

2.4 เทคนิคการทำ Model Optimization สำหรับการลดขนาดแบบจำลอง

ปัจจุบันโครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks : DNN) ถูกนำมาใช้งานอย่างแพร่หลายไม่ว่าจะเป็นในงานด้าน Computer Vision (CV) ไม่ว่าจะเป็นการจำแนกประเภท (Classification) การทำนาย (Prediction) หรือการตรวจจับวัตถุ (Object Detection) รวมทั้งการใช้งานด้าน Large Language Model (LLM) ที่ถูกนำมาประยุกต์ใช้และพัฒนาเป็นโปรแกรมประยุกต์ (Application) ให้กับผู้ใช้งานทั่วไป ซึ่งโปรแกรมต่าง ๆ เหล่านั้นถูกนำมาใช้งานในชีวิตประจำวันของมนุษย์ เพื่อเป็นเครื่องมือในการช่วยเหลือ หรืออำนวยความสะดวกในดำรงชีวิตหรือการทำงาน ให้สามารถเกิดประโยชน์ได้อย่างสูงสุด รวมทั้งการนำมาพัฒนาเพื่อเป็นเครื่องมือเพื่อใช้ในอุตสาหกรรมต่าง ๆ โดยเฉพาะอุตสาหกรรมที่ต้องใช้แรงงานมนุษย์ เช่น งานด้านการเกษตรกรรม เพื่อให้สามารถลดต้นทุนในการดำเนินการและเพิ่มประสิทธิภาพของผลผลิต

โครงข่ายประสาทเทียมเชิงลึกแต่ละแบบจำลองที่ถูกพัฒนาขึ้นจะมีความซับซ้อนมากขึ้นเพื่อมุ่งหวังให้มีประสิทธิภาพสูงขึ้น และมีความแม่นยำ เพื่อตอบสนองความต้องการของผู้ใช้งานในการใช้งานกับงานที่ต้องการการแก้ปัญหาที่ซับซ้อน ส่งผลให้แบบจำลองมีขนาดใหญ่ และมีจำนวน พารามิเตอร์จำนวนมาก ทำให้จำเป็นต้องใช้เครื่องคำนวณที่มีประสิทธิภาพ และมีหน่วยความจำสูง เช่น Personal Computer (PC) Workstation หรือ Server Computer เป็นต้น รวมทั้งต้องการใช้พลังงานในการคำนวณสูง ในปัจจุบันได้มีการนำเอาโครงข่ายประสาทเทียมเชิงลึกไปใช้ประยุกต์ใช้กับโปรแกรมประยุกต์บนสมาร์ทโฟน อุปกรณ์พกพา (Wearable Devices) หรือใช้กับงานด้าน IoT (Internet of Thing) ที่นำไปประยุกต์ใช้บน Edge Devices เพื่อให้สามารถประมวลผลแบบ Real-Time หรือนำประยุกต์ใช้บนอุปกรณ์สมองกลฝังตัว (Embedded System) ซึ่งอุปกรณ์เหล่านี้มีข้อจำกัดในเรื่องของทรัพยากรในการประมวลผล หน่วยความจำ และแหล่งพลังงานเนื่องจากเป็นอุปกรณ์พกพา ซึ่งถือเป็นเรื่องท้าทายสำหรับนักพัฒนาระบบ AI ที่ต้องออกแบบแบบจำลองให้มีขนาดเล็ก แต่ยังคงไว้ซึ่งประสิทธิภาพความแม่นยำให้สามารถใช้งานได้อย่างเหมาะสมกับทรัพยากรที่มีจำกัดบนอุปกรณ์

จากข้อจำกัดข้างต้น การจะนำโครงข่ายประสาทเทียมเชิงลึกไปใช้งานกับอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร จำเป็นจะต้องมีการทำ Model Optimization เพื่อลดขนาดของแบบจำลองลงให้มีขนาดที่เหมาะสมกับทรัพยากรที่มีอย่างจำกัดบนอุปกรณ์เหล่านั้น แต่ยังคงความสามารถในการประมวลผลและสามารถทำนายผลจากแบบจำลองได้อย่างมีประสิทธิภาพและแม่นยำ

ใกล้เคียงกับการใช้งานบนเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงที่ใช้ในการฝึกฝนหรือใช้ในการพัฒนาแบบจำลอง

ความแตกต่างระหว่าง Model Optimization กับการบีบอัดแบบจำลอง (Model Compression) แสดงดังตาราง 6

ตาราง 6 แสดงความแตกต่างของการทำ Model Optimization และ Model Compression

	Model Optimization	Model Compression
ความมุ่งหมาย	- การลดขนาดของแบบจำลองให้เหมาะสมกับอุปกรณ์ที่ใช้งาน - ปรับปรุงสมรรถนะในการประมวลผล เช่น Latency - ปรับปรุงประสิทธิภาพ และความแม่นยำในการทำนายผล	- ลดขนาดของแบบจำลอง ด้วยการลดจำนวนของพารามิเตอร์ หรือปรับเปลี่ยนชนิดของตัวแปรในการเก็บค่าพารามิเตอร์ - ลดความซับซ้อน (Complexity) ของแบบจำลอง
ผลลัพธ์ที่ต้องการ	- แบบจำลองที่ได้มีความเหมาะสมกับทรัพยากรของอุปกรณ์ สามารถทำงานได้อย่างมีประสิทธิภาพ - ใช้ทรัพยากร และหน่วยความจำในการประมวลผลน้อยลง - เวลาในการทำนายผล (Inference Time) น้อยลง	- แบบจำลองมีขนาดเล็กลง สามารถนำไปใช้งานกับอุปกรณ์ที่มีทรัพยากรจำกัดได้

ประโยชน์ของ การทำ Model Optimization

1) เพิ่มสมรรถนะของแบบจำลอง (Improved Performance)

แบบจำลองมีความแม่นยำสูงขึ้น (Accuracy) เนื่องจากมีความ Generalize กับข้อมูลใหม่ (Unseen data) ป้องกันการเกิด Over-Fitting และแบบจำลองยังใช้เวลาในการฝึกฝนและทำนายผล (Inference) ที่เร็วขึ้น ซึ่งเหมาะกับการใช้งานกับโปรแกรมประยุกต์ที่ต้องการประมวลผลแบบ Real-Time

2) การใช้ทรัพยากรอย่างมีประสิทธิภาพ (Resource Efficiency)

การทำ Model Optimization จะทำให้ช่วยลดการใช้งานหน่วยความจำ (Memory Footprint) ของแบบจำลองทำให้เหมาะสมกับการนำไปใช้งานบนอุปกรณ์ที่ทรัพยากรอย่างจำกัด

ลดพื้นที่สำหรับการจัดเก็บข้อมูล (Storage Space) และแบบจำลองยังใช้พลังในการประมวลผลที่น้อยกว่า ทำให้ลดการใช้พลังงานลง ซึ่งมีความสำคัญต่อการนำไปใช้งานบนสมาร์ทโฟน อุปกรณ์พกพา และ Edge Devices

3) สามารถขยายผลได้ (Scalability)

เนื่องจากแบบจำลองมีขนาดเล็ก และใช้ทรัพยากรน้อยทำให้มีความยืดหยุ่น สามารถนำไปใช้งานได้กับอุปกรณ์ หรือสภาพแวดล้อมที่หลากหลาย ตั้งแต่ On-Cloud On-premise บน Edge devices หรือแม้กระทั้งบนระบบสมองกลฝังตัว (Embedded System)

4) ลดต้นทุน (Cost Saving)

ช่วยลดต้นทุนด้านอุปกรณ์ (Hardware Cost) เนื่องจากแบบจำลองที่ผ่านการทำ Optimization มาแล้วสามารถทำงานได้อย่างมีประสิทธิภาพบนอุปกรณ์ที่มีพลังการคำนวณ และทรัพยากรไม่ต้องสูงมาก มีต้นทุนราคาของอุปกรณ์ที่ต่ำกว่า และช่วยลดต้นทุนด้านการปฏิบัติงาน (Operation Cost) เนื่องจากแบบจำลองที่ผ่านการทำ Optimization มาแล้วใช้พลังงานในการคำนวณที่น้อยกว่า และสามารถประมวลผลได้รวดเร็วกว่า

5) ประสบการณ์ของผู้ใช้งาน (User Experience)

เนื่องจากแบบจำลองสามารถประมวลผลได้อย่างรวดเร็ว ทำให้ช่วยเพิ่มประสบการณ์การใช้งาน โดยเฉพาะการใช้งานกับโปรแกรมประยุกต์แบบ Real-Time เช่น VDO streaming Gaming หรือ Augmented Reality

เทคนิคการทำ Model Optimization เพื่อลดขนาดของแบบจำลองจะเน้นการลดจำนวนโหนดในแบบจำลองเพื่อลดจำนวนพารามิเตอร์ (Weights และ Biases) การลดขนาดของตัวแปรในการเก็บข้อมูลของพารามิเตอร์เพื่อให้แบบจำลองใช้พื้นที่ในการเก็บข้อมูลและใช้หน่วยความจำน้อยลง และออกแบบแบบจำลองให้สามารถการลดจำนวนการคำนวณ หรือการดำเนินการทางคณิตศาสตร์ลง เพื่อให้แบบจำลองสามารถประมวลผลและทำนายผลได้รวดเร็วยิ่งขึ้น (ลด Latency) ใช้หน่วยความจำน้อยลง และยังคงประสิทธิภาพที่ดีในงานทำนาย การลดขนาดจำลองมีวิธีการหลัก ๆ อยู่ 5 วิธี ได้แก่

2.4.1 Model Pruning

การตัดแต่งแบบจำลองเป็นเทคนิคที่ช่วยลดขนาดของแบบจำลองโดยเฉพาะโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อให้แบบจำลองทำงานได้เร็วขึ้นและใช้ทรัพยากรในการประมวลผลน้อยลง วิธีการนี้ถูกพัฒนาขึ้นเพื่อลดจำนวนพารามิเตอร์หรือการเชื่อมต่อภายในแบบจำลอง (ลดจำนวน Neuron) ที่ไม่จำเป็นในการประมวลผลออกไป ในขณะที่ยังคง

ประสิทธิภาพและความแม่นยำของแบบจำลองไว้ให้ได้มากที่สุด ซึ่งเป็นประโยชน์อย่างยิ่งสำหรับการนำแบบจำลองโครงข่ายประสาทเทียมไปใช้งานในอุปกรณ์ที่มีข้อจำกัดเรื่องทรัพยากร เช่น สมาร์ทโฟน หรืออุปกรณ์ IoT

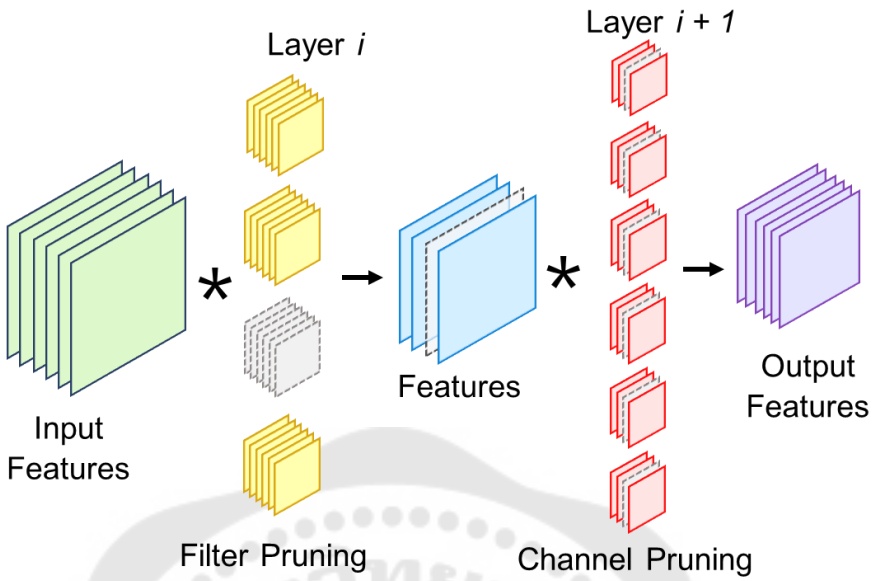
การตัดแต่งแบบจำลองมุ่งเน้นไปที่การลดจำนวนพารามิเตอร์ที่ไม่สำคัญออกจากแบบจำลอง ตัวอย่างเช่น ในโครงข่ายประสาทเทียมที่มีจำนวนของโหนดและน้ำหนักที่เชื่อมต่อระหว่างโหนดจำนวนมาก ซึ่งน้ำหนักบางค่าอาจมีค่าน้อยมากหรือไม่มีผลอย่างชัดเจนต่อการประมวลผลเพื่อทำนายผลลัพธ์ การไม่นำพารามิเตอร์ที่มีค่าน้อยมากๆ เหล่านั้นมาคำนวณ หรือนำออกไปจะช่วยลดขนาดของแบบจำลองและทำให้สามารถประมวลผลได้เร็วขึ้น เทคนิคนี้ยังช่วยลดการใช้หน่วยความจำและทำให้แบบจำลองสามารถถูกใช้ในอุปกรณ์ที่มีข้อจำกัดด้านการประมวลผลได้อย่างมีประสิทธิภาพ

การตัดแต่งแบบจำลองจะต้องหาจุดสมดุลระหว่างการลดขนาดของแบบจำลองกับการรักษาประสิทธิภาพในการประมวลผลของแบบจำลอง ถ้าแบบจำลองถูกตัดแต่งมากเกินไปก็อาจจะสูญเสียความสามารถในการทำนายทำให้ประสิทธิภาพลดลง แต่ถ้าแบบจำลองถูกตัดแต่งน้อยเกินไป ก็อาจไม่ได้ประโยชน์เท่าที่ควร ซึ่งต้องทดลองและปรับแต่งวิธีการในการตัดแต่งแบบจำลองให้เหมาะสมกับแบบจำลองและการใช้งานที่ต้องการ

2.4.1.1 **การตัดแต่งแบบ** Structured Pruning

Structured Pruning เป็นการลดขนาดของแบบจำลองเครือข่ายประสาทเทียม โดยลบพารามิเตอร์หรือการเชื่อมต่อบางส่วนที่ไม่จำเป็นออกไป ซึ่งจะตัดโครงสร้างของแบบจำลอง หรือลบพารามิเตอร์เป็นกลุ่ม ๆ เช่น การตัดทั้งฟิลเตอร์การตัดช่องสัญญาณ การตัดชั้นของแบบจำลอง การตัดทั้งโหนด หรือทั้งแถว ในชั้นคอนโวลูชัน วิธีนี้ทำให้แบบจำลองมีโครงสร้างที่เรียบง่ายและเหมาะสมมากขึ้นสำหรับการประมวลผลบนอุปกรณ์ที่มีทรัพยากรจำกัด เช่น สมาร์ทโฟน หรืออุปกรณ์ IoT ดังแสดงในภาพประกอบ 17

อีกวิธีหนึ่งคือการลบทั้งโหนดในชั้น Fully Connected โดยการวิเคราะห์น้ำหนักของการเชื่อมต่อแต่ละโหนด หากพบว่าโหนดใดมีค่าน้อยหรือไม่มีผลต่อการทำงานของแบบจำลอง โหนดนั้นสามารถถูกลบออกได้ การลบโหนดในลักษณะนี้จะช่วยลดความซับซ้อนของการคำนวณชั้น Fully Connected ที่มักจะมีจำนวนพารามิเตอร์มากที่สุดในแบบจำลอง ส่งผลให้แบบจำลองมีขนาดเล็กลงและใช้เวลาในการประมวลผลน้อยลง

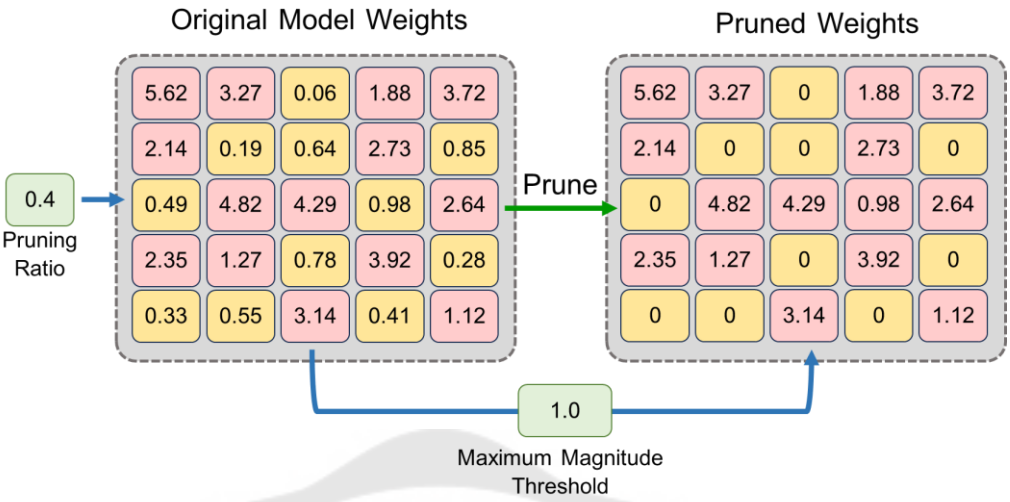


ภาพประกอบ 17 แสดงการตัดแต่งแบบจำลองแบบ Structure Pruning

การตัดแต่งแบบ Structured Pruning สามารถใช้กับการลบแถวหรือคอลัมน์ในเมทริกซ์ของน้ำหนักในการเชื่อมต่อระหว่างชั้นในแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ในกรณีนี้การลบแถวหรือคอลัมน์ที่มีค่าน้อยจะช่วยลดจำนวนการคำนวณที่ต้องทำในแต่ละขั้นตอนของการประมวลผล การลดจำนวนการคำนวณในลักษณะนี้ทำให้แบบจำลองสามารถประมวลผลได้เร็วขึ้นโดยยังคงรักษาประสิทธิภาพในการทำนายผลไว้ได้ใกล้เคียงเดิม

2.4.1.2 **การตัดแต่งแบบ** Unstructured Pruning

Unstructured Pruning เป็นการลดขนาดของแบบจำลองเครือข่ายประสาทเทียม โดยการลบพารามิเตอร์ที่ไม่จำเป็นออกทีละตัว ซึ่งแตกต่างจากการตัดแต่งแบบ Structured Pruning ที่ลบพารามิเตอร์เป็นกลุ่ม ๆ การตัดแต่งแบบ Unstructured Pruning จะเลือกตัดพารามิเตอร์ที่มีค่าน้อยหรือใกล้เคียงศูนย์ โดยไม่มีการคำนึงถึงโครงสร้างหรือรูปแบบของแบบจำลองที่แน่นอน วิธีการนี้ช่วยลดขนาดของแบบจำลองและทำให้การประมวลผลเร็วขึ้น ในขณะเดียวกันก็ยังคงความแม่นยำของแบบจำลองเอาไว้



ภาพประกอบ 18 แสดงการตัดแต่งแบบจำลอง โดยการตัดค่าน้ำหนัก (Weight Pruning)

ในการทำงานของแบบจำลองเครือข่ายประสาทเทียม พารามิเตอร์หรือน้ำหนักแต่ละตัวมีหน้าที่ในการเชื่อมต่อระหว่างโหนดหรือเชื่อมต่อระหว่างชั้น น้ำหนักที่มีค่าใกล้ศูนย์หรือไม่มีผลชัดเจนต่อการทำนายผลลัพธ์ สามารถถูกตัดออกได้เพื่อลดความซับซ้อนของแบบจำลอง มักใช้เพื่อปรับแต่งแบบจำลองที่ผ่านการฝึกฝนเสร็จแล้วให้มีขนาดเล็กลง โดยแสดงในภาพประกอบ 18

การตัดแต่งแบบ Unstructured Pruning ทำให้แบบจำลองเกิดความ “Sparsity” หรือความเบาบาง กล่าวคือ แบบจำลองจะมีจำนวนพารามิเตอร์ที่ใช้งานจริงน้อยลง ข้อดีของการทำให้แบบจำลองมีความ Sparsity คือ สามารถประหยัดทรัพยากรในการคำนวณ และลดเวลาในการประมวลผลได้ การคำนวณความ Sparsity ของแบบจำลองสามารถทำได้จากสมการ (16)

$$\text{Sparsity} = \frac{\text{จำนวนพารามิเตอร์ที่เป็นศูนย์}}{\text{จำนวนพารามิเตอร์ทั้งหมด}} \qquad (16)$$

ซึ่งค่า Sparsity ที่สูงแสดงว่าแบบจำลองมีพารามิเตอร์ที่ไม่จำเป็นมาก ทำให้แบบจำลองมีความเบาบางมากขึ้น และส่งผลให้ประสิทธิภาพในการประมวลผลเพิ่มขึ้นตามไปด้วย

ขั้นตอนในการทำ Unstructured Pruning เริ่มจากการฝึกแบบจำลองปกติให้มีความแม่นยำสูงสุดก่อน หลังจากนั้นจะทำการวิเคราะห์น้ำหนักหรือพารามิเตอร์ในแต่ละชั้นของแบบจำลอง พารามิเตอร์ที่มีค่าใกล้ศูนย์หรือไม่มีบทบาทสำคัญจะถูกลบออกโดยตรง หลังจากการลบพารามิเตอร์เหล่านี้ แบบจำลองจะถูกปรับปรุงและฝึกซ้ำอีกครั้งหนึ่งเพื่อให้กลับมามีความแม่นยำใกล้เคียงกับแบบจำลองเดิมที่ยังไม่ถูกตัดแต่ง

ในการเลือกพารามิเตอร์ที่จะถูกลบออกคือการใช้เทคนิคการวัดค่าความสำคัญของน้ำหนักหรือพารามิเตอร์ เช่น การใช้ L1 regularization หรือ L2 regularization ซึ่งเป็นเทคนิคที่ใช้ในการควบคุมขนาดของพารามิเตอร์โดยบังคับให้ค่าพารามิเตอร์มีความเบาบางมากขึ้น การใช้งานเทคนิคเหล่านี้ช่วยให้กระบวนการตัดแต่งพารามิเตอร์เป็นไปอย่างเป็นระบบและมีประสิทธิภาพมากขึ้น

อีกวิธีหนึ่งที่ซับซ้อนกว่าคือการดูว่าการเปลี่ยนแปลงของแต่ละการเชื่อมต่อส่งผลต่อความผิดพลาดของแบบจำลองอย่างไร วิธีนี้เรียกว่า Gradient-based Pruning เราจะคำนวณว่าถ้าเราเปลี่ยนค่าน้ำหนักของแต่ละการเชื่อมต่อไปนิดหน่อย จะทำให้ความผิดพลาดของแบบจำลองเปลี่ยนไปมากน้อยแค่ไหน การเชื่อมต่อที่มีผลน้อยต่อความผิดพลาดก็จะถูกตัดออกไป

ข้อดีของการตัดแต่งแบบ Unstructured Pruning คือ มันให้ความยืดหยุ่นสูง เราสามารถเลือกตัดเฉพาะบางส่วนที่ไม่สำคัญออกไปได้ โดยที่ยังคงรักษาโครงสร้างหลักของแบบจำลองไว้ วิธีนี้มักจะรักษาประสิทธิภาพของแบบจำลองได้ดีกว่าการตัดแต่งแบบ Structured Pruning โดยเฉพาะเมื่อเราต้องการลดขนาดของแบบจำลองลงมาก ๆ

แต่ข้อเสียของเทคนิคนี้ คือ การที่แบบจำลองอาจจะมีโครงสร้างที่ซับซ้อนและยากต่อการนำไปใช้งานในอุปกรณ์ที่มีข้อจำกัดด้านฮาร์ดแวร์ เนื่องจากการลบพารามิเตอร์แบบไม่คำนึงถึงโครงสร้าง ทำให้แบบจำลองที่ได้หลังจากการตัดแต่งมีการกระจายตัวของพารามิเตอร์ที่ไม่เป็นระเบียบ ดังนั้น การนำแบบจำลองไปใช้งานในอุปกรณ์ฝังตัวหรือในระบบซึ่งมีทรัพยากรจำกัดที่ต้องการประมวลผลแบบเรียลไทม์อาจทำได้ยากขึ้น โดยเฉพาะอุปกรณ์ที่ต้องการการคำนวณแบบขนาน เช่น GPU เพราะการคำนวณจะกระจัดกระจายไม่เป็นระเบียบ

2.4.2 Parameters Quantization

Model Parameters Quantization (Gholami et al., 2021) เป็นหนึ่งในเทคนิคที่ใช้ในกระบวนการลดขนาดของแบบจำลองประเภทเครือข่ายประสาทเทียม ซึ่งเทคนิคนี้ถูกนำมาใช้เพื่อลดความต้องการในการใช้หน่วยความจำและเพิ่มความเร็วในการคำนวณ โดยการลดความละเอียดของค่าพารามิเตอร์จากชนิดของข้อมูล (Data Types) ที่มีความละเอียดสูง เช่น 32-bit Floating Point (FP32) ไปเป็นรูปแบบของตัวแปรที่ใช้หน่วยความจำและการประมวลผลน้อยลง เช่น 16-bit Floating Point (FP16) หรือ 8-bit Integer (INT8) โดยที่ยังคงประสิทธิภาพการทำงานของแบบจำลองในระดับที่ยอมรับได้ (Rokh et al., 2023)

การทำ Quantization เป็นกระบวนการแปลงค่าตัวเลขจากช่วงที่กว้าง เช่น ค่าทศนิยมที่มีความละเอียดสูง (FP32) ให้เป็นช่วงที่เล็กลงโดยที่ข้อมูลจะถูกแทนที่ด้วยค่าที่มีความละเอียด

น้อยลง (FP16) หรือแปลงเป็นจำนวนเต็ม (Integer) กระบวนการนี้ช่วยลดการใช้หน่วยความจำและการคำนวณ แต่จะมีผลกระทบต่อความแม่นยำของแบบจำลองขึ้นอยู่กับลักษณะของข้อมูลและรูปแบบการใช้งาน

การปรับความละเอียดของพารามิเตอร์ของแบบจำลอง โดยการทำ Model Parameters Quantization มีหลายรูปแบบตามลักษณะการนำไปใช้งาน ซึ่งรูปแบบในการลดขนาดของแบบจำลองโครงข่ายประสาทเทียมโดยใช้เทคนิค Quantization สามารถแบ่งออกได้ 2 ประเภท ได้แก่ Quantization-Aware Training (QAT) และ Post-Training Quantization (PTQ)

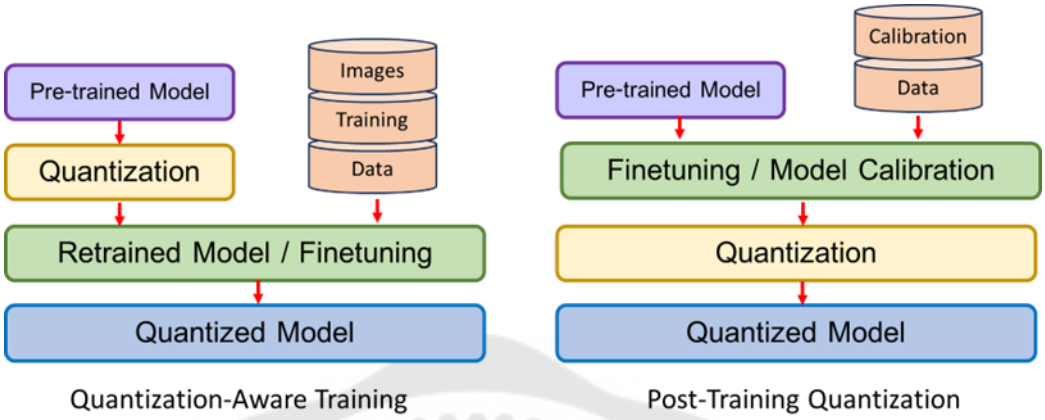
2.4.2.1 Quantization-Aware Training (QAT)

Quantization-Aware Training เป็นกระบวนการในการแปลงชนิดของค่าพารามิเตอร์ในขั้นตอนการฝึกฝนแบบจำลองตั้งแต่แรก ซึ่งจะคำนึงถึงผลกระทบจากการ Quantization โดยเฉพาะอย่างยิ่งเพื่อให้สามารถรับมือกับการแปลงค่าพารามิเตอร์จากตัวเลขทศนิยมที่มีความละเอียดสูง (Floating-Point) ไปเป็นจำนวนเต็ม (Integer) ได้ดียิ่งขึ้น ส่งผลให้เมื่อแบบจำลองถูกนำไปใช้งานจริงจะยังคงรักษาความแม่นยำสูง แม้ว่าจะผ่านการลดทอนความละเอียดของค่าพารามิเตอร์แล้วก็ตาม ซึ่งหมายความว่าในระหว่างการฝึกฝน (Training Phase) แบบจำลองจะรับรู้ถึงผลกระทบของการทำ Quantization ต่อประสิทธิภาพของแบบจำลอง และปรับค่าพารามิเตอร์ (Quantized Value) ที่ผ่านการทำ Quantization ให้ส่งผลการกระทบต่อประสิทธิภาพของแบบจำลองน้อยที่สุด ซึ่งวิธีการนี้ช่วยลดความคลาดเคลื่อน (Quantization Error) ที่เกิดจากการแปลงค่าเมื่อแบบจำลองถูกใช้งานจริง โดยจะให้ผลลัพธ์ที่มีความแม่นยำมากกว่า PTQ แต่เทคนิค QAT จะใช้เวลานานขึ้นและใช้ทรัพยากรมากขึ้นในการฝึกฝนแบบจำลอง เพราะต้องจำลองการทำ Quantization ในทุกรอบของการฝึกฝน นอกจากนี้ การตั้งค่าต่างๆ สำหรับ QAT อาจซับซ้อนและต้องการการปรับแต่งมากกว่าการฝึกฝนแบบปกติ ดังแสดงในภาพประกอบ 19

2.4.2.2 Post-Training Quantization (PTQ)

Post-Training Quantization เป็นการแปลงค่าพารามิเตอร์ของแบบจำลองหลังจากที่แบบจำลองได้ถูกฝึกฝนเรียบร้อยแล้ว โดยปกติแล้วแบบจำลองจะถูกฝึกฝน และบันทึกค่าของพารามิเตอร์ที่ถูกปรับค่าแล้วด้วยค่าพารามิเตอร์แบบ 32-bit Floating Point แล้วจะถูกแปลงไปเป็นชนิดของตัวแปรที่มีค่าความละเอียดน้อยลงในภายหลัง การทำ quantize แบบนี้เป็นวิธีที่สะดวกและไม่จำเป็นต้องใช้ข้อมูลการฝึกฝนอีกครั้ง กระบวนการนี้มักถูกนำไปใช้ในงานที่ต้องการลดขนาดของแบบจำลองให้สามารถทำงานได้บนอุปกรณ์ที่มีข้อจำกัดด้านพลังงานและทรัพยากร

เช่น บนอุปกรณ์ Edge Devices หรือ Mobile Devices ที่ต้องการใช้พลังงานน้อย และต้องการการประมวลผลที่รวดเร็ว ดังแสดงในภาพประกอบ 19



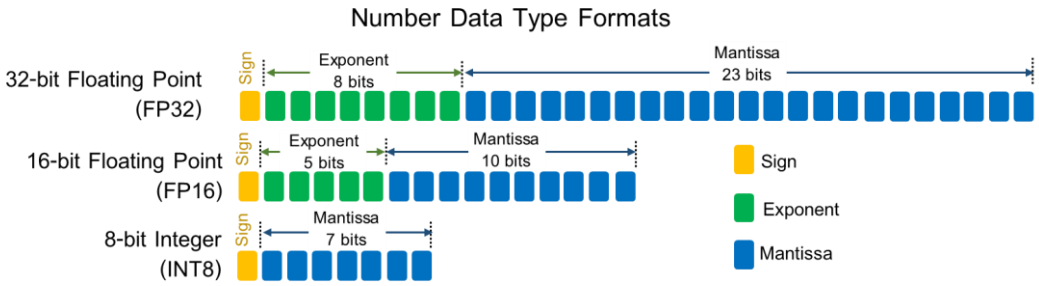
ภาพประกอบ 19 แสดงขั้นตอนการดำเนินการทำ Quantization แบบ Quantization-Aware Training และ Post-Training Quantization

การลดความละเอียดของข้อมูลโดยใช้เทคนิค PTQ เพื่อแปลงค่าจากรูปแบบที่มีความละเอียดสูง (FP32) ไปเป็นรูปแบบที่มีความละเอียดต่ำกว่า เช่น FP16 หรือ INT8 เพื่อลดขนาดของแบบจำลองและเพิ่มประสิทธิภาพการทำงาน มีการแปลงชนิดของข้อมูลหลายรูปแบบ ดังนี้

1) Float16 Quantization

Float16 Quantization เป็นกระบวนการที่ในการเปลี่ยนชนิดของตัวแปรโดยใช้เทคนิค PTQ จากรูปแบบ 32-bit Floating Point (FP32) หรือที่เรียกว่า Single-Precision Floating Point Format ไปเป็น 16-bit Floating Point (FP16) หรือ Half-Precision Floating Point Format โดยสามารถลดขนาดของค่าพารามิเตอร์ (จำนวนบิต) ลงครึ่งหนึ่งโดยไม่ต้องฝึกแบบจำลองใหม่ การทำ Quantization นี้ช่วยลดขนาดของแบบจำลองและเพิ่มประสิทธิภาพการประมวลผล โดยยังคงความแม่นยำของแบบจำลองไว้ในระดับที่ยอมรับได้

โครงสร้าง (Format) ของตัวแปรแบบ Floating Point โดยมีโครงสร้างของตัวแปรดังภาพประกอบ 20 ซึ่ง FP32 จะใช้ข้อมูล 1 บิต สำหรับเครื่องหมาย (Sign) ข้อมูล 8 บิต สำหรับ Exponent และข้อมูล 23 บิต สำหรับ Mantissa ส่วน FP16 จะใช้ข้อมูล 1 บิต สำหรับเครื่องหมาย (Sign) ข้อมูล 5 บิต สำหรับ Exponent และข้อมูล 10 บิต สำหรับ Mantissa โดยสามารถแปลงค่าจาก Floating Point เป็นค่าจำนวนจริง (Real Value) ได้จากสมการ (17)



ภาพประกอบ 20 แสดงโครงสร้างของตัวแปรชนิด FP32 FP16 และ INT8

$$\text{Real Value} = (-1)^{s} \times 2^{(e-bias)} \times (1+m) \qquad (17)$$

โดยที่

s คือ บิตของเครื่องหมาย (Sign)

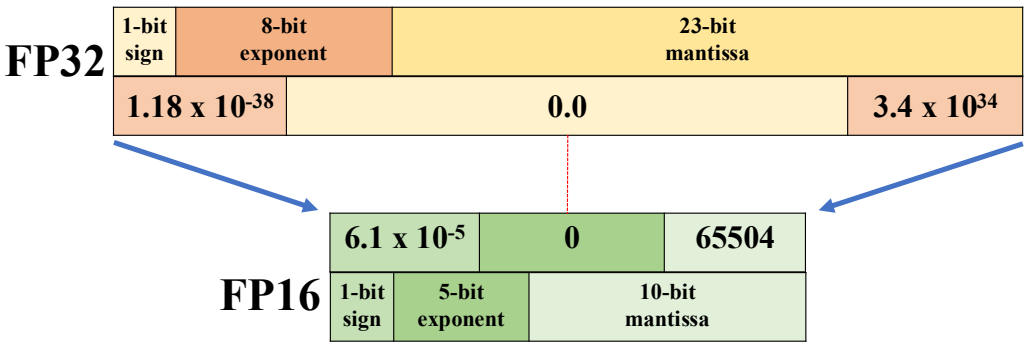
e คือ ค่าชี้กำลัง (Exponent)

m คือ ค่าเศษ (Mantissa)

$bias$ คือ ค่าชดเชยชี้กำลัง ใน FP32 bias=127 และใน FP16 bias=15

ขั้นตอนในการแปลงค่าจาก FP32 ไปเป็น FP16

ก) การปรับขนาดของเลขชี้กำลัง (Exponent Adjustment) เป็นการแปลงค่า Exponent bit โดยลดจำนวนจาก 8 บิต สำหรับ FP32 ให้เหลือ 5 บิต สำหรับ FP16 ซึ่งช่วงของการแปลงค่าจำนวนจริงของ FP32 ให้อยู่ในรูปแบบ FP16 เป็นไปตามภาพประกอบ 21



ภาพประกอบ 21 แสดงช่วงของจำนวนจริงของการแปลงจาก FP32 ไปเป็นรูปแบบ FP16

ในตัวแปร FP32 ค่าชี้กำลังที่ใช้ 8 บิต ซึ่งสามารถเก็บค่าตั้งแต่ 0 ถึง 255 แต่ใน FP16 ค่าชี้กำลังจะถูกลดเหลือเพียง 5 บิต ทำให้สามารถเก็บค่าได้ในช่วง 0 ถึง 15 ซึ่งหมายความว่า ค่าที่ใหญ่มากๆ หรือต่ำมากๆ อาจไม่สามารถแสดงใน FP16 ได้ จึงต้องมีการ ลดทอน (Clamping) ค่าที่อยู่นอกช่วงนี้ให้เป็นค่าที่สามารถแสดงได้ใน FP16โดยจะทำการแปลงค่าโดยใช้การคำนวณดังสมการ (18)

$$e_{FP16} = \max(\min(e_{FP32} - 112, 31), 0) \quad (18)$$

โดยที่

e_{FP16} คือ ค่าชี้กำลังของ FP16

e_{FP32} คือ ค่าชี้กำลังของ FP32

ข) การปัดเศษ (Mantissa Truncation) เป็นการลดค่า Mantissa bit โดยลดจำนวนจาก 23 บิต สำหรับ FP32 ให้เหลือ 10 บิต สำหรับ FP16

ค่าเศษ (Mantissa) ใน FP32 ใช้ 23 บิต เพื่อเก็บค่าเศษของทศนิยมที่มีความละเอียดสูง แต่ใน FP16 ค่าเศษจะถูกลดขนาดเหลือเพียง 10 บิต ทำให้ความละเอียดของข้อมูลที่สามารถแสดงได้ลดลง ซึ่งกระบวนการนี้จะมีการ ตัดทอน (truncating) ค่าทศนิยมที่ไม่สำคัญออก โดยทำการตัดค่าที่เกิน 10 บิตออกไป (ตัดค่าเศษตั้งแต่บิตที่ 11 ของ FP32 ทิ้งไป)

ตัวอย่างเช่น ต้องการแปลงค่า 9.7510 ซึ่งมีค่าเท่ากับ 0 10000010 00111000000000000000000 ในตัวแปรแบบ FP32 จะสามารถแปลงเป็น FP16 ได้ดังนี้

- ค่า Sign ยังคงเป็น 0 เนื่องจากค่าบวก

- ค่า Exponent ใน FP16 ค่าชี้กำลังจะมี 5 บิต และใช้ bias = 15 ดังนั้น เราจะคำนวณใหม่ จากสมการ (18) ซึ่งแทนค่า e_{FP32} = 10000010_2 = 130_{10}

$$e_{FP16} = \max(min(130 - 112, 31), 0) = 18$$

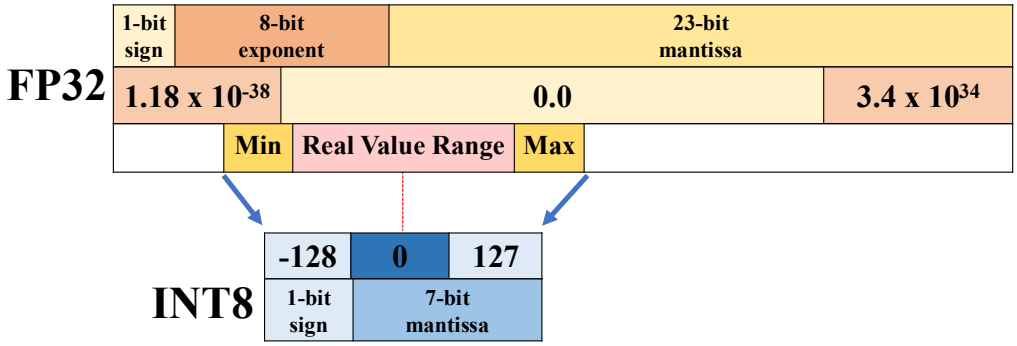
ซึ่ง ค่า 18 เขียนในรูปเลขฐานสองคือ 10010

- ค่า Mantissa ใน FP16 ค่า Mantissa จะถูกลดเหลือเพียง 10 บิตเท่านั้น ดังนั้นจากค่า Mantissa เดิม 00111000000000000000000 เราจะเก็บเพียง 10 บิตแรกเท่านั้น จะได้เท่ากับ 0011100000

ดังนั้น การแปลงค่า 0 10000010 00111000000000000000000 ในตัวแปรแบบ FP32 (9.7510) ไปเป็น FP16 มีค่าเท่ากับ 0 1001 0001110000

2) Dynamic Range Quantization

Dynamic Range Quantization เป็นกระบวนการที่ทำให้แบบจำลองแปลงค่าพารามิเตอร์จากรูปแบบตัวแปรแบบ FP32 ไปเป็น INT8 (8-bit Integer) โดยการแทนค่าจำนวนจริง (Real Value) ในรูปแบบทศนิยมช่วงกว้าง (FP32) ด้วยค่าจำนวนเต็มระหว่าง -128 ถึง 127 (INT8) แต่ยังคงรักษาช่วงของค่า (Dynamic Range) ไว้ให้ได้มากที่สุด ซึ่งช่วงของการแปลงค่าจำนวนจริงจากรูปแบบ FP32 ให้อยู่ในรูปแบบ INT8 แสดงดังภาพประกอบ 22



ภาพประกอบ 22 แสดงช่วงของจำนวนจริงของการแปลงจาก FP32 ไปเป็นรูปแบบ INT8

ในการดำเนินการ Quantization จะมีการใช้ค่า Scaling Factor และ Zero Point (Li et al., 2023) เพื่อช่วยในการแปลงค่าให้เหมาะสม จากสมการดังนี้

การหาค่า Scaling Factor จากสมการ (19)

$$S = \frac{R_{max} - R_{min}}{Q_{max} - Q_{min}} \tag{19}$$

โดยที่

Q_{max} คือ ค่าสูงสุดของการทำ Quantization

Q_{min} คือ ค่าต่ำสุดของการทำ Quantization

R_{max} คือ ค่าสูงสุดของค่าจำนวนจริง (Real Value)

R_{min} คือ ค่าต่ำสุดของค่าจำนวนจริง

สำหรับการแปลงค่าจาก FP32 เป็น INT8 Dynamic Range Quantization จะกำหนดค่า Q_{max} = 127 และ Q_{min} = -128 for INT8

และ การหาค่า Zero Point จากสมการ (20)

$$Z = INT\left(round\left(Q_{max} - \frac{R_{max}}{S}\right)\right) \tag{20}$$

ซึ่งค่าจำนวนเต็ม (Integer) ที่ได้จากการ Quantization จะหาได้จากสมการ(22)

$$Q = round(\frac{R}{S} + Z) \quad (21)$$

ตัวอย่าง การแปลงค่า FP32 ในเมทริกซ์ A ไปเป็น INT8 จะสามารถทำได้ดังนี้

1) ทำการ mapping ค่า R_{max} กับ ค่าQ_{max} และ mapping ค่า R_{min} กับ ค่า Q_{min} ดังภาพประกอบ 23

-249.13	49.23	-28.15	→	-128	
266.11	-106.08	123.45	→	127	
19.04	0	-3.08			

ภาพประกอบ 23 แสดงการ Mapping ค่ามากที่สุดและน้อยที่สุดของ FP32 กับค่า INT8

2) คำนวณหาค่าที่เหลือจากสมการ (19) (21) และ (21)

S = (266.11 – (-249.13)) / (127 - (-128)) = 2.02

Z = INT(round(127- (266.11 / 2.02))) = -5

จากนั้นคำนวณหาค่า Q ทั้งหมด จะได้ผลลัพธ์ดังภาพประกอบ 24

-249.13	49.23	-28.15	FP32 to INT8	-128	19	-19
266.11	-106.08	123.45		127	-58	56
19.04	0	-3.08		4	-5	-7

ภาพประกอบ 24 แสดงการแปลงค่า FP32 ไปเป็นค่า INT8

3) ซึ่งถ้าหากลอง Dequantization กลับไปเป็น FP32 จะได้ผลลัพธ์ดังภาพประกอบ 25

-128	19	-19	INT8 to FP32	-248.53	48.49	-28.29
127	-58	56		266.71	-107.09	123.25
4	-5	-7		18.18	0	-4.04

ภาพประกอบ 25 แสดงการแปลงค่าจาก INT8 กลับไปเป็นค่า FP32

จะเห็นได้ว่า เมื่อแปลงค่ากลับจาก INT8 เป็น FP32 แล้วพบว่า ค่าพารามิเตอร์มีความคลาดเคลื่อนไปจากความเป็นจริง เนื่องมาจากการปัดเศษเป็นจำนวนเต็ม จากการคำนวณในค่าของ S Z และ Q

ทั้งนี้ ในการแปลงค่าพารามิเตอร์ (Weights และ Biases) ของแบบจำลอง โดยใช้วิธี Dynamic Range Quantization นี้ จะเป็นการแปลงเฉพาะค่าพารามิเตอร์เพื่อใช้ในการเก็บข้อมูลเท่านั้น แต่ข้อมูลและการดำเนินการทางคณิตศาสตร์ในการประมวลผลแบบจำลองยังคงใช้รูปแบบของตัวแปรแบบ FP32 อยู่ ดังนั้น เมื่อแบบจำลองถูกนำมาใช้งานจะมีการแปลงค่าพารามิเตอร์ (ในรูปแบบ INT8) ที่ถูกเก็บไว้ในหน่วยความจำ (Storage Area) กลับมาเป็นรูปแบบของ FP32 เพื่อใช้ในการประมวลผลแบบจำลองอีกครั้งหนึ่ง ทำให้การประมวลผลแบบจำลองอาจจะไม่ได้รวดเร็วเท่าที่ควร

3) Full Integer Quantization

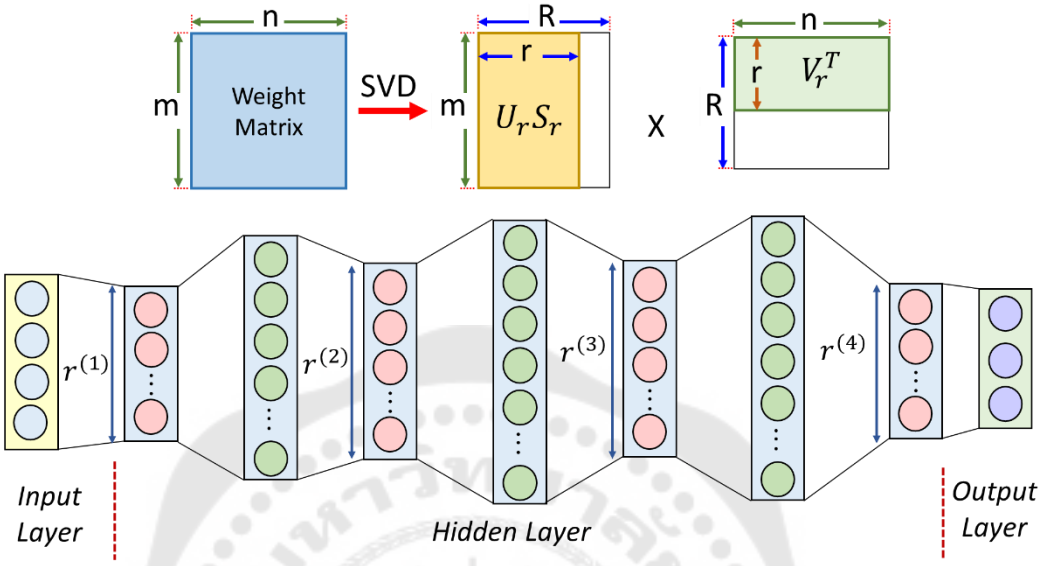
Full Integer Quantization หรือ Static Quantization เป็นวิธีการที่ออกแบบมาสำหรับการลดขนาดของแบบจำลอง เพื่อใช้กับอุปกรณ์ที่รองรับค่าแบบจำนวนเต็ม เช่น Edge TPU และไมโครคอนโทรลเลอร์ 8 บิต วิธีนี้จะทำการแปลงค่าตัวแปรแบบ Floating-point ทั้งหมด รวมถึงค่าข้อมูลรับเข้า ค่าข้อมูลส่งออก และผลลัพธ์ระหว่างการดำเนินการทางคณิตศาสตร์ในการประมวลผลของแบบจำลองให้เป็นจำนวนเต็ม (INT8) ซึ่งการทำ Full Integer Quantization จำเป็นต้องมีการปรับเทียบแบบจำลอง (Model Calibration) ด้วยชุดข้อมูลขนาดเล็กจากชุดการตรวจสอบ (Validation Set) โดยทั่วไปจะใช้ตัวอย่างระหว่าง 100 ถึง 500 ตัวอย่าง เพื่อกำหนดช่วงของค่าต่าง ๆ สำหรับเทนเซอร์ (Tensors) การปรับเทียบนี้จะเกี่ยวข้องกับการรันแบบจำลองผ่านการคำนวณการอนุมาน (Inference) จำนวนหนึ่ง เพื่อประมาณค่าต่ำสุดและค่าสูงสุดที่จำเป็นสำหรับการทำ Quantization

ในการศึกษาครั้งนี้ไม่ได้ใช้อุปกรณ์แบบ 8-bit Integer ในการประมวลผลแบบจำลอง ดังนั้น จึงเน้นเฉพาะการศึกษาเกี่ยวกับการทำ Float16 Quantization และ Dynamic Range Quantization เท่านั้น

2.4.3 Low-Rank Decomposition / Low-Rank Factorization

Low-rank Decomposition (Li et al., 2023) ใช้เทคนิคการลดขนาดของแบบจำลอง โดยมีพื้นฐานมาจากแนวคิดที่ว่าเมทริกซ์ขนาดใหญ่สามารถประมาณ (Approximate) ได้ด้วยผลคูณของเมทริกซ์ที่มีอันดับ (Rank) ต่ำกว่า ซึ่งสามารถนำแนวความคิดนี้มาใช้ในการย่อยเมทริกซ์น้ำหนัก (Weights Matrices) ในแบบจำลองโครงข่ายประสาทเทียมให้เป็นเมทริกซ์ย่อยที่มีอันดับ

ต่ำกว่า ทำให้ช่วยลดจำนวนพารามิเตอร์และความซับซ้อนของแบบจำลองโดยไม่สูญเสียประสิทธิภาพมากนัก



ภาพประกอบ 26 แสดงการตัวอย่างการทำ Matrix Decomposition

จากภาพประกอบ 26 จะเห็นได้ว่าหาก Weight Matrix มีขนาด m x n และมี Rank r สามารถย่อยให้กลายเป็น matrix ขนาดเล็กได้

ขั้นตอนในการทำ Low-rank Decomposition

1) Matrix Decomposition

ทำการย่อยเมทริกซ์น้ำหนักของแบบจำลองที่มีขนาดใหญ่ให้เป็นเมทริกซ์ขนาดเล็ก ตั้งแต่ 2 low-rank matrices ขึ้นไป โดยใช้เทคนิคต่างๆ เช่น Singular Value Decomposition (SVD) QR decomposition และ Non-negative Matrix Factorization (NMF)

2) Rank Selection

Rank ของเมทริกซ์ย่อยจะถูกเลือกจากการเปรียบเทียบระหว่าง ขนาดของแบบจำลอง และ ความแม่นยำ การเลือก Rank ที่ต่ำจะมีข้อดีในเรื่องของการลดขนาดของแบบจำลองได้ดี แต่จะมีข้อเสียใจเรื่องของประสิทธิภาพที่ลดลงและมีอัตราการสูญเสียเพิ่มขึ้น

3) Fine Tuning

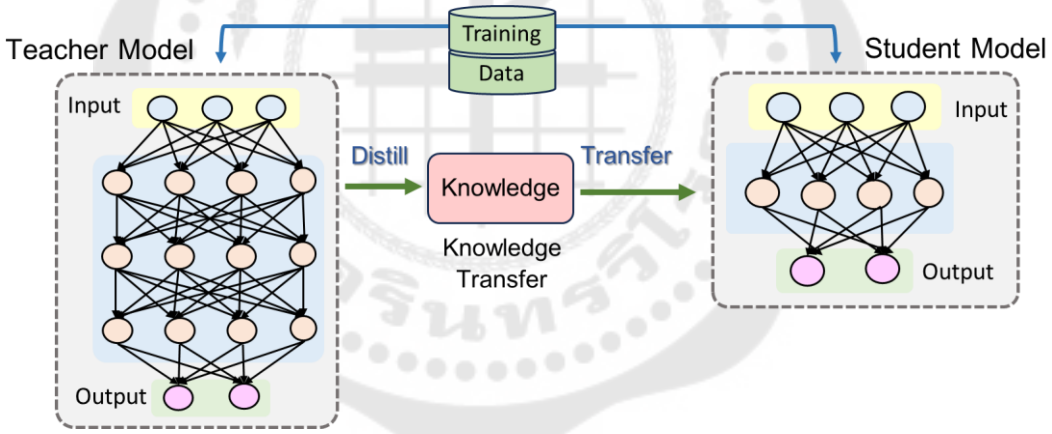
หลังจากที่ผ่านการทำย่อยเมทริกซ์ลงแล้ว แบบจำลองสามารถถูกนำกลับมาทำการปรับค่าพารามิเตอร์โดยละเอียด (Fine-tuning) อีกครั้ง เพื่อแก้ไขการสูญเสียความแม่นยำ ซึ่งการ Fine-

tuning จะช่วยให้เมทริกซ์ที่ถูกย่อยเป็นเมทริกซ์ขนาดเล็กปรับค่าต่างๆ ภายในโครงสร้าง และช่วยให้แบบจำลองมีประสิทธิภาพเพิ่มขึ้น

ถึงแม้ว่าการใช้เทคนิคการ Low-rank Decomposition จะช่วยย่อยเมทริกซ์น้ำหนักให้เป็นเมทริกซ์ย่อยที่มีอันดับต่ำกว่าจำนวนหลายเมทริกซ์ ทำให้แบบจำลองใช้พื้นที่ในการเก็บข้อมูลน้อยลง แต่ไม่ได้ช่วยในการเพิ่มความเร็วในการคำนวณเพื่อทำนายผล (Inference) มากนัก เนื่องจากเมื่อแบบจำลองถูกนำมาใช้งานจริงจะต้องมีการคำนวณให้เมทริกซ์ย่อย ๆ เหล่านั้นกำลังมาเป็นเมทริกซ์น้ำหนักที่มีขนาดเท่าเดิมก่อน ทำให้ไม่ส่งผลต่อประสิทธิภาพในด้านความเร็วในการประมวลผลมากนักเนื่องจากใช้จำนวนพารามิเตอร์ในการประมวลผลเท่าเดิม

ข้อจำกัดหนึ่งของการใช้เทคนิค Low-rank Decomposition คือ เมื่อคำนวณเมทริกซ์ย่อยที่มี Rank ต่ำกว่ากลับมาเป็นเมทริกซ์ต้นฉบับ จะพบว่าค่าที่ได้จากการคำนวณเป็นเมทริกซ์ต้นฉบับมีค่าความผิดพลาด (Error) เกิดขึ้น ซึ่งอาจจะส่งผลทำให้ประสิทธิภาพของแบบจำลองอาจจะให้ผลได้ดีไม่เท่ากับแบบจำลองที่ไม่ได้ใช้เทคนิค Low-rank Decomposition

2.4.4 Knowledge Distillation

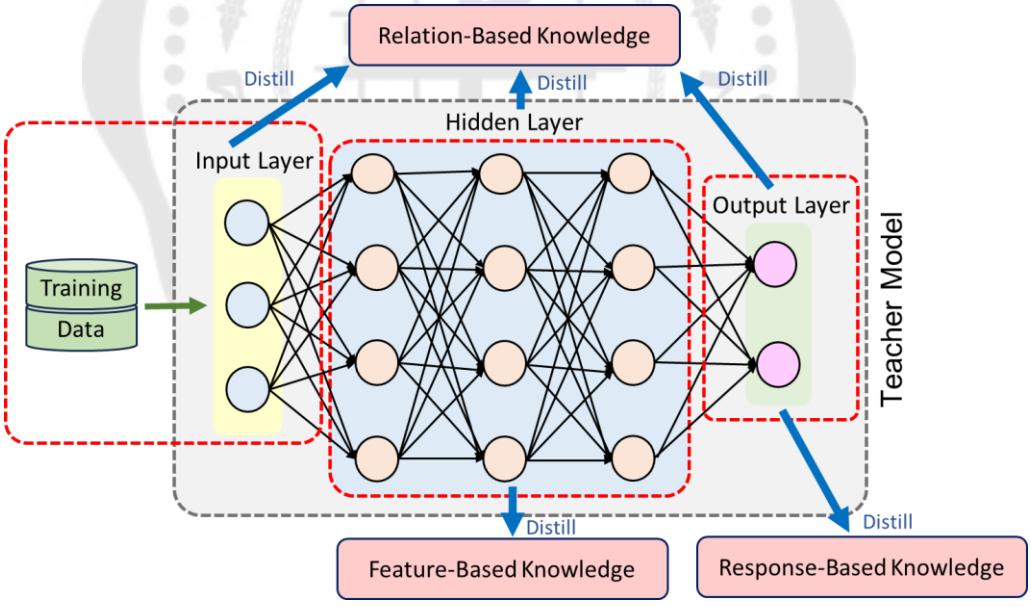


ภาพประกอบ 27 แสดงแนวความคิดในการพัฒนาแบบจำลองโดยวิธี Knowledge Distillation

Knowledge Distillation (Gou et al., 2021) เป็นกระบวนการที่ใช้ในการลดขนาดแบบจำลองการเรียนรู้เชิงลึก โดยการถ่ายโอนความรู้ของแบบจำลอง จากแบบจำลองขนาดใหญ่และมีโครงสร้างที่ซับซ้อน ซึ่งแบบจำลองนั้นถูกฝึกฝนจนมีประสิทธิภาพสูง (Teacher Model) ไปยังแบบจำลองขนาดเล็กที่มีโครงสร้างซับซ้อนน้อยกว่า (Student Model) ซึ่งสามารถใช้ได้จริงภายใต้ข้อจำกัดของทรัพยากรที่มีบนอุปกรณ์ โดยรูปแบบของแบบจำลองลักษณะนี้เรียกว่า “Teacher-Student Model” ดังแสดงในภาพประกอบ 27

Knowledge ในที่นี้จะหมายความถึง ประสิทธิภาพและพฤติกรรมในการประมวลผลและทำนายผลลัพธ์ที่แบบจำลองขนาดใหญ่ (Teacher Model) ได้เรียนรู้จากข้อมูลหลังการฝึกฝน เช่น การแทนค่าของพารามิเตอร์ภายในแบบจำลอง (Internal Representations) ความสัมพันธ์ระหว่างคุณลักษณะของข้อมูล และผลลัพธ์ที่ต้องการทำนาย รวมถึงความสัมพันธ์ระหว่างผลลัพธ์ในแต่ละคลาส ซึ่ง Knowledge เหล่านี้จะถูกคำนวณออกมาให้อยู่ในรูปแบบของค่า Distillation Loss จาก Teacher Model เพื่อนำไปใช้ในการฝึกฝน Student Model ต่อไป

Knowledge Distillation และ Transfer Learning เป็นเทคนิคการถ่ายทอดความรู้ในงานเรียนรู้เชิงลึกเหมือนกัน แต่มีวัตถุประสงค์ต่างกัน โดย Knowledge Distillation มุ่งเน้นการลดขนาดแบบจำลองโดยถ่ายทอดความรู้ในรูปแบบของพฤติกรรมการทำนายผลของแต่ละคลาสจากแบบจำลองขนาดใหญ่ไปยังแบบจำลองขนาดเล็ก (Student Model) เพื่อให้แบบจำลองเล็กสามารถทำงานได้เร็วขึ้นและใช้ทรัพยากรน้อยลง ในทางตรงกันข้าม Transfer Learning มุ่งเน้นการนำแบบจำลองที่ผ่านการฝึกฝนมาแล้วจากชุดข้อมูลหนึ่งมาใช้กับชุดข้อมูลใหม่ หรือปัญหาที่แตกต่างออกไป โดยไม่ต้องเริ่มฝึกฝนแบบจำลองใหม่ตั้งแต่ต้น



ภาพประกอบ 28 แสดงแหล่งความรู้ (Knowledge) ที่ใช้ในการฝึกฝน Student Model ของแต่ละรูปแบบในการใช้เทคนิค Knowledge Distillation

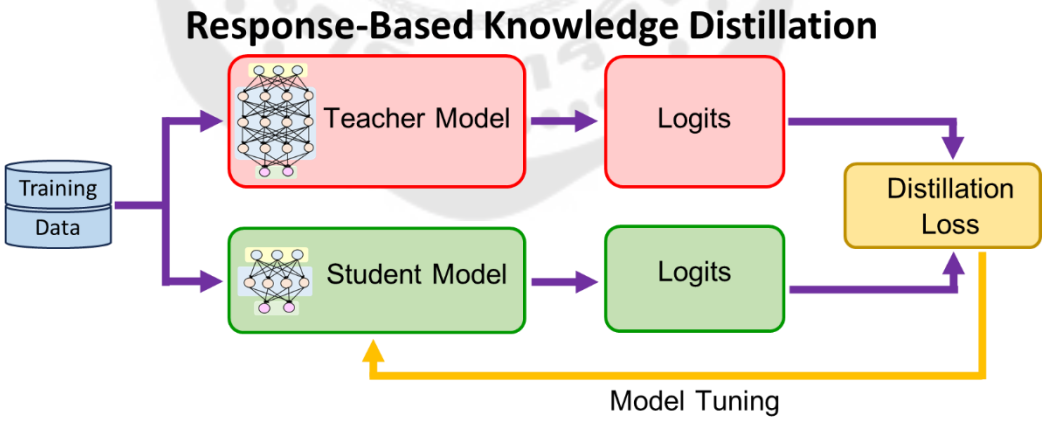
Knowledge Distillation จะใช้แหล่งความรู้จาก logits หรือฟังก์ชันที่ใช้ในการหาค่าความน่าจะเป็นของคลาสต่างๆ ใน Output Layer ของ Teacher Model เป็นแหล่งความรู้

(Knowledge Source) ที่จะถ่ายทอดให้กับ Student Model และจะทำการปรับแก้ค่าพารามิเตอร์ของ Student Model จากค่า Distillation Loss

รูปแบบของ Knowledge Distillation แบ่งเป็น 3 แบบ ได้แก่ Response-based knowledge Feature-based knowledge และ Relation-based knowledge ซึ่งแต่ละรูปแบบจะมีแหล่งของความรู้ที่ใช้ฝึกฝน Student Model แตกต่างกันออกไป ดังแสดงในภาพประกอบ 28

2.4.4.1 Respond-based Knowledge

Respond-based Knowledge เป็นวิธีที่ใช้ผลลัพธ์การทำนาย (Predictions) ที่ได้จาก Teacher Model มาเป็นตัวนำทางให้กับ Student Model โดยวิธีการนี้อิงกับผลลัพธ์ของ Output layer ในรูปแบบของ Softmax Layer ของ Teacher Model ซึ่งจะไม่ใช้แค่ผลลัพธ์ของคลาสที่ถูกต้อง แต่ยังใช้การแจกแจงความน่าจะเป็นทั้งหมดของคลาสอื่นๆ ในกระบวนการนี้ Student Model จะเรียนรู้จาก Soft Targets ที่ Teacher Model ทำนายออกมา ซึ่งจะสามารถทำได้โดยการใช้ Loss function ที่เรียกว่า “Distillation Loss” โดยจะวัดค่าความแตกต่างของผลลัพธ์การทำนายของ Student Model และ Teacher Model ซึ่ง Distillation Loss จะถูกปรับค่าให้ลดลง (Minimized) ระหว่างการนำแบบจำลอง Teacher Model มาทำกระบวนการ Knowledge Distillation ให้กับ Student Model เพื่อให้ Student Model สามารถที่จะทำนายผลได้ดีเทียบท่ากับ Teacher Model แต่มีขนาดเล็กกว่าและมีความซับซ้อนในเชิงโครงสร้างของแบบจำลองที่น้อยกว่า ดังแสดงในภาพประกอบ 29



ภาพประกอบ 29 แสดงการทำงานของ Response-based Distillation

Response-based Distillation ถูกใช้งานอย่างแพร่หลายในงานด้าน Machine Learning เช่น Image Classification Natural Language Processing และ Speech Recognition

ในงานด้าน Computer Vision โดยเฉพาะงานด้าน Image Classification โดยทั่วไปจะใช้ฟังก์ชัน Softmax ในการจำแนกคลาสของผลลัพธ์ แต่ในการทำ Knowledge Distillation จะใช้ฟังก์ชัน ที่เรียกว่า “Soft Target” เพื่อหาค่า Probability Distribution ของ Output Class ซึ่ง Soft Target จะมีการใช้ค่า Temperature (T) (Li et al., 2023) เพื่อควบคุมความสำคัญของ Soft Target แต่ละค่าตามสมการ (22)

$$q(Z_i, T) = \frac{e^{(Z_i/T)}}{\sum_j e^{(Z_j/T)}} \quad (22)$$

โดยที่

Z_i คือ logits ของ class ที่ i, j = 1 , 2 , ... , k

k คือ จำนวนคลาสทั้งหมด

T คือ Temperature Parameter ซึ่ง ถ้า T = 1 จะเป็นฟังก์ชัน Softmax ปกติ และในการระหว่างการฝึกฝนแบบจำลองจะมีการกำหนดค่า T ให้มีค่า T > 1

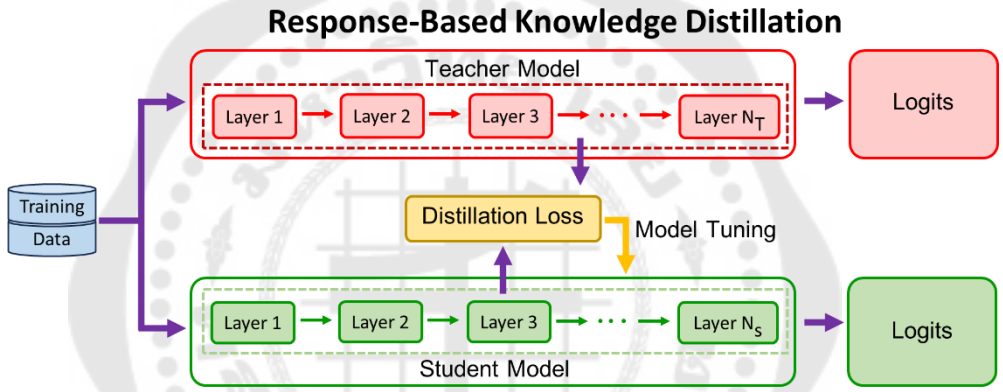
Response-based distillation ประโยชน์อย่างยิ่งเมื่อ Teacher Model มีจำนวน Output Class จำนวนมาก ซึ่งการฝึก Student Model จากเริ่มต้นจะมีต้นทุนสูงมากในการฝึกฝนแบบจำลองให้ประสิทธิภาพ แต่ด้วยการใช้ Response-based Distillation นั้น Student Model สามารถเรียนรู้เพื่อเลียนแบบพฤติกรรมของ Teacher Model โดยไม่ต้องเรียนรู้การตัดสินใจที่ซับซ้อนซึ่งแยกแยะระหว่าง Output Class ทั้งหมด

อย่างไรก็ตาม Response-based Distillation มีข้อจำกัดบางประการ เช่น เทคนิคนี้ถ่ายทอดความรู้ที่เกี่ยวข้องกับการคาดการณ์ผลลัพธ์จาก Output Layer ของ Teacher Model (ใช้ Distillation Loss) เท่านั้น และไม่ได้ถ่ายทอดค่าพารามิเตอร์ ภายในแบบจำลองของ Teacher Model โดยตรง ดังนั้น เทคนิคนี้อาจไม่เหมาะสำหรับงานที่ต้องการการตัดสินใจ หรือการทำนายผลลัพธ์จาก Feature ของข้อมูลรับเข้าที่มีความซับซ้อนมาก

2.4.4.2 Feature-based Knowledge

Feature-based Knowledge มุ่งเน้นไปที่การถ่ายทอดความรู้จากคุณลักษณะเชิงลึก (Deep Features) ที่ Teacher Model เรียนรู้จากข้อมูลต้นฉบับ ซึ่งคุณลักษณะเหล่านี้ คือ ข้อมูลที่ได้จากแต่ละชั้น (Layers) ภายในของ Teacher Model ที่ถูกแปลงจากข้อมูลดิบ (Raw Data / Raw Value) ให้กลายเป็นตัวแทนเชิงนามธรรมที่ใช้ในการจำแนกคลาส (Feature Representations) วิธีการนี้ทำให้ Student Model สามารถเรียนรู้ลักษณะของข้อมูลที่สำคัญจาก Teacher Model โดยไม่จำเป็นต้องเรียนรู้ทั้งหมดใหม่

แนวคิดหลักของ Feature-based Knowledge คือ การพยายามทำให้ Feature Representations ที่สร้างขึ้นโดย Student Model มีความคล้ายคลึงกับ Feature Representations ของแบบจำลอง Teacher Model มากที่สุด โดยทั่วไปแล้ว Feature Representations เหล่านี้จะถูกสร้างขึ้นในชั้นต่างๆ ของแบบจำลอง โดยเฉพาะอย่างยิ่งในชั้นกลาง (Intermediate Layers / Hidden Layers) ซึ่งมักจะเป็นที่ชั้นที่สกัดได้ข้อมูลที่สำคัญเกี่ยวกับลักษณะเฉพาะของข้อมูลที่ใช้ในการประมวลผล โดยในระหว่างกระบวนการทำ Knowledge Distillation จะใช้ Loss Function ที่วัดระยะห่าง หรือความแตกต่างระหว่างการ Feature Representations ของ Teacher Model กับ Student Model เช่น MSE หรือ Kullback-Leibler Divergence การทำงานของ Feature-based Knowledge แสดงดังภาพประกอบ 30



ภาพประกอบ 30 แสดงการทำงานของ Feature-based Knowledge

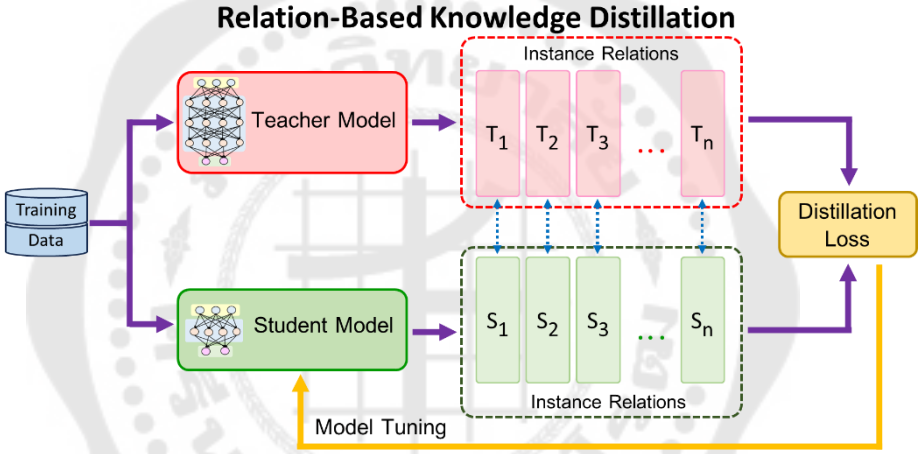
ข้อจำกัดที่สำคัญของ Feature-based knowledge คือ เทคนิคนี้จะต้องใช้ทรัพยากรมากกว่าเทคนิคอื่น ๆ เพราะจะต้องดึงการแทนค่าพารามิเตอร์ (Feature Representations) ภายใน Hidden Layers ของ Teacher Model ในแต่ละรอบการฝึก และ Feature-based knowledge อาจไม่เหมาะสำหรับงานที่โครงสร้างของ Hidden Layers ของ Teacher Model มีความแตกต่างหรือไม่สัมพันธ์กับ Student Model จนไม่สามารถคำนวณหาค่า Loss ได้

2.4.4.3 Relation-based Knowledge

Relation-based Knowledge เป็นหนึ่งในวิธีการสำคัญของ Knowledge Distillation ซึ่งเป็นเทคนิคในการถ่ายทอดความรู้จากแบบจำลองขนาดใหญ่ไปยังแบบจำลองขนาดเล็กโดยมุ่งเน้นการถ่ายทอดความสัมพันธ์ระหว่างตัวอย่างข้อมูล (Samples) รับเข้า หรือ Feature Maps ในแต่ละชั้นที่แบบจำลองใช้ในการเรียนรู้และตัดสินใจภายใน Teacher Model และภายใน Student Model เอง ซึ่งแตกต่างจาก Feature-based Knowledge จะเน้นการถ่ายทอด

ความสัมพันธ์ระหว่าง Teacher Model และ Student Model วิธีการนี้มีความสำคัญอย่างมากในการช่วยให้ Student Model สามารถเรียนรู้โครงสร้างความสัมพันธ์ของข้อมูลภายในโครงสร้างของแบบจำลองได้อย่างมีประสิทธิภาพ แม้จะมีขนาดและโครงสร้างของแบบจำลองที่มีความซับซ้อนน้อยกว่า Teacher Model

แนวคิดหลักของ Relation-based Knowledge คือ การพยายามทำให้ความสัมพันธ์ระหว่างข้อมูลหรือ Feature Maps ที่ Student Model เรียนรู้มีความคล้ายคลึงกับความสัมพันธ์ที่ Teacher Model เรียนรู้ โดยทั่วไปแล้วความสัมพันธ์เหล่านี้จะถูกพิจารณาในรูปแบบของระยะทาง (Distances) หรือมุม (Angles) ใน Feature Maps ซึ่งสะท้อนถึงโครงสร้างของข้อมูลที่แบบจำลองใช้ในกาประมวลผลเพื่อทำนายผลลัพธ์ ดังแสดงในภาพประกอบ 31



ภาพประกอบ 31 แสดงการทำงานของ Relation-based Knowledge

ข้อดีหลักประการหนึ่งของ Relation-based Knowledge คือ สามารถช่วยให้ Student Model เรียนรู้ความสัมพันธ์ระหว่าง Input Samples และ Output Labels ที่ Generalize และทนทานขึ้นกว่าการฝึกฝนแบบจำลองตั้งแต่เริ่มต้น นี่เป็นเพราะ Teacher Model ได้เรียนรู้ความสัมพันธ์ที่เกี่ยวข้องมากที่สุดระหว่าง Input Samples และ Output Labels จากข้อมูลแล้ว ซึ่งสามารถถ่ายทอดไปยัง student model

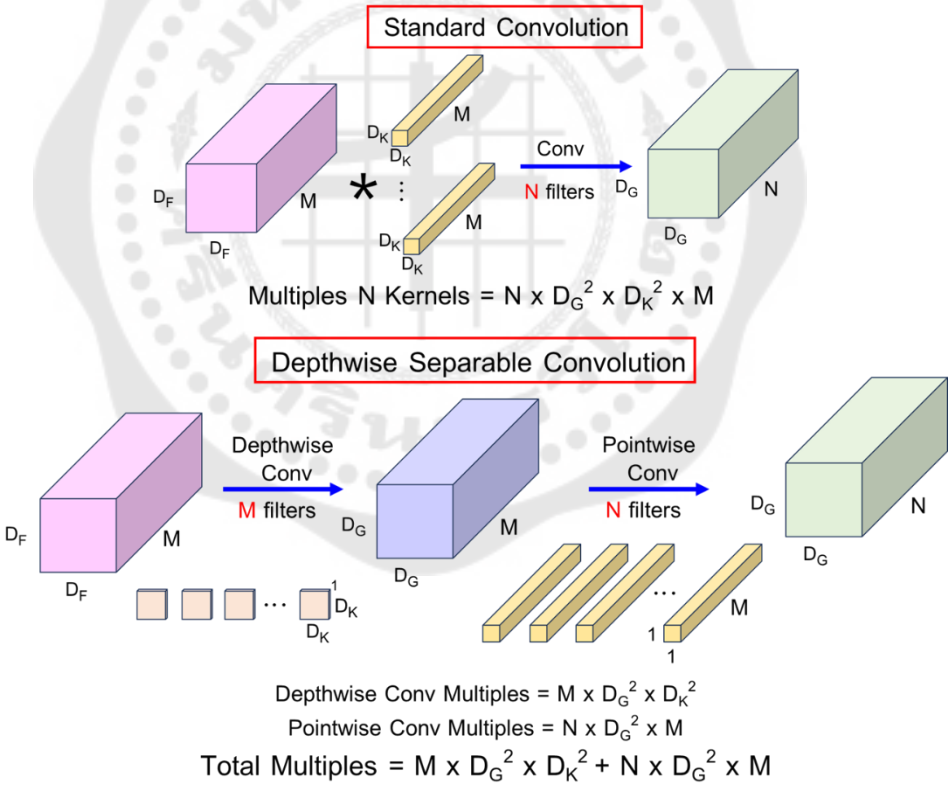
อย่างไรก็ตาม การสร้างเมทริกซ์ หรือ Tensors ความสัมพันธ์โดยเฉพาะสำหรับชุดข้อมูลขนาดใหญ่ อาจต้องใช้ทรัพยากรมาก นอกจากนี้ Relation-based Knowledge อาจไม่เหมาะสำหรับงานหรือข้อมูลที่ความสัมพันธ์ระหว่าง Input Samples และ Output Labels ไม่ได้ถูกกำหนดอย่างชัดเจนหรือยากต่อการแปลง

2.4.5 Lightweight Model Design / Lightweight Architectures

Lightweight Model Design เป็นเทคนิคในการลดขนาดของแบบจำลองโดยการออกแบบโครงสร้าง หรือสถาปัตยกรรมของ neural network ใหม่ เพื่อลดจำนวนพารามิเตอร์ และความซับซ้อนในการประมวลผล ตัวอย่างของ Lightweight Model เช่น MobileNet ที่ใช้สถาปัตยกรรมโครงสร้างของโครงข่ายแบบ Depthwise Separable Convolution SqueezeNet ที่ใช้สถาปัตยกรรมโครงสร้างของ Fire Module หรือ ShuffleNet ที่ใช้สถาปัตยกรรมโครงสร้างของโครงข่ายแบบ Group Convolution และ Channel Shuffle เป็นต้น

2.4.5.1 Depthwise Separable Convolution

โครงสร้างการดำเนินการคอนโวลูชันแบบ Depthwise Separable Convolution (Chollet, 2017) ซึ่งจะแตกต่างจากการใช้การดำเนินการคอนโวลูชันแบบปกติ โดย Depthwise Separable Convolution จะมี 2 ขั้นตอน ในการ Convolution ได้แก่



ภาพประกอบ 32 แสดงการเปรียบเทียบการดำเนินการทางคณิตศาสตร์ของการคอนโวลูชันแบบปกติ และ Depthwise Separable Convolution

1) การทำ Depthwise Convolution จะใช้ใช้ตัวกรองขนาดเล็ก (D_k x D_k) ที่มีความลึกเท่ากับ 1 ซึ่งจะได้ตัวกรองขนาด D_k x D_k x 1 และมีจำนวนของตัวกรองเท่ากับจำนวนช่องสัญญาณ (Channel) ของข้อมูลรับเข้าซึ่งหากข้อมูลรับเข้ามีจำนวนช่องสัญญาณ M ช่องสัญญาณ ก็จะได้จำนวนของตัวกรองเท่ากับ M ตัวกรอง ในการดำเนินการคอนโวลูชันกับข้อมูลรับเข้า ตัวกรองแต่ละตัวจะดำเนินการคอนโวลูชันกับแต่ละช่องของข้อมูลรับเข้าแยกจากกัน 1 ช่องต่อ 1 ตัวกรอง ทำให้ได้ผลลัพธ์ของการทำ Depthwise Convolution เท่ากับ D_k x D_k x M

2) Pointwise Convolution (หรือ 1x1 convolution) ใช้ตัวกรองขนาด 1x1 ที่มีขนาดความลึกเท่ากับ ช่องสัญญาณของข้อมูลรับเข้า ซึ่งจะได้ตัวกรองขนาด 1 x 1 x M และมีจำนวนของตัวกรองเท่ากับจำนวนของ Feature Map ที่ต้องการ (Filter = N) จากนั้นนำตัวกรองไปดำเนินการคอนโวลูชันกับผลลัพธ์ที่ได้จากการทำ Depthwise Convolution (D_k x D_k x M) จะได้ผลลัพธ์สุดท้ายของการทำ Depthwise Separable Convolution ขนาด D_G x D_G x N

หากพิจารณาเปรียบเทียบจำนวนการดำเนินการทางคณิตศาสตร์ระหว่างการดำเนินการคอนโวลูชันแบบปกติ กับการดำเนินการแบบ Depthwise Separable Convolutions ดังแสดงในภาพประกอบ 32 จะพบว่า

การดำเนินการคอนโวลูชันแบบปกติ มีจำนวนการดำเนินการทางคณิตศาสตร์ จำนวน D_k x D_k x M x N x D_F x D_F ครั้ง (Input มีขนาด D_F x D_F)

และการดำเนินการคอนโวลูชันแบบ Depthwise Separable Convolutions จะมีการดำเนินการทางคณิตศาสตร์ แยกออกเป็นการดำเนินการแบบ Depthwise Convolution จำนวน D_k x D_k x M x D_F x D_F ครั้ง และการดำเนินการแบบ Pointwise Convolution จำนวน M x N x D_F x D_F ครั้ง รวมการดำเนินการแบบ Depthwise Separable Convolutions จะมีการดำเนินการทางคณิตศาสตร์ทั้งสิ้น (D_k x D_k x M x D_F x D_F) + (M x N x D_F x D_F)

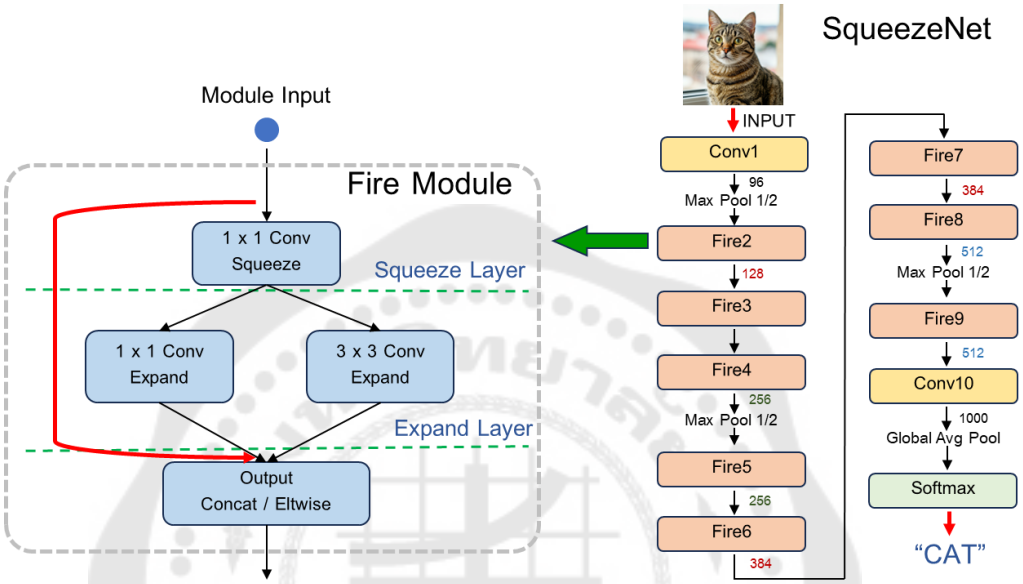
เปรียบเทียบประสิทธิภาพระหว่างการดำเนินการคอนโวลูชันแบบปกติ กับการดำเนินการแบบ Depthwise Separable Convolutions จะได้ดังสมการ (22) ซึ่งพบว่า จำนวนการดำเนินการทางคณิตศาสตร์ของ Depthwise Separable Convolutions มีจำนวนการคำนวณน้อยกว่ามาก

$$\frac{\text{จำนวนการดำเนินการ } \mathbf{Depthwise\ Separable\ Conv}}{\text{จำนวนการดำเนินการ } \mathbf{Standard\ Conv}} = \frac{(\mathrm{D}_k \times \mathrm{D}_k \times \mathrm{M} \times \mathrm{D}_F \times \mathrm{D}_F) + (\mathrm{M} \times \mathrm{N} \times \mathrm{D}_F \times \mathrm{D}_F)}{\mathrm{D}_k \times \mathrm{D}_k \times \mathrm{M} \times \mathrm{N} \times \mathrm{D}_F \times \mathrm{D}_F}$$

$$\frac{\text{จำนวนการดำเนินการ } \mathbf{Depthwise\ Separable\ Conv}}{\text{จำนวนการดำเนินการ } \mathbf{Standard\ Conv}} = \frac{1}{\mathrm{N}} + \frac{1}{\mathrm{D}_k{}^2} \qquad (23)$$

แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่นำการดำเนินการแบบ Depthwise Separable Convolution มาประยุกต์ใช้เพื่อสร้างเป็นโครงข่ายประสาทเทียมแบบคอนโวลูชัน ได้แก่ MobileNet (Adam et al., 2017) MobileNetV2 (Sandler et al., 2018) เป็นต้น

2.4.5.2 Fire Module

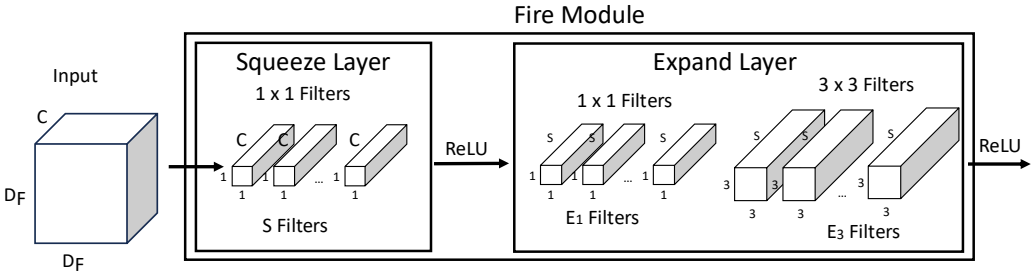


ภาพประกอบ 33 แสดงโครงสร้างของ Fire Module ภายในแบบจำลองแบบ SqueezeNet

Fire Module (Iandola et al., 2016) เป็นส่วนประกอบหลักในสถาปัตยกรรมของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน SqueezeNet ซึ่งออกแบบมาเพื่อลดจำนวนพารามิเตอร์แบบจำลองโดยยังคงประสิทธิภาพในการประมวลผลสูง โดยเฉพาะในการใช้ในงานด้านการจำแนกภาพ โมดูลนี้ประกอบไปด้วย 2 ส่วนหลัก คือ Squeeze Layer และ Expand Layer ดังภาพประกอบ 33ซึ่งทั้งสองส่วนทำงานร่วมกันเพื่อบีบอัดและขยายข้อมูลที่ทำให้การสกัดคุณลักษณะมีประสิทธิภาพมากขึ้น การทำงานของ Fire Module มีหลักการดังนี้

1) Squeeze Layer

ในการบีบอัดข้อมูลจะใช้ Squeeze Layer เป็นส่วนที่ทำหน้าที่ลดจำนวนช่องสัญญาณข้อมูล (Channels) ที่เข้าสู่ Fire Module โดยใช้การคอนโวลูชันด้วยตัวกรองขนาด 1 x 1 ซึ่งเป็นการใช้ตัวกรองที่ทำงานกับแต่ละจุดของภาพโดยไม่คำนึงถึงข้อมูลในบริเวณใกล้เคียง ตัวกรองขนาด 1 x 1 ช่วยลดจำนวนการคำนวณลง เพราะไม่ต้องคำนึงถึงความซับซ้อนของพื้นที่รอบจุดนั้น การลดขนาดช่องสัญญาณด้วยวิธีนี้เรียกว่าการบีบอัดข้อมูล



ภาพประกอบ 34 แสดงการดำเนินการของ Fire Module กับข้อมูลรับเข้า

ยกตัวอย่างเช่น ข้อมูลรับเข้า หรือ Feature Map มีขนาด D_F x D_F x C (ความสูง D_F ความกว้าง D_F และจำนวนช่องสัญญาณ C) และใน Squeeze Layer ใช้ตัวกรองขนาด 1 x 1 x C จำนวน S ตัว ทำให้ให้ได้ผลลัพธ์ของการดำเนินการเท่ากัน D_F x D_F x S และมีจำนวนการคำนวณเท่ากับ D_F x D_F x C x S ซึ่งจำนวนของตัวกรอง S ที่เลือกใช้ มักจะน้อยกว่า C (S < C) ทำให้เกิดการลดขนาดของข้อมูลเมื่อข้อมูลถูกบีบอัดด้วย Squeeze Layer ซึ่งข้อมูลนี้จะถูกส่งไปยัง Expand Layer เพื่อขยายออกอีกครั้ง ดังแสดงในภาพประกอบ 34

2) Expand Layer

การขยายข้อมูล หลังจาก Squeeze Layer ทำการบีบอัดข้อมูลแล้ว Expand Layer จะเข้ามาขยายข้อมูลออกโดยใช้การดำเนินคอนโวลูชัน 2 รูปแบบ คือ การดำเนินการคอนโวลูชันด้วยตัวกรองขนาด 1 x 1 จำนวน E_1 ตัวกรอง ที่เน้นการสกัดคุณลักษณะในแต่ละจุดของ Feature Map และ การดำเนินการคอนโวลูชันด้วยตัวกรองขนาด 3 x 3 จำนวน E_3 ตัวกรอง ที่ช่วยคำนวณหาลักษณะครอบคลุมบริเวณรอบจุดนั้นทำให้ได้ข้อมูลเชิงลึกมากขึ้น ซึ่งทั้ง 2 รูปแบบจะดำเนินการคอนโวลูชันขนาดกันแล้วนำข้อมูลมารวมกันเป็นผลลัพธ์ของแต่ละ Fire Module

การขยายข้อมูลใน Expand Layer จะมีจำนวนการคำนวณ 2ส่วน ได้แก่

- ตัวกรองขนาด 1 x 1 จำนวน E_1 ตัวกรอง จะมีจำนวนการคำนวณ เท่ากับ D_F x D_F x S x E_1 ครั้ง

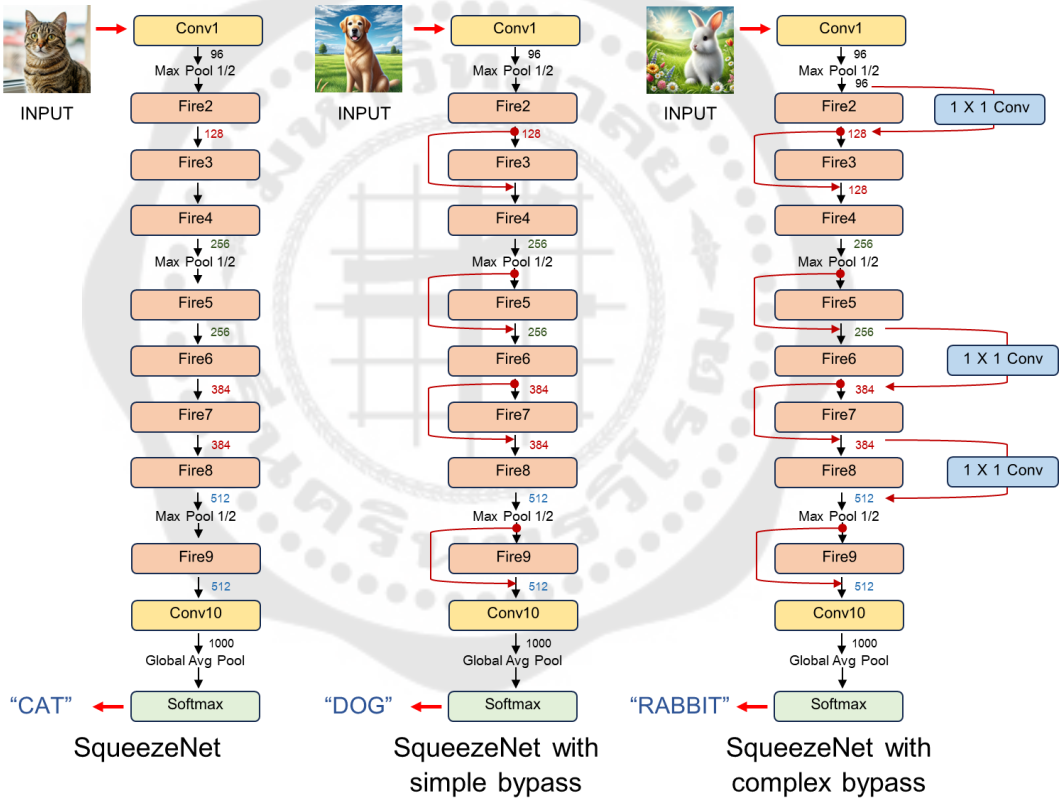
- ตัวกรองขนาด 1 x 1 จำนวน E_3 ตัวกรอง จะมีจำนวนการคำนวณ เท่ากับ D_F x D_F x S x E_3 x 9 ครั้ง

หลังจาก Expand Layer ขยายข้อมูลแล้ว ข้อมูลจะมีจำนวนช่องของ Feature Map มากขึ้น และพร้อมสำหรับการนำไปใช้ในการประมวลผลในชั้นถัดไป หรือนำไปใช้ในการจำแนกข้อมูล

หากเปรียบเทียบกับการดำเนินการคอนโวลูชันแบบปกติที่ต้องใช้ตัวกรองขนาดใหญ่ตั้งแต่ต้น เปรียบเทียบกับ Fire Module สามารถลดจำนวนพารามิเตอร์ได้อย่างมาก สมมติว่า

แบบจำลองปกติที่ใช้ตัวกรองขนาด 3 x 3 จำนวน F ตัวกรอง จำนวนการคำนวณทั้งหมดจะเท่ากับ D_F x D_F x C x F x 9 ในขณะที่ Fire Module ลดจำนวนการคำนวณเหลือเพียง D_F x D_F x C x S + D_F x D_F x S x (E_1 + (E_3 x 9)) การลดนี้ทำให้ Fire Module เหมาะสำหรับการสร้างแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ต้องการความเร็วในการประมวลผลและใช้หน่วยความจำน้อย

SqueezeNet เป็นแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ถูกออกแบบมาเพื่อลดขนาดของแบบจำลองโดยยังคงความแม่นยำสูง โครงสร้างของ SqueezeNet ในแต่ละแบบได้แก่ SqueezeNet SqueezeNet with simple bypass และ SqueezeNet with complex bypass แสดงในภาพประกอบ 35 (Iandola et al., 2016)



ภาพประกอบ 35 แสดงโครงสร้างของแบบจำลอง SqueezeNet แต่ละแบบ

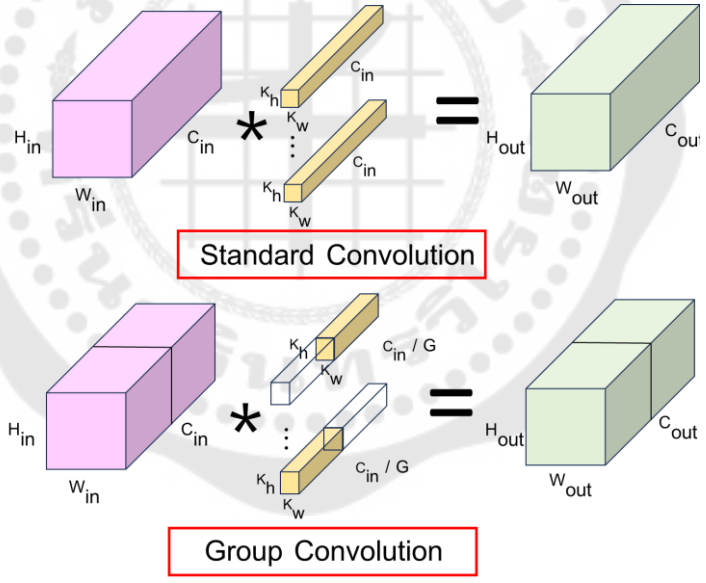
2.4.5.3 Group Convolution

Group Convolution (Zhang et al., 2018) เป็นเทคนิคในการดำเนินการแบบคอนโวลูชันที่แบ่งช่องสัญญาณของข้อมูลรับเข้าออกเป็นกลุ่มย่อย ๆ (Group) แล้วทำดำเนินการคอนโวลูชันแยกกันในแต่ละกลุ่ม สมมติว่าเรามี Input Feature Map ขนาด D_F x D_F x C (ความสูง และ

ความกว้าง เท่ากับ D_F และมีจำนวนช่องสัญญาณเท่ากับ C ช่อง) และต้องการใช้ตัวกรองขนาด K x K จำนวน F ตัวกรอง หากเราแบ่งเป็น G กลุ่ม แต่ละกลุ่มจะมี C/G ช่องสัญญาณต่อกลุ่ม และจะมีตัวกรอง F/G ตัวกรองต่อกลุ่ม ซึ่งในการดำเนินการคอนโวลูชันจะดำเนินการแยกแต่ละกลุ่มของช่องสัญญาณ กับแต่ละกลุ่มของตัวกรอง ดังภาพประกอบ 36

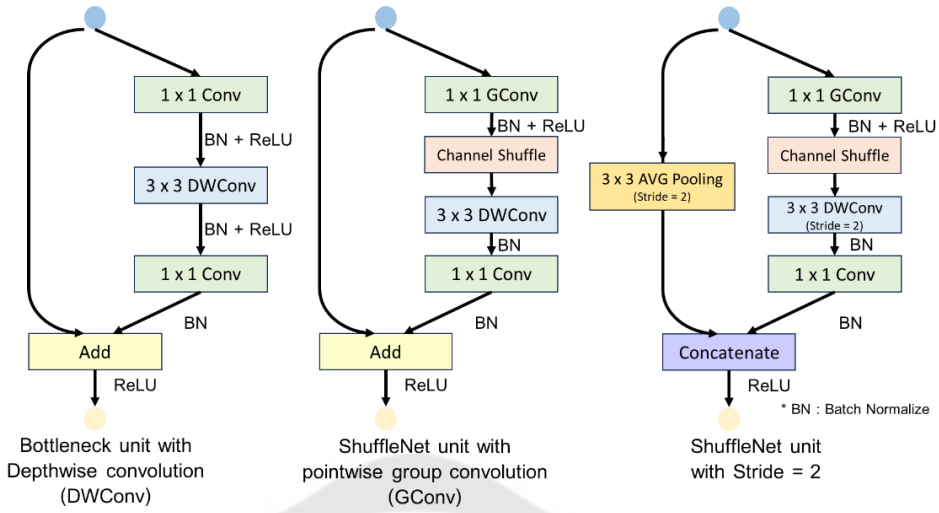
ในการเปรียบเทียบจำนวนการคำนวณระหว่างการดำเนินการคอนโวลูชันแบบปกติกับ Group Convolution เมื่อใช้การคอนโวลูชันแบบปกติจะเท่ากับ $D_F \times D_F \times C \times K^2 \times F$

และการคอนโวลูชันแบบ Group Convolution แต่ละกลุ่มจะมีจำนวนช่องสัญญาณเท่ากับ C/G ช่องสัญญาณ จำนวนการคำนวณในแต่ละกลุ่มจะลดลงเหลือ $D_F \times D_F \times (C / G) \times K^2 \times (F / G)$ และเมื่อรวมการคำนวณของทุกกลุ่ม จะมีจำนวนการคำนวณรวมเท่ากับ $D_F \times D_F \times C \times K^2 \times (F / G)$ ซึ่งจากสูตรนี้จะเห็นได้ว่าการแบ่งกลุ่มช่วยลดจำนวนการคำนวณได้ G เท่า เมื่อเปรียบเทียบกับจำนวนการคำนวณของการคอนโวลูชันแบบปกติ และ G มีค่าสูงขึ้น จำนวนการคำนวณต่อกลุ่มจะลดลง ซึ่งทำให้แบบจำลองทำงานได้เร็วขึ้นด้วย



ภาพประกอบ 36 แสดงภาพการเปรียบเทียบการคอนโวลูชันแบบปกติ กับ Group Convolution

ShuffleNet ใช้ Building Block ที่เรียกว่า ShuffleNet Unit ซึ่งประกอบด้วยสามส่วนหลักที่ใช้ 1x1 Group Convolution ซึ่งโครงสร้างนี้ช่วยลดจำนวนพารามิเตอร์และการคำนวณลงอย่างมาก ดังแสดงในภาพประกอบ 37



ภาพประกอบ 37 แสดงการใช้งาน Group Convolution ใน Builder Block ของแบบจำลอง ShuffleNet

2.5 งานวิจัยที่เกี่ยวข้อง

2.5.1 การใช้การเรียนรู้ของเครื่องและแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันในการจำแนกโรคพืช

Agarwal และคณะ (Agarwal et al., 2019) ได้ศึกษาเกี่ยวกับการใช้การเรียนรู้ของเครื่องในการจำแนกโรคในใบข้าวโพด โดยข้อมูลที่ใช้มาจากชุดข้อมูล PlantVillage ซึ่งประกอบด้วยภาพใบข้าวโพด ทั้งที่ปกติและที่เป็นโรค ทั้งหมด 3,852 ภาพ ทีมวิจัยได้แบ่งข้อมูลออกเป็นสองส่วน ส่วนหนึ่งสำหรับการฝึกฝนแบบจำลอง ประมาณ 80%และอีกส่วนหนึ่งสำหรับการทดสอบประมาณ 20% หลังจากการฝึกฝนแบบจำลอง

นักวิจัยใช้การเรียนรู้ของเครื่องแบบเดิม (SVM Naive Bayes Decision Tree Logistic Regression Random Forest และ KNN) โดยใช้กำหนดลักษณะพิเศษของรูปที่คนกำหนด เช่น สี รูปร่าง และลวดลาย นอกจากนี้ยังเทียบกับการใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน คือ VGG16 Inception V3 และแบบจำลองที่พัฒนาขึ้นเอง

ผลการทดลองแสดงให้เห็นว่า เมื่อเทียบกับการเรียนรู้ของเครื่องแบบเดิม พบว่าโครงข่ายประสาทเทียมแบบคอนโวลูชันประสิทธิภาพที่ดีกว่าในด้านความแม่นยำ โดยให้ผลค่า Accuracy ดังนี้ SVM ได้ 72.5% Naive Bayes 74.5% Decision Tree 72.5% Logistic Regression 88.5% Random Forest 92% และ KNN 90.5% และแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน VGG16 ให้ผล Accuracy 95% Inception ให้ผล 93.5% และแบบจำลองที่พัฒนาขึ้นสามารถจำแนกใบข้าวโพดที่มีโรคได้ด้วยความแม่นยำสูงถึง 94% ซึ่งถึงแม้ว่า VGG16 จะแม่นยำกว่านิด

หน่อย แต่ระบบใหม่นี้ใช้พื้นที่เก็บข้อมูลน้อยกว่าและทำงานเร็วกว่ามาก ซึ่งเหมาะกับการใช้งานจริงในสวนหรือไร่

Ashok และคณะ (Ashok et al., 2020) ได้นำเสนอการตรวจจับโรคในใบพืช โดยเฉพาะใบมะเขือเทศซึ่งเป็นพืชเศรษฐกิจที่สำคัญในหลายประเทศ ความเสียหายที่เกิดจากโรคพืชมีผลกระทบอย่างมากต่อผลผลิตและเศรษฐกิจ จึงเสนอการนำเทคโนโลยีปัญญาประดิษฐ์และการเรียนรู้เชิงลึกมาประยุกต์ใช้ในการจำแนกโรคใบพืช โดยใช้กระบวนการที่ครอบคลุมตั้งแต่การเตรียมข้อมูล การสกัดคุณลักษณะ การแยกส่วนภาพ และการจำแนกประเภทโรค

กระบวนการเริ่มจากการถ่ายภาพใบพืชที่มีทั้งส่วนที่เป็นโรคและส่วนที่ปกติ จากนั้นทำการเตรียมภาพด้วย Gaussian Filter เพื่อลดสัญญาณรบกวนและปรับปรุงคุณภาพภาพ จากนั้นใช้เทคนิค Discrete Wavelet Transform (DWT) และ Gray Level Co-occurrence Matrix (GLCM) เพื่อสกัดคุณลักษณะเชิงสถิติและเชิงพื้นที่ของภาพ กระบวนการแยกส่วนภาพ (Segmentation) ถูกนำมาใช้เพื่อตรวจหาขอบเขตของโรคในใบพืช โดยใช้การตั้งค่าช่วงความเข้มของสีและค่าจุดภาพที่คล้ายคลึงกัน

การจำแนกโรคใบมะเขือเทศใช้เครือข่ายประสาทเทียมแบบคอนโวลูชันซึ่งมีความสามารถในการวิเคราะห์ข้อมูลภาพในเชิงลึก CNN ถูกออกแบบให้สามารถปรับพารามิเตอร์ต่าง ๆ เพื่อลดความคลาดเคลื่อนในชุดข้อมูลฝึกอบรม ผลลัพธ์ของการจำแนกโรคถูกจัดเก็บในฐานข้อมูลเพื่อการตรวจวิเคราะห์ในอนาคต บทความยังเปรียบเทียบประสิทธิภาพของวิธีการที่นำเสนอ กับวิธีการเดิม เช่น AlexNet และ Artificial Neural Network (ANN) ผลการทดสอบแสดงให้เห็นว่า วิธีที่นำเสนอ (proposed method) มีความแม่นยำสูงกว่า โดยเฉพาะในด้านการตรวจจับโรคเป้าหมาย เช่น Target Spot และ Leaf Miner โดยมีความแม่นยำเฉลี่ย 98.12% ซึ่งถือเป็นความสำเร็จที่โดดเด่น ในขณะที่ AlexNet ให้ผลความแม่นยำ 95.75% และ Artificial Neural Network (ANN) ให้ผลความแม่นยำ 92.94% บทความยังเสนอแนวทางเพิ่มเติมสำหรับการวิจัยในอนาคต เช่น การใช้อัลกอริทึมที่มีประสิทธิภาพสูงขึ้น และการนำระบบนี้ไปปรับใช้ในสภาพแวดล้อมจริง เช่น การพัฒนาระบบการตรวจจับโรคแบบเรียลไทม์ (Real-Time) หรือการประยุกต์ใช้ในอุปกรณ์พกพา เพื่อให้เกษตรกรสามารถเข้าถึงเทคโนโลยีนี้ได้ง่ายขึ้น

2.5.2 การพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันขนาดเล็กสำหรับจำแนกโรคพืช

Waheed และคณะ (Waheed et al., 2020) ได้นำเสนอแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยการปรับปรุงแบบจำลอง DenseNet สำหรับจำแนกโรคในใบข้าวโพด โดยเปรียบเทียบกับ Pre-trained Model ได้แก่ VGG19Net XceptionNet EfficientNet และ

NASNet โดยใช้ข้อมูลภาพใบข้าวโพด 12,332 ภาพ แบ่งเป็น 4 ประเภท คือ ใบปกติ และโรค 3 ชนิด ได้แก่ ราสนิม จุดสีเทาบนใบ และใบไหม้แผลใหญ่ โดยแบ่งข้อมูลเป็นชุดฝึกฝน 90% และชุดทดสอบ 10% จากนั้นทำการเพิ่มจำนวนข้อมูล (Data Augmentation) ด้วยเทคนิคการหมุนภาพ ปรับขนาด และย้ายตำแหน่ง

ผลการทดลองพบว่าการปรับปรุงแบบจำลอง DenseNetที่นำเสนอสามารถจำแนกโรคได้ถูกต้อง 98.06% โดยใช้พารามิเตอร์เพียง 77,612 ตัว และใช้เวลาฝึกฝนเฉลี่ย 3 นาทีต่อรอบ (Epoch) เมื่อเทียบกับแบบจำลองอื่นๆ พบว่า EfficientNet ให้ความแม่นยำสูงสุดที่ 99.84% แต่ใช้พารามิเตอร์มากถึง 4.41 ล้านตัว และใช้เวลาฝึกฝน 4.88 นาทีต่อรอบ (Epoch) ส่วน VGG19Net XceptionNet และ NASNet ให้ความแม่นยำ 96.36% 93.52% และ 91.9% ตามลำดับ แต่ใช้พารามิเตอร์และเวลาฝึกฝนมากกว่าแบบจำลองลที่นำเสนออย่างมีนัยสำคัญ

ซึ่งจะเห็นได้ว่าแบบจำลอง DenseNet ที่ปรับปรุงสามารถจำแนกโรคใบข้าวโพดได้แม่นยำใกล้เคียงกับแบบจำลองลที่ซับซ้อนกว่า แต่ใช้ทรัพยากรน้อยกว่ามาก ทำให้เหมาะสมสำหรับการใช้งานจริงในอุปกรณ์ที่มีข้อจำกัดด้านประสิทธิภาพ เช่น โทรศัพท์มือถือ

Zeng และคณะ (Zeng et al., 2022) ได้นำเสนอแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเรียกว่า Lightweight Dense-Scale Network (LDSNet) เพื่อระบุโรคในใบข้าวโพดจากภาพถ่ายในสภาพแวดล้อมจริง โดยมีการใช้ข้อมูลจากภาพใบข้าวโพดที่เป็นโรคจากหลายแหล่ง ได้แก่ ข้อมูลจาก PlantVillage, มหาวิทยาลัยคอร์เนล และภาพที่ถ่ายจากสภาพแวดล้อมจริงโดยสมาร์ทโฟน โดยข้อมูลถูกแบ่งเป็น 70% สำหรับการฝึกฝนแบบจำลอง และ 30% สำหรับการทดสอบ การปรับแต่งข้อมูลได้ทำโดยการขยายชุดข้อมูลและปรับแต่งความสว่างและความคมชัด โดยใช้เทคนิค Contrast Limited Adaptive Histogram Equalization (CLAHE)

ผลการทดลองพบว่า LDSNet มีความแม่นยำในการระบุโรคใบข้าวโพดสูงถึง 95.4% ซึ่งดีกว่าแบบจำลองอื่น ๆ ที่นำมาเปรียบเทียบ เช่น AlexNet VGG16 VGG19 และ ResNet50 ที่มีความแม่นยำต่ำกว่า นอกจากนี้ LDSNet ยังมีขนาดเล็กกว่าแบบจำลองอื่น ๆ มาก โดยมีจำนวนพารามิเตอร์เพียง 590,844 พารามิเตอร์ เทียบกับ AlexNet ที่มีถึง 57 ล้านพารามิเตอร์

เมื่อเปรียบเทียบกับแบบจำลองขนาดเล็กอื่น ๆ เช่น MobileNet และ ShuffleNet พบว่า LDSNet ยังคงมีความแม่นยำสูงกว่า แม้จะมีขนาดเล็กกว่า นอกจากนี้ยังทดสอบการนำไปใช้งานจริงโดยติดตั้งบนโทรศัพท์มือถือ พบว่า LDSNet สามารถทำงานได้รวดเร็วและมีความแม่นยำสูง เหมาะสำหรับการใช้งานจริงในภาคสนาม สามารถนำไปใช้งานได้จริงบนอุปกรณ์พกพา ซึ่งจะเป็นประโยชน์อย่างมากสำหรับเกษตรกรในการตรวจหาโรคพืชได้อย่างรวดเร็วและแม่นยำ

Poornima Singh Thakur และคณะ (Thakur et al., 2023) ได้นำเสนอแบบจำลอง VGG-ICNN ซึ่งเป็นแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันแบบเบาที่ออกแบบมาเพื่อการระบุโรคในพืชโดยใช้ภาพใบพืช แบบจำลองนี้ผสมผสานชั้นคอนโวลูชันของ VGG16 และ Inception Block ของ GoogleNet เพื่อลดจำนวนพารามิเตอร์ให้อยู่ที่ประมาณ 6 ล้านพารามิเตอร์ ซึ่งน้อยกว่าแบบจำลองเชิงลึกแบบ Pre-trained อย่างมาก การทดลองใช้แบบจำลองกับชุดข้อมูลพืช 5 ชุดที่มีสภาวะแวดล้อมที่แตกต่างกัน พบว่า VGG-ICNN สามารถให้ความแม่นยำสูงถึง 99.16% บนชุดข้อมูล PlantVillage และ VGG-ICNN สามารถให้ความแม่นยำ 93.66% บนชุดข้อมูลใบข้าวโพด ซึ่งแบบจำลอง VGG-ICNN ถูกออกแบบมานี้เพื่อลดการใช้หน่วยความจำและพลังงาน ทำให้เหมาะสำหรับการใช้งานบนอุปกรณ์ที่มีทรัพยากรจำกัดเช่นสมาร์ทโฟน การวิจัยนี้แสดงให้เห็นถึงศักยภาพในการประยุกต์ใช้แบบจำลอง AI ในการตรวจจับโรคพืชสำหรับเกษตรอัจฉริยะ

2.5.3 การปรับขนาดของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันโดยใช้เทคนิคการทำ Quantization

Suharjito และคณะ (Suharjito et al., 2021) ได้นำเสนองานวิจัยที่เกี่ยวกับการพัฒนาแอปพลิเคชันบนสมาร์ทโฟนเพื่อจำแนกระดับความสุกของผลปาล์มน้ำมันโดยใช้การเรียนรู้เชิงลึกที่มีประสิทธิภาพสูงและสามารถทำงานบนอุปกรณ์ที่มีทรัพยากรคำนวณต่ำ การวิจัยนี้ใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันน้ำหนักเบา เช่น MobileNetV1 MobileNetV2 NASNet Mobile และ EfficientNetB0 ร่วมกับเทคนิค Transfer Learning จากชุดข้อมูล ImageNet โดยได้ Freeze การปรับค่าน้ำหนักของชั้นคอนโวลูชันบางส่วน และมีการปรับค่าน้ำหนักของของชั้นคอนโวลูชันในระหว่างการฝึกฝนแบบจำลองอีกส่วนหนึ่ง นอกจากนี้ยังได้มีการเสริมจำนวนของตัวอย่างด้วยการทำ Data Augmentation โดยใช้วิธี "9 - angle crop" ซึ่งเป็นนวัตกรรมการเพิ่มข้อมูลที่ช่วยลดการเกิด Overfitting และสามารถเพิ่มความแม่นยำของการจำแนกประเภทได้ ผลการทดลองแสดงให้เห็นว่า EfficientNetB0 มีความแม่นยำสูงสุดถึง 89.8% แม้ว่าจะใช้เวลามากกว่าแบบจำลองอื่นเล็กน้อย โดยใช้เวลาในการจำแนกรูปภาพ 96 มิลลิวินาทีต่อภาพ โดยมีความแม่นยำ 89.8% นอกจากนี้ แบบจำลองอื่นอย่าง MobileNetV1 และ MobileNetV2 ก็มีจุดเด่นในด้านความเร็วและความเหมาะสมกับการใช้งานที่ต้องการการตอบสนองแบบเรียลไทม์ โดยให้ความแม่นยำเท่ากัน คือ 81.1% และใช้เวลาในการคำนวณเพียง 18 มิลลิวินาที และ 24 มิลลิวินาที ตามลำดับ

แบบจำลองถูกนำไปพัฒนาเพิ่มเติมเพื่อใช้งานเป็นแอปพลิเคชันบนระบบปฏิบัติการ Android โดยใช้ไลบารี TensorFlow Lite ซึ่งสามารถรับภาพจากกล้องและแสดงผลการจำแนก

แบบเรียลไทม์ พบว่า แบบจำลอง EfficientNetB0 มีความแม่นยำลดลงเล็กน้อยเหลือ 89.3% เมื่อผ่านการ Quantization โดย Float16 ซึ่งใช้เวลาในการจำแนกรูปภาพเพียง 96 มิลลิวินาทีต่อภาพ ขณะที่ แบบจำลองอื่นอย่าง MobileNetV1 ให้ความแม่นยำเท่าเดิม คือ 81.1% ใช้เวลาในการคำนวณเพียง 39 มิลลิวินาที และ MobileNetV2 ก็มีความแม่นยำลดลงเหลือ 77.6% โดยใช้เวลาในการคำนวณเพียง 41 มิลลิวินาที

งานวิจัยนี้แสดงให้เห็นถึงศักยภาพในการนำแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันน้ำหนักเบาที่ผ่านการทำ Quantization มาใช้ในงานเกษตรกรรมโดยพัฒนาเป็นแอปพลิเคชันบนระบบปฏิบัติการ Android มีประสิทธิภาพการทำงานสูง และเป็นแนวทางในการพัฒนาเทคโนโลยีการจำแนกประเภทที่เหมาะสมสำหรับอุปกรณ์พกพาในอนาคตได้

Rabindra Nath Nandi และคณะ (Nandi et al., 2022) นำเสนอการพัฒนาและปรับขนาดแบบจำลองการตรวจจับโรคในผลไม้และใบฝรั่งโดยใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการใช้งานบนอุปกรณ์ Edge Devices เช่น สมาร์ทโฟนที่มีข้อจำกัดด้านทรัพยากร แบบจำลองที่ใช้ในการทดลองประกอบด้วย VGG-16 GoogleNet ResNet-16 MobileNet-V2 และ EfficientNet-b2 โดยมีการใช้เทคนิคการทำ Quantization แบบ Float16 และ Dynamic Range ซึ่งช่วยลดขนาดแบบจำลองได้อย่างมีประสิทธิภาพ GoogleNet ซึ่งเป็นแบบจำลองที่ผ่านการลดขนาดโดยใช้ Dynamic Range Quantization มีลด 75 % จาก 22.6 MB เหลือ 0.143 MB แต่ยังคงให้ความแม่นยำสูงถึง 97% ทำให้เป็นตัวเลือกที่เหมาะสมสำหรับการใช้งานที่ต้องการขนาดเล็กและประหยัดพื้นที่ ส่วน EfficientNet แม้ว่ามีขนาดที่ใหญ่กว่าอยู่ที่ 4.5 MB หลังจาการทำ Dynamic Range Quantization แต่ให้ความแม่นยำสูงสุดที่ 99% เหมาะสำหรับการใช้งานที่ต้องการผลลัพธ์ที่มีคุณภาพสูง การย่อขนาดแบบจำลองนี้ทำให้สามารถนำแบบจำลองการเรียนรู้เชิงลึกไปใช้งานในอุปกรณ์พกพาที่มีทรัพยากรจำกัดได้อย่างมีประสิทธิภาพ นอกจากนี้ บทความยังได้วางแผนการพัฒนาในอนาคตที่จะขยายการศึกษาไปยังชุดข้อมูลที่ซับซ้อนขึ้นเพื่อตอบโจทย์การใช้งานจริงในภาคเกษตร

2.5.4 การพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยใช้เทคนิค Knowledge Distillation สำหรับการจำแนกโรคพืช

Ali Ghofrani และคณะ (Ghofrani & Mahdian Toroghi, 2022) ได้นำเสนอวิธีการใช้การ Knowledge Distillation ในการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อจำแนกโรคพืช โดยใช้ชุดข้อมูล PlantVillage ซึ่งมุ่งเน้นการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันขนาดเล็กที่สามารถใช้งานบนอุปกรณ์พกพาที่มีข้อจำกัดด้านประสิทธิภาพ โดยมีการใช้แบบจำลองขนาดใหญ่ที่ถูกพัฒนาบนเซิร์ฟเวอร์เพื่อ Teacher Model ในการถ่ายทอด

ความรู้ให้กับแบบจำลองขนาดเล็กที่ทำงานบนอุปกรณ์ปลายทาง ซึ่งทำหน้าที่เป็น Student Model ในทำ Knowledge Distillation นี้ แบบจำลองขนาดใหญ่ ได้แก่ EfficientNet และ Xception ที่ถูกฝึกฝนและทดสอบด้วยชุดข้อมูลทดสอบได้ความแม่นยำ 99.73% และ 99.65% ตามลำดับ และใช้แบบจำลอง Vanilla Tiny MobileNet ที่พัฒนาขึ้น เป็น Student Model โดยมีความแม่นยำ 95.46% และถ่ายทอดประสิทธิภาพของแบบจำลอง โดยใช้เทคนิค Knowledge Distillation ซึ่งได้รับการปรับปรุงให้มีความแม่นยำใกล้เคียงกับแบบจำลองใหญ่ โดยไม่เพิ่มจำนวนพารามิเตอร์จนเกินขีดจำกัด การทดลองนี้ใช้ชุดข้อมูล PlantVillage ซึ่งประกอบด้วยภาพใบพืชหลากหลายชนิดที่มีโรคต่างๆ ส่งผลให้แบบจำลองขนาดเล็กหลังจากการถ่ายทอดความรู้มีความแม่นยำเพิ่มขึ้นถึง 2.12% เมื่อเปรียบเทียบกับการใช้ MobileNet ขนาดเล็กแบบปกติ นอกจากนี้ บทความยังได้อธิบายถึงเทคนิคการปรับข้อมูลด้วยการเพิ่มภาพเสริมที่ไม่ทำลายลักษณะการแยกแยะโรคที่สำคัญ เช่น การหมุนภาพหรือการปรับความสว่างและคอนทราสต์ ซึ่งช่วยให้แบบจำลองมีความสามารถในการแยกแยะโรคได้แม่นยำยิ่งขึ้น บทความนี้จึงแสดงถึงศักยภาพในการนำโครงสร้างการสกัดความรู้มาประยุกต์ใช้เพื่อพัฒนาแบบจำลอง AI ขนาดเล็กที่มีประสิทธิภาพสูงสำหรับการตรวจโรคพืชบนอุปกรณ์พกพา ทำให้เกษตรกรสามารถนำไปใช้ในฟาร์มได้อย่างสะดวก

จากการศึกษาเกี่ยวกับทฤษฎีที่เกี่ยวข้องกับการจำแนกโรคใบข้าวโพดในบทที่ ๒ โดยได้ศึกษาการประยุกต์ใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อใช้ในการจำแนกโรคพืชจากภาพถ่าย

2.6 บทสรุปจากการศึกษาทฤษฎีที่เกี่ยวข้องและทบทวนวรรณกรรม

จากการศึกษาเกี่ยวกับทฤษฎีที่เกี่ยวข้องกับการจำแนกโรคใบข้าวโพดในบทที่ ๒ โดยได้ศึกษาการประยุกต์ใช้การเรียนรู้ของเครื่องและการใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อใช้ในการจำแนกโรคพืชจากภาพถ่าย พบว่า การใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันมีประสิทธิภาพและมีความแม่นยำในการทำนายผลโรคพืชจากภาพถ่ายใบพืชที่ดีกว่าใช้การเรียนรู้ของเครื่องแบบดั้งเดิม แต่การนำแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันไปใช้งานจริงในภาคสนามแบบอุปกรณ์พกยังมีข้อจำกัดในเรื่องของขนาดของแบบจำลองที่มีขนาดใหญ่ และใช้ทรัพยากรในการประมวลผลที่สูง

การใช้เทคนิคในการการปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคพืชจะช่วยลดขนาดของแบบจำลองให้เหมาะสมกับทรัพยากรของอุปกรณ์ แต่ยังคงรักษาประสิทธิภาพในการทำนายผลของแบบจำลองในการจำแนกโรคพืชจากภาพถ่ายไว้ให้มีความแม่นยำที่สูงอยู่ ซึ่งจากการศึกษาทฤษฎีที่เกี่ยวข้องกับการลดขนาดของแบบจำลองโครงข่ายประสาทเทียมแบบคอน

โวลูชัน และจากการศึกษางานวิจัยที่เกี่ยวข้องที่ พบว่า เทคนิคที่เหมาะสมและมีประสิทธิภาพในการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อใช้ในการจำแนกโรคพืชจากภาพถ่ายที่สามารถนำมาประยุกต์ใช้กับงานวิจัยการปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน มี 3 วิธี ได้แก่ การเลือกใช้แบบจำลองแบบเบา การทำ Knowledge Distillation และการทำ Quantization แบบ Post-training Quantization

ในบทถัดไป จะกล่าวถึงกระบวนการ และวิธีการดำเนินการวิจัย ซึ่งประกอบด้วย การออกแบบกรอบแนวความคิดในการวิจัยครั้งนี้ ชุดข้อมูลภาพถ่ายใบข้าวโพดที่ใช้ในการวิจัย การฝึกฝนแบบจำลอง การปรับปรุงแบบจำลองให้เหมาะสมสำหรับการนำไปใช้งานบนอุปกรณ์พกพา และการประเมินผล และเปรียบเทียบประสิทธิภาพแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคพืช

บทที่ 3
กระบวนการ และวิธีการดำเนินการวิจัย

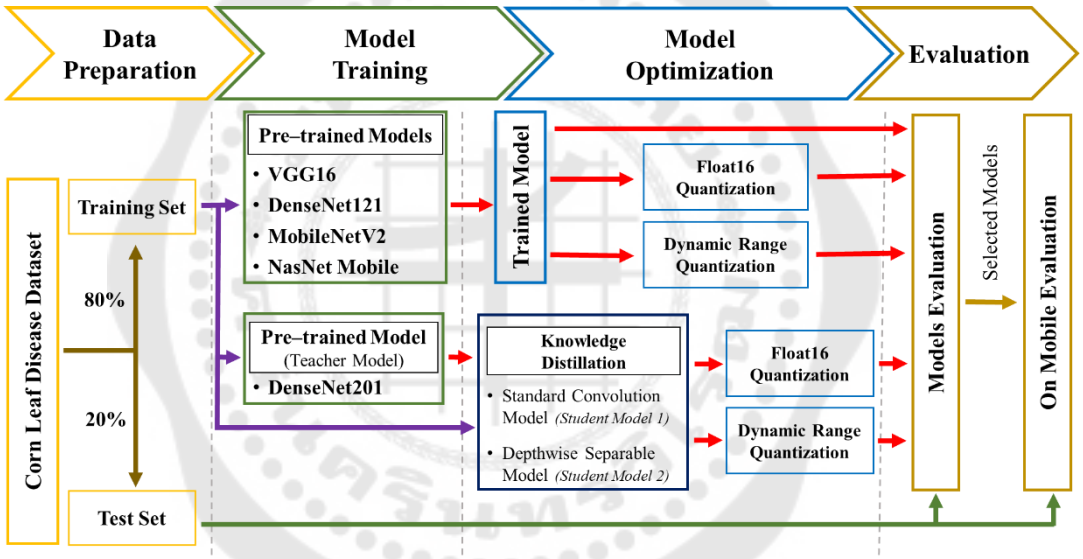
ในการวิจัยครั้งนี้ ผู้วิจัยได้ดำเนินการตามขั้นตอนดังนี้ flowchart

1. การกำหนดประชากรและกลุ่มตัวอย่าง

2. การฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการจำแนกโรคใบข้าวโพด

3. การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพด

4. การประเมินผล และเปรียบเทียบประสิทธิภาพแบบจำลองหลังจากผ่านกระบวนการปรับปรุงการปรับปรุงแบบจำลอง



ภาพประกอบ 38 แสดงกรอบแนวความคิดในการดำเนินการศึกษาวิจัย

กรอบแนวความคิดในการศึกษาวิจัยประกอบด้วยการการเตรียมข้อมูลโดยจะแยกชุดข้อมูลฝึกฝนและชุดข้อมูลทดสอบ จากนั้นจะนำชุดข้อมูลฝึกฝนไปทำการฝึกฝนแบบจำลอง แบบ Pre-trained แล้วนำแบบจำลองที่ได้จากการฝึกฝนไปผ่านการกระบวนการปรับปรุงแบบจำลอง (Model Optimization Techniques) ทำการทดสอบและประเมินผลแบบจำลองด้วยชุดข้อมูลทดสอบ ใน 3 ประเด็น คือ ความแม่นยำ ขนาดของแบบจำลอง และเวลาในการประมวลผล แล้วเลือกแบบจำลองที่มีขนาดเล็กที่สุด 3 แบบจำลอง และแบบจำลองที่มีความแม่นยำสูงสุด 2 แบบจำลอง ไปทดสอบการทำนายผลบนโทรศัพท์มือถือ เพื่อทดสอบประสิทธิภาพด้านความแม่นยำ และเวลาที่ใช้ในการประมวลผล ซึ่งแสดงดังภาพประกอบ 38

3.1 การกำหนดประชากรและกลุ่มตัวอย่าง

ในการวิจัยครั้งนี้ใช้ชุดข้อมูลจาก PlantVillage Dataset ซึ่งหล่งข้อมูลเปิดที่ใช้ในการศึกษาและพัฒนาเทคโนโลยีเพื่อการวินิจฉัยโรคพืชโดยใช้ภาพถ่ายใบพืช ข้อมูลชุดนี้ถูกรวบรวมโดย David P. Hughes จากมหาวิทยาลัยเพนน์สเตท (Penn State University) ประเทศสหรัฐอเมริกาและ Marcel Salathé จาก สถาบันเทคโนโลยีแห่งสหพันธ์สวิส โลซาน (École polytechnique fédérale de Lausanne: EPFL) ประเทศสวิสเซอร์แลนด์ ในปี 2015 โดยได้รับการสนับสนุนจากนักพืชศาสตร์และนักพยาธิพืชในการวินิจฉัยและระบุโรคที่ตรวจพบบนภาพถ่ายของใบพืช ทำให้ข้อมูลมีความถูกต้องและครบถ้วน ซึ่งมีเป้าหมายเพื่อช่วยเกษตรกรทั่วโลกในการวินิจฉัยโรคได้อย่างแม่นยำและทันท่วงที ผ่านการใช้เครื่องมือวิเคราะห์ภาพบนสมาร์ทโฟน หรือคอมพิวเตอร์

ข้อมูล PlantVillage ประกอบด้วยภาพถ่ายใบพืช ทั้งภาพใบที่ปกติปราศจากโรคพืชและภาพใบพืชที่เป็นโรค จำนวน 54,309 ภาพ โดยครอบคลุม 14 ชนิดพืช ได้แก่ แอปเปิล บลูเบอร์รี เชอร์รี่ ข้าวโพด องุ่น ส้ม พีช พริกหวาน มันฝรั่งราสเบอร์รี่ ถั่วเหลือง สควอช สตรอว์เบอร์รี และมะเขือเทศ นอกจากนี้ ยังมีภาพของโรคพืชชนิดต่าง ๆ ได้แก่ โรคเชื้อรา (Fungus) 17 ชนิด โรคแบคทีเรีย (Bacteria) 4 ชนิด โรครา (Mold : oomycete) 2 ชนิด โรคไวรัส (Virus) 2 ชนิด และโรคที่เกิดจากไร (Mite) 1 ชนิด รวมทั้งสิ้น 38 หมวดหมู่ (Class) ในการศึกษาวิจัยในครั้งนี้ได้เลือกใช้เฉพาะชุดข้อมูลใบข้าวโพดมาทำการศึกษาวิจัย

3.1.1 ชุดข้อมูลใบข้าวโพดของชุดข้อมูล PlantVillage

ชุดข้อมูลใบข้าวโพดในชุดข้อมูล PlantVillage ประกอบด้วยภาพถ่ายใบข้าวโพด รวมทั้งสิ้น 3,852 ภาพ ทั้งใบปกติที่ปราศจากโรคและใบข้าวโพดที่มีโรค จำนวน 3 หมวดหมู่ ได้แก่

1) โรคใบไหม้ทางตอนเหนือ (Northern Leaf Blight) ซึ่งเกิดจากเชื้อรา Exserohilum turcicum

2) โรคราสนิมข้าวโพด (Corn Rust) จากเชื้อรา Puccinia sorghi

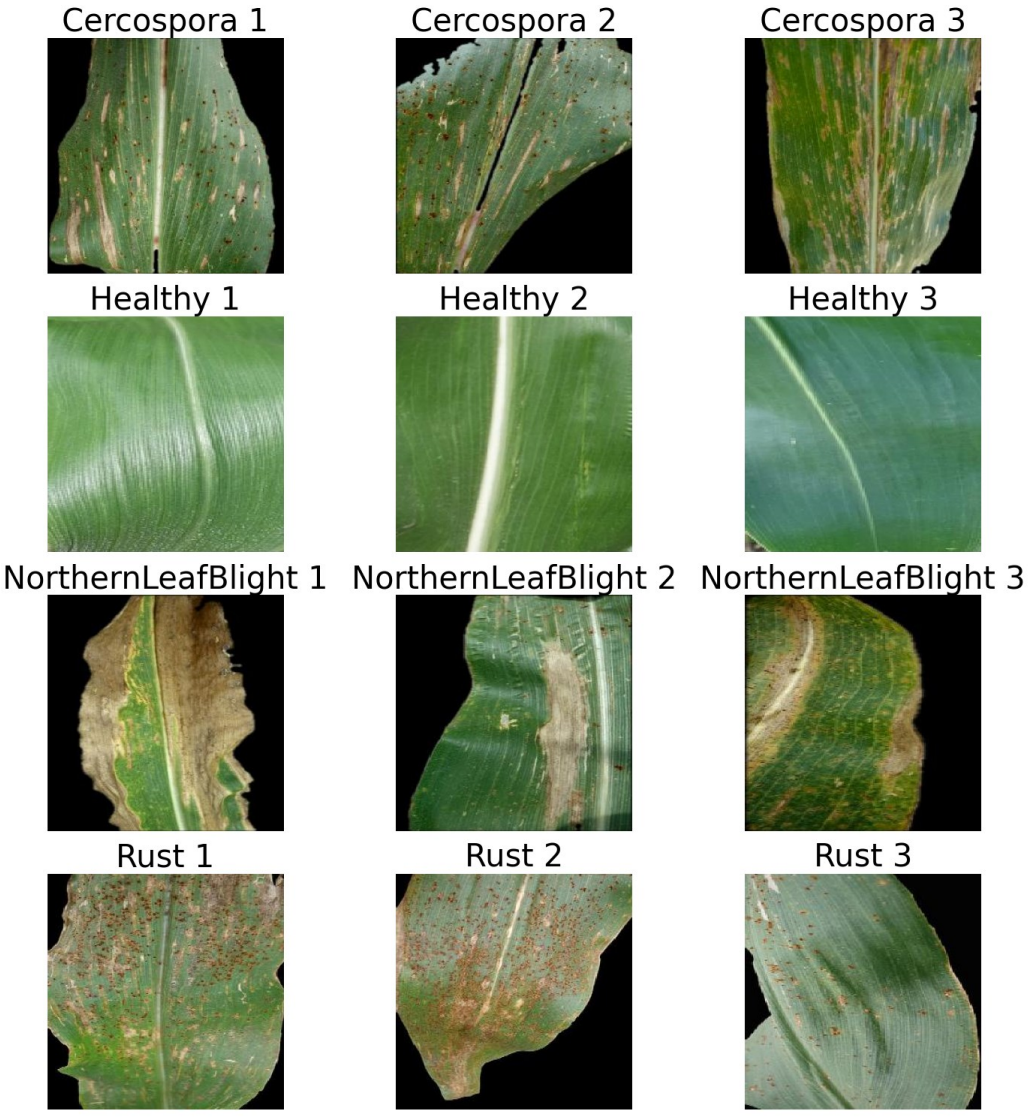
3) โรคใบไหม้จากเชื้อ Cercospora (Cercospora zeae-maydis)

ซึ่งรายละเอียดจำนวนของภาพถ่ายของใบข้าวโพดในแต่ละหมวดหมู่ แสดงในตาราง 7

ตาราง 7 แสดงจำนวนภาพของใบข้าวโพดในแต่ละโรค

ชนิดของโรค	จำนวนภาพ (ภาพ)
ใบปกติ	1,162
โรคใบไหม้ทางตอนเหนือ	985
โรคราสนิมข้าวโพด	1,192
โรคใบไหม้จากเชื้อ Cercospora	513
รวม	3,852

และตัวอย่างภาพของใบข้าวโพดในแต่ละหมวดหมู่ทั้งใบปกติที่ปราศจากโรค และใบข้าวโพดที่มีโรค แสดงดังภาพประกอบ 39



ภาพประกอบ 39 แสดงตัวอย่างภาพใบข้าวโพดที่ในชุดข้อมูลที่นำมาใช้ในการทดลอง

ซึ่งภาพแต่ละภาพในชุดข้อมูลจะถูกกำหนดให้อยู่ในหมวดหมู่ใดหมวดหมู่หนึ่งเท่านั้น (Single Class) โดยมีการกำหนดให้ Label ให้กับแต่ละหมวดหมู่ ได้แก่

1) Healthy สำหรับใบข้าวโพดปกติ

2) Cercospora สำหรับใบข้าวโพดที่มีโรคใบไหม้จากเชื้อ Cercospora

3) NorthernLeafBlight สำหรับใบข้าวโพดที่มีโรคใบไหม้ทางตอนเหนือ

4) Rust สำหรับใบข้าวโพดที่มีโรคราสนิมข้าวโพด

3.1.2 การแบ่งชุดข้อมูลฝึกฝน และการแบ่งชุดข้อมูลทดสอบ

ในการเตรียมข้อมูลสำหรับการจำแนกโรคใบข้าวโพดด้วยแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันในงานวิจัยนี้ได้ใช้ชุดข้อมูลใบข้าวโพด จาก PlantVillage Dataset เพื่อใช้ในการศึกษาพัฒนาระบบจำแนกโรคใบข้าวโพดด้วยแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ชุดข้อมูลใบข้าวโพดได้ถูกออกเป็นสองชุดหลัก คือ ชุดข้อมูลฝึกฝน (Training Set) และชุดข้อมูลทดสอบ (Test Set) ในอัตราส่วน 80:20 โดยที่ข้อมูลจำนวน 80% จะถูกใช้ในการฝึกฝนแบบจำลองเพื่อเรียนรู้คุณสมบัติที่สำคัญของข้อมูล ในขณะที่ข้อมูลอีก 20% ที่เหลือจะถูกใช้ในการประเมินผลประสิทธิภาพการทำงานของแบบจำลองหลังจากที่ผ่านการฝึกฝนเสร็จสิ้นแล้ว

ตาราง 8 แสดงจำนวนของตัวอย่างภาพในแต่ละหมวดหมู่

หมวดหมู่	จำนวนภาพ (ภาพ)	
	Training Set	Test Set
Healthy	930	232
Cercospora	410	103
NorthernLeafBlight	788	197
Rust	954	238
รวม	3,082	770

หลังจากแบ่งชุดข้อมูลใบข้าวโพดทั้ง 4 หมวดหมู่ ออกเป็น ชุดข้อมูลฝึกฝน และชุดข้อมูลทดสอบ ในอัตราส่วน 80:20 มีรายละเอียดของจำนวนตัวอย่างในแต่ละหมวดหมู่ตามตาราง 8 โดยแต่ละหมวดหมู่ประกอบด้วยจำนวนข้อมูลตัวอย่างที่แตกต่างกัน ดังนี้

1) Healthy

ประกอบด้วยภาพใบข้าวโพดที่ปกติ ปราศจากโรค มีจำนวนตัวอย่างทั้งหมด 1,162 ตัวอย่าง ซึ่งแบ่งเป็น 930 ตัวอย่าง ในชุดฝึกฝนและ 232 ตัวอย่าง ในชุดทดสอบ ข้อมูลคลาสนี้มี

จำนวนมากที่สุด เนื่องจากลักษณะใบข้าวโพดที่ปกติมีความหลากหลายและเป็นสิ่งสำคัญที่แบบจำลองต้องเรียนรู้เพื่อตรวจจับความผิดปกติของโรคได้อย่างแม่นยำ

2) Cercospora

ประกอบด้วยภาพใบข้าวโพดที่เป็นโรคใบไหม้จากเชื้อ Cercospora เป็นมีจำนวนตัวอย่างทั้งหมด 513 ตัวอย่าง ซึ่งแบ่งเป็น 410 ตัวอย่าง ในชุดฝึกฝนและ 103 ตัวอย่าง ในชุดทดสอบการแบ่งข้อมูลในคลาสนี้มีวัตถุประสงค์เพื่อให้แบบจำลองสามารถเรียนรู้ลักษณะเฉพาะของโรคใบไหม้จากเชื้อ Cercospora ได้อย่างมีประสิทธิภาพ

3) NorthernLeafBlight

ประกอบด้วยภาพใบข้าวโพดที่เป็นโรคโรคใบไหม้ มีจำนวนตัวอย่างทั้งหมด 985 ตัวอย่าง ซึ่งแบ่งเป็น 788 ตัวอย่าง ในชุดฝึกฝนและ 197 ตัวอย่าง ในชุดทดสอบ ข้อมูลในคลาสนี้แสดงถึงลักษณะเฉพาะของโรค Northern Leaf Blight ซึ่งเป็นโรคที่พบได้บ่อยในใบข้าวโพด ดังนั้นการมีข้อมูลฝึกฝนจำนวนมากเพียงพอจะช่วยให้แบบจำลองสามารถแยกแยะโรคนี้จากโรคอื่นได้อย่างชัดเจน

4) Rust

ประกอบด้วยภาพใบข้าวโพดที่เป็นโรคราสนิมข้าวโพด มีจำนวนตัวอย่างทั้งหมด 1,192 ตัวอย่าง แบ่งเป็น 954 ตัวอย่าง ในชุดฝึกฝน และ 238 ตัวอย่าง ในชุดทดสอบ คลาสนี้มีข้อมูลมากพอที่จะทำให้แบบจำลองเรียนรู้ลักษณะของโรค Rust ซึ่งเป็นโรคที่มีการแพร่ระบาดบ่อยในใบข้าวโพด โดยการจัดสรรข้อมูลให้มีความสมดุลช่วยลดโอกาสที่แบบจำลองจะเกิดความเอนเอียงไปที่คลาสใดคลาสหนึ่ง

จากตาราง 8 พบว่าจำนวนตัวอย่างในแต่ละหมวดหมู่มีจำนวนไม่เท่ากัน (Imbalance) ซึ่งสามารถแก้ไขโดยการใช้ Class Weight ในกระบวนการฝึกฝนแบบจำลอง เพื่อเพิ่มความแม่นยำของแบบจำลองให้มีประสิทธิภาพดีขึ้น ซึ่ง Class Weight สามารถคำนวณได้จากสมการ (24)

$$Weight\ for\ Class\ [i] = \frac{Total\ Samples}{Number\ of\ Classes \times Samples\ in\ Class\ [i]} \qquad (24)$$

โดยที่

$Weight\ for\ Class\ [i]$	คือ Class Weight ของ Class ที่ i
$Total\ Samples$	คือ จำนวนตัวอย่างทั้งหมด
$Number\ of\ Classes$	คือ จำนวน Class ทั้งหมด
$Samples\ in\ Class\ [i]$	คือ จำนวนตัวอย่างใน Class ที่ i

ซึ่งเมื่อคำนวณ Class Weight ของ Training Set แล้วจะได้ผลลัพธ์ตามตาราง 9

ตาราง 9 แสดง Class Weight ของตัวอย่างแต่ละ Class ในชุดข้อมูลฝึกฝน

Number of Classes	Class	Samples in Class	Weight for Class
1	Healthy	930	0.8285
2	Cercospora	410	1.8793
3	NorthernLeafBlight	788	0.9778
4	Rust	954	0.8077

Class Weight ที่ได้จะนำไปใช้กำหนดเป็นพารามิเตอร์ในการฝึกฝนแบบจำลองผ่าน TensorFlow Framework เพื่อเพิ่มประสิทธิภาพในการฝึกฝนแบบจำลองให้มีความแม่นยำมากขึ้น

3.2 การฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน สำหรับการจำแนกโรคใบข้าวโพด

3.2.1 TensorFlow Framework

ในการศึกษาครั้งนี้ได้ใช้ TensorFlow ซึ่งเป็น Framework ที่ถูกพัฒนาโดย Google สำหรับใช้เป็นเครื่องมือในการสร้างแบบจำลองการเรียนรู้ของเครื่องโดยเฉพาะในการทำงานกับข้อมูลที่เป็นภาพ เสียง และข้อมูลอื่น ๆ ที่ซับซ้อน เช่น ข้อความ ซึ่ง TensorFlow ใช้กระบวนการพัฒนาแบบจำลองโครงข่ายประสาทเทียม

TensorFlow ทำงานโดยการสร้างกราฟของการคำนวณ (Computational Graph) ซึ่งแต่ละโหนดในกราฟจะเป็นการคำนวณทางคณิตศาสตร์หรือฟังก์ชันเฉพาะต่าง ๆ ที่ใช้ในการประมวลผลข้อมูล โดยในกราฟนี้ ข้อมูลจะถูกส่งผ่านโหนดต่าง ๆ และทำให้การคำนวณแต่ละขั้นตอนเกิดขึ้น โดย TensorFlow ถูกออกแบบให้สามารถทำงานได้ทั้งบน CPU และ GPU ซึ่งช่วยเพิ่มความเร็วในการประมวลผล โดยเฉพาะเมื่อทำงานกับข้อมูลที่มีขนาดใหญ่หรือแบบจำลองที่ซับซ้อน

สิ่งที่ทำให้ TensorFlow โดดเด่นคือความยืดหยุ่นและความสามารถในการขยายตัว นักพัฒนาสามารถสร้างแบบจำลองที่ซับซ้อนได้โดยเริ่มต้นจากโครงสร้างพื้นฐาน เช่น โครงข่ายประสาทเทียมหรืออัลกอริทึมการเรียนรู้ของเครื่องอื่น ๆ นอกจากนี้ TensorFlow ยังรองรับการขยายตัวให้ทำงานได้บนเครื่องที่มีหลายหน่วยประมวลผลหรือแม้แต่กลุ่ม (Cluster) ของคอมพิวเตอร์ขนาดใหญ่ ทำให้สามารถทำงานกับข้อมูลที่มีขนาดมหาศาลได้อย่างมีประสิทธิภาพ

TensorFlow เป็นเครื่องมือที่ทรงพลังในการพัฒนาแบบจำลองการเรียนรู้ของเครื่องและ AI มีความยืดหยุ่นในการใช้งานและสามารถขยายตัวให้รองรับการทำงานบนแพลตฟอร์มต่าง ๆ ทั้ง CPU GPU และอุปกรณ์เคลื่อนที่ อีกทั้งยังมีชุมชนผู้ใช้งาน (Community) ที่ช่วยพัฒนาต่อเนื่อง ทำให้เป็นหนึ่งในเครื่องมือที่น่าสนใจสำหรับนักพัฒนาและนักวิจัย

TensorFlow ยังเป็นเครื่องมือที่มีการพัฒนาและเพิ่มคุณลักษณะใหม่ ๆ อย่างต่อเนื่อง จากการสนับสนุนโดย Google และชุมชนทั่วโลก มีการออกเวอร์ชันใหม่ที่ทำให้การทำงานมีประสิทธิภาพมากขึ้นและรองรับฮาร์ดแวร์ใหม่ ๆ นอกจากนี้ยังมีการพัฒนา TensorFlow Lite สำหรับการทำงานบนอุปกรณ์ที่มีข้อจำกัดเรื่องพลังประมวลผล เช่น โทรศัพท์มือถือ และ TensorFlow.js สำหรับการทำงานบนเว็บบราวเซอร์

TensorFlow สามารถใช้งานได้ทั้งผู้เริ่มต้นและผู้ที่มีประสบการณ์ในการเขียนโปรแกรม ด้วยการใช้งานง่ายผ่าน API นอกจากนี้ TensorFlow ยังมีฟังก์ชันที่ถูกพัฒนาขึ้นเพื่อรองรับงานหลายประเภท เช่น การประมวลผลภาพ การวิเคราะห์ข้อมูล และการประมวลผลภาษาธรรมชาติ โดยมีไลบรารีที่ชื่อว่า “Keras” ที่ทำหน้าที่เป็น API ระดับสูงเพื่อช่วยให้นักพัฒนาสามารถสร้างแบบจำลองได้ง่ายยิ่งขึ้นด้วยคำสั่งเพียงไม่กี่บรรทัด

Keras เป็นไลบรารีที่ใช้ในการสร้างและฝึกฝนแบบจำลองการเรียนรู้เชิงลึก ซึ่งเป็นที่นิยมในวงการวิจัยและพัฒนาเนื่องจากมีความง่ายในการใช้งานและความยืดหยุ่นในการปรับแต่งแบบจำลอง Keras ถูกพัฒนาขึ้นเพื่อให้ผู้ใช้งานสามารถสร้างแบบจำลองได้อย่างรวดเร็ว โดยมี API ที่เข้าใจง่ายและช่วยลดความซับซ้อนในการสร้างแบบจำลอง โดย Keras รองรับการทำงานกับ TensorFlow เป็น Backend ซึ่งทำให้สามารถใช้พลังการประมวลผลของ TensorFlow ในการฝึกฝนแบบจำลองที่ซับซ้อนได้อย่างมีประสิทธิภาพ

แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเป็นหนึ่งในสถาปัตยกรรมที่ Keras รองรับ ซึ่งเหมาะสำหรับการประมวลผลภาพ และ Keras ยังสนับสนุนการใช้งานการทำงานแบบขนาน (Parallel Computing) ซึ่งทำให้สามารถฝึกฝนแบบจำลองบน GPU ได้ โดยการใช้ Keras ร่วมกับ TensorFlow จะช่วยให้การทำงานกับแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเป็นไปอย่างรวดเร็ว โดยเฉพาะเมื่อทำงานกับข้อมูลภาพขนาดใหญ่

Keras มีความสามารถในการใช้ Pre-trained Model ที่มีการฝึกฝนมาก่อนแล้วโดยใช้ชุดข้อมูลขนาดใหญ่ ซึ่งช่วยประหยัดเวลาและทรัพยากรในการพัฒนาแบบจำลองการเรียนรู้เชิงลึก โดย Pre-trained Model เป็นแบบจำลองที่ได้รับการเรียนรู้คุณลักษณะที่สำคัญจากชุดข้อมูลที่มีความหลากหลาย ซึ่งทำให้มีความแม่นยำสูงในการจำแนกประเภทข้อมูลต่าง ๆ โดยเฉพาะอย่าง

ยิ่งในกรณีที่มีชุดข้อมูลตัวอย่างสำหรับการฝึกฝนแบบจำลองน้อยหรือมีลักษณะคล้ายกับชุดข้อมูลที่ใช้ในการฝึกฝนแบบจำลองก่อนหน้านี้ การใช้เทคนิค Transfer Learning ทำให้สามารถปรับแต่งแบบจำลอง Pre-trained Model ที่มีอยู่ให้เหมาะสมกับชุดข้อมูลใหม่ได้อย่างมีประสิทธิภาพ ตัวอย่างแบบจำลอง Pre-trained Model ที่มีอยู่ใน Keras เช่น VGG16 ResNet50 DenseNet InceptionV3 MobileNet หรือ NASNet Mobile ที่มีการออกแบบมาเพื่อเพิ่มประสิทธิภาพในการจำแนกประเภทภาพ แบบจำลองเหล่านี้สามารถนำไปใช้งานได้ทันทีใน Keras โดยมี API ที่เข้าใจง่าย สามารถเรียกใช้งานแบบจำลอง VGG16 พร้อมค่าพารามิเตอร์ที่ได้การฝึกฝนจากชุดข้อมูล ImageNet ทำให้ผู้ใช้สามารถเริ่มต้นการพัฒนาแบบจำลองได้เร็วขึ้น และสามารถปรับปรุงแบบจำลองให้มีความแม่นยำสูงไปใช้งานได้อย่างมีประสิทธิภาพ เป็นทางเลือกที่ดีสำหรับนักพัฒนาที่ต้องการสร้างแบบจำลองการจำแนกประเภทภาพภายในระยะเวลาที่จำกัดและต้องการผลลัพธ์ที่แม่นยำสูงโดยไม่ต้องฝึกฝนแบบจำลองที่ซับซ้อนใหม่ตั้งแต่เริ่มต้น

ในการลดขนาดแบบจำลอง Keras มีเครื่องมือและเทคนิคที่สามารถนำมาใช้ได้ เช่น การทำ Quantization และการตัดแต่งแบบจำลองซึ่งจะช่วยให้แบบจำลองที่สร้างขึ้นมีขนาดเล็กลงและสามารถประมวลผลได้เร็วขึ้น

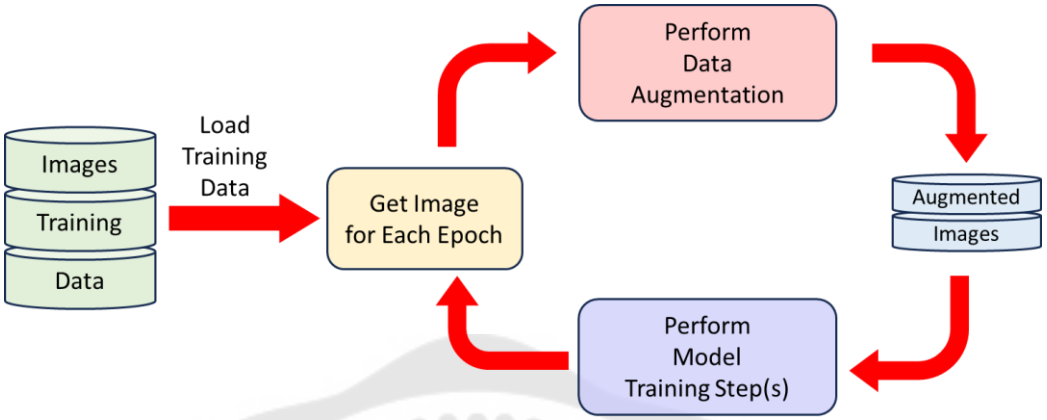
การใช้ Keras ใน TensorFlow จึงเป็นเครื่องมือที่มีประสิทธิภาพสำหรับการสร้างแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันเพื่อตรวจจับโรคในใบข้าวโพด โดยสามารถใช้หลักการทางคณิตศาสตร์และเทคนิคต่าง ๆ เพื่อปรับปรุงประสิทธิภาพของแบบจำลองและลดขนาดของแบบจำลองให้เหมาะสมกับการใช้งานจริงได้อย่างมีประสิทธิภาพ

3.2.2 การทำ On-the-Fly Data Augmentation โดยใช้ ImageDataGenerator

การสร้างแบบจำลองการเรียนรู้เชิงลึก โดยเฉพาะอย่างยิ่งสร้างแบบจำลองในการจำแนกภาพ นั้น การมีชุดข้อมูลที่หลากหลายและเพียงพอเป็นสิ่งสำคัญ เพื่อให้แบบจำลองเรียนรู้รูปแบบมุมมองที่แตกต่างกันของภาพ และเรียนรู้คุณลักษณะที่ซับซ้อนของภาพที่หลากหลาย ให้แบบจำลองสามารถจำแนกภาพได้อย่างมีประสิทธิภาพ ซึ่งการจัดเตรียมรวบรวมข้อมูลภาพตัวอย่างสำหรับฝึกฝนแบบจำลองในปริมาณมากและมีคุณภาพสูงนั้นมักใช้เวลาและทรัพยากรสูง จึงมักจะใช้การเพิ่มจำนวนของตัวอย่างภาพ และการเพิ่มรูปแบบของภาพโดยการทำ Data Augmentation จากชุดข้อมูลของภาพตัวอย่างที่มีอยู่เดิมเพื่อแก้ปัญหาดังกล่าว

การทำ Augmentation เป็นกระบวนการที่ช่วยสร้างข้อมูลตัวอย่างเสมือน (Synthetic Data) จากข้อมูลต้นฉบับที่มีอยู่เดิม โดยการสุ่ม (Randomization) โดยการประยุกต์ใช้เทคนิคต่าง ๆ เช่น การหมุนภาพ (Rotation) การตัดภาพ (Cropping) การเลื่อนตำแหน่ง (Shifting) การพลิก

ภาพ (Flipping) การปรับความสว่าง (Brightness) และการขยายภาพ (Zoom) เป็นต้น ซึ่งเทคนิคเหล่านี้จะช่วยเพิ่มความหลากหลายให้กับชุดข้อมูล โดยไม่จำเป็นต้องใช้ภาพใหม่



ภาพประกอบ 40 แสดงการทำ On-the-fly Data Augmentation

ในการศึกษาวิจัยครั้งนี้จะใช้รูปแบบการทำ On-the-Fly Data Augmentation (Cerqueira et al., 2024) ผ่านการเรียกใช้งานคลาส ImageDataGenerator ใน Keras ซึ่งเป็นเครื่องมือที่มีประสิทธิภาพในการจัดการข้อมูลภาพ โดยคลาส ImageDataGenerator จะสร้างภาพใหม่โดยการสุ่มพารามิเตอร์สำหรับการทำ Augmentation ที่ต้องการในแต่ละครั้งที่มีการโหลดข้อมูลเข้ามา ในระหว่างที่แบบจำลองกำลังเรียนรู้จากชุดภาพตัวอย่างในแบทช์ (Batch) นั้น ๆ ImageDataGenerator จะทำการประมวลผลภาพตามพารามิเตอร์ของการ Augmentation ที่กำหนดไว้และส่งภาพที่ถูกเปลี่ยนแปลงไปยังแบบจำลองในระหว่างการฝึกฝน ข้อดีของการทำ Data Augmentation แบบ On-the-Fly ยังช่วยประหยัดพื้นที่ในการจัดเก็บข้อมูล เนื่องจากเป็นการสร้างข้อมูลภาพขึ้นมาชั่วคราวขณะทำการฝึกฝนแบบจำลองเท่านั้น ไม่ได้สร้างและจัดเก็บภาพใหม่ที่เกิดจากการ Augmentation ไว้ในพื้นที่จัดเก็บข้อมูล (Storage) การทำงานของหลักการ On-the-Fly Data Augmentation แสดงในภาพประกอบ 40

ค่าพารามิเตอร์ของการทำ Augmentation ผ่านการใช้งาน ImageDataGenerator ที่เปลี่ยนแปลงไประหว่างการฝึกฝนแบบจำลอง จะทำให้ข้อมูลภาพตัวอย่างที่ใช้ในการฝึกฝนแบบจำลองถูกจัดการให้มีลักษณะที่แตกต่างกันออกไปในแต่ละรอบการฝึก (Epoch) ทำให้ได้ข้อมูลภาพตัวอย่างเสมือนที่หลากหลายมากขึ้น การทำเช่นนี้จะช่วยลดความเสี่ยงในการ Overfitting ซึ่งเป็นปัญหาที่เกิดขึ้นเมื่อแบบจำลองเรียนรู้จากข้อมูลเฉพาะเจาะจงมากเกินไป จนไม่สามารถทำงานได้ดีเมื่อเจอกับข้อมูลใหม่ที่ไม่เคยพบมาก่อน

ในการศึกษาวิจัยครั้งนี้ได้ใช้การทำ Data Augmentation กับเฉพาะชุดข้อมูลฝึกฝน โดยใช้เทคนิคและช่วงในการปรับค่าพารามิเตอร์ ดังตาราง 10

ตาราง 10 แสดงการทำ Data Augmentation ในระหว่างการฝึกฝนแบบจำลอง

เทคนิค	รายละเอียด	ค่าช่วงของพารามิเตอร์	คำสั่ง
การหมุนภาพ (Rotation)	หมุนภาพแบบสุ่มในช่วง 20 องศา ช่วยให้แบบจำลองเรียนรู้รูปแบบที่หลากหลายขึ้นในภาพที่อาจมีการเอียง	0 - 20 องศา	rotation_range=20
เลื่อนภาพในแนวนอน (Width Shifting)	เลื่อนภาพในแนวนอนโดยขยับซ้ายหรือขวาได้มากถึง 20% ของความกว้างภาพ	20% ของความกว้างภาพ	width_shift_range=0.2
เลื่อนภาพในแนวนอน (Height Shifting)	เลื่อนภาพในแนวตั้งโดยขยับขึ้นหรือลงได้มากถึง 20% ของความสูงภาพ	20% ของความสูงภาพ	height_shift_range=0.2
การบิดหรือเฉือนภาพ (Shearing)	ปรับภาพให้มีการบิดหรือเฉือนด้วยค่าความเข้มข้น 0.2 ซึ่งจะช่วยเพิ่มความแตกต่างของมุมมองในภาพ	ค่าความเข้มข้น 0.2	shear_range=0.2
การขยายภาพ (Zoom)	ซูมเข้าหรือออกจากภาพแบบสุ่มได้มากถึง 20% ช่วยให้แบบจำลองเข้าใจภาพที่มีขนาดหรือระยะที่แตกต่างกัน	0 – 20%	zoom_range=0.2
การพลิกภาพ (Flipping)	พลิกภาพในแนวนอนแบบสุ่ม ช่วยให้แบบจำลองเข้าใจภาพที่อาจกลับด้านในบางกรณี	True	horizontal_flip=True

3.2.3 แบบจำลอง Pre-trained ที่ใช้การศึกษาวิจัย

การใช้แบบจำลอง Pre-trained ใน Keras เป็นแนวทางที่ได้รับความนิยมอย่างสูงในวงการวิจัยและการพัฒนาแบบจำลองที่เกี่ยวข้องกับการประมวลผลภาพ เนื่องจากแบบจำลอง

Pre-trained เป็นแบบจำลองที่ได้รับการฝึกฝนมาแล้วบนชุดข้อมูลขนาดใหญ่ โดยเฉพาะชุดข้อมูล ImageNet ที่มีภาพจำนวนมากกว่า 14 ล้านภาพและจัดหมวดหมู่ของภาพเป็น 1,000 หมวดหมู่ ซึ่งช่วยให้แบบจำลองมีความสามารถในการเรียนรู้คุณลักษณะต่างๆ ของภาพได้ดี

การใช้แบบจำลอง Pre-trained ใน Keras ช่วยลดระยะเวลาในการฝึกฝนแบบจำลอง ทำให้การนำแบบจำลองไปประยุกต์ใช้งานเป็นไปได้อย่างรวดเร็วและมีประสิทธิภาพ การนำค่าพารามิเตอร์ของแบบจำลองที่ได้จากการฝึกแบบจำลองที่มีความลึกและซับซ้อนมาใช้ในการจำแนกประเภทภาพเป็นแนวทางที่มีประโยชน์อย่างมาก

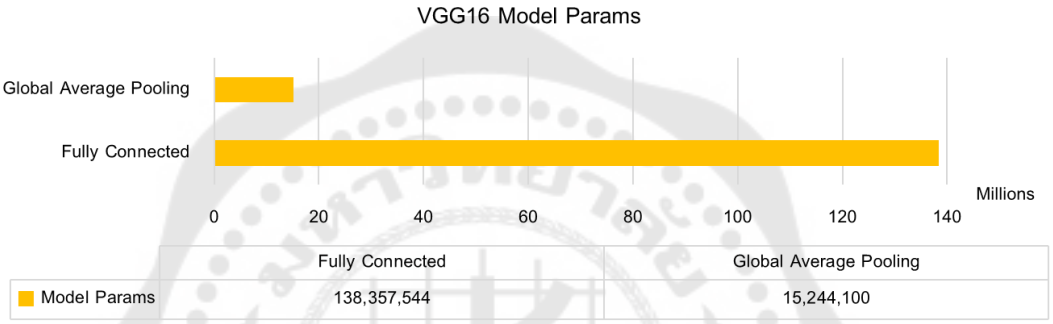
แบบจำลอง Pre-trained ของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันใน Keras ที่เลือกนำมาใช้ในการศึกษาวิจัยครั้งนี้ ได้แก่ VGG16 DenseNet121 MobileNetV2, และ NASNet Mobile ซึ่งแบบจำลองทั้งหมดประกอบกันด้วยโครงสร้างของโครงข่ายประสาทเทียมแบบคอนโวลูชันที่แตกต่างกันออกและได้รับการพัฒนาอย่างต่อเนื่อง เพื่อเพิ่มประสิทธิภาพในการตรวจจับและจำแนกประเภทของภาพ

3.2.3.1 แบบจำลอง VGG16

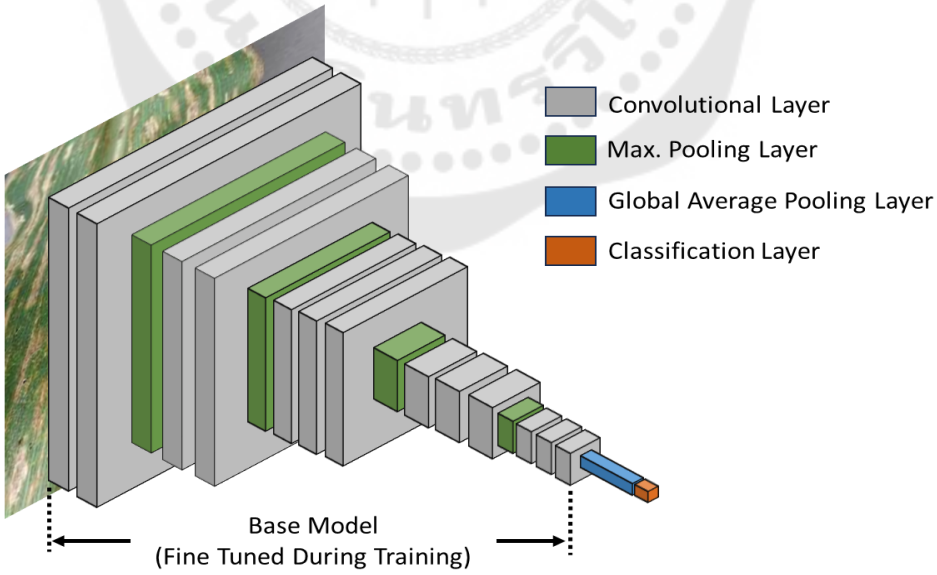
แบบจำลอง VGG16 เป็นสถาปัตยกรรมโครงข่ายประสาทเทียมแบบคอนโวลูชันที่เรียบง่ายและเน้นการใช้ Convolutional Layer มีตัวกรองขนาด 3x3 เรียงต่อกัน ซึ่งช่วยให้แบบจำลองสามารถจับคุณลักษณะที่ละเอียดและซับซ้อนได้ โดยประกอบด้วยชั้น Convolutional Layer ถึง 13 ชั้น และ Fully Connected Layer อีก 3 ชั้น ซึ่งทำให้มีพารามิเตอร์จำนวนมากถึง 138 ล้านตัว ส่งผลให้ต้องใช้พลังงานและทรัพยากรที่มากในการประมวลผล หากพิจารณาจำนวนของพารามิเตอร์ในแต่ละชั้นของโครงสร้างจะพบว่า พารามิเตอร์ส่วนใหญ่อยู่ในชั้น Fully Connected Layer ซึ่งถ้าหากว่าสามารถปรับปรุงโครงสร้างของชั้น Fully Connected Layer ของแบบจำลอง VGG16 ให้มีความซับซ้อนน้อยลงก็จะสามารถทำให้มีขนาดของแบบจำลองมีขนาดเล็กลงได้

การใช้แบบจำลอง Pre-trained VGG16 จึงต้องทำการปรับปรุงโครงสร้างของแบบจำลองโดยการตัดชั้น Fully Connected Layer ออกแล้วใช้ Global Average Pooling Layer (GAP) แทน ซึ่งการใช้ GAP แทน Fully Connected Layer มีข้อดีที่สำคัญในหลายประการ ประการแรก คือ การลดจำนวนพารามิเตอร์ของแบบจำลองลงอย่างมาก ซึ่ง GAP ทำหน้าที่ลดความซับซ้อนของข้อมูลด้วยการนำ Feature Map ทั้งหมดในชั้นสุดท้ายมาหาค่าเฉลี่ยแทนการใช้โครงสร้างแบบ Fully Connected ที่เชื่อมต่อทุกนิวรอนเข้าด้วยกัน ด้วยวิธีนี้แบบจำลองสามารถลดขนาดได้โดยที่ยังคงความสามารถในการจำแนกประเภทได้เป็นอย่างดี นอกจากนี้ การลดจำนวนพารามิเตอร์ยังช่วยลดปัญหาการเกิด Overfitting ได้อย่างมีประสิทธิภาพ ทำให้แบบจำลองที่ฝึกด้วยข้อมูล

จำนวนน้อยยังคงรักษาความแม่นยำในการประมวลผลได้มากขึ้น ประโยชน์อีกประการของ GAP คือการช่วยให้แบบจำลองมีโครงสร้างที่เรียบง่ายและสามารถทำงานได้อย่างรวดเร็วและประหยัดพลังงาน ซึ่งเหมาะสำหรับการนำไปใช้งานในระบบที่ต้องการประสิทธิภาพสูงแต่มีทรัพยากรจำกัด เช่น อุปกรณ์เคลื่อนที่หรืออุปกรณ์ IoT การใช้ GAP แทน Fully Connected Layer จึงเป็นการปรับแต่งที่มีความคุ้มค่า ช่วยให้แบบจำลองสามารถลดขนาดและเพิ่มประสิทธิภาพในการประมวลผลโดยที่ยังคงความแม่นยำในการจำแนกประเภทข้อมูล จำนวนของพารามิเตอร์ที่ลดลงแสดงในภาพประกอบ 41



ภาพประกอบ 41 แสดงภาพประกอบการเปรียบเทียบจำนวนพารามิเตอร์ระหว่างการใช้ Fully Connected และการใช้ Global Average Pooling



ภาพประกอบ 42 แสดงโครงสร้างของแบบจำลอง VGG16 หลังจากการปรับปรุงโครงสร้าง

หลังจากที่ปรับโครงสร้างแบบจำลองโดยเปลี่ยนชั้น Fully Connected เป็น GAP แล้ว ขั้นตอนต่อไปคือการเพิ่มชั้น Classification Layer ที่มีจำนวน 4 นิวรอน (เนื่องจากมี 4 คลาส) โดยใช้ฟังก์ชัน Softmax ในการประมวลผลเพื่อคำนวณค่าความน่าจะเป็นของแต่ละคลาส ฟังก์ชัน Softmax ทำหน้าที่เปลี่ยนผลลัพธ์ที่ได้จาก GAP ให้กลายเป็นค่าความน่าจะเป็นที่รวมกันได้ค่าเท่ากับ 1 ซึ่งเหมาะสำหรับการจำแนกประเภทของภาพ จะได้โครงสร้างหลังจากปรับแต่งโครงสร้างแล้วดังภาพประกอบ 42 ในการฝึกฝนแบบจำลอง Pre-trained VGG16 จะใช้ค่าพารามิเตอร์ของ Weights และ Biases ของ ImageNet และจะทำการปรับค่าพารามิเตอร์ทุกชั้นของโครงข่ายในแบบจำลอง

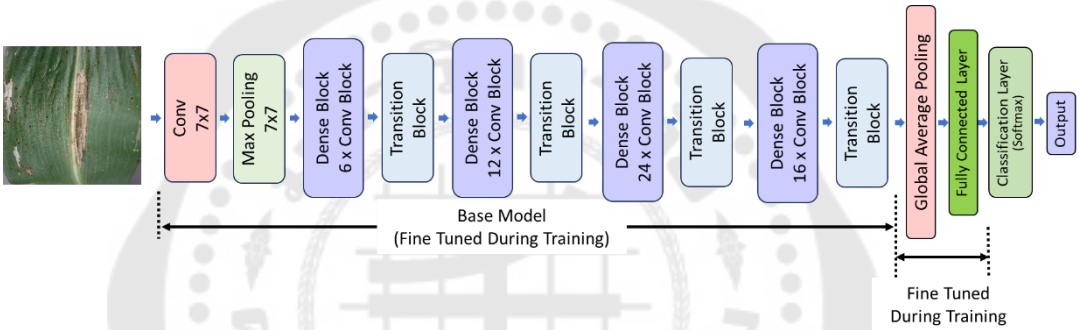
3.2.3.2 แบบจำลอง DenseNet121

DenseNet121 เป็นโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ออกแบบมาเพื่อลดปัญหาการสูญเสียข้อมูลเมื่อชั้นเพิ่มขึ้น โครงสร้างของ DenseNet121 ประกอบด้วยชั้นคอนโวลูชันหลายชั้น ที่ประกอบกันเรียกว่า"Dense Blocks" มีลักษณะเฉพาะ คือ การเชื่อมต่อแบบ “Dense Connectivity” ที่ช่วยให้ชั้นเชื่อมโยงกับทุกชั้นก่อนหน้า ซึ่งแตกต่างจากโครงข่ายประสาทเทียมทั่วไปที่การเชื่อมโยงจำกัดเพียงชั้นติดกันเท่านั้น แต่ละ Dense Block ประกอบด้วยชั้นคอนโวลูชันที่มีการส่งข้อมูลไปยังชั้นถัดไปอย่างสมบูรณ์ ทำให้การไหลของข้อมูลและ Gradients เป็นไปอย่างมีประสิทธิภาพ นอกจากนี้ DenseNet121 ใช้จำนวนพารามิเตอร์ที่น้อยลง เพราะชั้นใหม่ไม่ต้องเรียนรู้คุณลักษณะของ Feature Map ใหม่ทั้งหมด แต่ใช้ข้อมูล Feature Map ที่ส่งมาจากชั้นก่อนหน้าช่วยลดความซ้ำซ้อนของ Feature Map แบบจำลองนี้ยังประกอบด้วย Transition Layers ที่ช่วยลดขนาดของข้อมูลโดยการทำ Batch Normalization และ Down-sampling เช่น max pooling เพื่อคงความจำเป็นของพลังคำนวณต่ำ แต่ยังสามารถรับรู้คุณลักษณะที่ซับซ้อนได้ ข้อดีของ DenseNet121 ไม่เพียงแต่ลดการใช้พารามิเตอร์ แต่ยังลดปัญหาการสูญเสีย Gradients ทำให้แบบจำลองสามารถฝึกได้อย่างมีเสถียรภาพมากขึ้นในโครงสร้างที่ลึก เหมาะสำหรับการทำงานต่างๆ เช่น การจำแนกภาพ การตรวจจับวัตถุ และการวิเคราะห์ข้อมูลภาพที่มีรายละเอียดซับซ้อน ทั้งนี้ DenseNet121 ยังมีประสิทธิภาพสูง เมื่อใช้เป็นฐานข้อมูล Pre-trained model ทำให้สามารถนำไปปรับใช้ในงานอื่นๆ ที่ต้องการการปรับแต่งเฉพาะได้อย่างมีประสิทธิภาพ

หลังจากชั้น GAP ข้อมูลจะถูกส่งไปยังชั้น Classification Layer ใหม่ที่มีจำนวน 4 นิวรอน สำหรับการจำแนกประเภท ซึ่งจำนวนนี้สอดคล้องกับจำนวนคลาสที่ต้องการจำแนก โดยใช้ฟังก์ชัน Softmax เป็นตัวคำนวณความน่าจะเป็นของแต่ละคลาส Softmax จะทำให้ค่าที่ได้จาก GAP กลายเป็นค่าความน่าจะเป็นของแต่ละคลาส โดยการคำนวณความน่าจะเป็นของแต่ละคลาสจะ

ทำให้ได้ผลลัพธ์ที่รวมกันเท่ากับ 1 ซึ่งช่วยให้แบบจำลองสามารถตัดสินใจว่าภาพควรอยู่ในคลาสใดได้อย่างแม่นยำ การเปลี่ยนชั้น FC เป็น GAP และต่อด้วยชั้น Classification Layer ที่มี Softmax สำหรับ 4 หมวดหมู่ ตามลักษณะของใบข้าวโพดที่นำมาใช้ในการศึกษาวิจัย

การปรับโครงสร้างของแบบจำลอง DenseNet121 จะช่วยให้แบบจำลองมีประสิทธิภาพมากขึ้นในการจำแนกประเภท โดยยังคงรักษาความสามารถในการสกัดคุณลักษณะที่ซับซ้อนของภาพได้ดี การปรับแต่งนี้ทำให้แบบจำลองมีความเหมาะสมในการใช้งานบนอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร เช่น อุปกรณ์เคลื่อนที่หรือ IoT ซึ่งต้องการการประมวลผลที่เร็ว ประหยัดพลังงาน และมีขนาดแบบจำลองที่เล็ก โครงสร้างของแบบจำลอง DenseNet121 ที่ได้ปรับปรุงโครงสร้างแสดงในภาพประกอบ 43



ภาพประกอบ 43 แสดงโครงสร้างของแบบจำลอง DenseNet121 หลังจากการปรับปรุงโครงสร้าง

ในการฝึกฝนแบบจำลอง DenseNet121 ใช้ค่าพารามิเตอร์ของแบบจำลองที่ผ่านการฝึกจาก ImageNet ร่วมกับการทำ Fine-tuning ทุกชั้นของ DenseNet121 เป็นเทคนิคที่ช่วยให้แบบจำลองมีประสิทธิภาพสูงขึ้น แบบจำลองเริ่มต้นด้วยคุณลักษณะที่ได้รับการฝึกจากข้อมูลขนาดใหญ่ ทำให้สามารถแยกแยะลักษณะภาพได้ดี การปรับค่าพารามิเตอร์ในทุกชั้นช่วยให้แบบจำลองสามารถปรับค่าให้เข้ากับลักษณะเฉพาะของข้อมูลใหม่ได้มากขึ้น ขั้นตอนนี้ช่วยให้ DenseNet121 เรียนรู้ฟีเจอร์เฉพาะของโรคใบข้าวโพดได้แม่นยำและคงประสิทธิภาพสูง

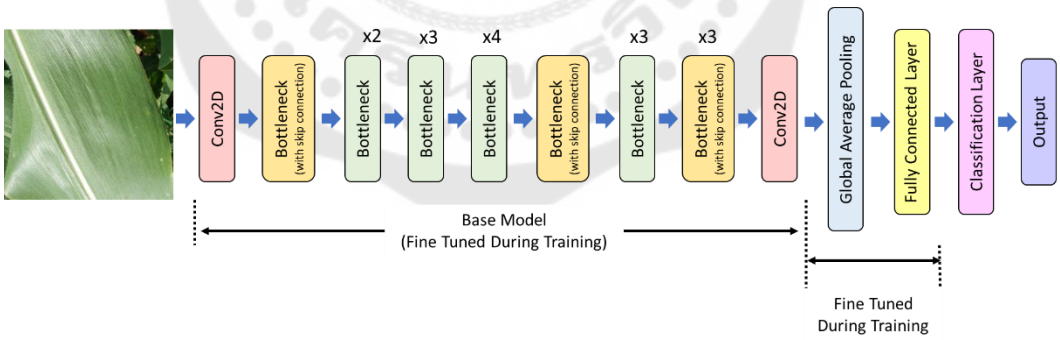
3.2.3.3 **แบบจำลอง** MobileNetV2

ในการศึกษาวิจัยครั้งนี้ต้องการศึกษาเกี่ยวกับประสิทธิภาพของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ใช้โครงสร้างของ Lightweight Model Design ที่ทำให้แบบจำลองมีขนาดเล็ก และสามารถประมวลผลได้อย่างรวดเร็ว ซึ่งแบบจำลอง MobileNetV2 เป็นแบบจำลอง Pre-trained สำหรับงานด้านการประมวลผลภาพที่มีประสิทธิภาพ ที่ถูกออกแบบมาสำหรับ

อุปกรณ์ที่มีข้อจำกัดด้านการประมวลผล เช่น โทรศัพท์มือถือ หรือ อุปกรณ์ Edge Devices เป็นต้น โดยมีโครงสร้างที่เน้นประสิทธิภาพสูงและใช้ทรัพยากรต่ำ แบบจำลองนี้ใช้เทคนิค Depthwise Separable Convolution เพื่อช่วยลดจำนวนพารามิเตอร์และการคำนวณให้เหลือน้อยที่สุด และ Inverted Residual Block ใช้เทคนิคการขยายและย่อข้อมูลในแต่ละบล็อกเพื่อประหยัดการคำนวณ แบบจำลอง MobileNetV2 จึงเหมาะอย่างยิ่งสำหรับการประยุกต์ใช้งานที่ต้องการทั้งความเร็วและความแม่นยำ

ในการปรับโครงสร้างแบบจำลอง Pre-trained ของ MobileNetV2 เพื่อนำไปใช้ในงานจำแนกภาพในการตรวจจับโรคจากใบข้าวโพด ขั้นตอนสำคัญ คือ การเพิ่มชั้น GAP ให้กับ Base model เพื่อทำหน้าที่รวบรวมและหาค่าเฉลี่ยคุณสมบัติจากชั้นคอนโวลูชันทั้งหมด โดยเฉลี่ยค่าตามความลึกของข้อมูลภาพ ทำให้แบบจำลองมีพารามิเตอร์น้อยลงและลดโอกาสเกิด Overfitting ซึ่งเป็นปัญหาที่มักพบในการใช้งานชั้น Fully Connected ที่มีพารามิเตอร์จำนวนมาก

เมื่อปรับ GAP layer เสร็จแล้ว ขั้นตอนต่อไปคือเพิ่มชั้น Classification layer สำหรับจำนวนหมวดหมู่ที่ต้องการ ซึ่งในที่นี้คือ 4 หมวดหมู่ โดยใช้ฟังก์ชัน SoftMax ในการคำนวณค่าของผลลัพธ์การจำแนกคลาสในรูปของความน่าจะเป็น ซึ่งฟังก์ชัน SoftMax จะแปลงผลลัพธ์จากชั้น GAP ให้เป็นค่าความน่าจะเป็นที่รวมกันได้เท่ากับ 1 ซึ่งเหมาะสำหรับงานจำแนก เช่น การตรวจจับโรคพืชที่มีหลายประเภท โครงสร้างของแบบจำลอง MobileNetV2 ที่ได้ปรับปรุงโครงสร้างแสดงในภาพประกอบ 44



ภาพประกอบ 44 แสดงโครงสร้างของแบบจำลอง MovileNetV2 หลังจากการปรับปรุงโครงสร้าง

ในการนำแบบจำลอง MobileNetV2 มาใช้งานในการศึกษาวิจัยครั้งนี้ เริ่มต้นด้วยการใช้ค่าพารามิเตอร์ของแบบจำลองที่ได้รับการฝึกฝนมาแล้วจากชุดข้อมูล ImageNet ซึ่งจะช่วยให้แบบจำลองมีพื้นฐานในการจดจำลักษณะต่างๆ ของภาพ ไม่ว่าจะเป็นรูปร่าง ขอบเขต และรูปแบบ

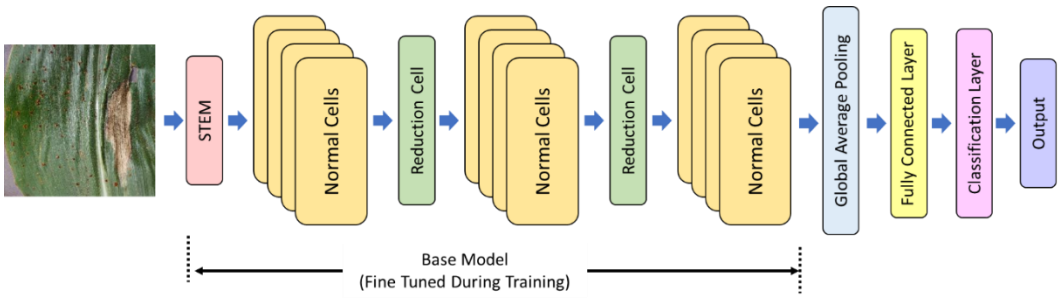
ต่างๆ ในธรรมชาติ ซึ่งทำให้การเรียนรู้ในขั้นถัดไปมีประสิทธิภาพมากขึ้น และในขั้นตอนการฝึกฝนแบบจำลองจะมีการทำ Fine-tuning โดยปรับค่าของทุกชั้นของแบบจำลอง เพื่อให้แบบจำลองสามารถเรียนรู้รายละเอียดเชิงลึกและปรับตัวให้เข้ากับข้อมูลที่นำมาพัฒนาแบบจำลองได้ดีขึ้น

3.2.3.4 แบบจำลอง NASNet Mobile

NASNet Mobile เป็นแบบจำลองที่ได้รับการออกแบบโดยการค้นหาโครงสร้างแบบอัตโนมัติผ่านเทคนิค Neural Architecture Search (NAS) ที่ใช้ในการค้นหาโครงสร้างของแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันให้เหมาะสมกับงานเฉพาะ ซึ่งโครงสร้างของ NASNet Mobile นี้ถูกออกแบบให้เหมาะสมกับการประมวลผลภาพเป็นหลัก โดยที่ NASNet Mobile เป็นแบบจำลองในรูปแบบย่อขนาดของ NASNet ที่ปรับแต่งเพื่อลดขนาดของโครงข่ายแบบจำลองลงและลดการคำนวณในการประมวลผล ทำให้เหมาะกับการใช้งานในระบบที่มีข้อจำกัดด้านทรัพยากร เช่น โทรศัพท์มือถือ หรืออุปกรณ์ IoT

NASNet Mobile ประกอบด้วยโครงสร้าง Building Block ที่เรียกว่า Normal cell และ Reduction cell ซึ่งบล็อกเหล่านี้จะทำหน้าที่ในการประมวลผลและสกัดคุณลักษณะของภาพออกมา โดยมีโครงสร้างเป็นแบบ Stack ซึ่งซ้อนทับกันไปมา การแบ่งโครงสร้างของ NASNet นี้จะช่วยให้แบบจำลองสามารถเรียนรู้คุณลักษณะของภาพได้อย่างละเอียดและลึกซึ้ง ช่วยให้แบบจำลองสามารถรับรู้ถึงความแตกต่างในคุณลักษณะของภาพแม้ว่าภาพเหล่านั้นจะมีลักษณะใกล้เคียงกันมากก็ตาม ซึ่งสามารถนำแบบจำลองนี้มาใช้กับการศึกษาวิจัยครั้งนี้ได้

ในการปรับโครงสร้างของ NASNet Mobile ให้เหมาะสมกับการจำแนกโรคจากใบพืชในระบบที่มี 4 กลุ่ม ได้ใช้ Base model ของแบบจำลอง Pre-trained ของ NASNet Mobile เป็นโครงสร้างหลักของแบบจำลองนี้ แต่ตัด Fully Connected Layer ออก และแทนที่ด้วยชั้น GAP เพื่อย่อขนาดข้อมูลจากการสกัดคุณลักษณะทั้งหมดให้อยู่ในรูปของเวกเตอร์ที่มีขนาดกะทัดรัด ช่วยลดจำนวนพารามิเตอร์ ทำให้แบบจำลองเล็กลงและลดความซับซ้อนในการคำนวณ เพื่อให้เหมาะสมกับการใช้งานที่มีข้อจำกัดด้านทรัพยากร หลังจาก GAP แล้ว จะทำการเชื่อมต่อชั้น Classification Layer ซึ่งเป็นชั้นที่ทำหน้าที่จำแนกประเภทของข้อมูลโดยตรง ซึ่งใช้นิวรอน จำนวน 4 นิวรอน ซึ่งสอดคล้องกับจำนวนหมวดหมู่ของโรคที่ต้องการจำแนก และใช้ฟังก์ชัน SoftMax สำหรับแปลงค่าผลลัพธ์ให้เป็นความน่าจะเป็นที่อยู่ในช่วงระหว่าง 0 ถึง 1 เพื่อนำไปใช้ในการทำนายผลว่าภาพที่ป้อนเข้าไปนั้นเป็นโรคใด โดยค่าความน่าจะเป็นที่ได้จะบ่งบอกถึงความเป็นไปได้ในการจำแนกภาพนั้นๆ ว่าอยู่ในคลาสใดมากที่สุด โครงสร้างของแบบจำลอง NASNet Mobile ที่ได้ปรับปรุงโครงสร้างแสดงในภาพประกอบ 45



ภาพประกอบ 45 แสดงโครงสร้างของแบบจำลอง NASNet Mobile หลังจากการปรับปรุงโครงสร้าง

ในการปรับ NASNet Mobile สำหรับการจำแนกโรคใบพืช เราเริ่มด้วยการใช้พารามิเตอร์จาก ImageNet ซึ่งเป็นชุดข้อมูลใหญ่ที่ช่วยให้แบบจำลองมีพื้นฐานที่ดี จากนั้นเราทำ Fine-tuning ทุกชั้นของแบบจำลอง เพื่อปรับค่าพารามิเตอร์ให้เหมาะสมกับข้อมูลโรคใบข้าวโพดที่เฉพาะเจาะจง การปรับค่าพารามิเตอร์ทุกชั้นของแบบจำลองนี้ช่วยให้แบบจำลองสามารถรับรู้คุณลักษณะเฉพาะของโรคในใบพืชได้ดีขึ้น ส่งผลให้แบบจำลองมีความแม่นยำในการจำแนกโรค นอกจากนี้ แบบจำลองที่ปรับนี้ยังสามารถนำไปใช้งานในสถานการณ์จริงได้อย่างมีประสิทธิภาพ

3.2.4 การป้องกันการเกิด Overfitting ด้วย การปรับแต่งค่า Learning Rate และใช้ Early Stopping

การป้องกันการ Overfitting เป็นกระบวนการสำคัญในศึกษาวิจัยเกี่ยวกับการฝึกฝนแบบจำลองโครงข่ายประสาทเทียม ซึ่งมีวัตถุประสงค์เพื่อให้แบบจำลองสามารถทำนายข้อมูลใหม่ได้อย่างแม่นยำ ไม่เพียงแต่จดจำข้อมูลฝึกฝนเท่านั้น การ Overfitting เกิดขึ้นเมื่อแบบจำลองเรียนรู้จากข้อมูลฝึกฝนมากเกินไปจนสามารถจำข้อมูลดังกล่าวได้ดีเกินไป ทำให้เมื่อทดสอบกับข้อมูลใหม่ แบบจำลองไม่สามารถทำนายได้อย่างแม่นยำเพราะไม่ได้เรียนรู้รูปแบบที่ครอบคลุมพอที่จะประยุกต์ใช้กับข้อมูลนอกเหนือจากข้อมูลที่ได้ฝึกฝนไปแล้ว ซึ่งในการศึกษาวิจัยครั้งนี้ได้เลือกใช้การป้องกันการเกิด Overfitting ด้วย การปรับแต่งค่า Learning Rate และใช้ Early Stopping

3.2.4.1 การปรับแต่งค่า Learning Rate ด้วยฟังก์ชัน ReduceLROnPlateau

การปรับค่า Learning Rate (LR) ถือเป็นปัจจัยสำคัญที่ส่งผลต่อการฝึกฝนแบบจำลอง LR คือ ตัวแปรที่กำหนดว่าแบบจำลองควรปรับพารามิเตอร์ของมันเร็วหรือช้าเพียงใด ซึ่งค่า LR ที่เหมาะสมช่วยให้แบบจำลองปรับตัวได้อย่างมีประสิทธิภาพ หาก LR สูงเกินไป แบบจำลองอาจเรียนรู้ได้เร็วเกินไปจนข้ามค่าที่ดีที่สุด (Optimal Point) ในขณะที่ค่า LR ต่ำเกินไปอาจทำให้

แบบจำลองค่อยๆ เคลื่อนที่ช้าและใช้เวลาในการฝึกฝนนานขึ้น การเลือก LR ที่เหมาะสมมักต้องทำการปรับจูนเพื่อหาค่าที่ทำให้ Loss Function ลดลงสม่ำเสมอ

ในการศึกษาวิจัยครั้งใช้การปรับ LR แบบ Adaptive Learning Rate ซึ่งค่า LR ลดลงอัตโนมัติ โดยใช้ฟังก์ชัน ReduceLROnPlateau ที่อยู่ใน Keras ซึ่งฟังก์ชัน ReduceLROnPlateau จะปรับค่า LR โดยลดลงเมื่อค่าการสูญเสียซึ่งในการวิจัยครั้งนี้จะใช้การติดตามค่า Validation Loss (Val Loss) ของแบบจำลอง ถ้าค่า Val Loss ไม่ลดลงตามจำนวน Epoch ที่กำหนด ฟังก์ชันจะปรับลดค่า LR ให้มีความละเอียดขึ้น เพื่อแบบให้จำลองเข้าใกล้ค่าที่เหมาะสมที่สุด เช่น ถ้าหากค่า Val Loss ไม่ลดลงใน 3 Epoch ให้ปรับลด LR ลง 20% เป็นต้น ซึ่งจะเป็นการลด LR ในการฝึกฝนแบบจำลองให้สอดคล้องกับสภาวะการปรับพารามิเตอร์ที่มีประสิทธิภาพและลดการแกว่งของค่าการสูญเสีย และการลดค่า LR ด้วยวิธีนี้ช่วยป้องกันแบบจำลองจากการติดอยู่ในจุดค่า Loss ที่คงที่ (Plateau) ทำให้ประสิทธิภาพการเรียนรู้ดีขึ้นในระยะยาว

3.2.4.2 **การใช้** Early Stopping

Early Stopping เป็นหนึ่งในเทคนิคที่สำคัญในกระบวนการฝึกฝนแบบจำลองโครงข่ายประสาทเทียม โดย Early Stopping จะช่วยให้เราสามารถควบคุมกระบวนการฝึกฝนแบบจำลองไม่ให้เกิดการ Overfitting มากเกินไป ซึ่งการศึกษาวิจัยครั้งนี้ได้ใช้ Early Stopping ในการหยุดการฝึกฝนแบบจำลองเพื่อป้องกันการเกิด Overfitting โดยการใช้งานฟังก์ชัน Early Stopping ใน Keras โดย ฟังก์ชัน EarlyStopping จะเฝ้าดูค่า Loss หากกระบวนการฝึกฝนแบบจำลองไม่มีการปรับปรุงในด้านค่า Loss ตามจำนวนรอบ Epoch ที่กำหนด ก็จะหยุดกระบวนการฝึกฝนแบบจำลองทันที เพื่อป้องกันการเกิด Overfitting จากการฝึกฝนแบบจำลอง

3.3 การปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพด

ในการศึกษาวิจัยในเกี่ยวกับการพัฒนาโครงข่ายประสาทเทียมแบบคอนโวลูชันในการตรวจจับโรคพืชจากภาพใบพืชถึงแม้จะได้ผลประสิทธิภาพที่สูง แต่การนำเอาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันไปใช้ในงานภาคเกษตรกรรมยังมีความท้าทายในการนำแบบจำลองไปใช้งานบนอุปกรณ์พกพา หรืออุปกรณ์ขนาดเล็กเพื่อใช้งานในภาคสนาม เนื่องจากอุปกรณ์เหล่านั้นมีข้อจำกัดในด้านการใช้พลังงานและทรัพยากรการประมวลผลสูง อีกทั้งในพื้นที่ทางการเกษตรที่ไม่สามารถเข้าถึงการเชื่อมต่ออินเตอร์เน็ต ทำให้จำเป็นต้องใช้การประมวลผลแบบจำลองบนอุปกรณ์พกพานั้น แทนการใช้ประมวลผลจาก Server ซึ่งอุปกรณ์เหล่านี้ต้องการแบบจำลองที่สามารถทำงานได้ดีในสภาวะที่มีข้อจำกัดของทรัพยากร ซึ่งได้แก่ หน่วยความจำ พลังงาน และพื้นที่จัดเก็บข้อมูล การสร้างแบบจำลองที่มีความแม่นยำในการตรวจจับสูงและมี

ประสิทธิภาพในการทำงานที่เพียงพอบนอุปกรณ์ขนาดเล็กจึงเป็นสิ่งท้าทาย วิธีการแก้ไขปัญหานี้ คือ การปรับปรุงแบบจำลองให้เหมาะสมที่สุด (Model Optimization) สำหรับการจำแนกโรคใบข้าวโพด โดยเน้นการลดขนาดของแบบจำลอง เพื่อทำให้สามารถประมวลผลได้รวดเร็วขึ้น โดยไม่ลดทอนความแม่นยำของการจำแนกโรคพืช

แนวทางที่ใช้ในการปรับปรุงประสิทธิภาพของแบบจำลองที่ใช้การศึกษาวิจัยในครั้งนี้ ได้แก่ การใช้แบบจำลองน้ำหนักเบา การทำ Knowledge Distillation และการทำ Quantization แบบ Post-training Quantization

3.3.1 การใช้แบบจำลองน้ำหนักเบา (Lightweight Model)

การนำแบบจำลองน้ำหนักเบามาใช้งานมีความสำคัญอย่างยิ่งเมื่อเราต้องการใช้โครงข่ายประสาทเทียมบนอุปกรณ์ที่มีข้อจำกัดด้านการประมวลผลและหน่วยความจำ เช่น สมาร์ทโฟน อุปกรณ์ IoT และระบบสมองกลฝังตัว การใช้แบบจำลองน้ำหนักเบาช่วยให้การประมวลผลภาพและการตัดสินใจเป็นไปได้อย่างมีประสิทธิภาพบนอุปกรณ์ที่ไม่สามารถรองรับแบบจำลองขนาดใหญ่ได้

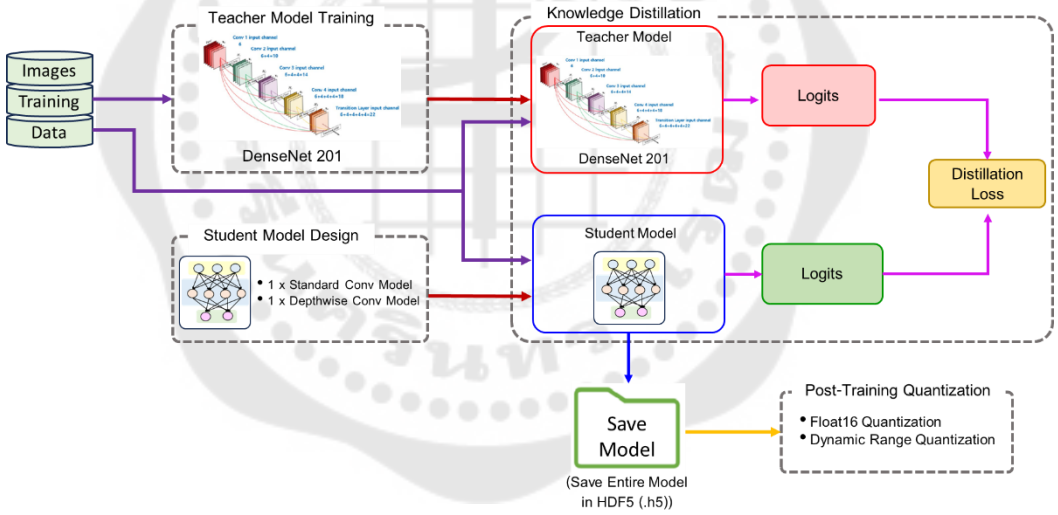
ในงานวิจัยครั้งนี้ได้เลือกแบบจำลองขนาดเบาที่ใช้โครงสร้างแบบ Depthwise Separable Convolution ซึ่งจะช่วยลดจำนวนของพารามิเตอร์ทำให้แบบจำลองมีขนาดเล็ก และช่วยลดจำนวนการดำเนินการทางคณิตศาสตร์ทำให้แบบจำลองประมวลผลได้อย่างรวดเร็ว และต้องการทรัพยากรในการประมวลผลน้อย โดยพิจารณาใช้แบบจำลองแบบ Pre-trained ในไลบรารี Keras API มาใช้ในการวิจัย จำนวน 3 แบบ ได้แก่ DenseNet121 MobileNetV2 NASNet Mobile และนอกจากนี้ยังการออกแบบแบบจำลอง Student Model ที่โดยใช้โครงสร้างของ Depthwise Separable Convolution จำนวน 1 แบบจำลอง เพื่อนำมาพัฒนาเป็นแบบจำลองสำหรับการวิจัยในการพัฒนาแบบจำลองจากการทำ Knowledge Distillation

3.3.2 การทำ Knowledge Distillation

ในการศึกษาวิจัยครั้งนี้จะใช้เทคนิคการทำ Knowledge Distillation แบบ Respond-based Knowledge ซึ่งในกระบวนการพิจารณาคัดเลือกแบบจำลองขนาดใหญ่สำหรับใช้เป็น Teacher Model สำหรับใช้ทำ Knowledge Distillation ทางผู้วิจัยได้ทดลองเลือกแบบจำลอง Pre-trained ในไลบรารี Keras API ที่มีความแม่นยำสำหรับการทำนายผลของชุดข้อมูล ImageNet ที่สูง มาเพื่อใช้สำหรับฝึกฝนเป็น Teacher Model โดยได้ทดลองฝึกฝนแบบจำลอง จำนวน 3 แบบจำลอง ได้แก่ EfficientNetV2M EfficientNetV2L และ DenseNet201 พบว่า และ แบบจำลอง DenseNet201 ให้ความแม่นยำสูงที่สุด 99.22% จึงได้เลือกใช้แบบจำลอง DenseNet201 มาใช้

เพื่อฝึกฝนเป็นแบบจำลอง Teacher Model สำหรับใช้ทำ Knowledge Distillation สำหรับการพัฒนาแบบจำลองในการวิจัยครั้งนี้

สำหรับ Student Model ในในการศึกษาวิจัยครั้งนี้ได้สร้างแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน ให้มีขนาดเล็กโดยกำหนดให้มีขนาดไม่เกิน 1 ใน 3 ของขนาดแบบจำลอง MobileNetV2 ซึ่งเป็นแบบจำลอง Pre-trained ที่มีขนาดเล็กที่สุดที่ใช้ในการวิจัยครั้งนี้ (มี 3.5 ล้านพารามิเตอร์ และมีขนาดประมาณ 14 MB) มีจำนวนพารามิเตอร์ไม่เกิน 1,200,000 พารามิเตอร์ หรือมีขนาดไม่เกิน 5 MB เพื่อใช้เป็น Student Model โดยประกอบสร้างแบบจำลองที่ใช้โครงสร้างของการคอนโวลูชันแบบปกติ จำนวน 1 แบบจำลอง และแบบจำลองที่ใช้โครงสร้างของ Depthwise Separable Convolution จำนวน 1 แบบจำลอง เพื่อนำแบบจำลอง Student Model ทั้งสองแบบที่ผ่านการทำ Knowledge Distillation แล้วมาเปรียบเทียบประสิทธิภาพในการทำนายผลว่าแบบใดจะให้การทำนายที่แม่นยำมากกว่า และแบบใดจะใช้เวลาในการประมวลผลเร็วกว่ากัน แนวความคิดในการทำ Knowledge Distillation ในการวิจัยครั้งนี้แสดงดังภาพประกอบ 46



ภาพประกอบ 46 แสดงขั้นตอนการทำ Knowledge Distillation เพื่อพัฒนาแบบจำลองขนาดเล็กในการวิจัยครั้งนี้

ในการวิจัยครั้งนี้ได้พิจารณาเลือกการทำ Knowledge Distillation แบบ Respond-based Knowledge ซึ่งใช้ความรู้ที่ถูกถ่ายโอนมาจากค่า Logits หรือค่าผลลัพธ์ที่ได้จากค่า Soft Target ของ Teacher Model โดยค่าผลลัพธ์นี้จะสะท้อนถึงความเชื่อมั่นที่แบบจำลองมีต่อคลาสต่าง ๆ ในชุดข้อมูล ซึ่งค่าความเชื่อมั่นนี้จะช่วยให้ Student Model เรียนรู้ได้อย่างมีประสิทธิภาพมากขึ้น

กระบวนการ Response-Based Knowledge Distillation

กระบวนการ Knowledge Distillation โดยใช้ Response-Based Approach สามารถแบ่งออกเป็นขั้นตอนหลัก ๆ ดังนี้

1) การเตรียม Teacher Model

การฝึกฝน DenseNet201 กับชุดข้อมูลที่มีการแยกโรคใบข้าวโพดอย่างละเอียดจนได้ค่าความแม่นยำสูง แบบจำลองนี้จะถูกใช้เพื่อสร้างค่าการตอบสนอง (Logits) สำหรับการถ่ายโอนความรู้

2) การออกแบบ Student Model

เลือกโครงสร้างแบบจำลองที่มีขนาดเล็กโดยมีเป้าหมายเพื่อลดจำนวนพารามิเตอร์และเพิ่มความเร็วในการประมวลผลของแบบจำลอง ซึ่งในการวิจัยครั้งนี้จะสร้างแบบจำลองที่ใช้โครงสร้างของการคอนโวลูชันแบบปกติ จำนวน 1 แบบจำลอง และแบบจำลองที่ใช้โครงสร้างของ Depthwise Separable Convolution จำนวน 1 แบบจำลอง

3) การกำหนด Loss Function

Loss Function ที่ใช้ในการฝึก Student Model จะประกอบด้วย 2 ส่วนหลัก คือ

ก. Distillation Loss ใช้ค่าการตอบสนองจาก Teacher Model ที่ถูกปรับผ่าน Softmax ด้วยค่าอุณหภูมิเพื่อสร้างความนุ่มนวลในค่าการตอบสนอง ค่านี้ช่วยให้ Student Model เข้าใจความเชื่อมั่นของ Teacher Model ในการแยกประเภทของคลาสต่าง ๆ โดย Distillation Loss เป็นองค์ประกอบหลักในกระบวนการ Knowledge Distillation ซึ่งมีหน้าที่ถ่ายโอนความรู้จาก Teacher Model ไปยัง Student Model โดยวัดความแตกต่างระหว่างค่าการตอบสนอง (Logits) ของทั้งสองแบบจำลองผ่านค่า Kullback-Leibler Divergence (KL Divergence) ค่าอุณหภูมิ T ถูกเพิ่มเข้ามาเพื่อทำให้ค่าการตอบสนองของ Teacher Model มีความนุ่มนวลมากขึ้น ซึ่งช่วยให้ Student Model เรียนรู้ความสัมพันธ์ระหว่างคลาสได้ดียิ่งขึ้น ค่า T ที่สูงเกินไป ค่าการตอบสนองจะคล้ายกันเกินไป แต่หากต่ำเกินไป จะทำให้แบบจำลองสูญเสียความละเอียดของข้อมูล เป้าหมายของ Distillation Loss จะช่วยให้ Student Model เข้าใจข้อมูลจากมุมมองของ Teacher Model โดยเรียนรู้ความเชื่อมั่นที่ Teacher Model มีต่อแต่ละคลาส โดยเน้นความสัมพันธ์ระหว่างคลาสมากกว่าการพยายามจับคู่ค่าที่เฉพาะเจาะจงเพียงอย่างเดียว

ข. Classification Loss ใช้ค่าความแม่นยำของ Student Model เมื่อเทียบกับค่า Ground Truth เพื่อให้แน่ใจว่า Student Model สามารถแยกแยะข้อมูลได้อย่างถูกต้อง

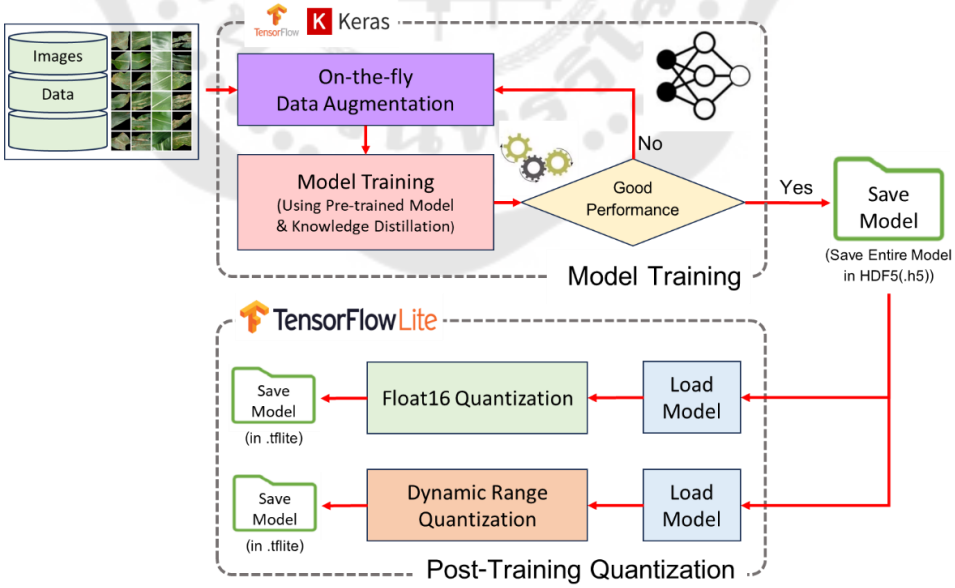
Loss Function ทั้งสองส่วนจะถูกนำมารวมกันด้วยอัตราส่วน ซึ่งเป็นค่าคงที่ระหว่าง 0 ถึง 1เพื่อควบคุมน้ำหนักของการเรียนรู้จาก Teacher Model และข้อมูลจริง

4) การฝึกฝน Student Model

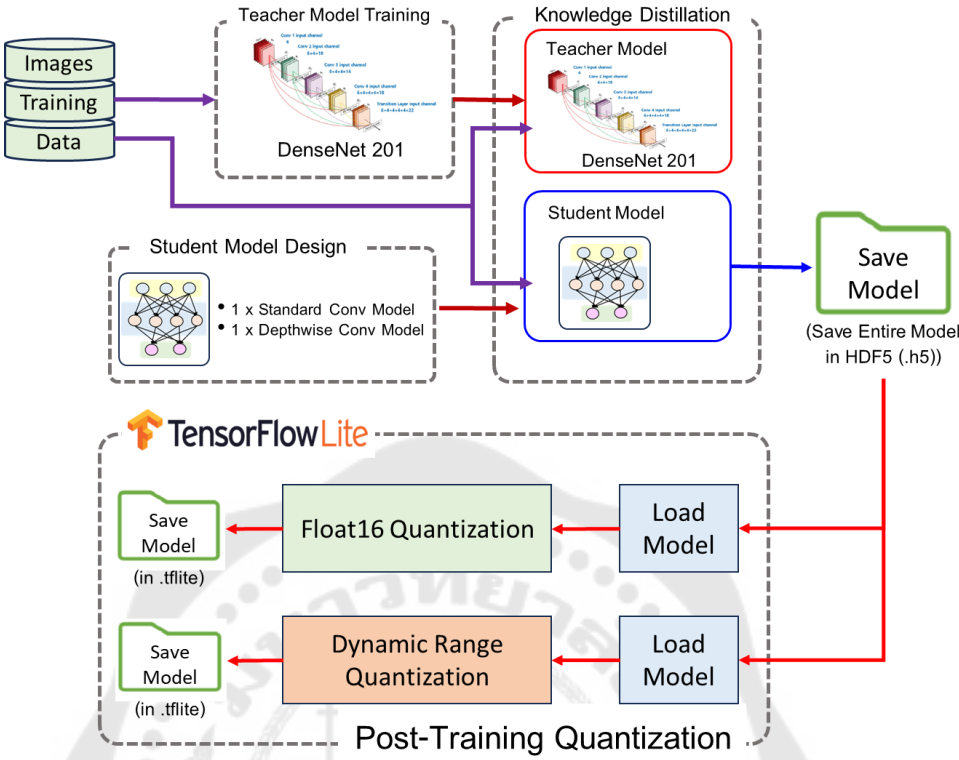
ในกระบวนการฝึก Student Model ค่าการตอบสนองที่ได้จาก Teacher Model จะถูกใช้เป็น Target สำหรับ Student Model ควบคู่ไปกับค่า Ground Truth กระบวนการนี้ช่วยให้ Student Model สามารถเรียนรู้ลักษณะข้อมูลในเชิงลึกได้มากขึ้น การใช้ค่าการตอบสนองจาก Teacher Model ช่วยให้ Student Model เข้าใจความสัมพันธ์ระหว่างคลาสต่าง ๆ ได้ดีขึ้น โดยเฉพาะในกรณีที่ข้อมูลมีความซับซ้อนหรือมีลักษณะใกล้เคียงกัน นอกจากนี้ การรวมค่า Ground Truth เข้ามาใน Loss Function ยังช่วยให้ Student Model สามารถปรับปรุงประสิทธิภาพได้ทั้งในแง่ของความแม่นยำและความสามารถในการแยกแยะข้อมูลใหม่ได้

3.3.3 การทำ Quantization แบบ Post-training Quantization

การทำ Quantization เป็นกระบวนการแปลงชนิดของข้อมูลของพารามิเตอร์ของแบบจำลองให้มีขนาดเล็กลง จากเดิมที่จัดเก็บในรูปแบบของ Floating Point ขนาด 32 บิต แปลงให้อยู่ในรูปแบบของ Floating Point ขนาด 16 บิต และ Integer ขนาด 8 บิต เพื่อลดขนาดของแบบจำลอง และทำให้แบบจำลองทำงานได้รวดเร็วขึ้น ซึ่งการแปลงค่านี้สามารถทำได้ทั้งขณะที่ทำการฝึกฝนแบบจำลอง หรือหลังจากที่ฝึกฝนแบบจำลองเสร็จแล้ว



ภาพประกอบ 47 แสดงขั้นตอนการพัฒนาแบบจำลองและลดขนาดแบบจำลอง Pre-trained โดยการทำ Post-Training Quantization



ภาพประกอบ 48 แสดงขั้นตอนการพัฒนาแบบจำลองด้วยวิธีการทำ Knowledge Distillation และลดขนาดแบบจำลองโดยการทำ Post-Training Quantization

ในการศึกษาวิจัยครั้งนี้เลือกใช้การทำ Quantization หลังจากที่ฝึกฝนแบบจำลองเสร็จแล้ว หรือ Post-training Quantization ซึ่งจะเป็นกระบวนการแปลงชนิดของพารามิเตอร์ของแบบจำลองที่ผ่านการฝึกฝนจนได้แบบจำลองที่มีประสิทธิภาพตามต้องการแล้ว ซึ่งจะเป็นการแปลงเฉพาะรูปแบบชนิดของพารามิเตอร์ของแบบจำลอง โดยไม่ได้ปรับโครงสร้างของแบบจำลองแต่อย่างใด แบบจำลองยังคงมีโครงสร้างของ Neuron และจำนวนของพารามิเตอร์เหมือนเดิมไม่มีการเปลี่ยนแปลง แต่แบบจำลองจะถูกทำให้มีขนาดเล็กลงโดยการแปลงค่าชนิดข้อมูลของพารามิเตอร์ของแบบจำลอง โดยจะใช้การแปลงแบบ Float16 Quantization (FP16) และการแปลงค่าแบบ Dynamic Range Quantization (INT8) กับทั้งแบบจำลอง Pre-trained (ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile) ดังแสดงขั้นตอนการทำงานดังภาพประกอบ 47 และแบบจำลองที่ถูกพัฒนาขึ้นจากการทำ Knowledge Distillation ที่ได้ผ่านการฝึกฝนกับชุดข้อมูลตัวอย่างแล้ว ดังแสดงขั้นตอนการทำงานดังภาพประกอบ 48

1) Float16 Quantization

Float16 Quantization เป็นกระบวนการแปลงข้อมูลของพารามิเตอร์ของแบบจำลองให้มีขนาดเล็กลง จากเดิมที่จัดเก็บในรูปแบบของ Floating Point ขนาด 32 บิต (FP32) แปลงให้อยู่ในรูปแบบของ Floating Point ขนาด 16 บิต (FP16) โดยกระบวนการนี้ช่วยลดขนาดของข้อมูลพารามิเตอร์ลงครึ่งหนึ่ง ทำให้แบบจำลองขนาดเล็กลงและสามารถโหลดเข้าไปประมวลผลในหน่วยความจำที่จำกัดได้ง่ายขึ้น

การใช้ Float16 Quantization ยังคงความแม่นยำในการคำนวณในระดับที่ดีเมื่อเทียบกับการทำงานด้วยค่า Integer (เช่น INT8) เนื่องจากยังมีค่าทศนิยมที่ละเอียดกว่าการแปลงเป็น Integer และยังมีข้อดีที่ลดความซับซ้อนในการคำนวณ เนื่องจากการแปลงค่าแบบ FP32 เป็น FP16 สามารถทำได้โดยตรงโดยไม่ต้องปรับเปลี่ยนโครงสร้างการคำนวณหลักของแบบจำลอง ซึ่ง FP16 นั้น มีค่า Exponent และ Bias ที่ต่ำกว่า FP32 จึงทำให้การแสดงค่าทศนิยมมีความละเอียดน้อยลง แต่ยังคงมีความแม่นยำที่เพียงพอต่อการคำนวณส่วนใหญ่

2) Dynamic Range Quantization

Dynamic Range Quantization เป็นกระบวนการแปลงข้อมูลของพารามิเตอร์ของแบบจำลอง จากเดิมที่จัดเก็บในรูปแบบของ Floating Point ขนาด 32 บิต (FP32) แปลงให้อยู่ในรูปแบบของ Integer ขนาด 8 บิต (INT8) เพื่อลดขนาดของแบบจำลอง

Dynamic Range Quantization จะใช้การปรับค่าให้อยู่ในช่วง (Range) ที่จำกัดในแต่ละชั้น โดยจะปรับค่า Weights และ Biases ให้เป็นค่า INT8 ซึ่งอยู่ในช่วง -128 ถึง 127 โดยจะปรับช่วงค่าเหล่านี้ให้อยู่ในรูปแบบที่ใกล้เคียงที่สุดกับข้อมูลต้นฉบับที่เป็น FP32 กระบวนการนี้ช่วยให้แบบจำลองขนาดเล็กลงอย่างมาก เนื่องจาก INT8 มีความต้องการทรัพยากรการเก็บข้อมูลน้อยกว่า FP32 ถึง 4 เท่า แต่ Dynamic Range Quantization อาจทำให้แบบจำลองสูญเสียความแม่นยำบางส่วน เนื่องจากความละเอียดของข้อมูลถูกลดลง

สิ่งสำคัญประการหนึ่งของการใช้ Dynamic Range Quantization จะเป็นการแปลงค่าเฉพาะค่าพารามิเตอร์ของแบบจำลอง (ค่า Weights และ Biases) เท่านั้น แต่ข้อมูลรับเข้า และการดำเนินการทางคณิตศาสตร์ยังคงใช้เป็นรูปแบบของ FP32 ส่งผลให้เมื่อนำแบบจำลองที่ผ่านการทำ Dynamic Range Quantization ไปใช้งานเวลาในการประมวลผลจะไม่รวดเร็วเท่าที่ควร เนื่องจากต้องแปลงค่าพารามิเตอร์ที่อยู่ในรูปแบบ INT8 ให้กลับไปอยู่ในรูปแบบ FP32 ก่อน จึงจะประมวลผลแบบจำลองได้

3.4 การประเมินผล และเปรียบเทียบประสิทธิภาพแบบจำลองหลังจากผ่านกระบวนการปรับปรุงแบบจำลอง

ในการวิจัยครั้งนี้จะเลือกแบบจำลอง Convolutional Neural Network (CNN) มาทดลองในการวิจัยครั้งนี้ ประกอบด้วยแบบจำลอง แบบ Pre-trained จากไลบรารีของ Keras API จำนวน 4 แบบจำลอง ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile รวมทั้งได้พัฒนาแบบจำลองขนาดเล็กโดยใช้วิธี Knowledge Distillation จำนวน 2 แบบจำลอง จะทำให้ได้แบบจำลองสำหรับการจำแนกโรคจากใบข้าวโพดทั้งสิ้น 6 แบบจำลอง จากนั้นนำแบบจำลองทั้งหมดมาลดขนาดโดยการทำ Quantization แบบ Post-training Quantization จำนวน 2 วิธี ได้แก่ Float16 Quantization และ Dynamic Range Quantization จะทำให้ได้แบบจำลองที่ลดขนาดแล้วอีก 12 แบบจำลอง ดังนั้นจะมีแบบจำลองที่ต้องเปรียบเทียบประสิทธิภาพรวมทั้งสิ้น 18 แบบจำลอง

3.4.1 การเปรียบเทียบความเหมาะสมแบบจำลอง

ในกระบวนการวิจัยและพัฒนาแบบจำลอง เพื่อใช้ในการจำแนกโรคในใบข้าวโพดบนอุปกรณ์ขนาดเล็ก สิ่งสำคัญที่ควรให้ความสนใจหลังการปรับปรุงประสิทธิภาพของแบบจำลองคือการประเมินผลและเปรียบเทียบประสิทธิภาพของแบบจำลองในด้านต่างๆ ได้แก่ ความแม่นยำ (Accuracy, Precision, Recall และ F1-Score) ขนาดของแบบจำลอง (Model Size) และเวลาในการประมวลผลภาพเฉลี่ยเมื่อทดสอบด้วย Test Set ซึ่งในส่วนนี้สามารถนำข้อมูลที่ได้จากการทดสอบมาตีความและนำเสนอได้ดังนี้

1) ความแม่นยำ (Accuracy)

การวัดผลความแม่นยำของแบบจำลองที่ผ่านกระบวนการปรับปรุงแบบจำลองนั้นควรพิจารณาทั้งค่า Accuracy (ความถูกต้องโดยรวม) Precision (ความแม่นยำในการทำนายแต่ละคลาส) Recall (อัตราการระบุข้อมูลที่ถูกต้องจริง) และ F1-Score ซึ่งเป็นค่าถ่วงน้ำหนักระหว่าง Precision และ Recall ค่าทั้งหมดนี้เป็นตัวชี้วัดที่ช่วยให้เห็นถึงความแม่นยำของแบบจำลองทั้งในแง่ของการระบุคลาสและการระบุข้อมูลที่มีลักษณะเฉพาะ ตัวอย่างเช่น ในการจำแนกโรคใบข้าวโพด ค่าความแม่นยำจะช่วยประเมินว่าแบบจำลองสามารถระบุใบที่ติดเชื้อและไม่ติดเชื้อได้แม่นยำเพียงใด โดยเฉพาะการใช้ F1-Score เพื่อหลีกเลี่ยงการวัดที่ไม่สมดุลหากมีคลาสที่แตกต่างกันมาก เช่น คลาสโรคบางชนิดที่พบไม่บ่อย ซึ่งในการวิจัยครั้งนี้จะพิจารณาค่า Accuracy

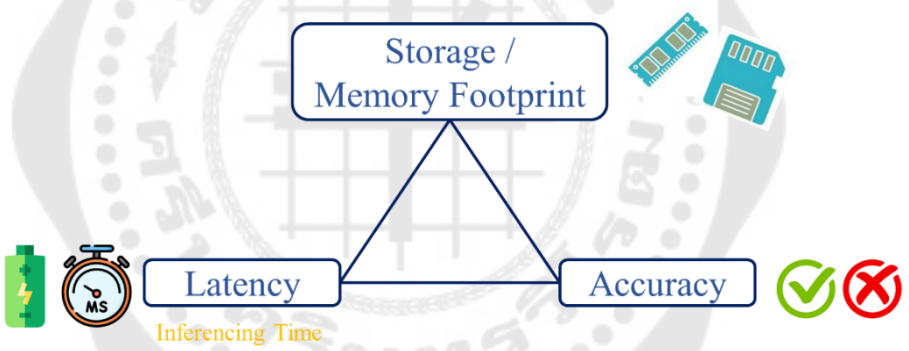
2) ขนาดของแบบจำลอง (Model Size)

ขนาดของแบบจำลองมีผลโดยตรงต่อการใช้งานบนอุปกรณ์ขนาดเล็ก (Edge Devices) เช่น โทรศัพท์มือถือหรือเซ็นเซอร์ที่มีข้อจำกัดด้านหน่วยความจำและการประมวลผล การปรับ

ขนาดแบบจำลองอาจทำได้โดยการใช้เทคนิคการทำให้แบบจำลองเบาขึ้น เช่น Pruning และ Quantization โดยเฉพาะการปรับขนาดเป็น Float16 หรือการใช้ Dynamic Range Quantization จะช่วยลดขนาดแบบจำลองได้อย่างมากโดยไม่สูญเสียความแม่นยำมากนัก นอกจากนี้ยังทำให้แบบจำลองสามารถถูกเก็บในหน่วยความจำได้มากขึ้นและสามารถประมวลผลได้อย่างรวดเร็ว

3) เวลาในการประมวลผลของแบบจำลอง (Inferencing Time)

สำหรับอุปกรณ์ที่มีความเร็วการประมวลผลจำกัด การวัดค่า Inferencing Time หรือเวลาที่แบบจำลองใช้ในการประมวลผลภาพแต่ละภาพเป็นสิ่งสำคัญ เนื่องจากเวลาในการประมวลผลนั้นเป็นตัวแปรที่บ่งชี้ถึงความเหมาะสมในการใช้งานจริง การทดสอบสามารถทำได้โดยใช้ Test Set และคำนวณเวลาเฉลี่ยในการประมวลผลเพื่อให้ได้ค่าที่เป็นกลาง เทคนิคการลดขนาดของแบบจำลอง เช่น Quantization หรือ Pruning อาจมีผลในการลดเวลาในการประมวลผลนี้เช่นกัน นอกจากนี้ การประเมิน Inferencing Time นั้นควรทำบนอุปกรณ์เป้าหมาย (เช่น อุปกรณ์ขนาดเล็ก) เพื่อให้ผลการทดสอบใกล้เคียงกับการใช้งานจริงมากที่สุด



ภาพประกอบ 49 แสดงการรักษาสมดุลของการลดขนาดของแบบจำลองใน 3 ด้าน ได้แก่ ความแม่นยำ ขนาดของแบบจำลอง และ เวลาในการประมวลผลของแบบจำลอง

ในการพัฒนาและลดขนาดของแบบจำลองเพื่อใช้งานกับอุปกรณ์ประมวลผลขนาดเล็กนั้นจะต้องพิจารณาสมดุลกันของทั้ง 3 ด้านที่ได้กล่าวมาข้างต้น ซึ่งขึ้นอยู่กับลักษณะงานที่จะนำแบบจำลองไปใช้งาน หรือมุ่งเน้นการลดขนาดของแบบจำลองให้มีขนาดเล็กที่สุดโดยไม่ได้คำนึงถึงประสิทธิภาพความแม่นยำในการทำนายผลของแบบจำลอง ดังแสดงในภาพประกอบ 49

3.4.2 **การใช้เมทริกซ์ตัดสินใจแบบถ่วงน้ำหนัก** (Weighted Decision Matrix)

การเลือกแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่เหมาะสมสำหรับการจำแนกโรคใบข้าวโพดเป็นขั้นตอนที่สำคัญ โดยเฉพาะเมื่อเป้าหมายคือการลดขนาดแบบจำลองเพื่อให้สามารถใช้งานได้อย่างมีประสิทธิภาพในสภาพแวดล้อมที่มีทรัพยากรจำกัด เช่น อุปกรณ์

พกพาหรือในพื้นที่ชนบท การใช้ เมทริกซ์ตัดสินใจแบบถ่วงน้ำหนัก (WDM) ถือเป็นเครื่องมือวิเคราะห์ที่มีประสิทธิภาพในการเปรียบเทียบและคัดเลือกแบบจำลอง โดยพิจารณาจากเกณฑ์สำคัญ เช่น ความแม่นยำ ขนาดแบบจำลอง และเวลาประมวลผล

Weighted Decision Matrix เป็นเทคนิคในการตัดสินใจแบบหลายเกณฑ์ที่ช่วยให้นักวิจัยสามารถจัดลำดับความสำคัญของเกณฑ์ต่าง ๆ ได้อย่างเป็นระบบ วิธีการนี้เริ่มจากการกำหนดเกณฑ์ที่ใช้วิเคราะห์ เช่น ความแม่นยำ ซึ่งมีความสำคัญสำหรับการจำแนกโรคได้อย่างถูกต้อง ขนาดแบบจำลองที่สะท้อนถึงทรัพยากรที่ใช้ และเวลาประมวลผลที่ส่งผลต่อการใช้งานแบบ Real-Time

จากนั้นแต่ละเกณฑ์จะได้รับค่าความสำคัญหรือน้ำหนัก (Weight) ตามความเหมาะสม เช่น หากเป้าหมายคือการใช้งานในอุปกรณ์พกพา น้ำหนักของเกณฑ์ที่เกี่ยวข้องกับขนาดแบบจำลองอาจมีค่าสูงกว่า ในขณะที่การใช้งานในสภาพแวดล้อมที่ต้องการผลลัพธ์ทันที น้ำหนักของเวลาประมวลผลอาจมีความสำคัญกว่า

ตาราง 11 ตารางแสดงระดับความสำคัญระหว่างเกณฑ์

ระดับ	Intensity of Importance	ความเข้มข้นของความสำคัญ
1	Equal importance	สำคัญเท่ากัน
2	Equal to moderate importance	สำคัญเท่ากันถึงปานกลาง
3	Moderate importance	สำคัญปานกลาง
4	Moderate to strong importance	สำคัญปานกลางถึงค่อนข้างมาก
5	Strong importance	สำคัญค่อนข้างมาก
6	strong to very strong importance	สำคัญค่อนข้างมากถึงมากกว่า
7	Very strong importance	สำคัญมากกว่า
8	Very to extremely strong importance	สำคัญมากกว่าถึงมากที่สุด
9	Extremely importance	สำคัญมากที่สุด

ที่มา : (Saaty, 1987)

Analytic Hierarchy Process (AHP) (ธวัชระพงษ์, 2022) เป็นเครื่องมือเชิงวิเคราะห์ที่ใช้สำหรับการตัดสินใจในสถานการณ์ที่มีหลายเกณฑ์หรือหลายทางเลือกที่ต้องพิจารณา วิธีการนี้เหมาะสมสำหรับการกำหนดค่าน้ำหนักในเมทริกซ์ตัดสินใจแบบถ่วงน้ำหนัก โดย AHP มีจุดเด่นใน

การช่วยประเมินความสำคัญของแต่ละเกณฑ์อย่างเป็นระบบ การคำนวณค่าน้ำหนักของหลักเกณฑ์ (Computation of the Criterion Weights) WDM โดยใช้ AHP มี 3 ขั้นตอน ดังนี้

1) การพัฒนาตารางเปรียบเทียบเป็นคู่ (Pairwise Comparison Matrix)

สร้างตารางเปรียบเทียบหลักเกณฑ์แต่ละหลักเกณฑ์เป็นคู่ ๆ โดยใช้ระดับความสำคัญที่กำหนดไว้ ตั้งแต่ 1 – 9 โดยแสดงความสัมพันธ์ระหว่าง 2 หลักเกณฑ์ ดังแสดงในตาราง 11

การเปรียบเทียบเกณฑ์ในการวิจัยครั้งนี้จะมีการเปรียบเทียบใน 3 หลักเกณฑ์ ได้แก่ ความแม่นยำ ขนาดแบบจำลอง และเวลาประมวลผล ซึ่งกำหนดระดับความสำคัญระหว่างเกณฑ์แต่ละคู่ดังตาราง 12

ตาราง 12 การกำหนดระดับความสำคัญของเกณฑ์แต่ละคู่

	ความแม่นยำ	**ขนาดแบบจำลอง**	**เวลาประมวลผล**
ความแม่นยำ	1	3	5
ขนาดแบบจำลอง	1/3	1	3
เวลาประมวลผล	1/5	1/3	1

2) การคำนวณค่าน้ำหนักของหลักเกณฑ์ (Computation of the Criterion Weights)

ค่าถ่วงน้ำหนัก สามารถคำนวณได้ตามลำดับโดย ขั้นตอนที่ 1) เริ่มจากการหาผลรวมค่าระดับความสำคัญแบบ Pairwise ในแต่ละคอลัมน์จากตาราง 13 ซึ่งจะได้ค่าผลรวมของเกณฑ์ในคอลัมน์ความแม่นยำ ขนาดแบบจำลอง และเวลาประมวลผล เป็น 1.53 4.33 และ 9.00 ตามลำดับ จากนั้น หารค่าในตารางด้วยผลรวมของแต่ละคอลัมน์ (Normalized Matrix) เช่น ในคอลัมน์ความแม่นยำ จะได้ค่า

1 / 1.53 = 0.6522

0.2 / 1.53 = 0.2174

0.1429 / 1.53 = 0.1304

โดยคำนวณในคอลัมน์ ขนาดแบบจำลอง และเวลาประมวลผล จนครบทั้งหมด

และ คำนวณหาน้ำหนักของแต่ละเกณฑ์ โดยการหาค่าเฉลี่ยของแต่ละแถวของ Normalized Matrix โดยค่าน้ำหนักของแต่ละเกณฑ์หาค่าได้ ดังนี้

ค่าน้ำหนักของความแม่นยำ = (0.6522 + 0.6923 + 0.5556) / 3 = 0.6333

ค่าน้ำหนักของขนาดแบบจำลอง = (0.2174 + 0.2308 + 0.3333) / 3 = 0.2605

ค่าน้ำหนักของเวลาประมวลผล = (0.1304 + 0.0769 + 0.1111) / 3 = 0.1062

ซึ่งค่าน้ำหนักของทุกเกณฑ์รวมกันจะมีค่าเท่ากับ 1 และค่าที่คำนวณได้ของแต่ละเกณฑ์แสดงในตาราง 13

ตาราง 13 แสดงค่าการคำนวณการหาค่าถ่วงน้ำหนักของแต่ละตัวเกณฑ์

Criteria	(1) ผลรวมแต่ละคอลัมน์			(2) Normalized Matrix			(3) น้ำหนัก (Weight)
	ความแม่นยำ	ขนาดแบบจำลอง	เวลาประมวลผล	ความแม่นยำ	ขนาดแบบจำลอง	เวลาประมวลผล	
ความแม่นยำ	1	3	5	0.6522	0.6923	0.5556	0.6333
ขนาดแบบจำลอง	1/3	1	3	0.2174	0.2308	0.3333	0.2605
เวลาประมวลผล	1/5	1/3	1	0.1304	0.0769	0.1111	0.1062
ผลรวม	1.53	4.33	9.00	1	1	1	1

3) การประมาณค่าความสอดคล้อง (Consistency Ratio)

การประเมินค่าถ่วงน้ำหนักของแต่ละเกณฑ์ที่ได้จากการคำนวณว่ามีความสอดคล้องกันเพียงพอที่จะนำไปใช้วิเคราะห์หรือไม่ ขึ้นอยู่กับค่า Consistency Ratio (CR) ซึ่งเป็นดัชนีที่บ่งบอกถึงระดับความสอดคล้องของค่าถ่วงน้ำหนัก หากค่า CR < 0.10 แสดงว่าค่าถ่วงน้ำหนักมีความสอดคล้องกันในระดับที่ยอมรับได้ สามารถนำไปใช้วิเคราะห์ได้ แต่หากค่า CR $\geq$ 0.10 แสดงว่าค่าถ่วงน้ำหนักยังมีความไม่สอดคล้องกัน จึงจำเป็นต้องทบทวนกระบวนการกำหนดระดับความสำคัญของแต่ละตัวเกณฑ์ และคำนวณค่าค่าถ่วงน้ำหนักใหม่อีกครั้ง จากนั้นจะคำนวณค่า CR ใหม่

โดยค่า CR สามารถคำนวณได้จากการนำค่า Consistency Index (CI) หารด้วยค่า Random Index (RI) ตามสมการ (25)

$$CR = \frac{CI}{RI} \quad (25)$$

โดยที่

ค่า RI (Random Index) ซึ่งค่า RI ขึ้นอยู่กับจ้านวนของ หลักเกณฑ์ที่ใช้เปรียบเทียบ เมื่อจำนวนของเกณฑ์ = 3 จะมีค่าเท่ากับ 0.58 (Ammarapala et al., 2018)

ค่า CI สามารถค้านวณได้จาก สมการ (26)

$$CI = \frac{\lambda - 1}{n-1} \quad (26)$$

โดยที่

λ คือ ค่าเฉลี่ยจากการวิเคราะห์ความสอดคล้อง หาค่าได้ด้วยการคำนวณดังนี้

(1) การนำค่าน้ำหนักของแต่ละตัวเกณฑ์จากตาราง 13 ในข้อ 2). มาคูณกับค่าระดับความสำคัญในแต่ละคอลัมน์

ความแม่นยำ = (0.7235 * 1) + (0.1932 * 5) + (0.0833 * 5) = 2.2726

ขนาดของแบบจำลอง = (0.7235 * 1/5) + (0.1932 * 1) + (0.0833 * 3) = 0.5878

เวลาประมวลผล = (0.7235 * 1/5) + (0.1932 * 0.3333) + (0.0833 * 1) = 0.2511

(2) แล้วหารเฉลี่ยด้วยค่าถ่วงน้ำหนักของแต่ละเกณฑ์

ความแม่นยำ = 2.2726 / 0.7235 = 3.1411

ขนาดของแบบจำลอง = 0.5878 / 0.1932 = 3.0427

เวลาประมวลผล = 0.2511 / 0.0833 = 3.0137

(3) หาค่า λ โดยหาเฉลี่ยของผลรวมในข้อ (2)

λ = (3.1411 + 3.0427 + 3.0137) / 3 = 3.0658

และเมื่อแทนค่า λ ลงในสมการ (26) จะได้ค่า

$$CI = \frac{3.0658 - 1}{3 - 1} = 0.0329$$

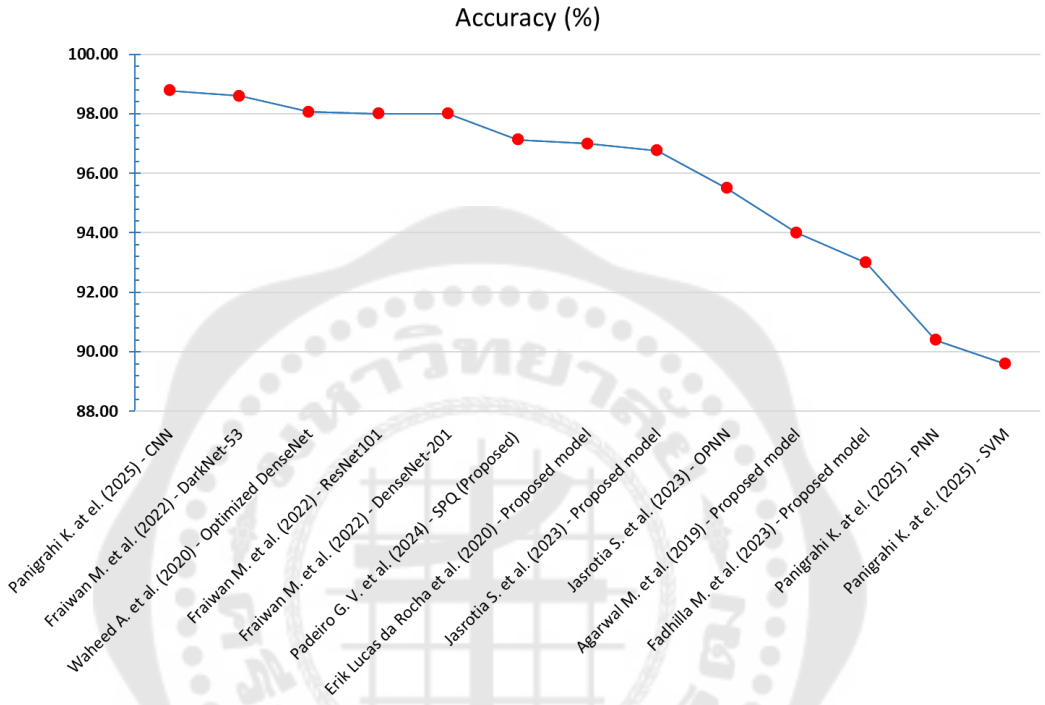
และสุดท้ายจะคำนวณค่า CR จากสมการ (25) ได้

$$CR = \frac{0.0329}{0.58} = 0.0567$$

ซึ่งแสดงให้เห็นความสอดคล้องกันของค่าถ่วงน้ำหนักของแต่ละตัวเกณฑ์ (CR < 0.10) ดังนั้นสามารถนำค่าถ่วงน้ำหนักที่ได้จากการคำนวณในขั้นตอน 2) ไปใช้ในการประเมินเปรียบเทียบเพื่อเลือกแบบจำลองที่เหมาะสมที่สุดในการจำแนกโรคใบข้าวโพดได้

3.4.2.1 การกำหนดคะแนนในเกณฑ์ความแม่นยำ

จากการศึกษางานวิจัยที่เกี่ยวข้องใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน จำแนกโรคใบข้าวโพดที่ผ่านมาได้ พบว่า เมื่อนำผลความแม่นยำของแบบจำลองของงานวิจัยที่ผ่านมา มาพล็อตกราฟ ได้ผลดังภาพประกอบ 50ซึ่งพบว่าจะอยู่ในช่วง 89.6 – 98.78%



ภาพประกอบ 50 กราฟแสดงความแม่นยำในการทำนายผลของแบบจำลองที่ได้จากการศึกษางานวิจัยที่เกี่ยวข้องที่ผ่านมา

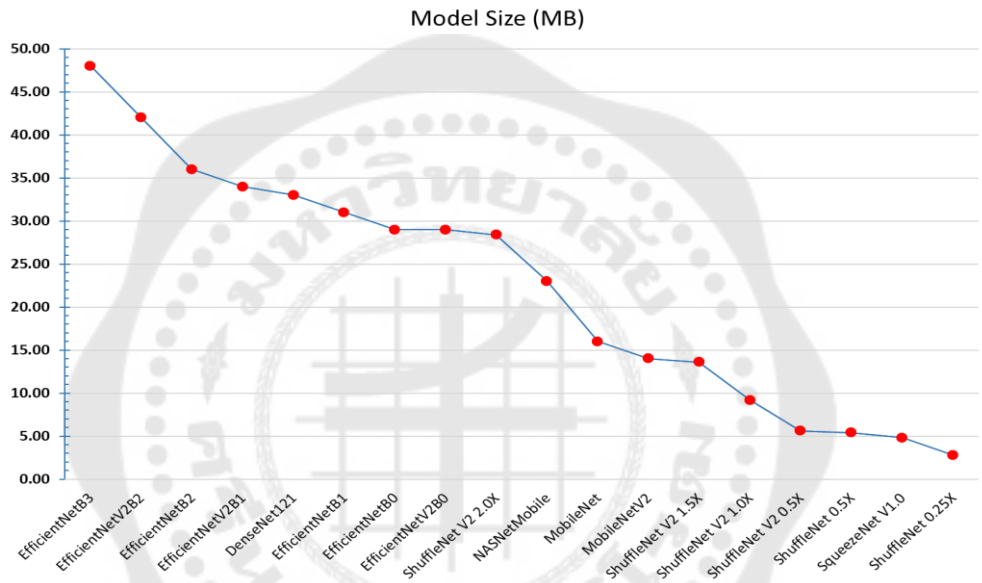
ตาราง 14 ระดับการให้คะแนนตามเกณฑ์ความแม่นยำจากการเรียงลำดับแบบจำลอง

ความแม่นยำของแบบจำลอง	**คะแนน (0-100)**
99.50 – 100%	100
99.00 – 99.49%	95
98.00 – 99.99%	90
97.00 – 97.99%	85
96.00 – 96.99%	80

ดังนั้น สามารถกำหนดช่วงคะแนน 0 - 100 สำหรับความแม่นยำของผลการทดสอบแบบจำลองด้วยชุดข้อมูลทดสอบ ตามตาราง 14

3.4.2.2 การกำหนดคะแนนในเกณฑ์ขนาดแบบจำลอง

จากการศึกษางานวิจัยที่เกี่ยวข้องใช้แบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน จำแนกโรคใบข้าวโพดที่ผ่านมาได้ พบว่า เมื่อนำขนาดของแบบจำลองของงานวิจัยที่ผ่านมามาพล็อตกราฟ ได้ผลดังภาพประกอบ 51 โดยมีขนาดของแบบจำลองอยู่ระหว่าง 2.8 – 48 MB ซึ่งหากมีการลดขนาดโดยใช้เทคนิค Post-training Quantization แบบ Dynamic Range Quantization จะสามารถลดขนาดได้มากที่สุด ถึง 4 เท่า ทำให้ช่วงของขนาดแบบจำลองจะอยู่ในช่วง 0.7 – 12 MB



ภาพประกอบ 51กราฟแสดงขนาดของแบบจำลองจากการศึกษางานวิจัยที่เกี่ยวข้องที่ผ่านมา

ตาราง 15 ระดับการให้คะแนนตามเกณฑ์ขนาดแบบจำลอง (Model Size)

ขนาดแบบจำลอง (MB)	**คะแนน** (0-100)
น้อยกว่า 3 MB	100
3 – 5.99 MB	95
6 – 9.99 MB	90
10 – 15 MB	85
มากกว่า 15 MB	80

ดังนั้น สามารถกำหนดช่วงคะแนน 0 – 100 สำหรับขนาดของแบบจำลองโดยมีการให้คะแนนตามตาราง 15

3.4.2.3 การกำหนดคะแนนในเกณฑ์เวลาประมวลผล

เนื่องจากเวลาที่ใช้ในการประมวลผลของแบบจำลองจะแปรผันตามอุปกรณ์ที่ใช้ในการประมวลผล จึงไม่สามารถกำหนดเกณฑ์ช่วงเวลาที่ใช้ในการประมวลผลแบบจำลองที่แน่นอนได้ ดังนั้น จึงใช้เกณฑ์การให้คะแนน โดยเรียงลำดับเวลาที่ใช้ในการประมวลผลแบบจำลอง ซึ่งสามารถกำหนดช่วงคะแนน 0 – 100 สำหรับเวลาประมวลผล โดยมีการให้คะแนนตามตาราง 16

ตาราง 16 ระดับการให้คะแนนตามเกณฑ์เวลาประมวลผล (Inferencing Time)

ลำดับเวลาประมวลผล	**คะแนน** (0-100)
เวลาในการประมวลผล น้อยที่สุดอันดับ 1	100
เวลาในการประมวลผล น้อยที่สุดอันดับ 2	95
เวลาในการประมวลผล น้อยที่สุดอันดับ 3	90
เวลาในการประมวลผล น้อยที่สุดอันดับ 4	85
เวลาในการประมวลผล น้อยที่สุดอันดับ 5 ขึ้นไป	80

หลังจากให้คะแนนสำหรับแต่ละแบบจำลองทั้งหมดในแต่ละเกณฑ์แล้วให้คำนวณการถ่วงน้ำหนักของแต่ละแบบจำลอง เพื่อประเมินแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดในการวิจัยครั้งนี้

3.5 บทสรุปของกระบวนการ และวิธีการดำเนินการวิจัย

ในบทที่ 3 ได้นำเสนอวิธีการดำเนินงานเพื่อพัฒนาและปรับปรุงแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด โดยเริ่มจากการใช้ข้อมูลจาก PlantVillage Dataset ซึ่งประกอบด้วยภาพถ่ายใบข้าวโพด 3,852 ภาพที่แบ่งเป็น 4 หมวดหมู่ ได้แก่ ใบปกติ โรคใบไหม้จากเชื้อ Cercospora โรคใบไหม้ทางตอนเหนือ และโรคราสนิมข้าวโพด ข้อมูลถูกแบ่งเป็นชุดฝึกฝน 80% และชุดทดสอบ 20% เพื่อนำไปใช้ในการฝึกฝนและประเมินแบบจำลอง CNN ได้รับการพัฒนาโดยใช้ TensorFlow และ Keras Framework พร้อมด้วยการทำ On-the-fly Data Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล และการปรับแต่งค่า Learning Rate รวมถึงการใช้ Early Stopping เพื่อป้องกัน Overfitting

ในขั้นตอนถัดมา มีการปรับปรุงแบบจำลองให้เหมาะสมสำหรับการใช้งานบนอุปกรณ์พกพา โดยใช้เทคนิคต่างๆ ผสมผสานร่วมกัน ได้แก่ การออกแบบ Lightweight Model การทำ Knowledge Distillation และการทำ Post-Training Quantization โดยเป็นการพัฒนาแบบจำลอง Pre-trained ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile แล้วใช้การทำ Post-Training Quantization เพื่อลดขนาดแบบจำลองโดยยังคงความแม่นยำในการจำแนกโรค อีกทั้งมีการพัฒนา Student Model ที่มีขนาดเล็กลงสำหรับการวิจัย ประกอบด้วย แบบจำลองที่ใช้โครงสร้างของการคอนโวลูชันแบบปกติ จำนวน 1 แบบจำลอง และแบบจำลองที่ใช้โครงสร้างของ Depthwise Separable Convolution จำนวน 1 แบบจำลอง

ในการเปรียบเทียบประสิทธิภาพและการประเมินผลแบบจำลองด้วยเกณฑ์สำคัญ ได้แก่ ความแม่นยำ ขนาดแบบจำลอง และเวลาประมวลผล โดยใช้เมทริกซ์ตัดสินใจแบบถ่วงน้ำหนัก เพื่อเปรียบเทียบแบบจำลองต่างๆ และประเมินค่าแบบจำลองที่เหมาะสมที่สุดสำหรับการนำไปใช้ในงานจริงบนอุปกรณ์ที่มีทรัพยากรจำกัด

ในบทถัดไปจะเป็นการผลการทดลองที่ได้จากการศึกษาวิจัยการปรับปรุงแบบจำลองให้เหมาะสมที่สุดสำหรับการจำแนกโรคใบข้าวโพดโดยใช้โครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยมีเนื้อหาเกี่ยวกับการเปรียบเทียบผลการทดสอบประสิทธิภาพความแม่นยำของแบบจำลองที่ได้ฝึกฝนกับชุดข้อมูล PlantVillage Dataset โดยประกอบด้วยผลการทดสอบประสิทธิภาพของแบบจำลองทั้งแบบจำลองที่ไม่ได้รับการปรับปรุง และได้รับการปรับปรุงแบบจำลองแล้ว การเปรียบเทียบขนาดของแบบจำลอง การเปรียบเทียบความเร็วในการประมวลผลของแบบจำลอง และการใช้เมทริกซ์ตัดสินใจแบบถ่วงน้ำหนักเพื่อประเมินค่าแบบจำลองที่เหมาะสมที่สุดสำหรับการนำไปใช้ในงานจริงบนอุปกรณ์ที่มีทรัพยากรจำกัด

บทที่ 4
การทดลอง และผลลัพธ์ของการวิจัย

จากเนื้อหาในบทที่ 3 ได้กล่าวถึงแนวความคิดในการการดำเนินการวิจัยเพื่อพัฒนาและปรับปรุงแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด โดยใช้ข้อมูลจาก PlantVillage Dataset ซึ่งประกอบด้วยภาพถ่ายใบข้าวโพดจำนวน 3,852 ภาพ แบ่งออกเป็น 4 หมวดหมู่ ได้แก่ ใบปกติ, โรคใบไหม้จากเชื้อ Cercospora, โรคใบไหม้ทางตอนเหนือ และโรคราสนิมข้าวโพด ข้อมูลถูกแบ่งออกเป็น ชุดข้อมูลฝึกฝน จำนวน ภาพ (80% ของภาพทั้งหมด) และ ชุดข้อมูลทดสอบ จำนวน ภาพ (20% ของภาพทั้งหมด) เพื่อใช้ในการฝึกและประเมินแบบจำลอง โดยการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด ซึ่งในการกระบวนการฝึกฝนแบบจำลองนั้น เพื่อป้องกัน Overfitting จะใช้กระบวนการ On-the-fly Data Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล มีการใช้ Early Stopping และมีการปรับลดค่า Learning Rate ระหว่างการฝึกอบรมแบบจำลอง

และในบทนี้จะกล่าวถึงการทดลองในพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด และการปรับขนาดของแบบจำลองให้มีความเหมาะสมกับโทรศัพท์มือถือ โดยมีเนื้อหา ดังนี้

1. การทดลอง
2. การปรับปรุงแบบจำลองให้เหมาะสมที่สุด
3. ผลการทดลอง
4. การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือ
5. การเลือกแบบจำลองที่เหมาะสมที่สุดโดยใช้ ใช้ Weighted Decision Matrix

4.1 การทดลอง

4.1.1 การฝึกฝนแบบจำลอง Pre-trained Model

ในการวิจัยครั้งนี้ ได้เลือกแบบจำลอง Pre-trained จำนวน 4 แบบจำลอง ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile

ในการนำแบบจำลองมาใช้ในการทดลองได้ทำการปรับปรุงแบบจำลองแบบ Pre-Trained โดยปรับในส่วนของ Fully Connected Layer โดยให้ใช้ Global Average Pooling เพื่อลดจำนวนของพารามิเตอร์ของแบบจำลอง และปรับในส่วนของ Classification Layer ให้จำแนกออกมาเป็น 4 หมวดหมู่ ได้แก่ ใบปกติ โรคใบไหม้จากเชื้อ Cercospora โรคใบไหม้ทางตอนเหนือ และโรคราสนิมข้าวโพด

ในการฝึกฝนแบบจำลอง Pre-trained ทั้ง 4 แบบจำลอง จะเริ่มต้นจากค่าพารามิเตอร์ของแบบจำลองด้วยค่าพารามิเตอร์ของที่ผ่านการฝึกฝนมาแล้วบนชุดข้อมูล ImageNet และในระหว่างการฝึกฝนแบบจะลองจะมีการปรับค่าพารามิเตอร์ในทุกชั้นของชั้นคอนโวลูชันของทุกแบบจำลอง โดยทำการ Fine-Tuning แบบเต็มรูปแบบ (Full Fine-tuning) วิธีการนี้แตกต่างจากการ Fine-tuning แบบบางส่วนที่มักจะแช่แข็ง (Freeze) ชั้นต้นๆ ของแบบจำลองไว้ เนื่องจากชั้นเหล่านั้นของชั้นคอนโวลูชันจะจับคุณลักษณะพื้นฐานทั่วไปของภาพ เช่น เส้น ขอบ และรูปทรงเรขาคณิตง่ายๆ การปรับค่าพารามิเตอร์ทุกชั้น จะช่วยให้แบบจำลองสามารถเรียนรู้และปรับตัวให้เข้ากับลักษณะเฉพาะของโรคใบข้าวโพดได้อย่างลึกซึ้ง

การเลือกใช้ Optimizer เลือกใช้ ADAM Optimizer สำหรับทุกแบบจำลอง Pre-trained และมีการทดลองเลือกค่า Learning Rate เริ่มต้นในการฝึกฝนแบบจำลองเพื่อให้ได้แบบจำลองที่มีประสิทธิภาพเหมาะสมที่สุด และในระหว่างการฝึกฝนแบบจำลองให้มีการลดค่า Learning Rate ด้วยฟังก์ชัน ReduceLROnPlateau ในไลบรารี keras ซึ่งมีการตั้งค่าพารามิเตอร์เกี่ยวกับ Optimizer ของการฝึกแบบจำลอง Pre-trained แต่ละแบบ ดังตาราง 17

ตาราง 17 แสดงการปรับค่าพารามิเตอร์ในการฝึกฝนแบบจำลองแบบ Pre-trained

แบบจำลอง	Optimizer	Loss Function	Learning Rate	Reduce Learning Rate Factor	Early Stopping	Batch Size
VGG16	Adam optimizer	categorical cross-entropy loss	0.0001	0.2	patience=10	32
DenseNet121	Adam optimizer	categorical cross-entropy loss	0.0001	0.2	patience=10	32
MobileNetV2	Adam optimizer	categorical cross-entropy loss	0.0001	0.5	patience=10	32
NASNet Mobile	Adam optimizer	categorical cross-entropy loss	0.00001	0.5	patience=10	32

4.1.2 การฝึกฝนแบบจำลองสำหรับใช้เป็น Teacher Model

การฝึกฝนแบบจำลองสำหรับใช้เป็น Teacher Model สำหรับการพัฒนาแบบจำลองโดยใช้วิธี Knowledge Distillation ได้เลือกแบบจำลองแบบ Pre-trained ที่อยู่ในไลบรารี Keras ที่มีประสิทธิภาพในการทดสอบกับชุดข้อมูล ImageNet ซึ่งได้เลือกแบบจำลองมาฝึกฝนเพื่อใช้เป็น Teacher Model โดยพิจารณาจาก Top-5 Accuracy ของแบบจำลองที่ได้จากการทดสอบแบบจำลอง กับชุดข้อมูล ImageNet จำนวน 3 แบบจำลอง มาฝึกฝนกับชุดข้อมูลใบข้าวโพด และได้ทดลองฝึกฝนแบบจำลองโดยใช้ค่าพารามิเตอร์ ดังตาราง 18

ตาราง 18 แสดงการปรับค่าพารามิเตอร์ในการฝึกฝนแบบจำลองเพื่อใช้เป็น Teacher Model

แบบจำลอง	Optimizer	Loss Function	Learning Rate	Reduce Learning Rate Factor	Early Stopping	Batch Size
DenseNet201	Adam optimizer	categorical cross-entropy loss	0.00005	0.5	patience=10	128
EfficientNetV2L	Adam optimizer	categorical cross-entropy loss	0.00005	0.2	patience=3	128
EfficientNetV2M	Adam optimizer	categorical cross-entropy loss	0.0001	0.2	patience=3	128

และเมื่อทดสอบแบบจำลองที่ผ่านการฝึกฝนกับชุดข้อมูลฝึกฝน โดยใช้ชุดข้อมูลทดสอบของชุดข้อมูลใบข้าวโพดได้ผลการทดสอบ ดังตาราง 19

ตาราง 19 แสดงผลการทดสอบแบบจำลองเพื่อใช้เป็น Teacher Model กับชุดข้อมูลทดสอบ

แบบจำลอง	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
DenseNet201	99.22	99.23	99.22	99.22
EfficientNetV2L	98.44	98.44	98.44	98.44
EfficientNetV2M	98.44	98.48	98.44	98.45

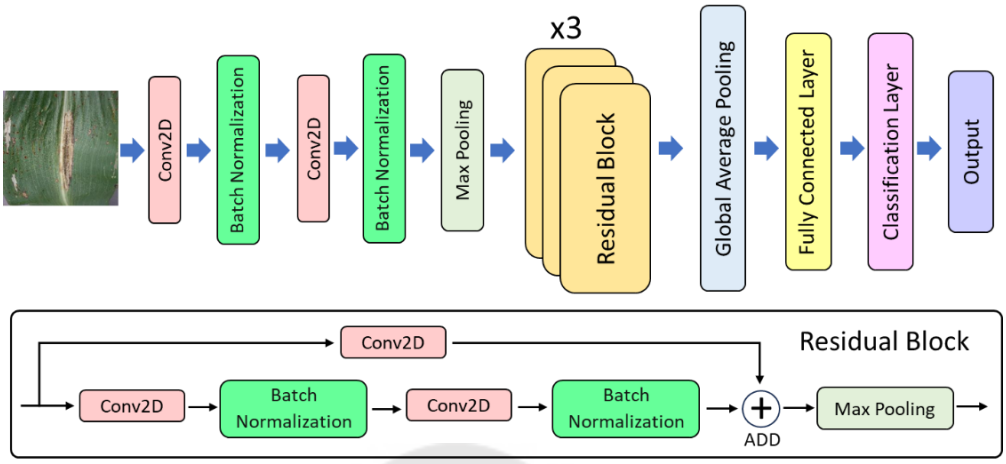
จากผลการทดสอบแบบจำลองที่ผ่านการฝึกฝนเพื่อใช้เป็น Teacher Model ในกระบวนการ Knowledge Distillation พบว่าแบบจำลองทั้ง 3 แบบที่ได้คัดเลือกมาแสดงประสิทธิภาพที่โดดเด่นในการจำแนกโรคใบข้าวโพด โดยเฉพาะอย่างยิ่ง DenseNet201 ที่แสดงผลลัพธ์ยอดเยี่ยมด้วย Accuracy สูงถึง 99.22% พร้อมทั้งค่า Precision Recall และ F1-Score ที่สอดคล้องกันที่ 99.23% 99.22% และ 99.22% ตามลำดับ ขณะที่แบบจำลอง EfficientNetV2L และ EfficientNetV2M ให้ผลลัพธ์ที่ใกล้เคียงกันมากโดยมี Accuracy เท่ากันที่ 98.44% ทั้งสองแบบจำลอง โดย EfficientNetV2M มีค่า Precision และ F1-Score ที่สูงกว่าเล็กน้อยที่ 98.48% และ 98.45% ตามลำดับ เมื่อเทียบกับ EfficientNetV2L ที่มีค่าเท่ากันทั้ง Precision Recall และ F1-Score ที่ 98.44% ผลการทดสอบนี้แสดงให้เห็นว่าแบบจำลอง DenseNet201 มีประสิทธิภาพสูงที่สุดในการจำแนกโรคใบข้าวโพดจากชุดข้อมูลทดสอบ จึงเป็นตัวเลือกที่เหมาะสมที่สุดในการนำไปใช้เป็น Teacher Model สำหรับกระบวนการ Knowledge Distillation เพื่อถ่ายทอดความรู้ไปยัง Student Model ที่มีขนาดเล็กกว่า ซึ่งจะช่วยให้แบบจำลองขนาดเล็กสามารถเรียนรู้และจำแนกโรคใบข้าวโพดได้อย่างมีประสิทธิภาพใกล้เคียงกับแบบจำลองขนาดใหญ่

4.1.3 ออกแบบแบบจำลองสำหรับใช้เป็น Student Model

ในการออกแบบแบบจำลองสำหรับใช้เป็น Student Model ในกระบวนการ Knowledge Distillation มีเป้าหมายสำคัญคือการสร้างแบบจำลองที่มีขนาดเล็กกะทัดรัดแต่ยังคงความสามารถในการจำแนกโรคใบข้าวโพดได้อย่างแม่นยำ ในการวิจัยนี้ได้ออกแบบ Student Model ไว้ 2 แบบที่มีโครงสร้างแตกต่างกัน โดยทั้งสองแบบจำลองได้รับการออกแบบให้มีจำนวนพารามิเตอร์น้อยกว่า Teacher Model อย่างมีนัยสำคัญ แต่ยังคงความสามารถในการสกัดคุณลักษณะที่สำคัญของโรคใบข้าวโพดได้ แบบจำลองทั้งสองมีการใช้เทคนิคการออกแบบโครงสร้างแบบใช้การคอนโวลูชันแบบปกติ และการใช้ Depthwise Separable Convolution แบบจำลองทั้งสองได้รับการออกแบบให้มีขนาดเล็กแต่มีประสิทธิภาพสูง สามารถทำงานได้อย่างรวดเร็วและใช้พลังงานน้อยเมื่อนำไปใช้งานบนอุปกรณ์พกพา ซึ่งเป็นคุณสมบัติสำคัญสำหรับการใช้งานในภาคสนาม

4.1.3.1 Standard Convolution Student Model (STD Conv Student)

แบบจำลองนี้ใช้ชั้นคอนโวลูชันแบบมาตรฐาน (Standard Convolution) ในการสกัดคุณลักษณะ (Feature Extraction) จากภาพตัวอย่าง เพื่อแยกประเภทภาพออกเป็น 4 คลาส โดยรับภาพขนาด 224×224 pixel ที่มี 3 ช่องสี (RGB) เป็นอินพุต โดยมีโครงสร้างของแบบจำลอง ดังภาพประกอบ 52



ภาพประกอบ 52 แสดงโครงสร้างของ Standard Convolution Student Model

แบบจำลองประกอบด้วยบล็อกแรกที่มีชั้น Convolutional จำนวน 2 ชั้น โดยแต่ละชั้นมีฟิลเตอร์จำนวน 24 ตัว ขนาด 3×3 และใช้ padding แบบ 'same' เพื่อรักษาขนาดของข้อมูลให้คงเดิม ฟังก์ชันกระตุ้น ReLU ถูกใช้เพื่อเพิ่มความสามารถในการเรียนรู้รูปแบบที่ซับซ้อน การทำ Batch Normalization ระหว่างชั้นช่วยให้การเทรนมีความเสถียรและเร็วขึ้น และจบด้วย Max Pooling ที่ลดขนาดข้อมูลลงครึ่งหนึ่ง ซึ่งช่วยลดจำนวนพารามิเตอร์และเพิ่มประสิทธิภาพในการเรียนรู้

ในแบบจำลองนี้ใช้ Residual Block จำนวน 3 บล็อก ซึ่งเป็นแนวคิดที่ได้รับความนิยมจากแบบจำลอง ResNet โดยแต่ละบล็อกมีการเพิ่มจำนวนฟิลเตอร์เป็นสองเท่าของบล็อกก่อนหน้า (48 96 และ 192 ตามลำดับ) แนวคิดหลักของ Residual Block คือ การมี Shortcut Connection ที่ข้ามผ่านชั้น Convolutional ไป ทำให้ Gradient สามารถไหลผ่านโครงข่ายได้ดีขึ้นในระหว่างการฝึกฝนแบบจำลอง ช่วยแก้ปัญหา Gradient Vanishing ในโครงข่ายที่ลึก การใช้ Conv2D ขนาด 1×1 ใน Shortcut Connection มีไว้เพื่อปรับขนาดของข้อมูลให้ตรงกับผลลัพธ์จากเส้นทางหลัก ก่อนที่จะนำมารวมกันด้วยการใช้ Add ตามด้วย Max Pooling ที่ช่วยลดขนาดของข้อมูลในแต่ละบล็อก

หลังจากผ่าน Residual Block ทั้งหมด แบบจำลองใช้ Global Average Pooling เพื่อลดมิติของข้อมูลก่อนส่งต่อไปยังชั้น Dense ขนาด 512 โหนด ที่มีฟังก์ชันกระตุ้น ReLU ซึ่งช่วยในการเรียนรู้ลักษณะเฉพาะระดับสูงของภาพ สุดท้าย คือ ชั้น output ขนาด 4 โหนด (ตามจำนวนคลาสที่ต้องการจำแนก) ด้วยฟังก์ชันกระตุ้น Softmax ที่แปลงผลลัพธ์ให้เป็นความน่าจะเป็นที่ภาพจะอยู่ในแต่ละคลาส โดยผลรวมของความน่าจะเป็นทั้งหมดเท่ากับ 1

โครงสร้างแบบจำลองนี้ถูกออกแบบให้มีความสมดุลระหว่างความซับซ้อนและประสิทธิภาพ โดยการเพิ่มจำนวนฟิลเตอร์อย่างต่อเนื่องในแต่ละ Residual Block ช่วยให้แบบจำลองสามารถเรียนรู้ลักษณะเฉพาะที่ซับซ้อนขึ้นเรื่อย ๆ ในขณะที่การใช้ Residual Connection ช่วยให้สามารถเทรนโครงข่ายที่ลึกได้อย่างมีประสิทธิภาพ โดยไม่เกิดปัญหา Gradient Vanishing การออกแบบนี้มีความยืดหยุ่นและสามารถปรับแต่งได้ง่ายสำหรับงานแยกประเภทภาพที่หลากหลาย

แบบจำลอง CNN ที่นำเสนอโดยใช้แนวคิด Residual Network ที่มีการกระจายพารามิเตอร์อย่างมีประสิทธิภาพ โดยมีพารามิเตอร์ทั้งหมด 788,012 พารามิเตอร์ ซึ่งถือว่าประหยัดทรัพยากรเมื่อเทียบกับแบบจำลองสำหรับการรู้จำภาพขนาดใหญ่อย่าง VGG16 (138 ล้านพารามิเตอร์) หรือ ResNet-50 (25 ล้านพารามิเตอร์) โดยมีรายละเอียดจำนวนของพารามิเตอร์ในแต่ละชั้น ตามตาราง 20

การเพิ่มจำนวนฟิลเตอร์เป็นเท่าตัวในแต่ละชั้นมีเหตุผลทางทฤษฎีที่สำคัญ 3 ประการ คือ

1) ชดเชยการลดลงของขนาดพื้นที่จากการใช้ Max Pooling

2) รองรับการเรียนรู้ลักษณะเฉพาะที่ซับซ้อนขึ้นตามลำดับชั้น

3) ชดเชยข้อมูลที่สูญหายจากการลดขนาด

ตาราง 20 แสดงจำนวนพารามิเตอร์ในแต่ละชั้นของแบบจำลอง Standard Convolution Student Model

Layer Name	Layer Type	Output Shape	Parameters
input_layer	InputLayer	(None,224,224,3)	-
conv2d	Conv2D	(None,224,224,24)	672
batch_normalization	BatchNormalization	(None,224,224,24)	96
conv2d_1	Conv2D	(None,224,224,24)	5,208
batch_normalization_1	BatchNormalization	(None,224,224,24)	96
max_pooling2d	MaxPooling2D	(None,224,224,24)	-
conv2d_3	Conv2D	(None,112,112,24)	10,416
batch_normalization_2	BatchNormalization	(None,112,112,48)	192
conv2d_4	Conv2D	(None,112,112,48)	20,784
batch_normalization_3	BatchNormalization	(None,112,112,48)	192
conv2d_2	Conv2D	(None,112,112,48)	1,200

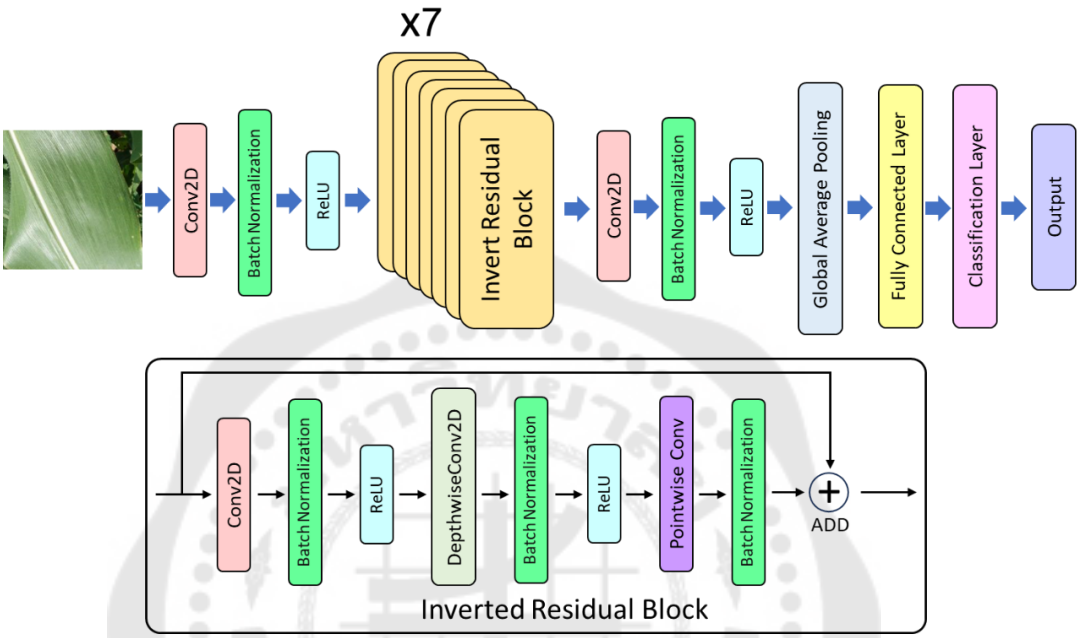
ตาราง 20 (ต่อ)

Layer Name	Layer Type	Output Shape	Parameters
add	Add	(None,112,112,48)	-
max_pooling2d_1	MaxPooling2D	(None,56,56,48)	-
conv2d_6	Conv2D	(None,56,56,96)	41,568
batch_normalization_4	BatchNormalization	(None,56,56,96)	384
conv2d_7	Conv2D	(None,56,56,96)	83,040
batch_normalization_5	BatchNormalization	(None,56,56,96)	384
conv2d_5	Conv2D	(None,56,56,96)	4,704
add_1	Add	(None,56,56,96)	-
max_pooling2d_2	MaxPooling2D	(None,28,28,96)	-
conv2d_9	Conv2D	(None,28,28,192)	166,080
batch_normalization_6	BatchNormalization	(None,28,28,192)	768
conv2d_10	Conv2D	(None,28,28,192)	331,968
batch_normalization_7	BatchNormalization	(None,28,28,192)	768
conv2d_8	Conv2D	(None,28,28,192)	18,624
add_2	Add	(None,28,28,192)	-
max_pooling2d_3	MaxPooling2D	(None,14,14,192)	-
global_average_pooling2d	GlobalAveragePooling2D	(None,192)	-
dense	Dense	(None,512)	98,816
dense_1	Dense	(None,4)	2,052

4.1.3.2 Depthwise Separable Convolution Student Model (DW Conv Student)

แบบจำลอง Depthwise Separable Convolution Student Model ที่นำเสนอในงานวิจัยนี้เป็นการพัฒนาต่อยอดจากสถาปัตยกรรม MobileNetV2 ดั้งเดิม ให้มีขนาดของแบบจำลองเล็กลงอย่างมาก โดยมุ่งเน้นการเพิ่มประสิทธิภาพการประมวลผลและลดการใช้ทรัพยากรสำหรับการคำนวณ โครงสร้างหลักของแบบจำลองนี้อาศัยหลักการสำคัญของ MobileNetV2 คือ การใช้ Inverted Residual Block และ Depthwise Separable Convolution ซึ่งเป็นเทคนิคพื้นฐานที่ช่วยลดจำนวนพารามิเตอร์และการคำนวณโดยไม่สูญเสียความแม่นยำในการจำแนกภาพมากเกินไป

โครงสร้างของแบบจำลองถูกออกแบบให้รับภาพอินพุตขนาด 224x224 พิกเซล ที่มี 3 ช่องสี (RGB) และสามารถจำแนกภาพออกเป็น 4 คลาสที่แตกต่างกัน ตามการจำแนกโรคของใบข้าวโพดที่ใช้ในการทดลอง แสดงดังภาพประกอบ 53



ภาพประกอบ 53 แสดงโครงสร้างแบบจำลอง Depthwise Separable Convolution Student Model

ชั้นแรกของแบบจำลองเริ่มต้นด้วยชั้นคอนโวลูชันมาตรฐานที่มี 32 ฟิลเตอร์ ขนาด 3x3 พร้อมกับ stride เท่ากับ 2 ซึ่งจะลดขนาดภาพลงเหลือครึ่งหนึ่ง (112x112) ในขั้นตอนแรก ช่วยลดปริมาณการคำนวณในชั้นถัดไป หลังจากนั้นแบบจำลองจะประกอบด้วย 7 ชุดของ Inverted Residual Block ที่มีการกำหนดค่าพารามิเตอร์แตกต่างกันไปตามลำดับชั้น แต่ละชุดมีการกำหนดค่า 4 พารามิเตอร์หลัก คือ ค่า expansion factor (t) จำนวนช่องสัญญาณเอาต์พุต (c) จำนวนบล็อกที่ซ้ำกัน (n) และค่า stride สำหรับบล็อกแรก (s) เพื่อให้ได้สมดุลระหว่างความซับซ้อนของแบบจำลองและความสามารถในการสกัดคุณลักษณะที่สำคัญจากภาพ

หัวใจสำคัญของแบบจำลอง คือ การใช้ Inverted Residual Block ที่มีโครงสร้างพิเศษต่างจาก Residual Block แบบดั้งเดิมของ ResNet โดยแทนที่จะบีบข้อมูลก่อนแล้วค่อยขยายกลับ (compress-then-expand) กลับกลายเป็นการขยายข้อมูลก่อนแล้วค่อยบีบข้อมูล (expand-then-compress) ซึ่งในแต่ละ Inverted Residual Block จะประกอบด้วย 3 ส่วนหลัก คือ Expansion

Layer Depthwise Convolution และ Pointwise Linear Projection โดยส่วนแรก Expansion Layer ใช้คอนโวลูชัน 1x1 เพื่อเพิ่มจำนวนช่องสัญญาณให้มากขึ้นตามค่า expansion factor ช่วยให้มีพื้นที่มากขึ้นสำหรับการสกัดคุณลักษณะที่ซับซ้อน จากนั้นส่วน Depthwise Convolution จะทำการคอนโวลูชันแยกตามช่องสัญญาณโดยใช้เคอร์เนลขนาด 3x3 ซึ่งช่วยลดการคำนวณลงอย่างมากเมื่อเทียบกับการคอนโวลูชันแบบปกติ และสุดท้ายส่วน Pointwise Linear Projection จะใช้คอนโวลูชัน 1x1 อีกครั้งเพื่อลดจำนวนช่องสัญญาณกลับมา ซึ่งลักษณะพิเศษของส่วนนี้คือการไม่ใช้ฟังก์ชันกระตุ้น ReLU เพื่อป้องกันการสูญเสียข้อมูลสำคัญในมิติที่มีความละเอียดต่ำ

นอกจากนี้ Inverted Residual Block ยังมีการเชื่อมต่อแบบ Residual Connection (หรือที่เรียกว่า Skip Connection) โดยจะทำการเชื่อมต่อระหว่างอินพุตและเอาต์พุตของบล็อกเมื่อเข้าเงื่อนไขสองข้อคือ ค่า stride ต้องเท่ากับ 1 (ไม่มีการลดขนาดภาพ) และจำนวนช่องสัญญาณอินพุตต้องเท่ากับจำนวนช่องสัญญาณเอาต์พุต การเชื่อมต่อแบบนี้ช่วยแก้ปัญหา Gradient Vanishing ในแบบจำลองที่มีความลึกมาก ช่วยให้การฝึกฝนแบบจำลองมีเสถียรภาพและมีประสิทธิภาพมากขึ้น

สำหรับแต่ละบล็อกในแบบจำลอง DW Conv Student จะมีการใช้ฟังก์ชันกระตุ้น ReLU6 แทนที่ ReLU แบบดั้งเดิม โดย ReLU6 เป็นการปรับปรุงจาก ReLU ปกติด้วยการจำกัดค่าสูงสุดไว้ที่ 6 (f(x) = min(max(0, x), 6)) ซึ่งมีข้อดีในการช่วยให้แบบจำลองทำงานได้ดีบนอุปกรณ์ที่มีความแม่นยำต่ำ (Low-precision) และเหมาะสมกับการประมวลผลบนอุปกรณ์พกพาที่มีข้อจำกัดด้านทรัพยากร นอกจากนี้ยังมีการใช้ BatchNormalization ในทุกชั้นของแบบจำลองเพื่อเพิ่มความเสถียรในการฝึกสอน เร่งความเร็วในการเรียนรู้ และช่วยลดปัญหา overfitting โดยทำหน้าที่เป็นตัวควบคุม (Regularizer) ไปในตัว

โครงสร้างของแบบจำลอง Depthwise Separable Convolution Student Model ยังมีการจัดวางชั้นต่าง ๆ อย่างเหมาะสมเพื่อให้เกิดประสิทธิภาพสูงสุด โดยทั้ง 7 ชุดของ Inverted Residual Block ถูกออกแบบให้มีการลดขนาดภาพลงอย่างค่อยเป็นค่อยไปจาก 112x112 เป็น 56x56, 28x28 และสุดท้ายเป็น 14x14 ซึ่งช่วยให้แบบจำลองสามารถเรียนรู้คุณลักษณะทั้งในระดับต่ำ (low-level features) เช่น เส้น ขอบ และพื้นผิว ไปจนถึงคุณลักษณะระดับสูง (high-level features) เช่น โครงสร้างของวัตถุ ได้อย่างมีประสิทธิภาพ การลดขนาดภาพนี้ยังช่วยลดจำนวนการคำนวณในชั้นท้ายๆ ของแบบจำลองอีกด้วย

เพื่อเพิ่มความสามารถในการสกัดคุณลักษณะระดับสูง หลังจากผ่าน 7 ชุดของ Inverted Residual Block แล้ว แบบจำลองยังมีการเพิ่มชั้นคอนโวลูชันขนาด 1x1 อีกหนึ่งชั้นที่มี 320

ฟิลเตอร์ ซึ่งช่วยเพิ่มความสามารถในการจำแนกคุณลักษณะที่ซับซ้อนก่อนที่จะส่งต่อไปยังชั้นสุดท้ายของแบบจำลอง ต่อจากนั้นจะใช้ GlobalAveragePooling2D เพื่อลดมิติของข้อมูลจาก (14, 14, 320) เหลือเพียงเวกเตอร์ขนาด 320 หน่วย ซึ่งช่วยลดจำนวนพารามิเตอร์ในเลเยอร์สุดท้ายอย่างมากและทำให้แบบจำลองมีความทนทานต่อการเปลี่ยนแปลงตำแหน่งของวัตถุในภาพ และสุดท้ายคือ Classification Layer ที่มีโหนดเท่ากับจำนวนคลาสที่ต้องการจำแนก (4 คลาส) พร้อมฟังก์ชันกระตุ้น Softmax เพื่อแปลงค่าให้เป็นความน่าจะเป็นสำหรับการจำแนกประเภทของใบข้าวโพดที่เป็นโรค

แบบจำลอง DW Conv Student ที่พัฒนาขึ้นมีโครงสร้างที่ซับซ้อนประกอบด้วยชั้นการประมวลผลจำนวนมาก ซึ่งถูกออกแบบโดยการปรับปรุงแบบจำลอง MobileNetV2 ให้มีขนาดเล็กลงอย่างมาก โดยมีจำนวนพารามิเตอร์ที่ต้องเรียนรู้ทั้งสิ้นประมาณ 838,980 พารามิเตอร์ ซึ่งถือว่าน้อยมากเมื่อเทียบกับแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันทั่วไป โดยเกิดจากการใช้เทคนิค Depthwise Separable Convolution ซึ่งแบ่งการคอนโวลูชันออกเป็นสองขั้นตอน คือ Depthwise Convolution และ Pointwise Convolution ทำให้ลดการคำนวณลงอย่างมาก โดยมีรายละเอียดจำนวนของพารามิเตอร์ในแต่ละชั้น ตามตาราง 21

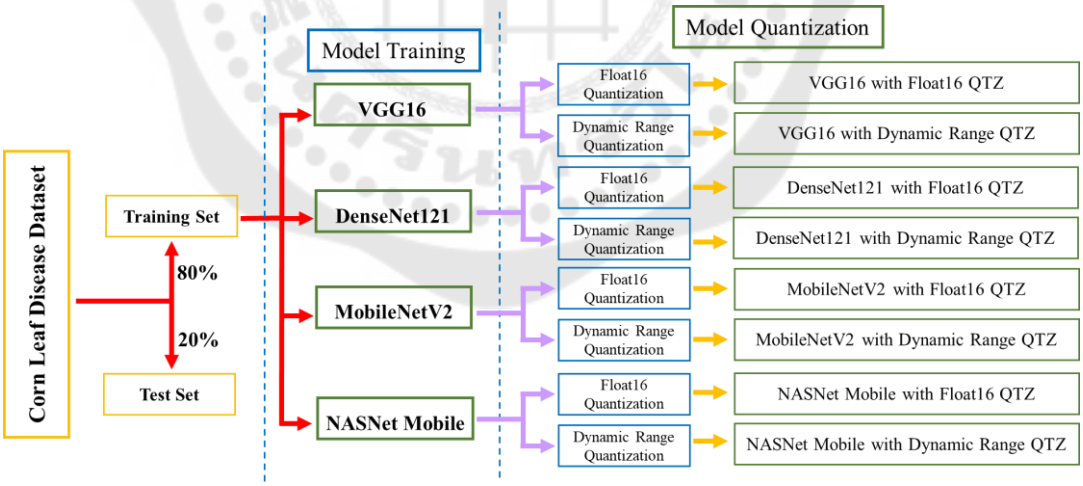
ตาราง 21 แสดงรายละเอียดจำนวนของพารามิเตอร์ในแต่ละชั้นของแบบจำลอง Depthwise Separable Convolution Student Model

Layer Name	Blocks	Total Parameters
Initial Convolutional Layer		992
Inverted Residual Block Stage 1	1	992
Inverted Residual Block Stage 2	2	15,024
Inverted Residual Block Stage 3	3	58,704
Inverted Residual Block Stage 4	3	114,480
Inverted Residual Block Stage 5	2	137,008
Inverted Residual Block Stage 6	2	263,168
Inverted Residual Block Stage 7	1	194,848
Final Layers and Classification Layers		53,764
Total Parameters		838,980

การพัฒนาแบบจำลองสำหรับการจำแนกโรคใบข้าวโพดสองแบบที่มีขนาดกะทัดรัด ได้แก่ STD Conv Student และ DW Conv Student ซึ่งทั้งสองแบบจำลองถูกออกแบบให้เป็น Student Model ในกระบวนการ Knowledge Distillation โดยมีเป้าหมายให้มีประสิทธิภาพใกล้เคียงกับ Teacher Model แต่มีขนาดเล็กกว่ามาก แบบจำลอง STD Conv Student ใช้ Standard Convolution ในการสกัดคุณลักษณะจากภาพ มีจำนวนพารามิเตอร์ทั้งสิ้น 788,012 พารามิเตอร์ ส่วนแบบจำลอง DW Conv Student พัฒนาจากสถาปัตยกรรม MobileNetV2 ใช้เทคนิค Depthwise Separable Convolution และ Inverted Residual Block ซึ่งช่วยลดการคำนวณลงอย่างมาก มีจำนวนพารามิเตอร์ 838,980 พารามิเตอร์ ทั้งสองแบบจำลองใช้เทคนิคต่างๆ เช่น Residual Connection Batch Normalization และ Global Average Pooling เพื่อเพิ่มประสิทธิภาพในการเรียนรู้และความแม่นยำ ทั้งสองแบบจำลองถูกออกแบบให้มีจำนวนพารามิเตอร์ไม่เกิน 1 ล้านพารามิเตอร์ เพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพบนอุปกรณ์พกพา ซึ่งเป็นคุณสมบัติสำคัญสำหรับการใช้งานในภาคสนาม ทำให้เหมาะสมต่อการนำไปใช้จำแนกโรคใบข้าวโพดในพื้นที่จริง

4.2 การปรับปรุงแบบจำลองให้เหมาะสมที่สุด

4.2.1 การทำ Quantization กับแบบจำลอง Pre-trained Model



ภาพประกอบ 54 แสดงขั้นตอนการทำ Quantization กับแบบจำลอง Pre-trained Model

กระบวนการในการทำ Quantization กับแบบจำลอง Pre-trained Model นั้นจะนำแบบจำลองที่ได้จากการฝึกฝนโดยใช้ชุดข้อมูลฝึกฝน แล้วทดสอบประสิทธิภาพแล้วมาทำ Quantization โดยกระบวนการในการพัฒนาแบบจำลองเป็นดังภาพประกอบ 54

ในกระบวนการปรับรูปชนิดของข้อมูลของพารามิเตอร์ของแบบจำลองในการทดลองนี้จะใช้ฟังก์ชันการปรับรูปแบบข้อมูลจากไลบรารี TensorFlow Lite ซึ่งเป็นเทคนิคที่มีประสิทธิภาพในการลดขนาดของแบบจำลอง โดยการแปลงค่าพารามิเตอร์ของแบบจำลองจากรูปแบบที่มีความละเอียดสูง (Floating-point 32 บิต) ให้อยู่ในรูปแบบที่ใช้พื้นที่จัดเก็บน้อยลง ซึ่งในงานวิจัยนี้ได้ใช้เทคนิค Post-training Quantization ได้แก่ Float16 Quantization และ Dynamic Range Quantization

4.2.1.1 Float16 Quantization (FP16)

การทำ Float16 Quantization เป็นการแปลงค่าพารามิเตอร์ของแบบจำลองจากรูปแบบ Float32 (32-bit) ให้เป็น Float16 (16-bit) ซึ่งช่วยลดขนาดของแบบจำลองลงได้ประมาณ 50% โดยที่ความแม่นยำในการจำแนกยังคงใกล้เคียงเดิม

4.2.1.2 Dynamic Range Quantization (DR)

การทำ Dynamic Range Quantization เป็นการแปลงค่าพารามิเตอร์ของแบบจำลองให้อยู่ในรูปแบบของ Integer 8-bit (INT8) ซึ่งช่วยลดขนาดของแบบจำลองลงได้ถึง 75% เมื่อเทียบกับแบบจำลองต้นฉบับ

4.2.2 การทำ Knowledge Distillation

Knowledge Distillation เป็นเทคนิคที่มีความสำคัญอย่างยิ่งในการพัฒนาแบบจำลองโครงข่ายประสาทเทียมให้มีขนาดเล็กลงแต่ยังคงรักษาประสิทธิภาพไว้ได้ หลักการของเทคนิคนี้คือการถ่ายทอดความรู้จากแบบจำลองขนาดใหญ่ที่มีความซับซ้อนและมีประสิทธิภาพสูง (Teacher Model) ไปยังแบบจำลองที่มีขนาดเล็กกว่า (Student Model) เพื่อให้แบบจำลองขนาดเล็กสามารถเลียนแบบพฤติกรรมและความสามารถในการจำแนกของแบบจำลองขนาดใหญ่ได้ ในการวิจัยนี้ได้เลือกใช้แบบจำลอง DenseNet201 เป็น Teacher Model เนื่องจากมีประสิทธิภาพสูงในการจำแนกโรคใบข้าวโพด โดยมีค่า Accuracy สูงถึง 99.22% พร้อมทั้งค่า Precision Recall และ F1-Score ที่สอดคล้องกันที่ 99.23% 99.22% และ 99.22% ตามลำดับ ซึ่งแสดงให้เห็นว่าแบบจำลองนี้มีความแม่นยำสูงมากในการจำแนกโรคใบข้าวโพดทั้ง 4 ประเภท

สำหรับ Student Model ในงานวิจัยนี้ได้ออกแบบไว้ 2 รูปแบบที่มีโครงสร้างแตกต่างกัน ได้แก่ STD Conv Student และ DW Conv Student โดยทั้งสองแบบจำลองมีจำนวนพารามิเตอร์ที่น้อยกว่า Teacher Model อย่างมาก คือประมาณ 788,012 และ 838,980 พารามิเตอร์ตามลำดับ เมื่อเทียบกับ DenseNet201 ที่มีพารามิเตอร์มากกว่า 20 ล้านพารามิเตอร์

การทำ Knowledge Distillation ในงานวิจัยนี้ใช้เทคนิคที่เรียกว่า Response-based Knowledge Distillation ซึ่งเป็นการถ่ายทอดความรู้ผ่านผลลัพธ์ของการทำนาย (Soft Targets)

จาก Teacher Model ไปยัง Student Model โดยใช้ Temperature Scaling เพื่อปรับการกระจายตัวของ Soft Targets ให้เหมาะสม กระบวนการนี้ช่วยให้ Student Model ไม่เพียงแต่เรียนรู้ว่าภาพใดเป็นโรคประเภทใด แต่ยังเรียนรู้ถึงความไม่แน่นอนและความสัมพันธ์ระหว่างคลาสต่างๆ ที่ Teacher Model ได้เรียนรู้ไว้ และในระหว่างการฝึกฝนแบบจำลองมีการใช้เทคนิคการเพิ่มข้อมูล (Data Augmentation) และการปรับ Learning rate เพื่อเพิ่มประสิทธิภาพในการเรียนรู้

หลังจากฝึกฝนแบบจำลอง Student Model ทั้ง 2 แบบจำลอง ได้แก่ STD Conv Student และ DW Conv Student แล้ว จะใช้เทคนิค Post-training Quantization ได้แก่ Float16 Quantization และ Dynamic Range Quantization กับแบบจำลองทั้งสอง เพื่อปรับขนาดของแบบจำลองให้มีขนาดเล็กลง แต่ยังคงมีประสิทธิภาพที่ดีอยู่

4.3 ผลการทดลอง

4.3.1 ผลการฝึกผลแบบจำลอง

จากการทดลองฝึกฝนแบบจำลองแบบ Pre-trained จำนวน 4 แบบจำลอง ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile บนแพลตฟอร์ม Google Colab โดยใช้ GPU T4 Runtime ได้ผลการฝึกฝนแบบจำลอง ดังตาราง 22

ตาราง 22 แสดงผลการฝึกฝนแบบจำลอง Pre-trained Model

Model	Training Scores (%)				Training Time (HH:MM:SS)
	Accuracy	Precision	Recall	F1-Score	
VGG16	98.44	98.49	98.44	98.45	01:09:06
DenseNet121	99.09	99.09	99.09	99.09	00:58:29
MobileNetV2	98.44	98.46	98.44	98.44	00:46:30
NASNet Mobile	98.18	98.21	98.18	98.19	02:17:59

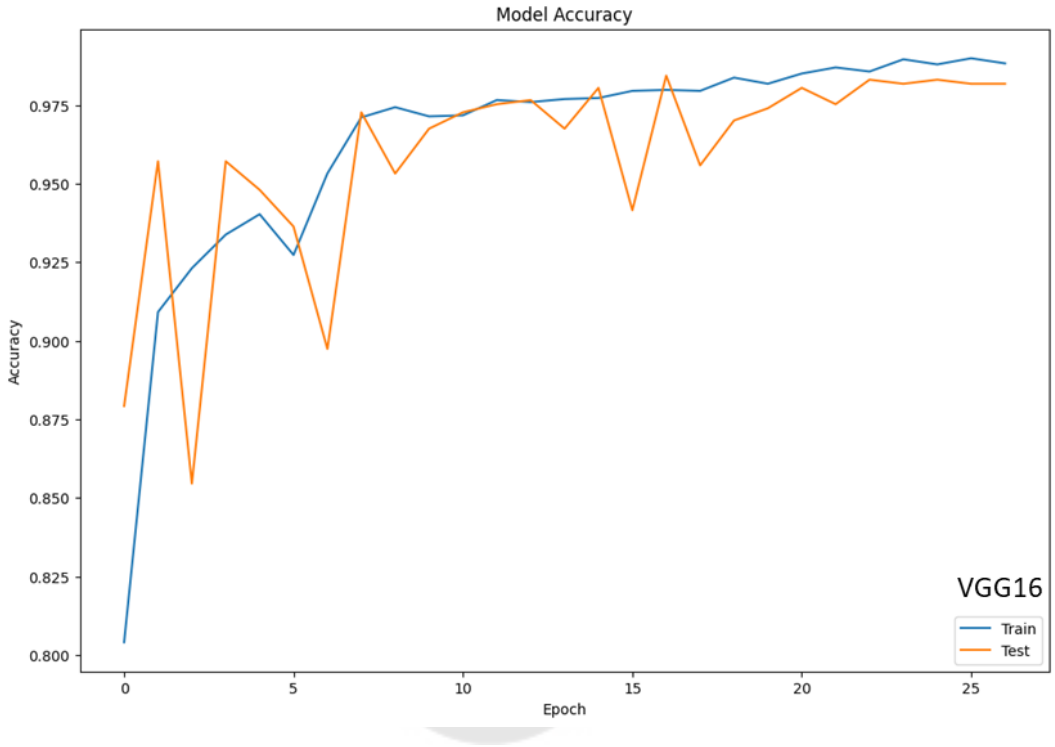
จากตาราง 22แสดงให้เห็นว่า DenseNet121 ให้ผลลัพธ์ที่ดีที่สุดในด้านความแม่นยำ โดยมีค่า Training Accuracy Precision Recall และ F1-Score เท่ากันที่ 99.09% ซึ่งสูงกว่าแบบจำลองอื่นๆ ทั้งหมด ค่า และใช้เวลาในการฝึกฝนประมาณ 58 นาที 29 วินาที

แบบจำลองที่มีค่าความแม่นยำรองลงมา คือ MobileNetV2 และ VGG16 มีค่า Training Accuracy เท่ากันที่ 98.44% แสดงให้เห็นถึงความสมดุลในการจำแนกทุกคลาสของโรคใบข้าวโพด MobileNetV2 ยังใช้เวลาในการฝึกฝนเพียง 46 นาที 30 วินาที ซึ่งถือว่ารวดเร็วเมื่อเทียบ

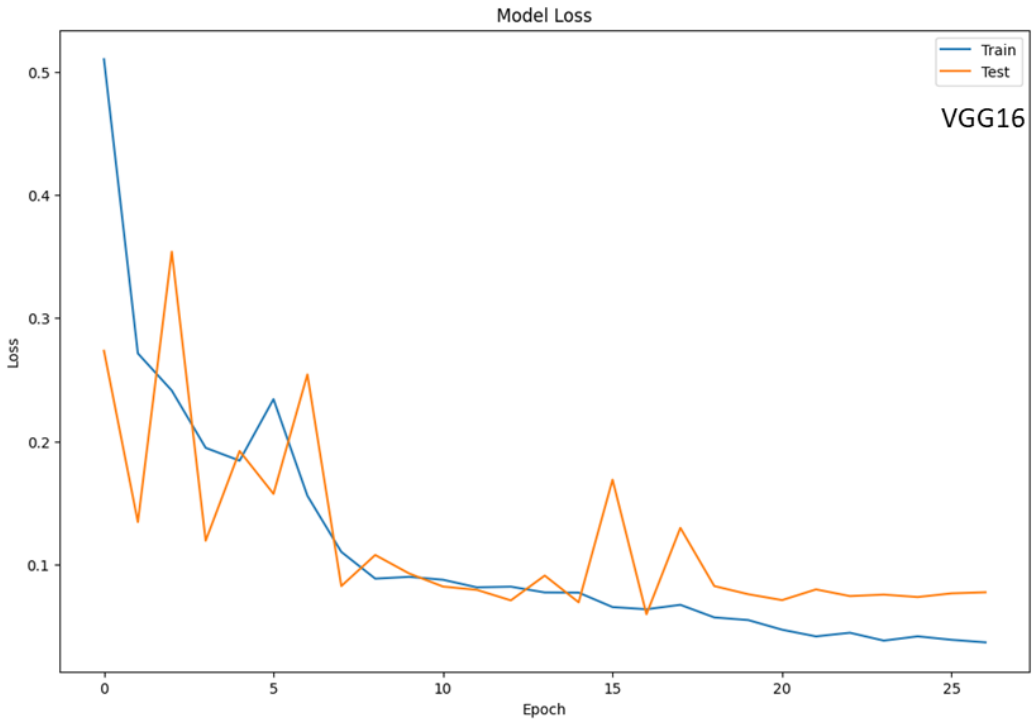
กับแบบจำลองอื่นๆ และ VGG16 เป็นแบบจำลองที่มีโครงสร้างดั้งเดิมและซับซ้อนกว่า ใช้เวลาในการฝึกฝน 1 ชั่วโมง 9 นาที 6 วินาที ซึ่งสะท้อนถึงความซับซ้อนและจำนวนพารามิเตอร์ที่มากกว่าของแบบจำลองนี้

ส่วน NASNet Mobile มีค่า Training Accuracy ต่ำที่สุดในบรรดาแบบจำลองทั้งหมดที่ 98.18% และใช้เวลาในการฝึกฝนนานที่สุดถึง 2 ชั่วโมง 17 นาที 59 วินาที ซึ่งเป็นเวลาที่มากกว่าแบบจำลอง

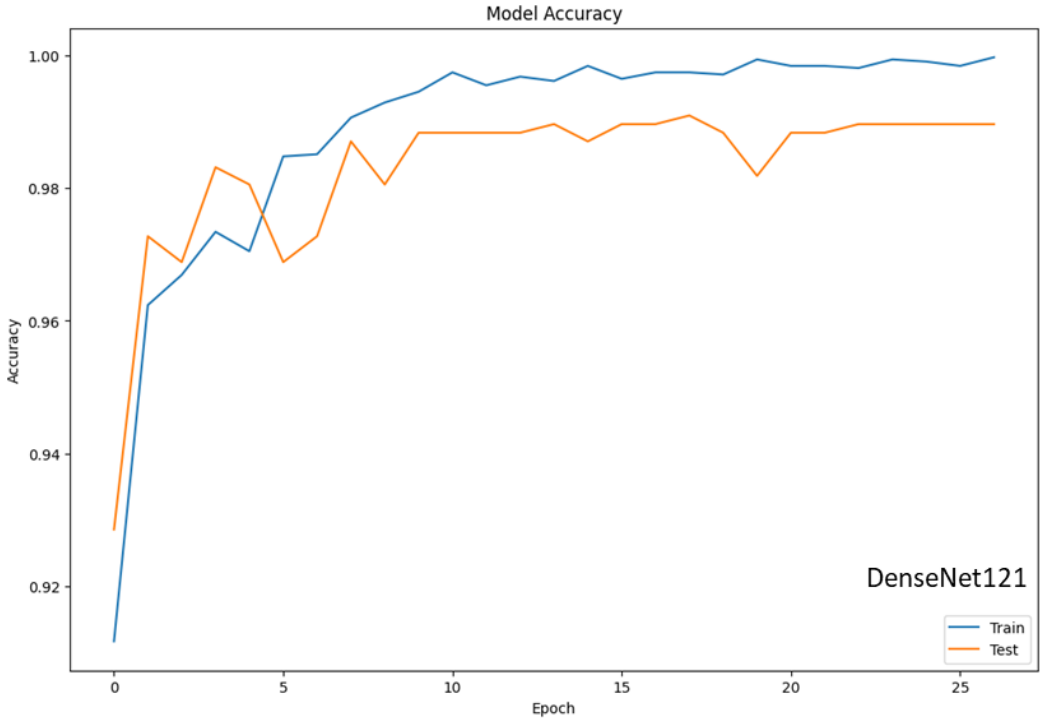
กราฟแสดง Training Accuracy และ Training Loss ของแต่ละแบบจำลองในขณะฝึกฝน แสดงดังภาพประกอบ 55 – ภาพประกอบ 62



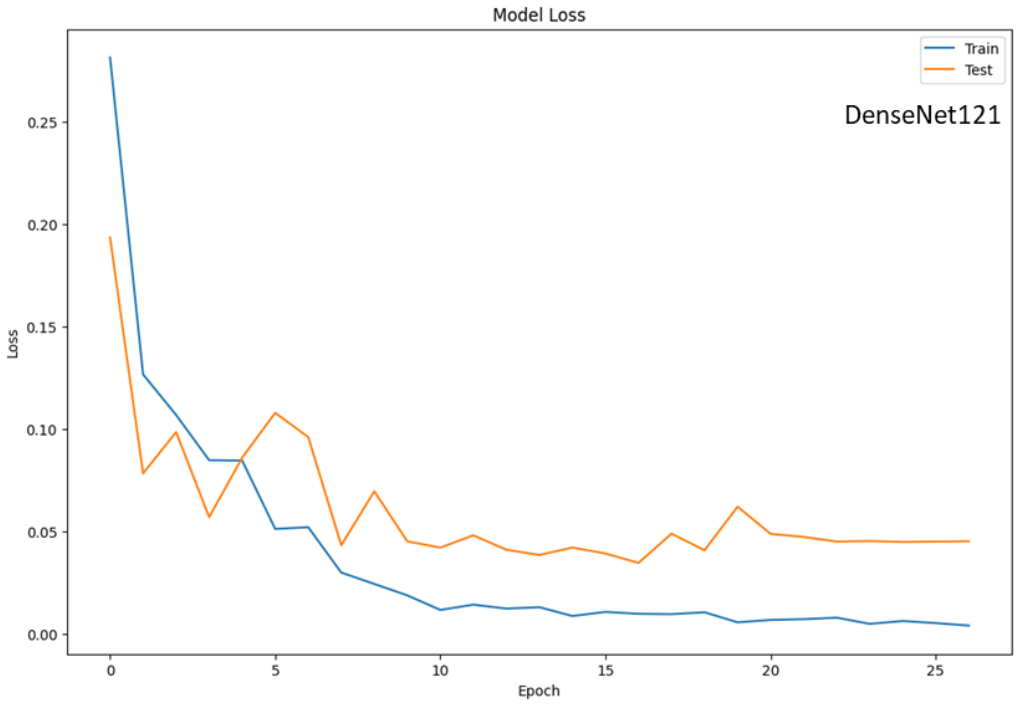
ภาพประกอบ 55 แสดง Training Accuracy ของแบบจำลอง VGG16



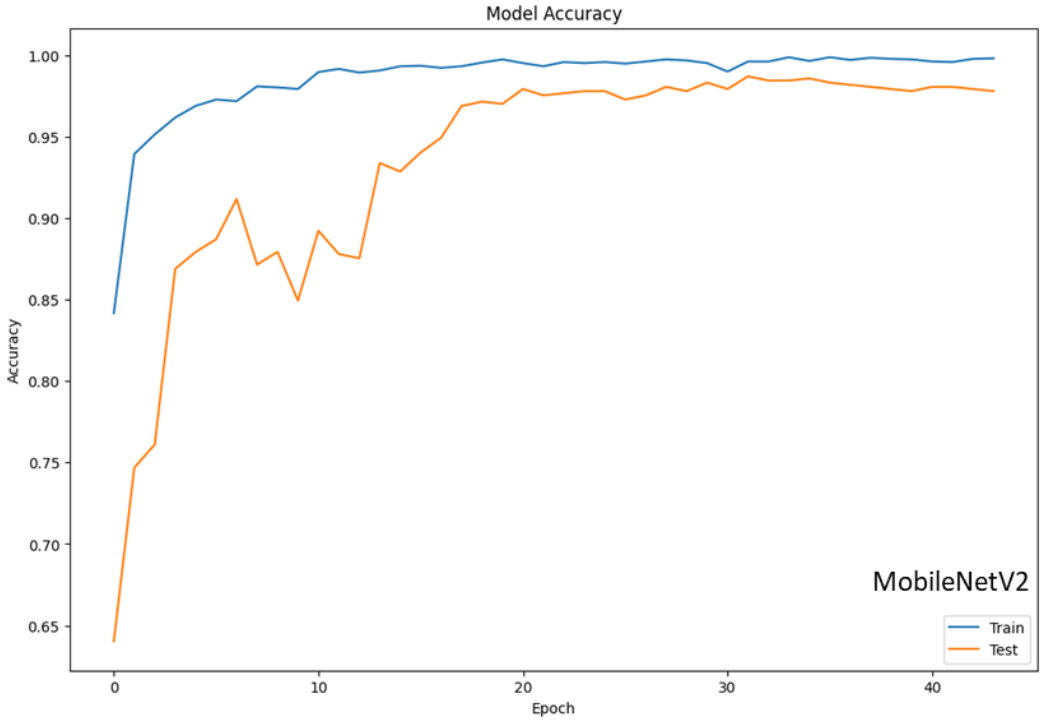
ภาพประกอบ 56 แสดง Training Loss ของแบบจำลอง VGG16



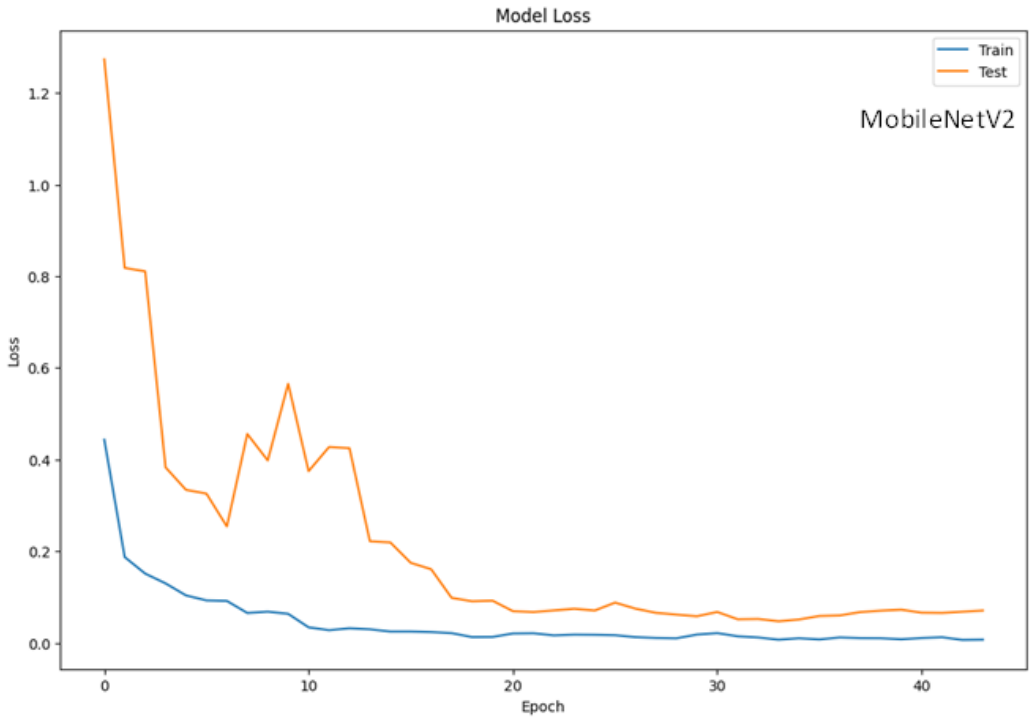
ภาพประกอบ 57 แสดง Training Accuracy ของแบบจำลอง DenseNet121



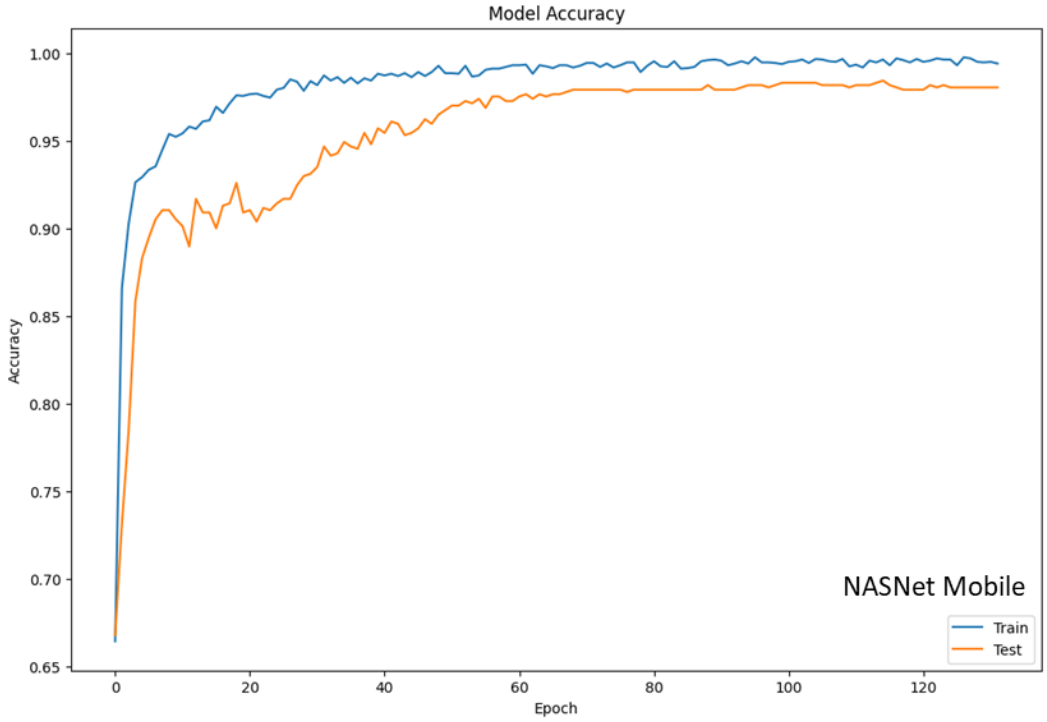
ภาพประกอบ 58 แสดง Training Loss ของแบบจำลอง DenseNet121



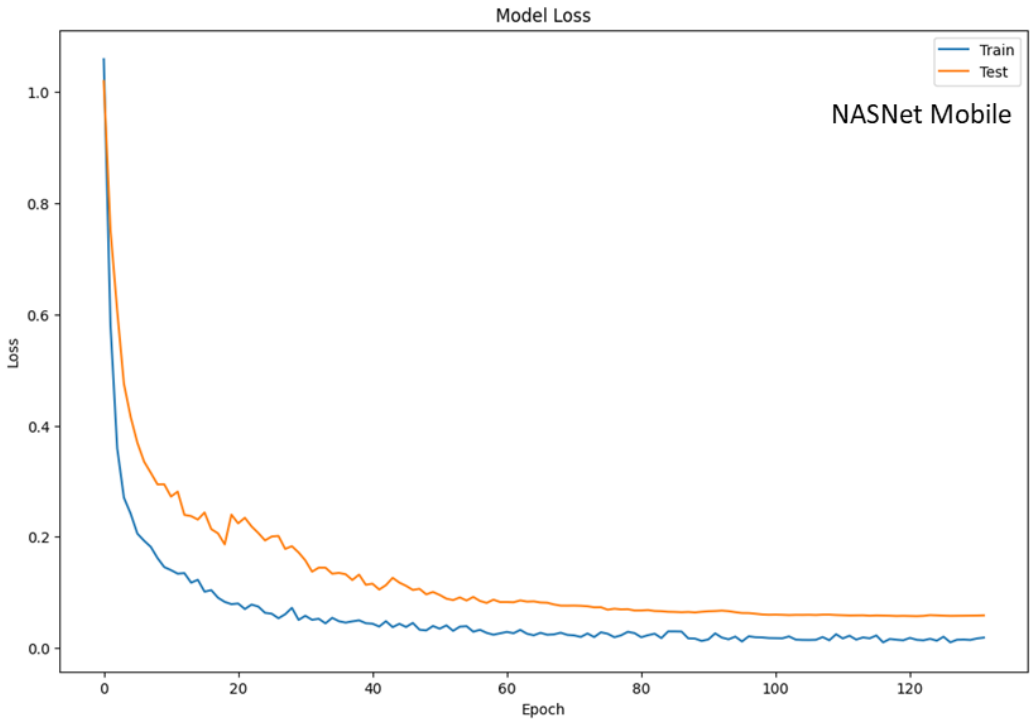
ภาพประกอบ 59 แสดง Training Accuracy ของแบบจำลอง MobileNetV2



ภาพประกอบ 60 แสดง Training Loss ของแบบจำลอง MobileNetV2



ภาพประกอบ 61 แสดง Training Accuracy ของแบบจำลอง NASNet Mobile



ภาพประกอบ 62 แสดง Training Loss ของแบบจำลอง NASNet Mobile

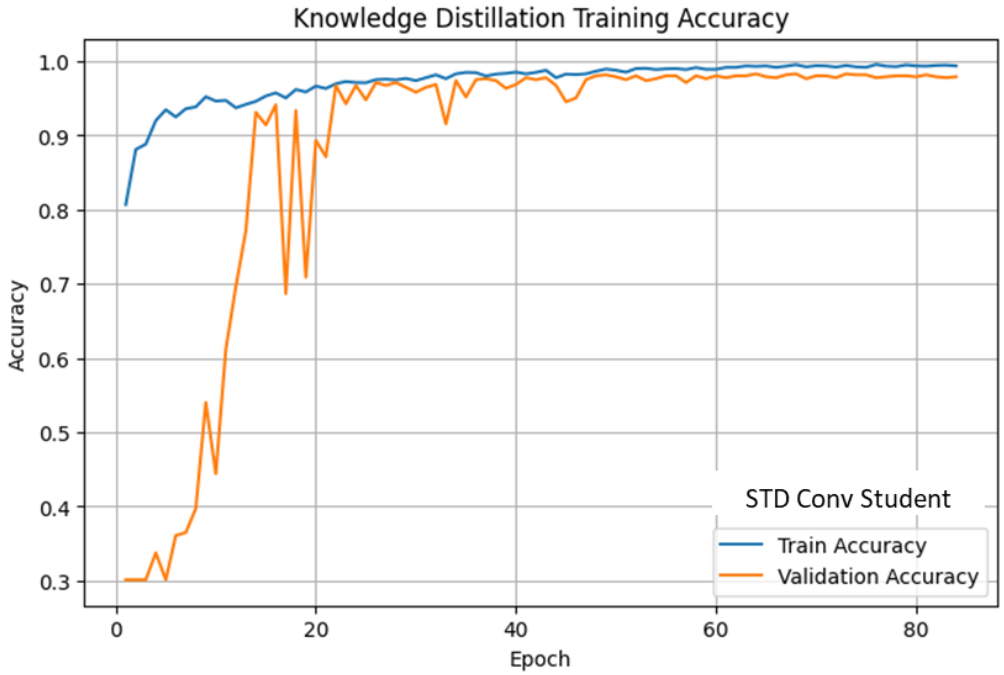
ผลการฝึกฝนแบบจำลองโดยใช้เทคนิคการทำ Knowledge Distillation กับแบบจำลองที่พัฒนาขึ้นจำนวน 2 แบบจำลอง ได้แก่ STD Conv Student และ DW Conv Student ให้ผลการฝึกฝน ดังตาราง 23

ตาราง 23 แสดงผลการฝึกฝนแบบจำลองโดยใช้เทคนิคการทำ Knowledge Distillation

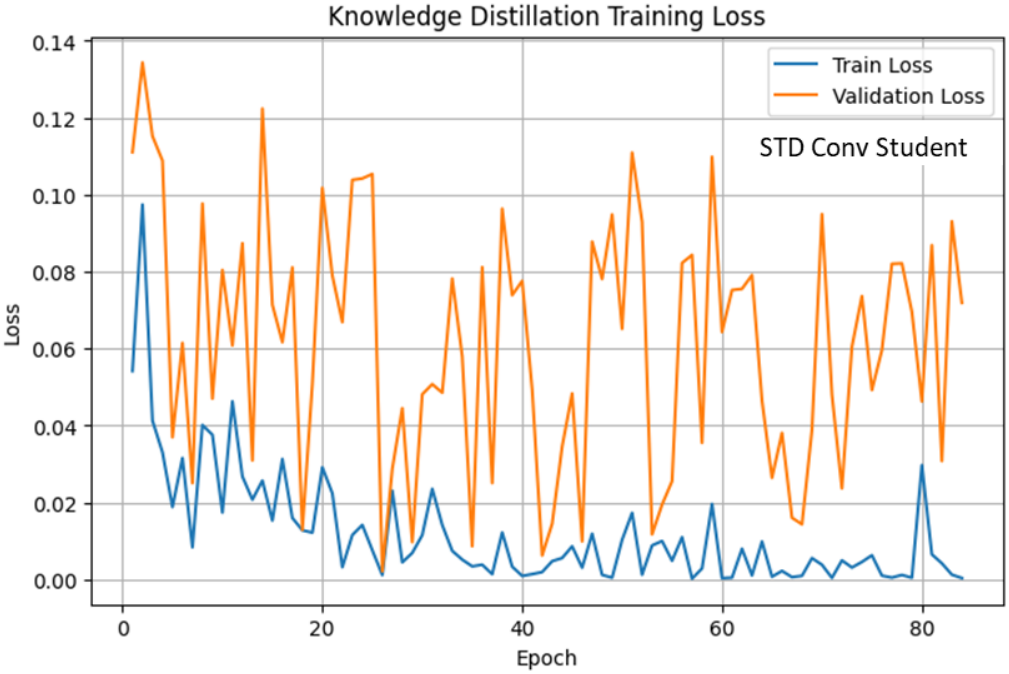
Model	Training Scores (%)			
	Accuracy	Precision	Recall	F1-Score
STD Conv Student	98.31	98.31	98.31	98.31
DW Conv Student	98.70	98.70	98.70	98.70

ผลการฝึกฝนพบว่า DW Conv Student มีประสิทธิภาพเหนือกว่าเล็กน้อย ด้วยค่า Training Accuracy Precision Recall และ F1-Score ที่เท่ากัน 98.70% เทียบกับ STD Conv Student ที่มีผลค่า Training Accuracy Precision Recall และ F1-Score ที่ 98.31%

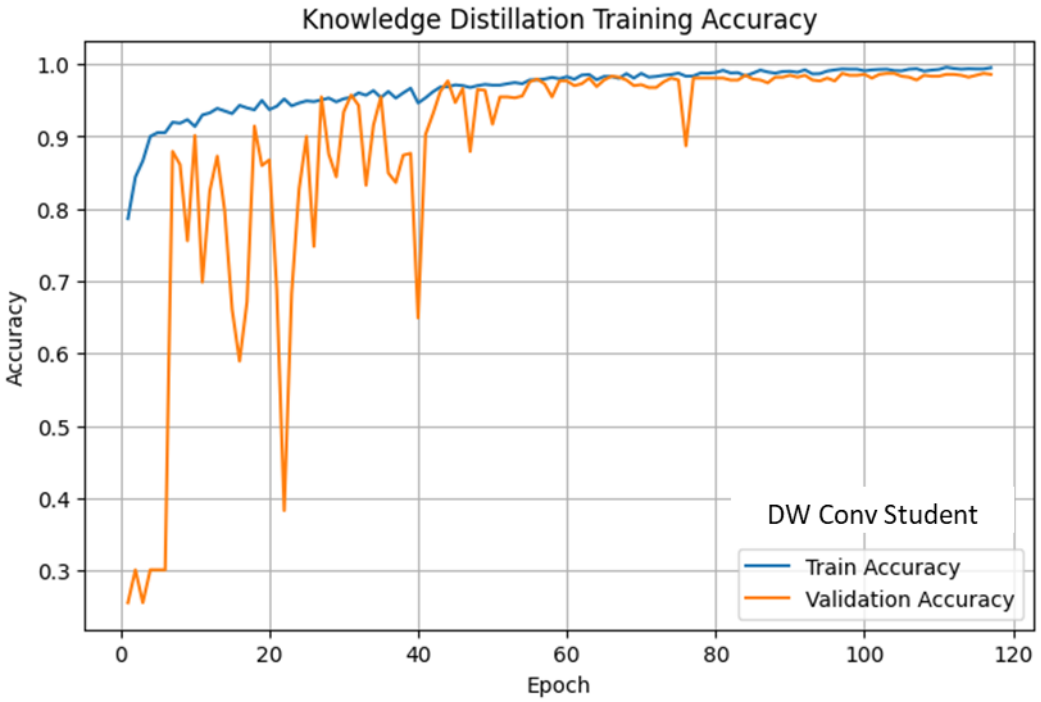
กราฟแสดง Training Accuracy และ Training Loss ของแต่ละแบบจำลองในขณะฝึกฝนโดยใช้เทคนิคการทำ Knowledge Distillation แสดงดังภาพประกอบ 63– ภาพประกอบ 66



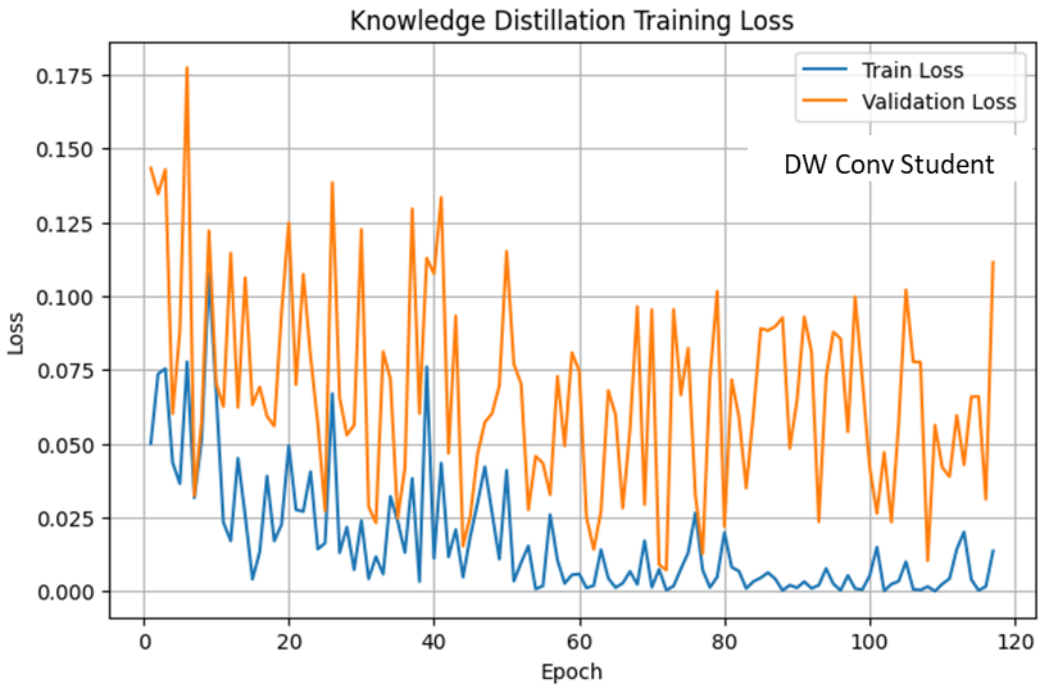
ภาพประกอบ 63 แสดง Training Accuracy ของแบบจำลอง STD Conv Student



ภาพประกอบ 64 แสดง Training Loss ของแบบจำลอง STD Conv Student



ภาพประกอบ 65 แสดง Training Accuracy ของแบบจำลอง DW Conv Student



ภาพประกอบ 66 แสดง Training Loss ของแบบจำลอง DW Conv Student

4.3.2 ผลการทดสอบแบบจำลองกับชุดข้อมูลทดสอบ

หลังจากที่ได้ทำการฝึกฝนแบบจำลองด้วยชุดข้อมูลฝึกฝนจนได้ค่าพารามิเตอร์ที่เหมาะสมแล้ว ขั้นตอนต่อไปเป็นการนำแบบจำลองเหล่านั้นมาทดสอบกับชุดข้อมูลทดสอบเพื่อประเมินความสามารถในการทำนายของแบบจำลอง การทดสอบนี้ได้ดำเนินการบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime ผลการทดสอบที่แสดงในตาราง 24 โดยแสดงข้อมูลในด้านขนาดของแบบจำลอง ความแม่นยำในการจำแนกภาพโรคจากใบข้าวโพด และเวลาที่ใช้ในการประมวลผลต่อภาพ ซึ่งปัจจัยเหล่านี้มีความสำคัญในการประเมินความเหมาะสมของแบบจำลองสำหรับการนำไปใช้งานจริงในภาคสนามที่อาจมีข้อจำกัดด้านทรัพยากรในการประมวลผล และการตอบสนองต่อการประมวลผลภาพที่รวดเร็ว

เมื่อพิจารณาด้านความแม่นยำในการจำแนกภาพของแบบจำลองที่ไม่ผ่านการทำปรับปรุงแบบจำลอง พบว่าแบบจำลอง MobileNetV2 มีความแม่นยำในการจำแนกภาพโรคจากใบข้าวโพดสูงสุด โดยได้ผลการทดสอบ มี ค่า Accuracy Precision Recall และ F1-Score ที่เท่ากัน 99.09% ตามมาด้วยแบบจำลอง DenseNet121 และ DW Conv Distillation from DenseNet201 ตามลำดับ

และเมื่อมีการปรับปรุงแบบจำลองโดยการทำ Quantization พบว่า แบบจำลอง MobileNetV2 ที่ผ่านการทำ Dynamic Range Quantization ให้ผลลัพธ์ที่โดดเด่นที่สุด ด้วยค่า Accuracy Precision Recall และ F1-Score ที่เท่ากัน 99.22% ซึ่งสูงกว่าแบบจำลองอื่นๆ ทั้งหมด ซึ่งน่าสนใจว่าหลังจากการทำ Quantization ค่าความแม่นยำของ MobileNetV2 กลับเพิ่มขึ้นเล็กน้อยจาก 99.09% เป็น 99.22%

ในด้านขนาดของแบบจำลอง การทำ Quantization แสดงให้เห็นผลลัพธ์ที่น่าประทับใจอย่างยิ่ง โดยเฉพาะ Dynamic Range Quantization ที่สามารถลดขนาดของแบบจำลองลงได้ถึง 75% เมื่อเทียบกับแบบจำลองต้นฉบับ ตัวอย่างเช่น STD Conv Student ที่ทำ Dynamic Range Quantization มีขนาดเพียง 0.811 MB และ DW Conv Student ที่ทำ Dynamic Range Quantization มีขนาดเพียง 0.974 MB ซึ่งเป็นขนาดที่เล็กมากและเหมาะสมอย่างยิ่งสำหรับการใช้งานบนอุปกรณ์พกพา

ด้านเวลาในการประมวลผล (Inference Time) ก็มีการปรับปรุงอย่างมีนัยสำคัญหลังจากการทำ Quantization โดยเฉพาะ Float16 Quantization ที่ช่วยลดเวลาในการประมวลผลลงอย่างมาก โดยเฉพาะแบบจำลอง MobileNetV2 ที่ทำ Float16 Quantization มีเวลาในการประมวลผลเพียง 26.01 มิลลิวินาที (millisecond) ต่อภาพ เร็วขึ้นประมาณ 6 เท่า เมื่อเทียบกับเวลา

ประมวลผลของแบบจำลองปกติ และ DW Conv Distillation ที่ทำ Float16 Quantization มีเวลาเพียง 25.59 มิลลิวินาทีต่อภาพ ซึ่งเร็วกว่าแบบจำลองอื่น ๆ อย่างมีนัยสำคัญ

Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลองโดยใช้ชุดข้อมูลทดสอบแสดงดังภาพประกอบ 67 ก) - ฉ)

4.4 การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือ

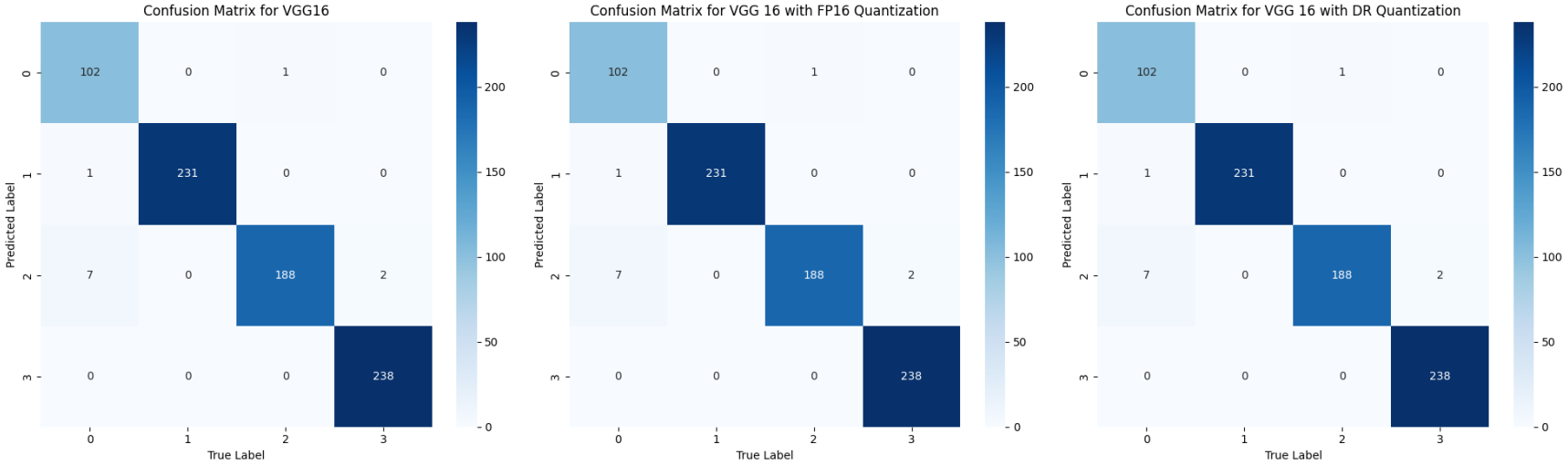
การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือเป็นขั้นตอนสำคัญในการประเมินความเป็นไปได้ในการนำแบบจำลองไปใช้งานจริงในภาคสนาม เนื่องจากแม้ว่าแบบจำลองจะแสดงประสิทธิภาพที่ดีในสภาพแวดล้อมการทดสอบบน Google Colab แต่การทำงานบนอุปกรณ์โทรศัพท์มือถือซึ่งมีข้อจำกัดด้านทรัพยากรและกำลังประมวลผลอาจให้ผลลัพธ์ที่แตกต่างกัน การทดสอบบนอุปกรณ์จริงจึงช่วยยืนยันความเหมาะสมของแบบจำลองสำหรับการใช้งานในสถานการณ์จริง

ตาราง 24 แสดงผลการประเมินแบบจำลองกับชุดข้อมูลทดสอบ

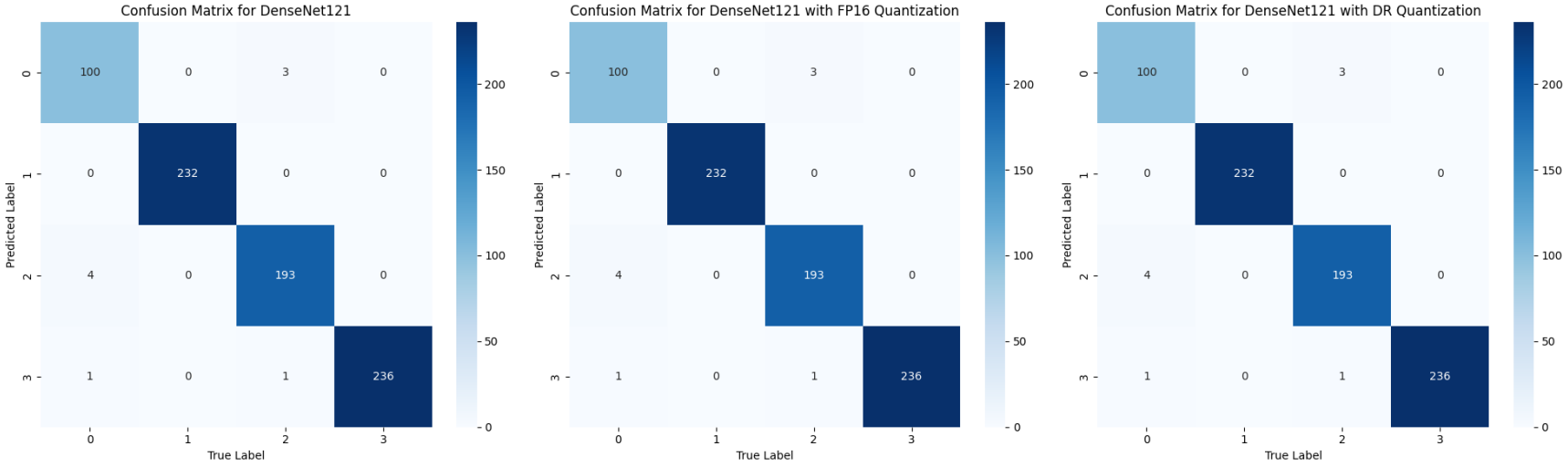
แบบจำลอง	model size (MB)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (msec/image)
VGG16	58.15	98.57	98.63	98.57	98.58	1187.42
VGG 16 with FP16 Quantization	29.1	98.57	98.63	98.57	98.58	714.58
VGG 16 with DR Quantization	14.6	98.57	98.63	98.57	98.58	643.52
DenseNet121	30.87	98.83	98.84	98.83	98.84	371.82
DenseNet121 with FP16 Quantization	15.4	98.83	98.84	98.83	98.84	150.71
DenseNet121 with DR Quantization	8.1	98.83	98.84	98.83	98.84	159.17
MobileNetV2	13.63	99.09	99.09	99.09	99.09	155.71
MobileNetV2 with FP16 Quantization	6.8	99.09	99.09	99.09	99.09	26.01
MobileNetV2 with DR Quantization	3.7	99.22	99.22	99.22	99.22	35.25
NASNet Mobile	20.43	97.92	97.94	97.92	97.93	245.82
NASNet Mobile with FP16 Quantization	10.4	97.92	97.94	97.92	97.93	52.21
NASNet Mobile with DR Quantization	5.8	97.92	97.95	97.92	97.93	73.00
STD Conv Distillation from DenseNet201	3.01	98.05	98.23	98.05	98.08	238.09
STD Conv Distillation from DenseNet201 with FP16 Quantization	1.5	98.05	98.23	98.05	98.08	94.24

ตาราง 24 (ต่อ)

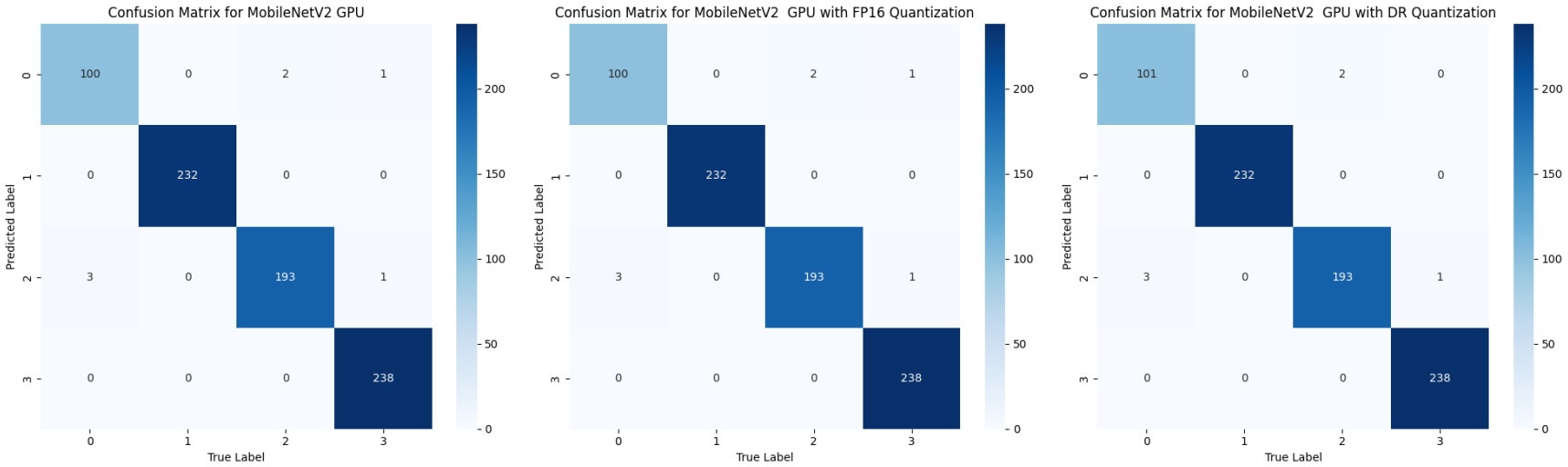
แบบจำลอง	model size (MB)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (msec/image)
STD Conv Distillation from DenseNet201 with DR Quantization	0.811	98.18	98.33	98.18	98.21	82.16
DW Conv Distillation from DenseNet201	3.2	98.70	98.73	98.70	98.71	163.70
DW Conv Distillation from DenseNet201 with FP16 Quantization	1.6	98.70	98.73	98.70	98.71	25.59
DW Conv Distillation from DenseNet201 with DR Quantization	0.974	98.70	98.73	98.70	98.71	43.15



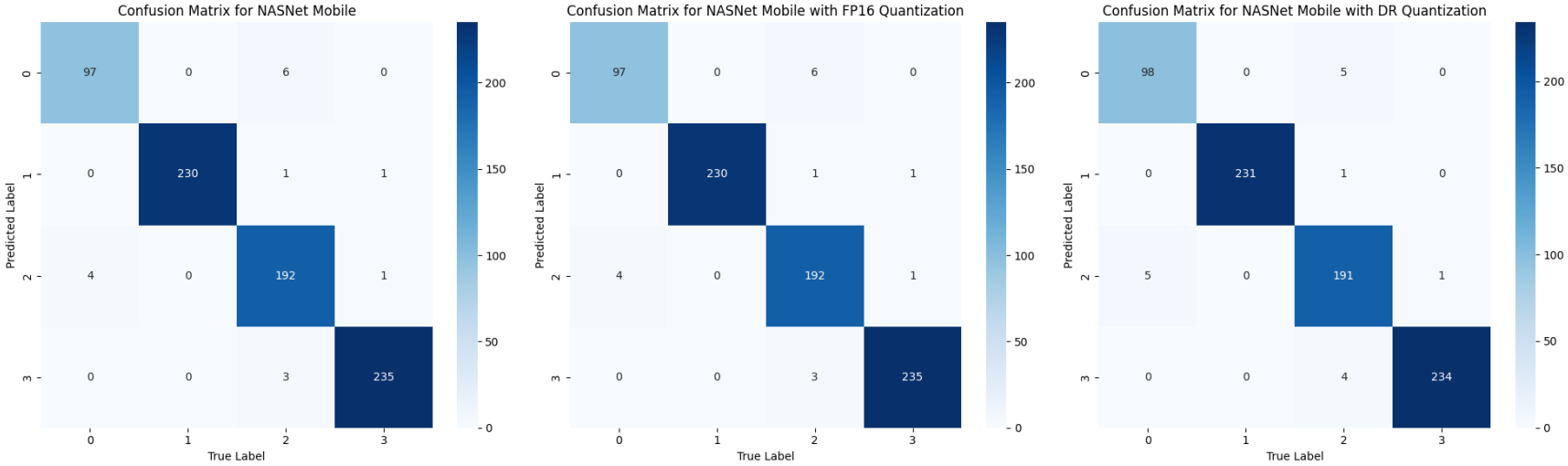
ก) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง VGG16



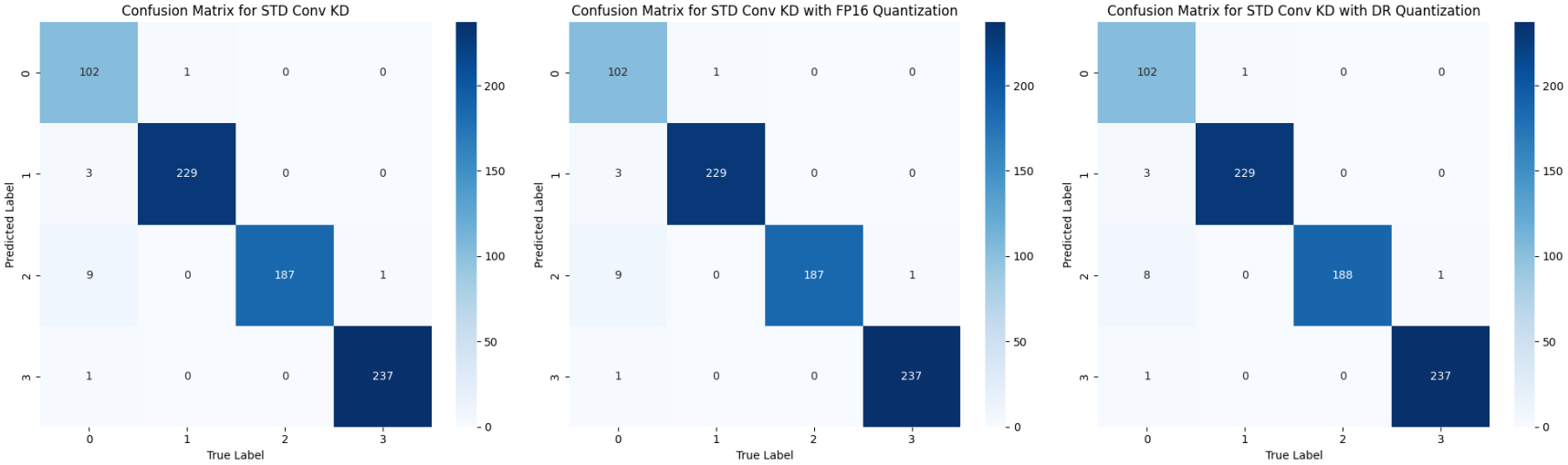
ข) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง DenseNet121



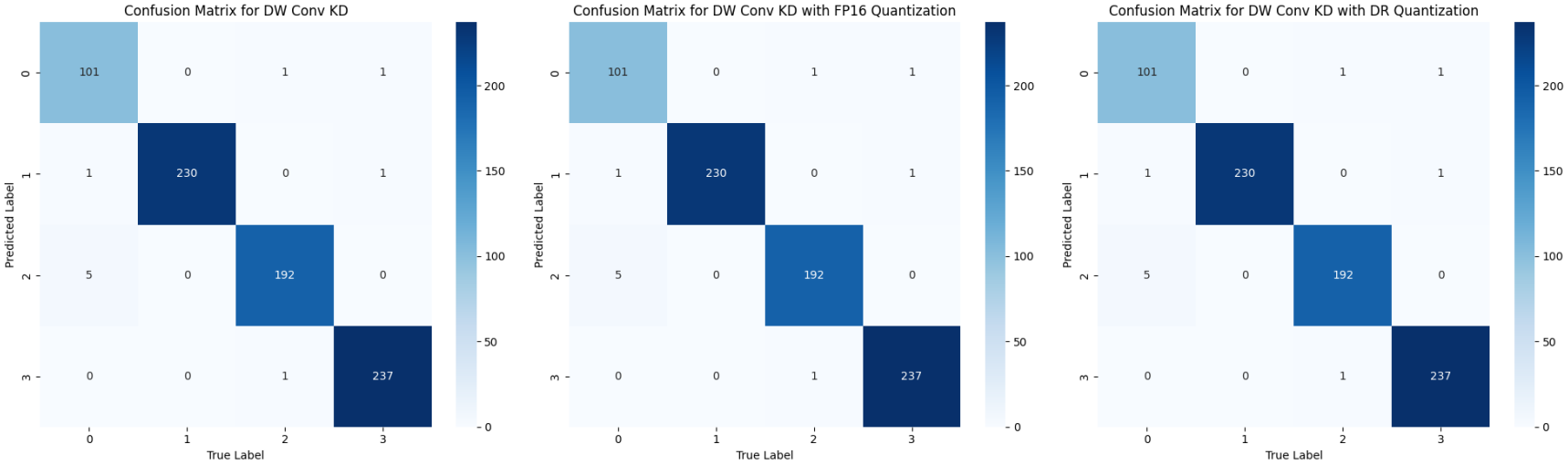
ค) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง MobileNetV2



ง) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง NASNet Mobile



จ) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง STD Conv Distillation from DenseNet201



ฉ) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง DW Conv Distillation from DenseNet201

ภาพประกอบ 67 แสดง Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลองโดยใช้ชุดข้อมูลทดสอบ

ในการทดสอบแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันที่ถูกปรับให้เหมาะสมแล้ว มาทดสอบบนโทรศัพท์มือถือ ได้เลือกแบบจำลองที่ถูกปรับปรุงโดยใช้เทคนิค Post-Training Quantization ด้วยวิธีการ Float16 Quantization แต่เนื่องจากข้อจำกัดของเครื่องมือที่ใช้ในการพัฒนาแอปพลิเคชันโดยใช้ Flutter (version 3.108.0) และภาษา Dart (version 3.108.2) ซึ่งรองรับการทำงานกับ TensorFlow Lite เฉพาะที่มีรูปแบบข้อมูลพารามิเตอร์เป็นแบบ Floating-point ไม่สามารถรองรับแบบจำลองที่ผ่านปรับปรุงโดยใช้เทคนิคการทำ Dynamic Range Quantization ที่ใช้ข้อมูลแบบ Integer จึงต้องเลือกแบบจำลองที่ถูกปรับปรุงโดยใช้เทคนิค Post-Training Quantization ด้วยวิธีการ Float16 Quantization มาทดสอบบนอุปกรณ์โทรศัพท์มือถือ ซึ่งแบบจำลองที่เลือกมาทดสอบจำนวน 6 แบบจำลอง ได้แก่

1. VGG 16 with FP16 Quantization
2. DenseNet121 with FP16 Quantization
3. MobileNetV2 with FP16 Quantization
4. NASNet Mobile with FP16 Quantization
5. STD Conv KD with FP16 Quantization
6. DW Conv KD with FP16 Quantization

อุปกรณ์โทรศัพท์มือถือที่ใช้ในการทดสอบครั้งนี้ คือ SAMSUNG Galaxy A06 ซึ่งมีคุณลักษณะทางเทคนิคตามตาราง 25

ตาราง 25 แสดงคุณลักษณะของโทรศัพท์มือถือที่ใช้ทดสอบการประมวลผล

Title	Details
Model	SAMSUNG Galaxy A06
Screen Size	6.7 inch
Chip	Mediatek Helio G85
Display	720 x 1600 (HD+) LCD 60Hz
Memory	RAM 4GB / ROM 64GB
Operating System	Android 14
Front Camera	8MP
Back Camera	50MP (Main) + 2MP (Depth)

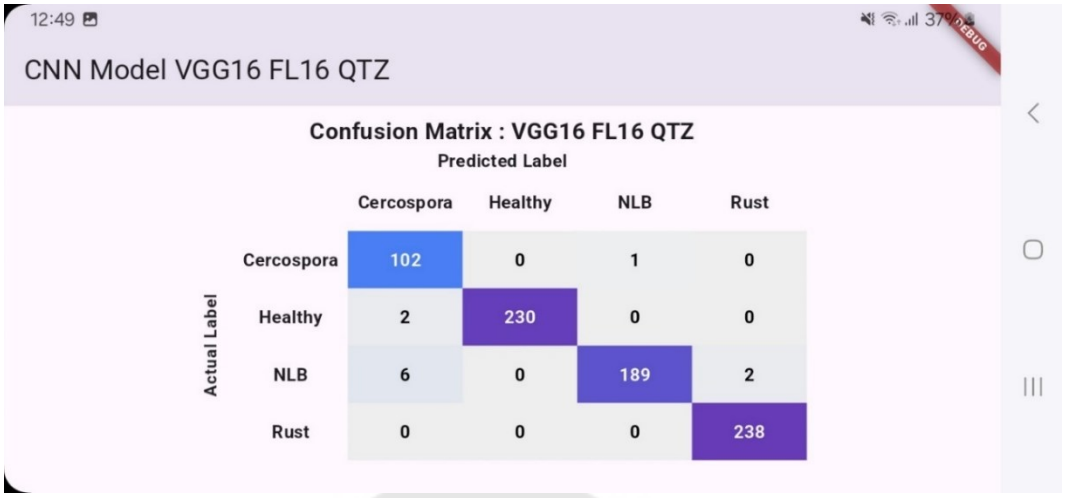
ผลการทดสอบบนโทรศัพท์มือถือตามตาราง 26 แสดงให้เห็นความแม่นยำในการทำนายผลของแบบจำลองบนโทรศัพท์มือถือมีความแตกต่างจากการทดสอบบน Google Colab เล็กน้อย โดย DenseNet121 with FP16 Quantization แสดงความแม่นยำสูงสุดที่ 99.35% รองลงมา คือ VGG16 with FP16 Quantization ที่ 98.57% และแบบจำลองที่ให้ความแม่นยำน้อยที่สุด คือ DW Conv KD with FP16 Quantization ที่ 97.92%

เวลาที่ใช้ในการประมวลผลบนโทรศัพท์มือถือมีค่าสูงกว่าการทดสอบบน Google Colab โดย VGG16 with FP16 Quantization ใช้เวลาในการประมวลผลมากที่สุด 2,746.73 มิลลิวินาทีต่อภาพ ในขณะที่ DW Conv KD with FP16 Quantization ใช้เวลาน้อยที่สุดเพียง 480.49 มิลลิวินาทีต่อภาพ แสดงให้เห็นถึงประสิทธิภาพของแบบจำลองที่พัฒนาด้วยเทคนิค Knowledge Distillation ร่วมกับ Depthwise Separable Convolution ในการทำงานบนอุปกรณ์ที่มีข้อจำกัดด้านทรัพยากร

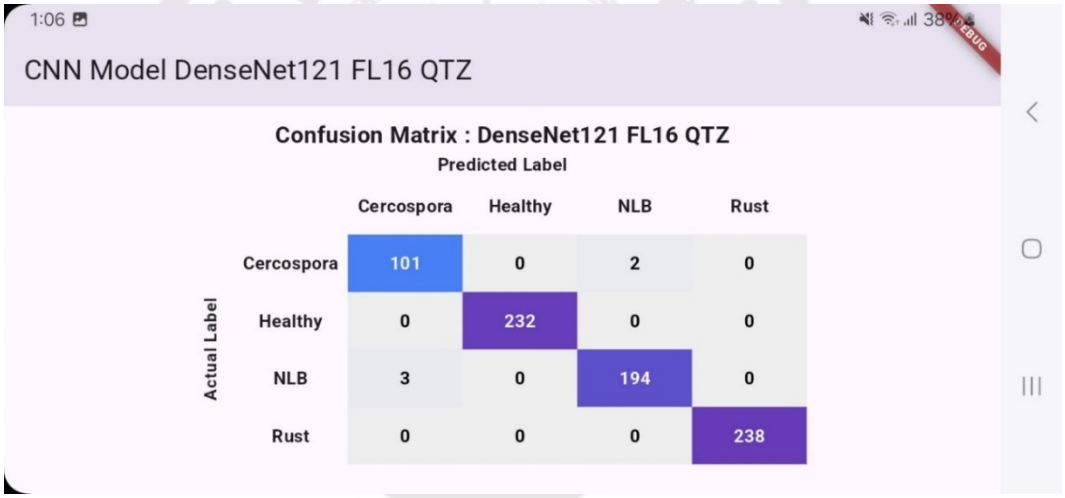
ตาราง 26 แสดงผลการทดสอบแบบจำลองกับชุดข้อมูลทดสอบบนอุปกรณ์โทรศัพท์มือถือ

Model	model size (MB)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (msec/image)
VGG 16 with FP16 Quantization	29.1	98.57	98.57	98.57	98.57	2746.73
DenseNet121 with FP16 Quantization	15.4	99.35	99.35	99.35	99.35	1013.64
MobileNetV2 with FP16 Quantization	6.8	98.05	98.05	98.05	98.05	503.68
NASNet Mobile with FP16 Quantization	10.4	98.05	98.05	98.05	98.05	603.63
STD Conv KD with FP16 Quantization	1.5	98.18	98.18	98.18	98.18	701.29
DW Conv KD with FP16 Quantization	1.6	97.92	97.92	97.92	97.92	480.49

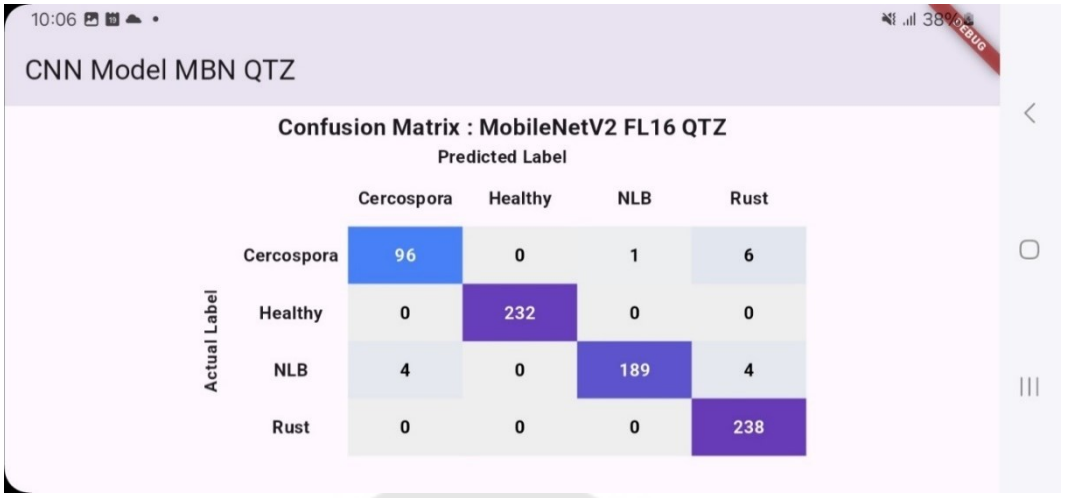
Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลองโดยใช้ชุดข้อมูลทดสอบแสดงดังภาพประกอบ 68 ก) - ฉ)



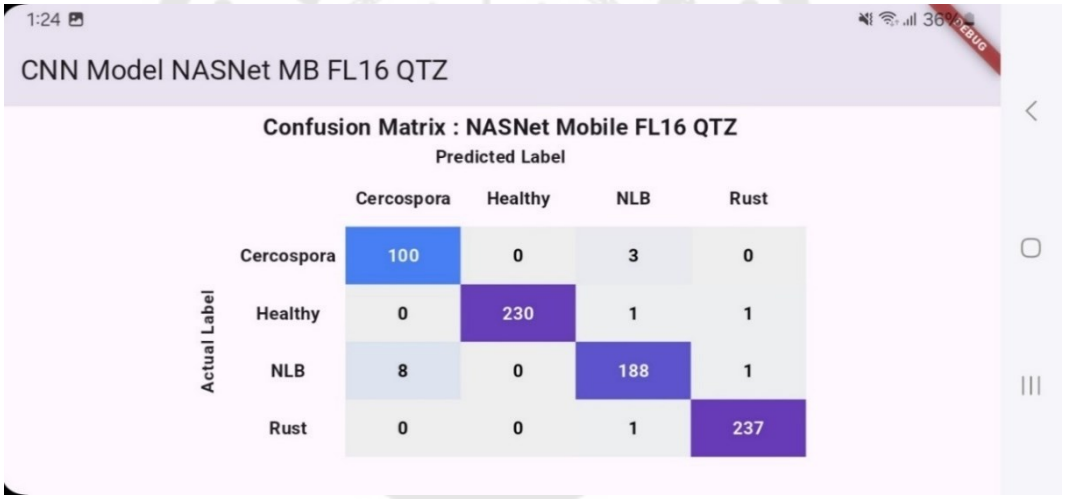
ก) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง VGG16 with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ



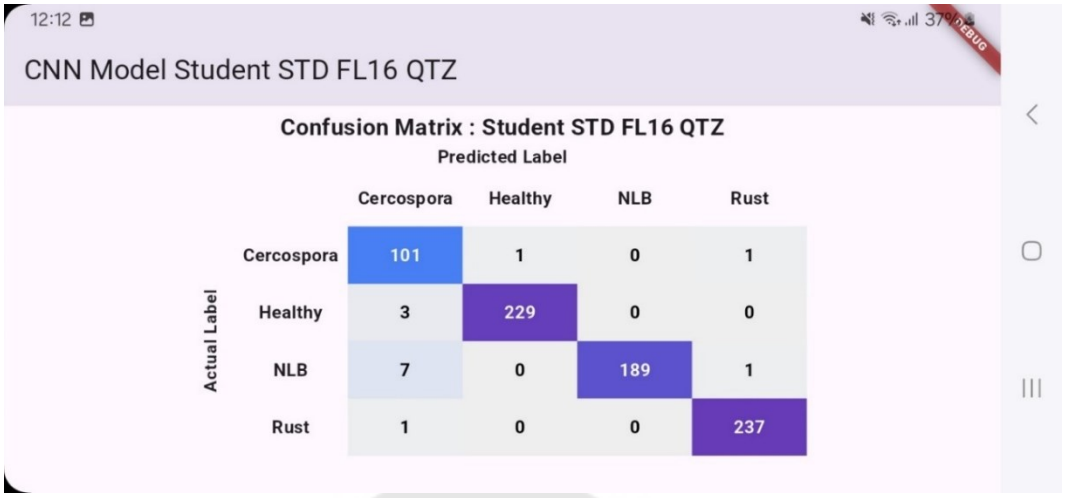
ข) Confusion Matrix ของผลลัพธ์การทำนายของ DneseNet121 with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ



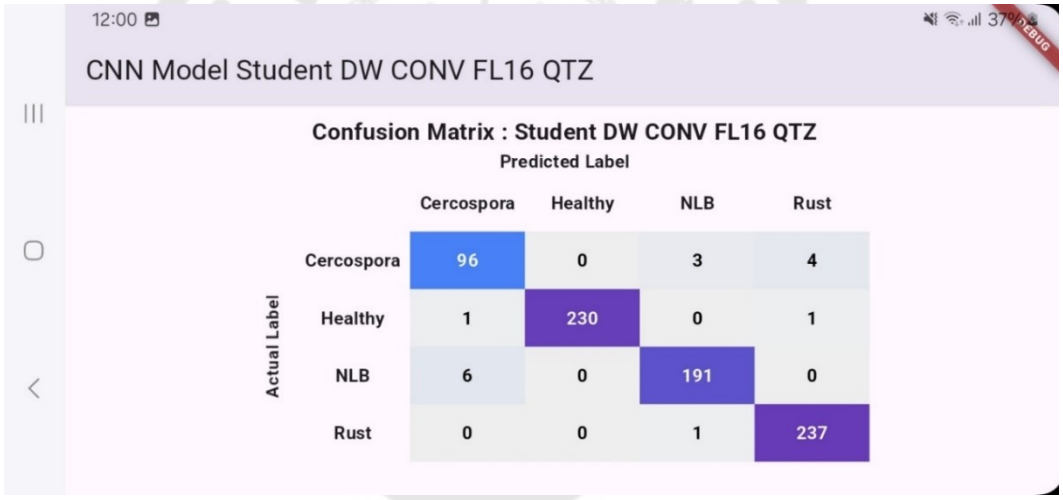
ค) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง MobileNetV2 with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ



ง) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง NASNet Mobile with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ



จ) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง STD Conv Distillation from DenseNet201 with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ



ฉ) Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง DW Conv Distillation from DenseNet201 with Float16 Quantization เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือ

ภาพประกอบ 68 แสดง Confusion Matrix ของผลลัพธ์การทำนายของแบบจำลอง โดยใช้ชุดข้อมูลทดสอบบนอุปกรณ์โทรศัพท์มือถือ

4.5 การเลือกแบบจำลองที่เหมาะสมที่สุดโดยใช้ Weighted Decision Matrix

ในการเลือกแบบจำลองที่เหมาะสมที่สุดจะใช้การวิธีการเลือกโดยใช้ Weighted Decision Matrix ตามที่ได้กล่าวไปในบทที่ 3 มาประยุกต์ใช้เพื่อประเมินและเปรียบเทียบ

ประสิทธิภาพของแบบจำลองทั้งหมดที่ได้พัฒนาขึ้น โดยพิจารณาจากหลายปัจจัยที่มีความสำคัญแตกต่างกัน โดยใช้หลักเกณฑ์ 3 ประการ ได้แก่ ความแม่นยำในการจำแนก ขนาดของแบบจำลอง และเวลาที่ใช้ในการประมวลผล ซึ่งแต่ละเกณฑ์ได้กำหนดค่าน้ำหนักที่แตกต่างกันตามความสำคัญ โดยใช้กระบวนการลำดับชั้นเชิงวิเคราะห์ในการคำนวณค่าน้ำหนักของแต่ละเกณฑ์ เกณฑ์ด้านความแม่นยำในการจำแนกมีความสำคัญสูงสุด ด้วยค่าน้ำหนัก 0.6333 รองลงมา คือ เกณฑ์ด้านขนาดของแบบจำลอง มีค่าน้ำหนัก 0.2605 และเกณฑ์ด้านเวลาที่ใช้ในการประมวลผล มีค่าน้ำหนักน้อยที่สุด 0.1062 ซึ่งแสดงให้เห็นว่าในการเลือกแบบจำลองสำหรับการจำแนกโรคในใบข้าวโพด ความแม่นยำในการจำแนกมีความสำคัญมากที่สุด แต่ยังต้องคำนึงถึงขนาดของแบบจำลองและเวลาที่ใช้ในการประมวลผลด้วย เพื่อให้สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพบนอุปกรณ์มือถือ ในแต่ละเกณฑ์กำหนดคะแนนในช่วง 0 – 100 คะแนน ผลการวิเคราะห์ประสิทธิภาพในการทำนายผลของแบบจำลองด้วยการใช้ Weighted Decision Matrix โดยพิจารณาจากผลการทำนายของแบบจำลองที่ทดสอบบนโทรศัพท์มือถือ แสดงดังตาราง 27

4.6 บทสรุปของการทดลองและผลการทดลอง

ในบทที่ 4 นี้ได้นำเสนอเนื้อหาการทดลองและผลลัพธ์ของงานวิจัยเกี่ยวกับการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด โดยใช้ข้อมูลจากชุดข้อมูล PlantVillage ซึ่งประกอบด้วยภาพถ่ายใบข้าวโพด 3,852 ภาพ แบ่งเป็น 4 หมวดหมู่ ได้แก่ ใบปกติ โรคใบไหม้จากเชื้อ Cercospora โรคใบไหม้ทางตอนเหนือ และโรคราสนิมข้าวโพด การทดลองแบ่งชุดข้อมูลเป็นชุดฝึกฝน 80% และชุดทดสอบ 20% ในการทดลองได้ทดสอบแบบจำลอง Pre-trained จำนวน 4 แบบจำลอง คือ VGG16, DenseNet121, MobileNetV2 และ NASNet Mobile นอกจากนี้ ได้ออกแบบแบบจำลองขนาดเล็ก 2 แบบสำหรับใช้เป็น Student Model ในกระบวนการ Knowledge Distillation ได้แก่ Standard Convolution Student Model (STD Conv Student) และ Depthwise Separable Convolution Student Model (DW Conv Student) และหลังจากการฝึกฝนแบบจำลองแล้วได้ทำการปรับปรุงแบบจำลองโดยใช้เทคนิค Post-training Quantization จำนวน 2 วิธี ได้แก่ Float16 Quantization และ Dynamic Range Quantization เพื่อลดขนาดของแบบจำลอง

ตาราง 27 แสดงผลการวิเคราะห์ประสิทธิภาพในการทำนายผลของแบบจำลองด้วยการใช้ Weighted Decision Matrix

model	Accuracy (%)	Accuracy Score	Weight	Model Size (MB)	Model Size Score	Weight	Inference Time (msec/image)	Inference Time Score	Weight	Total Score
VGG 16 with FP16 Quantization	98.57	90	0.6333	29.1	80	0.2605	2746.73	80	0.1062	86.333
DenseNet121 with FP16 Quantization	99.35	95	0.6333	15.4	80	0.2605	1013.64	80	0.1062	89.4995
MobileNetV2 with FP16 Quantization	98.05	90	0.6333	6.8	90	0.2605	503.68	95	0.1062	90.531
NASNet Mobile with FP16 Quantization	98.05	90	0.6333	10.4	85	0.2605	603.63	90	0.1062	88.6975
STD Conv KD with FP16 Quantization	98.18	90	0.6333	1.5	100	0.2605	701.29	85	0.1062	92.074
DW Conv KD with FP16 Quantization	97.92	85	0.6333	1.6	100	0.2605	480.49	100	0.1062	90.5005

ผลการฝึกฝนแบบจำลองพบว่า MobileNetV2 ให้ผลลัพธ์ดีที่สุดในด้านความแม่นยำ โดยมีค่า Accuracy อยู่ที่ 99.09% ผลการทดสอบกับชุดข้อมูลทดสอบพบว่า MobileNetV2 ที่ผ่านการทำ Dynamic Range Quantization ให้ผลลัพธ์โดดเด่นที่สุด ด้วยค่า Accuracy 99.22% ด้านขนาดของแบบจำลอง การทำ Quantization สามารถลดขนาดลงได้ถึง 75% ของขนาดของแบบจำลองปกติ

การทดสอบประสิทธิภาพบนโทรศัพท์มือถือ พบว่า DenseNet121 with FP16 Quantization แสดงความแม่นยำสูงสุดที่ 99.35% แต่ DW Conv KD with FP16 Quantization ใช้เวลาประมวลผลน้อยที่สุดเพียง 480.49 มิลลิวินาทีต่อภาพ และการเลือกแบบจำลองที่เหมาะสมที่สุดใช้ Weighted Decision Matrix โดยพิจารณาจากความแม่นยำในการจำแนก ขนาดของแบบจำลอง และเวลาที่ใช้ในการประมวลผล ผลการวิเคราะห์พบว่า STD Conv KD with FP16 Quantization มีคะแนนรวมสูงสุด 92.074 คะแนน รองลงมาคือ MobileNetV2 with FP16 Quantization และ DW Conv KD with FP16 Quantization โดยทั้งสามแบบจำลองมีจุดเด่นต่างกัน คือ STD Conv KD with FP16 Quantization มีความสมดุลของทั้งความแม่นยำและขนาดแบบจำลอง MobileNetV2 with FP16 Quantization มีจุดเด่นด้านความเร็วในการประมวลผล และ DW Conv KD with FP16 Quantization มีขนาดเล็กและเร็วที่สุด แต่ความแม่นยำต่ำกว่า

ในบทที่ 5 จะสรุปผลการวิจัยทั้งหมดที่ได้ดำเนินการในการพัฒนาแบบจำลองโครงข่ายประสาทเทียมสำหรับการจำแนกโรคในใบข้าวโพด รวมถึงการอภิปรายผลและข้อค้นพบสำคัญจากการทดลอง นอกจากนี้จะนำเสนอข้อเสนอแนะสำหรับการพัฒนาต่อยอดงานวิจัยในอนาคต ซึ่งอาจรวมถึงการปรับปรุงแบบจำลองให้มีประสิทธิภาพดียิ่งขึ้น หรือการนำไปประยุกต์ใช้กับพืชชนิดอื่นๆ ตลอดจนแนวทางการพัฒนาเป็นแอปพลิเคชันสำหรับเกษตรกรใช้งานจริงในภาคสนาม

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

ในบทที่ 4 ได้กล่าวถึงกระบวนการดำเนินการวิจัย ในการพัฒนาและฝึกฝนแบบจำลองสำหรับการจำแนกโรคจากใบข้าวโพดด้วยชุดข้อมูลด้วยชุดข้อมูลใบข้าวโพดจากชุดข้อมูล PlantVillage กระบวนการปรับปรุงแบบจำลองให้เหมาะสมที่สุดด้วยวิธีการ Knowledge Distillation และการปรับลดขนาดข้อมูลของแบบจำลองด้วยเทคนิค Post-training Quantization ได้แก่ Float16 Quantization และ Dynamic Range Quantization จากนั้นทดสอบประสิทธิภาพของแบบจำลองด้วยชุดข้อมูลทดสอบในแพลตฟอร์ม Google Colab และเลือกแบบจำลองที่ผ่านการปรับปรุงด้วยวิธีการ Float16 Quantization มาทดสอบบนอุปกรณ์โทรศัพท์มือถือซึ่งเป็นส่วนสำคัญในการประเมินความเป็นไปได้ในการนำแบบจำลองไปใช้งานจริงในภาคสนาม จากนั้นประเมินเปรียบเทียบประสิทธิภาพของผลการทำงานของแบบจำลองบนอุปกรณ์โทรศัพท์มือถือด้วย Weighted Decision Matrix เพื่อหาแบบจำลองที่มีความเหมาะสมที่สุดสำหรับการจำแนกโรคจากใบข้าวโพด

และในบทนี้จะกล่าวถึงการสรุปผลการวิจัยในการพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด โดยมีเนื้อหา ดังนี้

1. สรุปผลการวิจัย
2. อภิปรายผลการวิจัย
3. ข้อเสนอแนะ

5.1 สรุปผลการวิจัย

การวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชันสำหรับการจำแนกโรคในใบข้าวโพด ที่มีความเหมาะสมสำหรับการใช้งานบนโทรศัพท์มือถือในภาคสนาม โดยมุ่งเน้นการพัฒนาแบบจำลองที่มีความแม่นยำสูง มีขนาดเล็ก และประมวลผลได้รวดเร็ว ผลการวิจัยสามารถสรุปได้ดังนี้

5.1.1 การพัฒนาและฝึกฝนแบบจำลองโครงข่ายประสาทเทียมแบบคอนโวลูชัน

ในการวิจัยครั้งนี้ ผู้วิจัยได้ดำเนินการทดลองพัฒนาแบบจำลองสำหรับการจำแนกโรคในใบข้าวโพดจากภาพถ่าย โดยใช้ชุดข้อมูล PlantVillage ซึ่งประกอบด้วยภาพใบข้าวโพดจำนวน 3,852 ภาพ แบ่งออกเป็น 4 หมวดหมู่ ได้แก่ ใบปกติ โรคใบไหม้จากเชื้อ Cercospora โรคใบไหม้ทางตอนเหนือ และโรคราสนิมข้าวโพด แบ่งเป็นชุดข้อมูลฝึกฝนจำนวน 80% ของแต่ละหมวดหมู่

และชุดข้อมูลทดสอบจำนวน 20% ของแต่ละหมวดหมู่ การพัฒนาแบบจำลองแบ่งออกเป็น 2 แนวทางหลัก ดังนี้

5.1.1.1 การพัฒนาจากแบบจำลอง Pre-trained Model

ผู้วิจัยได้ทดลองฝึกฝนแบบจำลอง Pre-trained Model จำนวน 4 แบบจำลอง ได้แก่ VGG16 DenseNet121 MobileNetV2 และ NASNet Mobile ด้วยการทำการ Fine-tuning แบบเต็มรูปแบบ (Full Fine-tuning) โดยปรับค่าพารามิเตอร์ทุกชั้นของแบบจำลอง ผลการฝึกฝนแบบจำลองบนแพลตฟอร์ม Google Colab โดยใช้ GPU T4 Runtime พบว่า

DenseNet121 ให้ผลลัพธ์ดีที่สุดในด้านความแม่นยำ โดยมีค่า Training Accuracy Precision Recall และ F1-Score ที่ 99.09% และแบบจำลอง NASNet Mobile ให้ค่า Training Accuracy ต่ำที่สุดที่ 98.18%

แบบจำลอง MobileNetV2 ใช้เวลาในการฝึกฝนน้อยที่สุดเพียง 46 นาที 30 วินาที และแบบจำลอง NASNet Mobile ใช้เวลาฝึกฝนนานที่สุดถึง 2 ชั่วโมง 17 นาที 59 วินาที

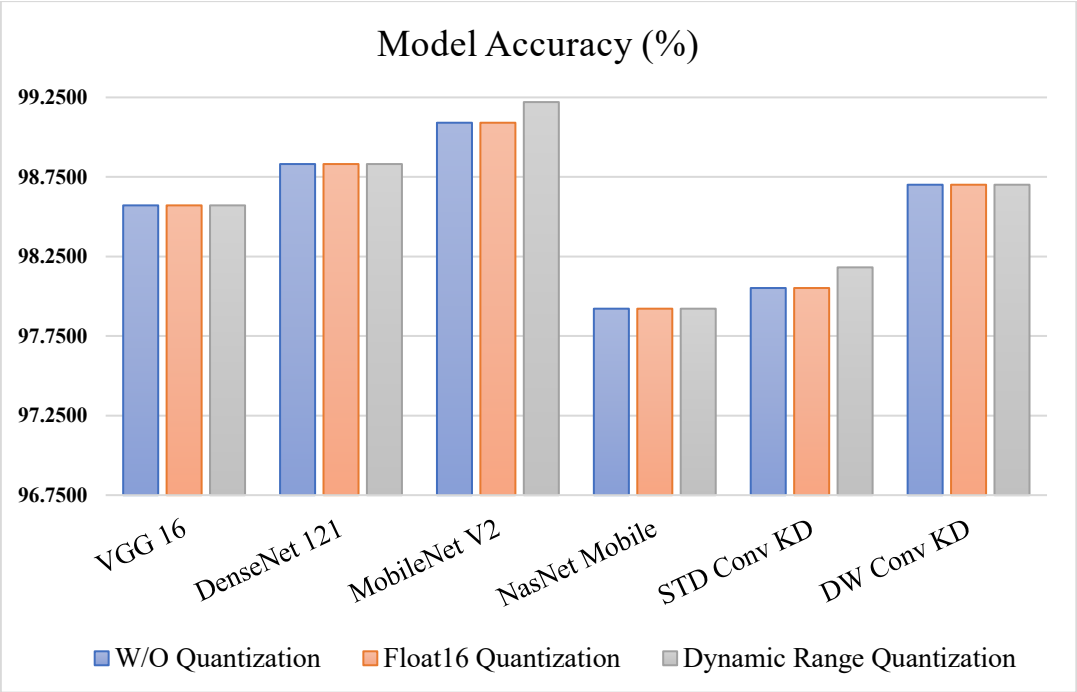
5.1.1.2 การพัฒนาแบบจำลองด้วยเทคนิค Knowledge Distillation

ผู้วิจัยได้พัฒนาแบบจำลองขนาดเล็กโดยใช้เทคนิค Knowledge Distillation ซึ่งเป็นการถ่ายทอดความรู้จากแบบจำลองขนาดใหญ่ (Teacher Model) ไปยังแบบจำลองขนาดเล็ก (Student Model) โดยเลือกใช้ DenseNet201 เป็น Teacher Model ซึ่งมีค่า Accuracy สูงถึงร้อยละ 99.22 เมื่อทดสอบกับชุดข้อมูลทดสอบ สำหรับ Student Model ได้ออกแบบไว้ 2 รูปแบบที่มีโครงสร้างแตกต่างกัน ได้แก่

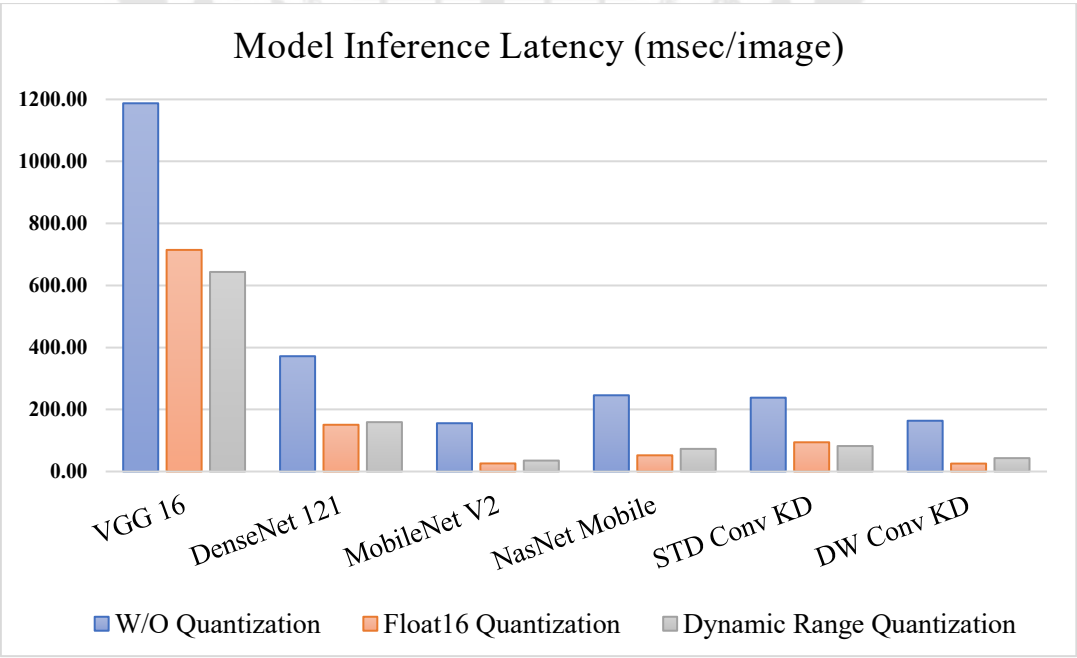
Standard Convolution Student Model (STD Conv Student) ซึ่งใช้ Standard Convolution ในการสกัดคุณลักษณะจากภาพ มีการใช้ Residual Block เพื่อเพิ่มประสิทธิภาพในการเรียนรู้รูปแบบที่ซับซ้อน มีจำนวนพารามิเตอร์ทั้งสิ้น 788,012 พารามิเตอร์ ผลการฝึกฝนพบว่ามีค่า Accuracy Precision Recall และ F1-Score เท่ากับ 98.31%

Depthwise Separable Convolution Student Model (DW Conv Student) ซึ่งพัฒนาจากสถาปัตยกรรม MobileNetV2 ใช้เทคนิค Depthwise Separable Convolution และ Inverted Residual Block ซึ่งช่วยลดการคำนวณลงอย่างมาก มีจำนวนพารามิเตอร์ 838,980 พารามิเตอร์ ผลการฝึกฝนพบว่ามีค่า Accuracy Precision Recall และ F1-Score เท่ากับ 98.70%

การเปรียบเทียบความแม่นยำในการทำนายผล และเวลาที่ใช้ในการประมวลของแบบจำลอง เมื่อทดสอบบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime กับชุดข้อมูลทดสอบภาพใบข้าวโพดแสดงดังภาพประกอบ 69 และภาพประกอบ 70 ตามลำดับ



ภาพประกอบ 69 แสดงการเปรียบเทียบความแม่นยำ (Accuracy) ในการทำนายผลของแบบจำลองบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime กับชุดข้อมูลทดสอบ



ภาพประกอบ 70 แสดงการเปรียบเทียบเวลาในการทำนายผลของแบบจำลองบนแพลตฟอร์ม Google Colab โดยใช้ CPU Runtime กับชุดข้อมูลทดสอบ

5.1.2 การปรับปรุงแบบจำลองให้เหมาะสมที่สุดด้วยการทำ Quantization

ในการลดขนาดของแบบจำลองให้เหมาะสมกับการใช้งานบนโทรศัพท์มือถือได้ใช้เทคนิค Post-training Quantization จำนวน 2 วิธี ได้แก่

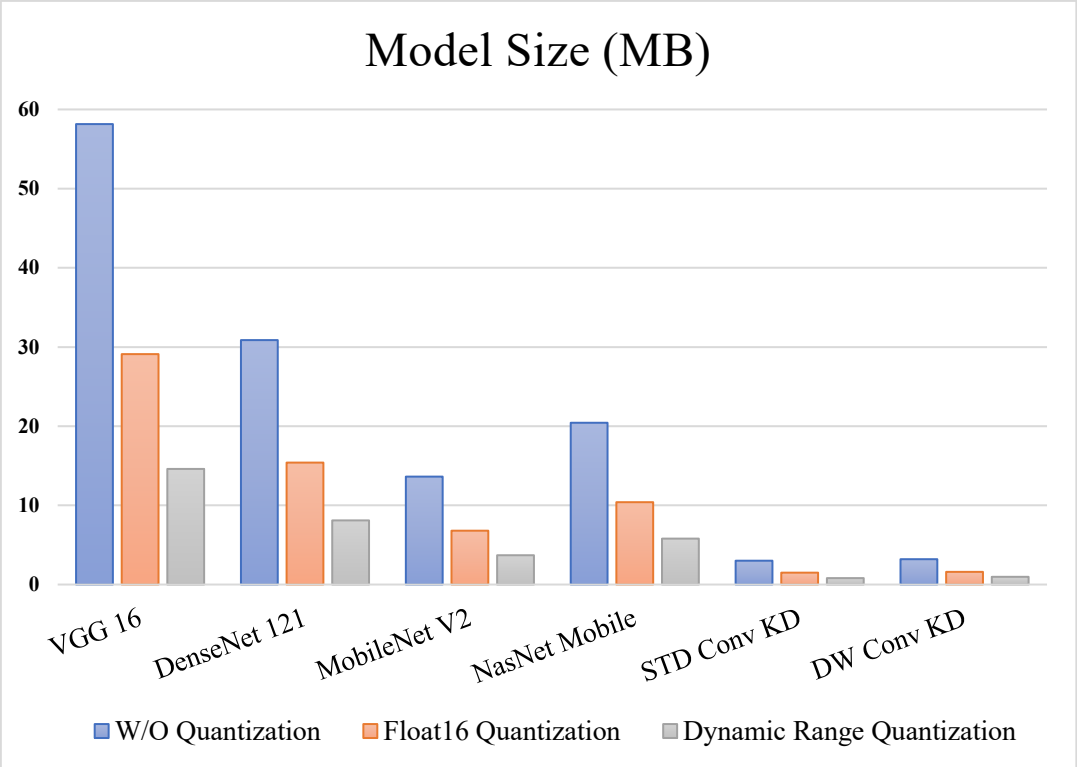
5.1.2.1 Float16 Quantization

Float16 Quantization เป็นการแปลงค่าพารามิเตอร์ของแบบจำลองจากรูปแบบ Float32 (32-bit) ให้เป็น Float16 (16-bit) ช่วยลดขนาดของแบบจำลองลงได้ประมาณร้อยละ 50 โดยที่ความแม่นยำในการจำแนกยังคงใกล้เคียงเดิม

5.1.2.2 Dynamic Range Quantization

Dynamic Range Quantization เป็นการแปลงค่าพารามิเตอร์ของแบบจำลองให้อยู่ในรูปแบบของ Integer 8-bit (INT8) ช่วยลดขนาดของแบบจำลองลงได้ถึงร้อยละ 75 เมื่อเทียบกับแบบจำลองต้นฉบับ

การเปรียบเทียบขนาดของแบบจำลองที่ลดลงจากการปรับปรุงแบบจำลองด้วยวิธี Post-training Quantization แสดงดังภาพประกอบ 71



ภาพประกอบ 71 แสดงการเปรียบเทียบขนาดของแบบจำลองเมื่อผ่านการปรับปรุงปรับปรุงแบบจำลองด้วยวิธี Post-training Quantization

ผลการทดสอบกับชุดข้อมูลทดสอบบน Google Colab พบว่า MobileNetV2 ที่ผ่านการทำ Dynamic Range Quantization ให้ผลลัพธ์โดดเด่นที่สุด ด้วยค่า Accuracy Precision Recall และ F1-Score เท่ากับ99.22% ขณะที่มีขนาดเพียง 3.7 MB และใช้เวลาในการประมวลผลเพียง 35.25 มิลลิวินาทีต่อภาพ ส่วนแบบจำลองที่พัฒนาด้วยเทคนิค Knowledge Distillation และผ่านการทำ Dynamic Range Quantization มีขนาดเล็กมากเพียง 0.811 MB สำหรับ STD Conv Student และ 0.974 MB สำหรับ DW Conv Student

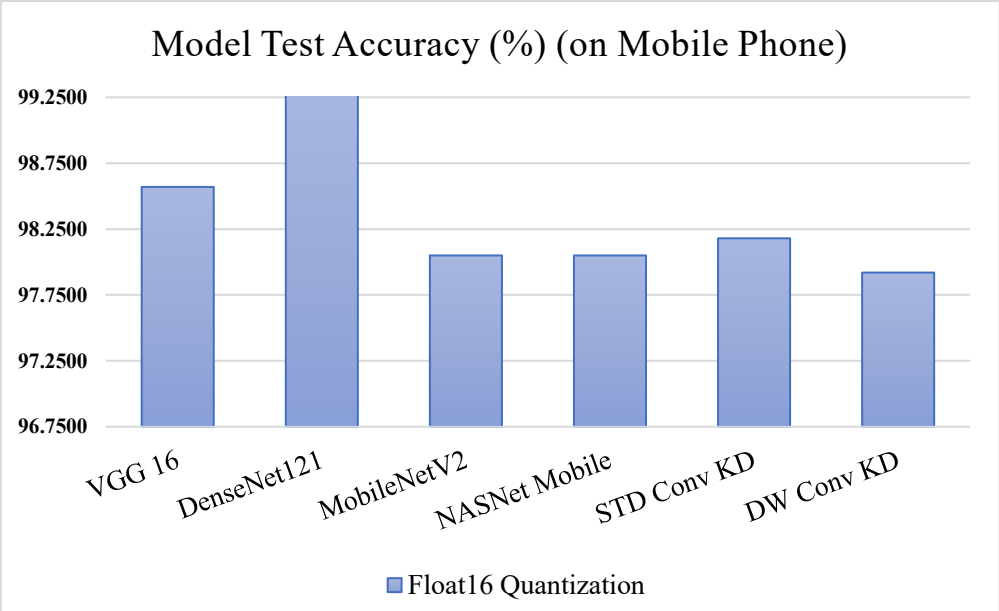
5.1.3 การทดสอบประสิทธิภาพของแบบจำลองบนโทรศัพท์มือถือ

ผู้วิจัยได้ทดสอบประสิทธิภาพของแบบจำลองที่ผ่านการปรับปรุงด้วยวิธีการ Float16 Quantization บนโทรศัพท์มือถือ ผลการทดสอบพบว่า

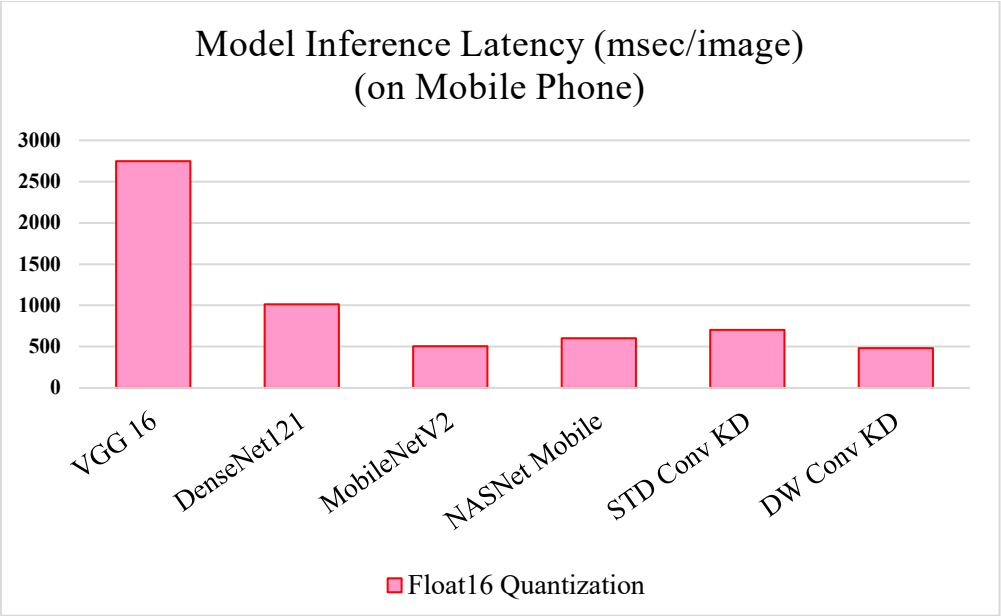
DenseNet121 with FP16 Quantization แสดงความแม่นยำสูงสุดที่ 99.35% แต่ใช้เวลาในการประมวลผลค่อนข้างนานที่ 1,013.64 มิลลิวินาทีต่อภาพ

DW Conv KD with FP16 Quantization ใช้เวลาประมวลผลน้อยที่สุดเพียง 480.49 มิลลิวินาทีต่อภาพ แต่ให้ความแม่นยำต่ำที่สุดที่ร้อยละ 97.92

การเปรียบเทียบความแม่นยำและเวลาที่ใช้ในการประมวลของแบบจำลอง เมื่อทดสอบบนอุปกรณ์โทรศัพท์มือถือกับชุดข้อมูลทดสอบภาพใบข้าวโพด แสดงดังภาพประกอบ 72 และภาพประกอบ 73 ตามลำดับ



ภาพประกอบ 72 แสดงการเปรียบเทียบความแม่นยำ (Accuracy) ในการทำนายผลของแบบจำลองบนอุปกรณ์โทรศัพท์มือถือกับชุดข้อมูลทดสอบ



ภาพประกอบ 73 แสดงการเปรียบเทียบเวลาที่ใช้ในการทำนายผลของแบบจำลองบนอุปกรณ์โทรศัพท์มือถือ

5.1.4 การเลือกแบบจำลองที่เหมาะสมที่สุดโดยใช้ Weighted Decision Matrix

ในการเลือกแบบจำลองที่เหมาะสมที่สุดสำหรับการใช้งานบนโทรศัพท์มือถือ ผู้วิจัยได้ใช้ Weighted Decision Matrix โดยพิจารณาจากหลักเกณฑ์ 3 ประการ ได้แก่ ความแม่นยำในการจำแนก ขนาดของแบบจำลอง และเวลาที่ใช้ในการประมวลผล ผลการวิเคราะห์พบว่า

STD Conv KD with FP16 Quantization มีคะแนนรวมสูงสุด 92.074 คะแนน โดยมีจุดเด่นที่ความสมดุลระหว่างความแม่นยำ (98.18%) ขนาดที่เล็กมาก (1.5 MB) และเวลาในการประมวลผลที่ยอมรับได้ (701.29 มิลลิวินาทีต่อภาพ)

MobileNetV2 with FP16 Quantization มีคะแนนรวม 90.531 คะแนน โดยมีจุดเด่นที่ความเร็วในการประมวลผล (503.68 มิลลิวินาทีต่อภาพ) และความแม่นยำที่ดี (98.05%)

DW Conv KD with FP16 Quantization มีคะแนนรวม 90.5005 คะแนน โดยมีจุดเด่นที่เวลาในการประมวลผลที่เร็วที่สุด (480.49 มิลลิวินาทีต่อภาพ) และขนาดที่เล็กมาก (1.6 MB) แต่มีความแม่นยำที่ต่ำกว่าแบบจำลองอื่นๆ (ร้อยละ 97.92)

จากผลการวิจัยทั้งหมดสรุปได้ว่า การพัฒนาแบบจำลองสำหรับการจำแนกโรคในใบข้าวโพดโดยใช้เทคนิค Knowledge Distillation ร่วมกับการทำ Quantization สามารถสร้างแบบจำลองที่มีขนาดเล็กที่จำนวนพารามิเตอร์ไม่เกิน 1 ล้านพารามิเตอร์ และมีขนาดไม่เกิน 1.6 MB โดยที่ยังคงรักษาความแม่นยำในการจำแนกโรคได้สูงถึง 97.92% – 98.18% และมีความเร็ว

ในการประมวลผลที่เหมาะสมกับการใช้งานบนโทรศัพท์มือถือ ซึ่งแบบจำลองที่เหมาะสมที่สุดคือ STD Conv KD with FP16 Quantization ที่มีความสมดุลระหว่างความแม่นยำ ขนาด และความเร็วในการประมวลผล เหมาะสำหรับการนำไปพัฒนาเป็นแอปพลิเคชันจำแนกโรคใบข้าวโพดบนโทรศัพท์มือถือเพื่อใช้งานในภาคสนาม

5.2 อภิปรายผลการวิจัย

5.2.1 รูปแบบความผิดพลาดในการจำแนกโรค

จากการวิเคราะห์ Confusion Matrix ของแบบจำลองทั้งหมด พบว่ามีรูปแบบความผิดพลาดในการจำแนกโรคที่คล้ายคลึงกัน โดยเฉพาะการสับสนระหว่างคลาสโรคใบไหม้ทางตอนเหนือ กับโรคราสนิมข้าวโพด ซึ่งเป็นความผิดพลาดที่พบได้บ่อยที่สุดในแบบจำลองส่วนใหญ่ ทั้งนี้อาจเนื่องมาจากลักษณะทางกายภาพของโรคทั้งสองมีความคล้ายคลึงกันในบางแง่มุม โดยเฉพาะในระยะเริ่มต้นของการเกิดโรค ทั้งสองโรคมักแสดงอาการเป็นจุดหรือแผลสีน้ำตาลบนใบ ซึ่งต้องอาศัยความชำนาญในการสังเกตลักษณะเฉพาะและรูปร่างของแผลเพื่อแยกแยะความแตกต่าง

นอกจากนี้ ยังพบว่าแบบจำลองทุกประเภทมีความแม่นยำสูงมากในการระบุใบข้าวโพดปกติ โดยแทบไม่มีความผิดพลาดในการจำแนกใบปกติเป็นใบที่เป็นโรค หรือจำแนกใบที่เป็นโรคเป็นใบปกติ ซึ่งเป็นข้อดีอย่างมากสำหรับการนำไปใช้งานจริง เนื่องจากช่วยลดความเสี่ยงในการตรวจไม่พบโรค (False Negative) ซึ่งอาจส่งผลเสียต่อการจัดการโรคในแปลงปลูก

5.2.2 ประสิทธิภาพของแบบจำลอง Pre-trained

จากการวิเคราะห์ Confusion Matrix ของแบบจำลองทั้งหมดที่พัฒนาขึ้นในงานวิจัยนี้ พบประเด็นที่น่าสนใจหลายประการเกี่ยวกับความสามารถในการจำแนกโรคใบข้าวโพดของแต่ละแบบจำลอง ซึ่งช่วยให้เข้าใจจุดแข็งและข้อจำกัดของแบบจำลองแต่ละประเภทได้ชัดเจนยิ่งขึ้น

แบบจำลอง MobileNetV2 แสดงประสิทธิภาพโดดเด่นในการจำแนกโรคใบข้าวโพดด้วยค่า Accuracy สูงถึง 99.09% และยังมีความสมดุลในการจำแนกทุกคลาสของโรค จากการวิเคราะห์ Confusion Matrix พบว่า MobileNetV2 มีอัตราการจำแนกถูกต้อง (True Positive Rate) ที่สูงและสม่ำเสมอในทุกคลาส โดยเฉพาะความสามารถในการแยกแยะระหว่างโรคใบไหม้ทางตอนเหนือกับโรคราสนิมข้าวโพดได้ดีกว่าแบบจำลองอื่นๆ ทั้งนี้อาจเป็นเพราะสถาปัตยกรรมของ MobileNetV2 ที่ออกแบบมาให้มีประสิทธิภาพสูงแม้จะมีขนาดเล็ก โดยใช้เทคนิค Depthwise Separable Convolution และ Inverted Residual Block ซึ่งช่วยให้สามารถสกัดคุณลักษณะสำคัญของโรคได้อย่างมีประสิทธิภาพ

ในขณะที่ DenseNet121 แม้จะมีค่า Accuracy รวมสูงถึง 99.09% เช่นกัน แต่จากการพิจารณา Confusion Matrix พบว่า มีความผิดพลาดในการจำแนกโรคใบไหม้จากเชื้อ Cercospora เป็นโรคราสนิมข้าวโพดอยู่บ้าง ซึ่งแตกต่างจาก MobileNetV2 ที่แทบไม่มีความผิดพลาดในการจำแนกโรคใบไหม้จากเชื้อ Cercospora ความแตกต่างนี้อาจเกิดจากกลไกการเรียนรู้ของ DenseNet ที่มีการเชื่อมต่อแบบ Dense Connection ซึ่งอาจให้น้ำหนักกับลักษณะบางประการของโรคที่คล้ายคลึงกันมากเกินไป

VGG16 แม้จะเป็นแบบจำลองที่มีขนาดใหญ่และซับซ้อน แต่กลับมีความผิดพลาดในการจำแนกระหว่างโรคใบไหม้ทางตอนเหนือกับโรคราสนิมข้าวโพดค่อนข้างมาก เมื่อเทียบกับแบบจำลองอื่นๆ สะท้อนให้เห็นว่าความซับซ้อนและจำนวนพารามิเตอร์ที่มากไม่ได้รับประกันประสิทธิภาพที่ดีกว่าเสมอไป โดยเฉพาะในงานเฉพาะทางอย่างการจำแนกโรคพืช ซึ่งต้องการความละเอียดในการสกัดคุณลักษณะเฉพาะมากกว่าความสามารถในการเรียนรู้รูปแบบทั่วไป

NASNet Mobile มีรูปแบบความผิดพลาดที่น่าสนใจ คือ มักจำแนกโรคราสนิมข้าวโพดเป็นโรคใบไหม้ทางตอนเหนือมากกว่าแบบจำลองอื่นๆ ซึ่งเป็นโรคพืชที่มีลักษณะคล้ายคลึงกัน

5.2.3 ประสิทธิภาพของแบบจำลองที่พัฒนาด้วย Knowledge Distillation

แบบจำลอง DW Conv Student ที่พัฒนาด้วยเทคนิค Knowledge Distillation จาก DenseNet201 แสดงประสิทธิภาพที่ดีที่สุด โดยมีค่า Accuracy ถึง 99.35% ในชุดข้อมูลฝึกฝนซึ่งโดดเด่นกว่าแบบจำลองอื่น ๆ

เมื่อเปรียบเทียบกับแบบจำลอง STD Conv Student ที่ผ่านการปรับปรุงด้วย Float16 Quantization ซึ่งมีค่า Accuracy Precision Recall และ F1-Score ที่เท่ากันคือ 98.18% และมีเวลาในการประมวลผลเฉลี่ย 701.29 มิลลิวินาทีต่อภาพ และพบว่าแบบจำลอง DW Conv Student มีความรวดเร็วในการประมวลผลเร็วที่สุดเฉลี่ย 480.49 มิลลิวินาทีต่อภาพ แต่กลับมีความแม่นยำในการทำนายผลลดลงอย่างมาก เมื่อเปรียบเทียบกับความแม่นยำจากการทดสอบด้วย Google Colab จาก 99.09% ลดลงเหลือ 97.92%

5.2.4 ผลของการทำ Quantization ต่อความแม่นยำในการจำแนก

การทำ Quantization โดยเฉพาะ Dynamic Range Quantization แสดงผลที่น่าสนใจอย่างยิ่ง คือ ไม่เพียงแต่ช่วยลดขนาดของแบบจำลองลงอย่างมากถึง 75% แต่ในบางกรณียังช่วยเพิ่มความแม่นยำในการจำแนกอีกด้วย ดังจะเห็นได้จาก MobileNetV2 ที่ผ่านการทำ Dynamic Range Quantization มีค่า Accuracy เพิ่มขึ้นจาก 99.09% เป็น 99.22%

จากการวิเคราะห์ Confusion Matrix พบว่า MobileNetV2 ที่ผ่านการทำ Dynamic Range Quantization มีความสามารถในการจำแนกโรคใบไหม้ทางตอนเหนือดีขึ้นอย่างชัดเจน

โดยไม่มีการจำแนกผิดพลาดเลยในคลาสนี้ ซึ่งแตกต่างจาก MobileNetV2 แบบปกติที่มีความผิดพลาดในการจำแนกโรคใบไหม้ทางตอนเหนือเป็นโรคราสนิมข้าวโพดอยู่บ้าง ปรากฏการณ์นี้อาจอธิบายได้ว่า การลดความละเอียดของค่าพารามิเตอร์ลงเป็น 8 บิต ในกรณีของ Dynamic Range Quantization อาจช่วยลดปัญหา Overfitting และเพิ่มความสามารถในการ Generalization ของแบบจำลอง ทำให้สามารถตัดสินใจได้ชัดเจนมากขึ้นในกรณีที่มีความคลุมเครือของการทำนายผล

ในทางตรงกันข้าม STD Conv Student ที่ผ่านการทำ Dynamic Range Quantization มีค่า Accuracy เพิ่มขึ้นเล็กน้อยจาก 98.05% เป็น 98.18% แต่ยังมีความผิดพลาดในการจำแนกโรคใบไหม้ทางตอนเหนือกับโรคราสนิมข้าวโพดเช่นเดิม แสดงให้เห็นว่าผลของการทำ Quantization อาจแตกต่างกันไปตามสถาปัตยกรรมของแบบจำลอง

5.2.5 ข้อสังเกตเกี่ยวกับชุดข้อมูลที่ใช้ในการวิจัย

จากการวิเคราะห์ผลการจำแนกของแบบจำลองทั้งหมด พบข้อสังเกตที่น่าสนใจเกี่ยวกับชุดข้อมูล PlantVillage ที่ใช้ในการวิจัยครั้งนี้ คือ แบบจำลองทุกประเภทสามารถจำแนกใบข้าวโพดปกติได้อย่างแม่นยำสูงมาก แทบไม่มีความผิดพลาดในการจำแนกใบปกติเป็นใบที่เป็นโรค ซึ่งแสดงให้เห็นว่าภาพใบข้าวโพดปกติในชุดข้อมูลนี้มีลักษณะที่แตกต่างจากใบที่เป็นโรคอย่างชัดเจน ซึ่งในสถานการณ์จริง ใบข้าวโพดปกติอาจมีลักษณะที่หลากหลายมากกว่านี้ เช่น มีร่องรอยของแมลง มีสีใบที่แตกต่างกันตามสายพันธุ์หรืออายุของพืช ซึ่งอาจทำให้การจำแนกในสถานการณ์จริงมีความซับซ้อนมากกว่าผลการทดลองในงานวิจัยนี้

นอกจากนี้ ยังมีภาพที่ทุกแบบจำลองให้ผลการทำนายผิดหมวดหมู่เหมือนกันที่ทุกแบบจำลอง จำนวน 2 ภาพ ได้แก่

1) Actual Class: Northern Leaf Blight
   VGG16 Predicted Class: Cercospora
   DenseNet121 Predicted Class: Cercospora
   MobileNetV2 GPU Predicted Class: Cercospora
   MobileNetV2 TPU Predicted Class: Cercospora
   NASNet Mobile Predicted Class: Cercospora

   ภาพตัวอย่างที่ทำนายผิดหมวดหมู่ภาพนี้ แสดงดังภาพประกอบ 74

2) Actual Class: Cercospora
   VGG16 Predicted Class: Northern Leaf Blight

DenseNet121 Predicted Class: Northern Leaf Blight

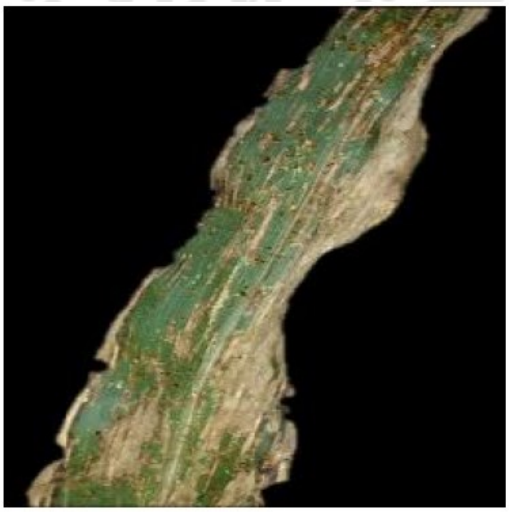
MobileNetV2 GPU Predicted Class: Northern Leaf Blight

MobileNetV2 TPU Predicted Class: Northern Leaf Blight

NASNet Mobile Predicted Class: Northern Leaf Blight

ภาพตัวอย่างที่ทำนายผิดหมวดหมู่ภาพนี้ แสดงดังภาพประกอบ 75

ภาพประกอบ 74 แสดงภาพหมวดหมู่จริง Northern Leaf Blight แต่ทุกแบบจำลองทำนายเป็นหมวดหมู่ Cercospora



ภาพประกอบ 75 แสดงภาพหมวดหมู่จริง Cercospora แต่ทุกแบบจำลองทำนายเป็นหมวดหมู่ Northern Leaf Blight

จากการสังเกตภาพทั้งสองด้วยตาเปล่าพบว่าลักษณะของใบข้าวโพดที่เป็นโรค Cercospora และ Northern Leaf Blight มีลักษณะที่คล้ายคลึงกันมาก ซึ่งการแก้ปัญหาการทำนายโรคผิดพลาดจากภาพทั้ง 2 ภาพนี้ ทำได้โดยการทำ Feature Selection ที่ให้แบบจำลองสามารถทำนายผลได้อย่างมีประสิทธิภาพมากขึ้น

Feature Selection เป็นกระบวนการคัดเลือกคุณลักษณะที่มีความสำคัญและให้ข้อมูลที่เป็นประโยชน์สูงสุดต่อการจำแนกประเภท โดยลดคุณลักษณะที่ไม่จำเป็นหรือก่อให้เกิดสัญญาณรบกวน (noise) ออกไป (Guyon & Elisseeff, 2003) ในกรณีของการจำแนกโรคพืชที่มีอาการคล้ายคลึงกัน เทคนิคนี้จะช่วยให้แบบจำลองสามารถเน้นไปที่คุณลักษณะเฉพาะที่สามารถแยกแยะความแตกต่างระหว่างโรคต่างๆ ได้อย่างชัดเจนยิ่งขึ้น

การประยุกต์ใช้ Feature Selection ในการจำแนกโรคพืชสามารถดำเนินการได้หลายวิธี เช่น การใช้ Principal Component Analysis (PCA) เพื่อลดมิติข้อมูลและเลือกเฉพาะ components ที่มีความแปรปรวนสูง การใช้ Recursive Feature Elimination (RFE) เพื่อคัดเลือกคุณลักษณะที่มีผลต่อการจำแนกมากที่สุด หรือการใช้เทคนิค Mutual Information เพื่อวัดความสัมพันธ์ระหว่างคุณลักษณะกับผลลัพธ์การจำแนก

สำหรับการแก้ไขปัญหาเฉพาะในการจำแนกโรค Cercospora และ Northern Leaf Blight ควรมุ่งเน้นไปที่การคัดเลือกคุณลักษณะที่เกี่ยวข้องกับรูปแบบการกระจายของจุดโรค ขนาดและรูปร่างของแผลโรค สีและเนื้อผิวของบริเวณที่เป็นโรค รวมถึงตำแหน่งการเกิดโรคบนใบ การใช้เทคนิค Grad-CAM (Gradient-weighted Class Activation Mapping) (Selvaraju et al., 2017) จะช่วยให้เข้าใจว่าแบบจำลองใช้บริเวณใดของภาพในการตัดสินใจ และสามารถปรับปรุงการเลือกคุณลักษณะให้ตรงกับลักษณะทางพยาธิวิทยาของโรคแต่ละชนิดได้อย่างแม่นยำยิ่งขึ้น การรวมเทคนิค Feature Selection เหล่านี้เข้าด้วยกันจะช่วยเพิ่มความแม่นยำในการจำแนกโรคและลดปัญหาการทำนายผิดพลาดในกรณีที่โรคมีลักษณะคล้ายคลึงกันได้อย่างมีประสิทธิภาพ

5.2.6 เปรียบเทียบกับงานวิจัยที่เกี่ยวข้อง

เมื่อเปรียบเทียบผลการวิจัยครั้งนี้กับงานวิจัยก่อนหน้าที่เกี่ยวข้องกับการจำแนกโรคในใบข้าวโพดโดยใช้เทคนิคการเรียนรู้เชิงลึก พบว่าแบบจำลองที่พัฒนาขึ้นในงานวิจัยนี้มีประสิทธิภาพสูงและมีขนาดเล็กกว่าอย่างมีนัยสำคัญ งานวิจัยของ Chen และคณะ (Chen et al., 2020) พัฒนาแบบจำลองสำหรับตรวจจับโรคพืชจากภาพถ่ายโดยใช้นำแบบจำลอง Pre-trained จากชุดข้อมูล ImageNet มาปรับใช้กับชุดข้อมูลเฉพาะของโรคในพืช ได้แก่ ข้าวและข้าวโพด และได้ปรับปรุงโครงสร้างของแบบจำลอง VGG เพื่อเพิ่มประสิทธิภาพและลดจำนวนพารามิเตอร์ เรียกว่า

INC-VGGN มีความสามารถในการจำแนกโรคพืชเมื่อทดลองกับชุดข้อมูลทดสอบและภาพถ่ายจริง พบว่าแบบจำลองมีความแม่นยำเฉลี่ยถึง 92% สำหรับโรคในข้าว และ 80.38% สำหรับโรคในข้าวโพดซึ่งต่ำกว่าแบบจำลองทุกประเภทในงานวิจัยนี้

ในขณะที่งานวิจัยของ Fraiwan และคณะ (Fraiwan et al., 2022) นำเสนอการใช้แบบจำลอง CNN จำนวน 10 แบบจำลอง เช่น ResNet101, DarkNet-53, และ Inceptionv3 ที่ผ่านการฝึกฝนมาแล้วจากชุดข้อมูล ImageNet มาปรับแต่งให้เหมาะสมกับชุดข้อมูลใบข้าวโพดจำนวน 3,852 ภาพ พบว่า DarkNet-53 ที่นำเสนอในงานวิจัยมีความแม่นยำสูงสุดที่ 98.6% ซึ่งมีความแม่นยำใกล้เคียงกับแบบจำลอง DW Conv KD ในงานวิจัยนี้ (ความแม่นยำ 98.7%) แต่แบบจำลองของ DarkNet-53 มีขนาดใหญ่กว่าถึง 100 เท่า

นอกจากนี้ งานวิจัยของ Fadhilla และคณะ (Fadhilla et al., 2023) ซึ่งใช้ได้นำเสนอการจำแนกโรคในใบข้าวโพดโดยอัตโนมัติจากภาพถ่ายด้วยแบบจำลอง CNN ที่พัฒนาขึ้นมา ผลการทดสอบพบว่า แบบจำลองที่นำเสนอมีความแม่นยำสูงถึง 93% และได้คะแนน Precision และ Recall สูงสุดถึง 100% ในบางคลาส โดยเฉลี่ย F1-score อยู่ที่ประมาณ 99% ซึ่งสูงกว่าแบบจำลอง MobileNetV2 with DR Quantization เล็กน้อย แต่แบบจำลอง ที่นำเสนอมีขนาดใหญ่กว่าถึง 17 เท่า

การเปรียบเทียบนี้แสดงให้เห็นว่าการใช้เทคนิค Knowledge Distillation ร่วมกับการทำ Quantization เป็นแนวทางที่มีประสิทธิภาพอย่างยิ่งในการพัฒนาแบบจำลองขนาดเล็กสำหรับการจำแนกโรคพืชที่สามารถนำไปใช้งานได้จริงบนอุปกรณ์พกพา โดยยังคงรักษาความแม่นยำในระดับที่ยอมรับได้

โดยสรุป การวิเคราะห์ Confusion Matrix และผลการทดลองอย่างละเอียดในงานวิจัยนี้ช่วยให้เข้าใจทั้งจุดแข็งและข้อจำกัดของแบบจำลองแต่ละประเภท นำไปสู่การเลือกแบบจำลองที่เหมาะสมที่สุดสำหรับการนำไปใช้งานจริงในการจำแนกโรคใบข้าวโพดบนโทรศัพท์มือถือ ซึ่งจะเป็นประโยชน์อย่างยิ่งต่อเกษตรกรในการตรวจวินิจฉัยโรคได้อย่างรวดเร็วและแม่นยำในภาคสนาม นำไปสู่การจัดการโรคที่มีประสิทธิภาพและลดความสูญเสียผลผลิตข้าวโพด

5.2.7 การเลือกใช้เทคนิคการลดขนาดของแบบจำลองให้เหมาะสมที่สุด

จากการศึกษาวิจัยที่เกี่ยวข้องกับการลดขนาดของแบบจำลองโครงข่ายประสาทเทียมสำหรับการจำแนกโรคใบข้าวโพด พบว่า การทำ Parameters Quantization การใช้เทคนิค Knowledge Distillation การใช้ Lightweight Model Design และ การผสมผสานเทคนิคต่าง ๆ

นั้นให้ผลลัพธ์ที่มีประสิทธิภาพทั้งในด้านความแม่นยำ ความเร็ว และขนาดของแบบจำลอง ซึ่งมีความสำคัญอย่างยิ่งสำหรับการนำไปใช้งานบนอุปกรณ์พกพาที่มีทรัพยากรจำกัด

การทำ Parameters Quantization โดยเฉพาะแบบ Post-Training ได้แก่ Float16 Quantization และ Dynamic Range Quantization เป็นกระบวนการลดความละเอียดของพารามิเตอร์ของแบบจำลอง ซึ่งช่วยลดขนาดของแบบจำลองลงอย่างมีนัยสำคัญ โดยไม่สูญเสียความแม่นยำมากนัก และมีข้อได้เปรียบในเรื่องของต้นทุนการพัฒนาแบบจำลอง สามารถปรับรูปแบบชนิดข้อมูลของค่าพารามิเตอร์ให้อยู่ในรูปแบบที่มีความละเอียดต่ำได้ โดยไม่ต้องฝึกฝนแบบจำลองใหม่

ขณะเดียวกันการทำ Knowledge Distillation ก็เป็นอีกแนวทางที่มีประสิทธิภาพสูง โดยเป็นกระบวนการถ่ายทอดความรู้จากแบบจำลองขนาดใหญ่ หรือ Teacher Model ไปยังแบบจำลองขนาดเล็ก หรือ Student Model ที่ถูกออกแบบมาให้มีจำนวนพารามิเตอร์น้อยกว่ามาก แต่สามารถเลียนแบบการทำนายผลลัพธ์ของ Teacher Model ได้อย่างมีประสิทธิภาพใกล้เคียงกัน ในการทดลองพบว่าแบบจำลอง STD Conv Student Model และ DW Conv Student Model ที่ได้รับการกลั่นความรู้สามารถรักษาความแม่นยำในระดับสูงกว่า 98% ได้แม้จะมีขนาดเล็กกว่า 1 MB ซึ่งแสดงให้เห็นถึงความเหมาะสมของวิธีนี้ในการสร้างแบบจำลองที่มีขนาดกะทัดรัดสำหรับใช้งานบนอุปกรณ์พกพา (Gou et al., 2021)

ส่วนการใช้เทคนิค Lightweight Model Design ซึ่งเป็นแบบจำลองที่ใช้โครงสร้างพิเศษอย่าง Depthwise Separable Convolution ก็เป็นเทคนิคที่ช่วยลดจำนวนพารามิเตอร์ลงโดยตรงจากการออกแบบสถาปัตยกรรมแบบจำลอง เช่น ในแบบจำลอง DW Conv Student Model ที่ใช้ Depthwise Separable Convolution ซึ่งสามารถลดจำนวนพารามิเตอร์ลงได้หลายเท่า ทำให้มีการประมวลผลได้รวดเร็ว โดยยังคงความสามารถในการเรียนรู้ลักษณะของโรคได้อย่างมีประสิทธิภาพ

สิ่งที่น่าสนใจที่สุดจากการศึกษา คือ การนำเทคนิคในการลดขนาดของแบบจำลองมาใช้แบบผสมผสานร่วมกัน ได้แก่ การใช้เทคนิค Knowledge Distillation ร่วมกับ Float16 Quantization ใน STD Conv Student Model ทำให้ได้แบบจำลองที่มีขนาดเพียง 1.5 MB แต่สามารถทำความแม่นยำได้ถึง 98.18% และเมื่อประเมินด้วย Weighted Decision Matrix ที่พิจารณาทั้งความแม่นยำ ความเร็ว และขนาดแบบจำลอง พบว่า แบบจำลองดังกล่าวมีคะแนนรวมสูงสุดถึง 92.074 คะแนน ซึ่งยืนยันว่า การผสมผสานเทคนิค Knowledge Distillation ร่วมกับ Float16 Quantization สามารถสร้างแบบจำลองที่มีประสิทธิภาพสูงและเหมาะสมที่สุดสำหรับ

การใช้งานบนอุปกรณ์พกพา เป็นแนวทางที่ให้ผลลัพธ์ที่ดีที่สุดในการพัฒนาแบบจำลองที่เหมาะสมสำหรับการใช้งานในภาคสนาม โดยเฉพาะในสถานการณ์ที่มีข้อจำกัดด้านทรัพยากร (Cheng et al., 2017)

5.3 ข้อเสนอแนะ

จากผลการวิจัยที่ได้ดำเนินการพัฒนาแบบจำลองสำหรับการจำแนกโรคจากใบข้าวโพด ผู้วิจัยมีข้อเสนอแนะเพื่อการพัฒนางานวิจัยในอนาคต ดังนี้

5.3.1 การเพิ่มความหลากหลายของชุดข้อมูล

ชุดข้อมูล PlantVillage ที่ใช้ในงานวิจัยนี้ เป็นภาพถ่ายในสภาพแวดล้อมที่ควบคุม ซึ่งมีความแตกต่างจากสภาพแวดล้อมจริงในภาคสนาม การวิจัยในอนาคตควรพัฒนาชุดข้อมูลที่มีความหลากหลายมากขึ้น โดยเฉพาะอย่างยิ่งภาพถ่ายในสภาพแวดล้อมจริง ที่มีความแตกต่างในด้านแสง มุมมองการถ่ายภาพ และพื้นหลังที่ซับซ้อน

นอกจากนี้ ควรเพิ่มจำนวนตัวอย่างของโรคในระยะต่างๆ ของการเกิดโรค ตั้งแต่ระยะเริ่มต้นที่ยังไม่แสดงอาการชัดเจน จนถึงระยะที่มีความรุนแรงมาก เพื่อให้แบบจำลองสามารถเรียนรู้ลักษณะของโรคในทุกระยะได้อย่างแม่นยำ การรวบรวมภาพถ่ายใบข้าวโพดที่มีการซ้อนทับกันของโรคหลายชนิด หรือมีร่องรอยของแมลงศัตรูพืชร่วมด้วย จะช่วยให้แบบจำลองมีความสามารถในการจำแนกที่ครอบคลุมสถานการณ์จริงมากขึ้น

5.3.2 การปรับปรุงแบบจำลองให้เหมาะสมกับโรคของใบข้าวโพดที่เกิดในประเทศไทย

เพื่อให้การจำแนกโรคข้าวโพดในประเทศไทยมีประสิทธิภาพและตอบโจทย์การใช้งานจริง ควรพัฒนาแบบจำลองให้ครอบคลุมโรคที่สำคัญเพิ่มเติม เช่น โรคใบไหม้แผลใหญ่ ราน้ำค้าง กาบและลำต้นเน่า และฝักเน่า รวมถึงอาการขาดธาตุอาหารซึ่งมักคล้ายกับโรคพืช ทั้งนี้ ควรเก็บข้อมูลจากสายพันธุ์ข้าวโพดที่นิยมปลูกในไทย และสภาพแวดล้อมที่หลากหลาย พร้อมออกแบบให้ใช้งานได้แม้ในพื้นที่ออฟไลน์ บูรณาการกับระบบเตือนภัย และสนับสนุนการจัดการโรคแบบผสมผสาน เพื่อช่วยเกษตรกรลดการสูญเสีย เพิ่มคุณภาพผลผลิต และส่งเสริมความยั่งยืนในภาคเกษตรกรรมไทย

บรรณานุกรม

Adam, H., Andreetto, M., Chen, B., Howard, A., Kalenichenko, D., Wang, W.,Zhu, M. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *Comput. Vis and Pat. Recogn*.

Agarwal, M., Bohat, V. K., Ansari, M. D., Sinha, A., Gupta, S. K., & Garg, D. (2019). A Convolution Neural Network based approach to detect the disease in Corn Crop. *IEEE Potentials*.

Ammarapala, V., Chinda, T., Pongsayaporn, P., Ratanachot, W., Punthutaecha, K., & Janmonta, K. (2018). Cross-border shipment route selection utilizing analytic hierarchy process (AHP) method. *Songklanakarin Journal of Science and Technology*, *40*(1), 31-37.

Ashok, S., Kishore, G., Rajesh, V., Suchitra, S., Sophia, S. G. G., & Pavithra, B. (2020, 10-12 June 2020). Tomato Leaf Disease Detection Using Deep Learning Techniques. 2020 5th International Conference on Communication and Electronics Systems (ICCES),

Cerqueira, V., Santos, M., Baghoussi, Y., & Soares, C. (2024). On-the-fly Data Augmentation for Forecasting with Deep Learning. *arXiv preprint arXiv:2404.16918*.

Chen, J., Chen, J., Zhang, D., Sun, Y., & Nanehkaran, Y. A. (2020). Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture*, *173*, 105393.

Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A Survey of Model Compression and Acceleration for Deep Neural Networks.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition,

Erenstein, O., Jaleta, M., Sonder, K., Mottaleb, K., & Prasanna, B. M. (2022). Global maize production, consumption and trade: trends and R&D implications. *Food Security*, *14*(5), 1295-1319.

Fadhilla, M., Suryani, D., Labella, A., & Gunawan, H. (2023). Corn Leaf Diseases Recognition Based on Convolutional Neural Network. *IT Journal Research and Development*, *8*, 14-21.

Fraiwan, M., Faouri, E., & Khasawneh, N. (2022). Classification of Corn Diseases From Leaf Images Using Deep Transfer Learning. *Plants*, *11*, 2668.

Ghofrani, A., & Mahdian Toroghi, R. (2022). Knowledge distillation in plant disease recognition. *Neural Computing and Applications*, *34*(17), 14287-14296.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2021). A Survey of Quantization Methods for Efficient Neural Network Inference. *CoRR*.

Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, *3*(null), 1157–1182.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2016). Densely Connected Convolutional Networks.

Hughes, D. P., & Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *ArXiv*, *abs/1602.07360*.

Kilaru, R., & MurthyRaju, K. (2022). Prediction of Maize Leaf Disease Detection to improve Crop Yield using Machine Learning based Models. 4th International Conference on Recent Trends in Computer Science and Technology, ICRTCST 2021 - Proceedings,

Li, Z., Li, H., & Meng, L. (2023). Model Compression for Deep Neural Networks: A Survey. *MDPI*, *12*(3).

Mishra, R., Gupta, H. P., & Dutta, T. (2020). A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions.

Nandi, R. N., Palash, A. H., Siddique, N., & Zilani, M. G. (2022). *Device-friendly Guava fruit and leaf disease detection using deep learning*.

Nelson, M. (2018). *Neural Networks and Deep Learning*.

Rokh, B., Azarpeyvand, A., & Khanteymoori, A. (2023). A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification. *ACM Trans. Intell. Syst. Technol.*, *14*(6), Article 97.

Saaty, R. W. (1987). The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, *9*(3), 161-176.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017, 22-29 Oct. 2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. 2017 IEEE International Conference on Computer Vision (ICCV),

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. In.

Suharjito, Elwirehardja, G. N., & Prayoga, J. S. (2021). Oil palm fresh fruit bunch ripeness classification on mobile devices using deep learning approaches. *Computers and Electronics in Agriculture*, *188*, 106359.

Thakur, P. S., Sheorey, T., & Ojha, A. (2023). VGG-ICNN: A Lightweight CNN model for crop disease identification. *Multimedia Tools and Applications*, *82*(1), 497-520.

USDA. (2024). *Production - Corn*. Retrieved 03 Jan 2025 from

Waheed, A., Goyal, M., Gupta, D., Khanna, A., Hassanien, A. E., & Pandey, H. M. (2020). An optimized dense convolutional neural network model for disease recognition and classification in corn leaf. *Computers and Electronics in Agriculture*, *175*.

Yao, J., Tran, S. N., Sawyer, S., & Garg, S. (2023). Machine learning for leaf disease classification: data, techniques and applications. *Artificial Intelligence Review*, *56*.

Zaniolo, L., Garbin, C., & Marques, O. (2023). Zaniolo, Luiz and Garbin, Christian and Marques, Oge. *IEEE Potentials*, *42*, 39-45.

Zeng, W., Li, H., Gensheng, H., & Liang, D. (2022). Lightweight dense-scale network (LDSNet) for corn leaf disease identification. *Computers and Electronics in Agriculture*, *197*.

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. Proceedings of the IEEE conference on computer vision and pattern recognition,

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning Transferable Architectures for Scalable Image Recognition.

ธวัชระพงษ์, ว. ส. (2022). การกำหนดค่าน้ำหนักหลักเกณฑ์เพื่อการตัดสินใจ (Determining weights of criteria for decision Making). วารสารวิชาการเทคโนโลยีอุตสาหกรรม.