การวิเคราะห์โรคใบข้าวด้วยพรีเทรนซีเอ็นเอ็นโมเดล

RICE LEAF DISEASE ANALYSIS USING PRE-TRAINED CNN MODEL

วิทวัส เหมหงษา

WITTAWAT HAMHONGSA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2566

KMITL-2023-SC-M-002-050

RICE LEAF DISEASE ANALYSIS USING PRE-TRAINED CNN MODEL

WITTAWAT HAMHONGSA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE

DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE SCHOOL OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2023

KMITL-2023-SC-M-002-050

หัวข้อวิทยานิพนธ์ การวิเคราะห์โรคใบข้าวด้วยพรีเทรนซีเอ็นเอ็นโมเดล
ชื่อนักศึกษา วิทวัส เหมหงษา
รหัสประจำตัว 60605074
ปริญญา วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชา วิทยาการคอมพิวเตอร์
พ.ศ. 2566
อาจารย์ที่ปรึกษาวิทยานิพนธ์ ดร. รุ่งรัตน์ เวียงศรีพนาวัลย์

บทคัดย่อ

อุปสรรคหนึ่งที่สำคัญในการปลูกข้าวคือการเกิดโรคใบข้าว ด้วยโรคใบข้าวแต่ละโรคไม่ได้มีวิธีการรักษาที่เหมือนกัน การที่เกษตรกรยังขาดความรู้ในการวิเคราะห์โรคข้าวและอาศัยการคาดเดาโดยการสังเกตด้วยตาเปล่าจากประสบการณ์อาจทำให้เกิดข้อผิดพลาดในการกำจัดโรคข้าว หรือ อาจใช้ยาฆ่าโรคที่มีฤทธิ์รุนแรงเกินไป การปรึกษาผู้เชี่ยวชาญเป็นการแก้ปัญหาที่ดีแต่เกษตรกรบางรายอาจเข้าถึงผู้เชี่ยวชาญได้ยาก ด้วยในปัจจุบันปัญญาประดิษฐ์โดยเฉพาะการเรียนรู้เชิงลึกสามารถนำมาใช้เป็นเครื่องมือในการวิเคราะห์วัตถุจากภาพได้ งานวิจัยนี้จึงนำโมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล (ซีเอ็นเอ็น) แบบพรีเทรนของการเรียนร์เชิงลึก มาใช้ในการจำแนกโรคใบข้าวที่เกิดขึ้นในประเทศไทยจำนวน 5 โรค ได้แก่ โรคไหม้ โรคใบจุดสีน้ำตาล โรคใบขีดสีน้ำตาล โรคใบสีส้ม โรคขอบใบแห้ง และข้าวที่ไม่เป็นโรค โมเดลที่นำมาใช้จะอยู่ในกลุ่ม เอ็กซ์เซ็พชัน เรซเน็ต อินเซ็พชัน อินเซ็พชันเรซเน็ต เด็นท์เน็ต ชุดข้อมูลที่ใช้นำมาจาก 3 แหล่งข้อมูล ได้แก่ 1) ฐานข้อมูล UCI 2) ชุดข้อมูลตัวอย่างของโรคใบข้าวของ Sethy และ 3) ชุดข้อมูลโรคข้าวที่ผู้วิจัยได้จัดทำขึ้นจากการเก็บภาพใบข้าวที่เป็นโรคในพื้นที่อำเภอบางเลน จังหวัดนครปฐม ระหว่างปี พ.ศ. 2561 -2563 ซึ่งได้รับการรับรองภาพจากนักวิชาการข้าว ผลการทดลองเบื้องต้นแสดงให้เห็นว่าการใช้โมเดลซีเอ็นเอ็นแบบพรีเทรนสในการจำแนกภาพที่มีนั้นต้องการภาพจำนวนที่มากขึ้น ด้วยข้อจำกัดที่ไม่สามารถถ่ายภาพเพิ่มได้เนื่องจากสถานการณ์โควิด ผู้วิจัยจึงมีการนำหลักการของการทำดาต้าอ็อกเมนเตชันสำหรับรูปภาพมาใช้ในการเพิ่มจำนวนรูปภาพในชุดข้อมูล ผลการทดลองที่มีการเพิ่มภาพเข้าไปพบว่าสามารถเพิ่มประสิทธิภาพในการจำแนกโรคข้าวได้เพิ่มขึ้นสูงสุดถึง 16.533% ในโมเดลเรซเน็ตห้าสิบวีสอง และ มีค่าที่เพิ่มขึ้นอยู่ระหว่าง 13.178 – 16.533 % ในโมเดลอื่น ยกเว้นเด็นท์เน็ตสองศูนย์หนึ่งซึ่งมีค่าเพิ่มขึ้นโดยเฉลี่ยเพียง 8.503%

คำสำคัญ: การเรียนรู้แบบเชิงลึก โครงข่ายประสาทเทียมแบบคอนโวลูชัน คอมพิวเตอร์วิชัน โรคใบข้าว

Thesis Title	Rice Leaf Disease Analysis Using Pre-trained CNN Model
Student Name	Wittawat Hamhongsa
Student ID	60605074
Degree	Master of Science (Computer Science)
Department	Computer Science
Year	2023
Thesis Advisor	Dr. Rungrat Wiangsripanawan

Abstract

Thailand is one of the top exporters of rice in the world. Nevertheless, one of the major obstacles in rice cultivation is the incidence of rice leaf disease. Not all rice leaf diseases have the same treatment. The lack of knowledge in rice disease analysis and relying on experience based on naked eye observations may lead to errors in rice disease treatment such as using a too strong disinfectant. Although consulting experts is possible, some farmers may find it difficult. At present, artificial intelligence, especially deep learning can be used as a tool to analyze objects from images. In this research, pretrained convolutional neural network (CNN) models are used to classify 5 rice leaf diseases that frequently found in Thailand namely blast disease, brown spot disease, bacterial leaf blight disease, rice tungro disease and narrow brown spot disease and healthy leaves. The pretrained models used are in the group of Xception, ResNet, Inception, InceptionResNet and DenseNet. The dataset was from 3 data sources: 1) the UCI database, 2) the sample dataset of Sethy's rice leaf disease, and 3) the diseased rice dataset which photographed and collected by the researcher in BangLen District, Nakhon Pathom Province during the year 2018 -2020 and certified by the rice scholars. The preliminary result showed that pretrained CNN models required a larger number of images for a better classification. With restrictions under the COVID situation, the researcher then applied the image data augmentation technique to increase the

number of images in the dataset instead of taking more photos. The experimental result showed that the accuracy in rice disease classification was averagely increased up to 16.533% in the ResNet50v2 and the results of other models had increased between 13.178 – 16.533%. Only the DenseNet201 that its average accuracy increased only 8.503%.

Keywords: Deep learning, Computer vision, Convolutional neural networks, Rice leaf disease

กิตติกรรมประกาศ

งานวิจัยนี้จะสำเร็จลุล่วงไม่ได้หาขาดบุคคลสำคัญที่ช่วยเหลือ และให้คำปรึกษากับข้าพเจ้า ได้แก่ ดร. รุ่งรัตน์ เวียงศรีพนาวัลย์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้คำแนะนำช่วยเหลือมาโดยตลอด ทั้งในด้านการให้ความรู้ ด้านวิชาการ และการใช้ชีวิต รวมถึงสุขภาพความเป็นอยู่ ขอขอบคุณ คุณวันพร เข็มมุกด์ ผู้อำนวยการกลุ่มวิทยาการอารักขาข้าว กรมการข้าว สังกัดกระทรวงเกษตรและสหกรณ์ ที่ได้ให้ความรู้เกี่ยวกับโรคข้าว และตรวจสอบชุดข้อมูลภาพโรคข้าวให้จนสำเร็จลุล่วง ขอบคุณ ผศ.ดร. ชัยพร ใจแก้ว ประธานกรรมสอบวิทยานิพนธ์และผู้ทรงคุณวุฒิจากภายนอกสถาบันฯ ผศ.ดร. วรางคณา กิ้มปาน อาจารย์บัณฑิตประจำ ที่ได้ให้เกียรติมาเป็นคณะกรรมการในการสอบวิทยานิพนธ์ และให้คำแนะที่เป็นประโยชน์ในการทำวิจัยในครั้งนี้ด้วย ขอขอบคุณชาวนาในพื้น อำเภอบางเลน จังหวัดนครปฐม ที่คอยให้ความช่วยเหลือ และให้ความร่วมมือในการเก็บชุดข้อมูลภาพ รวมถึงเพื่อน พี่ น้อง ที่ช่วยจัดทำชุดข้อมูลรวมทั้งให้คำแนะนำและเป็นกำลังใจให้กับข้าพเจ้า และที่ขาดไม่ได้คือ ขอขอบพระคุณครอบครัวของข้าพเจ้า ได้แก่บิดา และมารดา ที่ให้การสนับสนุนและเป็นกำลังใจมาโดยตลอด รวมถึงการเลี้ยงดูและอบรมสั่งสอนข้าพเจ้าให้เติบโตมาเป็นคนดีไม่สร้างความเดือดร้อนให้กับสังคมจนถึงทุกวันนี้ และคณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้มอบทุนการศึกษายกเว้นค่าเล่าเรียนเป็นเวลา 4 ภาคการศึกษา ให้กับข้าพเจ้า รวมถึงการสนับสนุนทุนสำหรับการตีพิมพ์ผลงานทางวิชาการส่วนหนึ่งจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังด้วย

ทั้งนี้ขอขอบคุณผู้มีส่วนเกี่ยวข้องทุกท่านทั้งที่ได้กล่าวมา และไม่ได้กล่าวมาด้วย ณ ที่นี้

นายวิทวัส เหมหงษา

สารบัญ

สารบัญตาราง

สารบัญรูป

คำย่อ/สัญลักษณ์

CNN	Convolutional Neural Network
SVM	Support Vector Machaine
ReLU	Rectified Linear Unit

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ประเทศไทยนับว่าเป็นประเทศที่ส่งออกผลผลิตและผลิตภัณฑ์ทางการเกษตรในอันดับต้น ๆ ของโลกโดยเฉพาะข้าว จากการส่งออกข้าวไทยในครึ่งแรกของปี 2565 มีปริมาณ 3.99 ล้านตัน คิดเป็นมูลค่า 2,106 ล้านเหรียญสหรัฐ หรือราว 70,340 ล้านบาท (รัฐบาลไทย 2565) จากตัวเลขของปริมาณการส่งออกแสดงให้เห็นว่าข้าวเป็นพืชเศรษฐกิจที่หล่อเลี้ยงประเทศไทย และคนไทยก็ยังบริโภคข้าวเป็นอาหารหลัก (Berno, Dentice et al. 2019) ดังนั้นข้าวจึงเป็นพืชที่สำคัญมากต่อประเทศไทย

ปัจจัยหนึ่งที่ก่อให้เกิดสาเหตุดังกล่าวเกิดจาก เกษตรกรที่ไม่ได้มาจากพื้นฐานเกษตรกร หรือเกษตรกรสมัยใหม่ที่ไม่มีพื้นฐานเกษตรแต่เป็นนักลงทุนที่มองภาคการเกษตรเป็นโอกาส ซึ่งเกษตรกรกลุ่มนี้ต้องการความรู้และเทคโนโลยีเข้ามาช่วย แต่ในปัจจุบันประเทศไทยมีเทคโนโลยีที่จะเข้ามาช่วยสนับสนุนเกษตรกรกลุ่มนี้ในลักษณะที่เป็นรูปธรรมน้อยมาก และในส่วนของการวิเคราะห์โรคข้าว ปัจจุบันเกษตรกรใช้ประสบการณ์ในการวิเคราะห์โรคข้าวโดยการสังเกตด้วยตาเปล่าซึ่งอาจจะทำให้เกิดความผิดพลาดได้ หรือเกษตรกรต้องพึ่งพาผู้เชี่ยวชาญ ซึ่งการปรึกษาและติดต่อผู้เชี่ยวชาญนั้นอาจจะต้องใช้เวลาและค่าใช้จ่ายในการวิเคราะห์ตรวจหาโรคด้วย รวมถึงเกษตรบางรายอาจเข้าถึงผู้เชี่ยวชาญได้ยาก และเนื่องจากการวิเคราะห์โรคข้าวที่ไม่มีความแม่นยำจึงนำไปสู่การใช้สารเคมีที่ไม่ตรงตามโรคข้าวที่เป็น หรือใช้สารเคมีเกินความจำเป็น เช่น เกษตรกรวิเคราะห์ลักษณะของใบข้าวโดยที่ไม่แน่ชัดว่าเกิดโรคไหม้ หรือโรคใบจุดสีน้ำตาลซึ่งเป็นโรคที่มีลักษณะที่ค่อนข้างคล้ายกันจนบางครั้งไม่สามารถแยกแยะด้วยตาเปล่าได้ ดั้งนั้นเกษตรกรจึงใช้สารเคมีกำจัดทั้งสองโรคหรือเลือกกำจัดโรคใดโรคหนึ่ง ซึ่งส่งผลให้ไม่สามารกำจัดโรคใบข้าวได้ และเสียค่าใช้จ่ายเพิ่มมากขึ้นรวมถึงส่งผลกระทบกับสิ่งแวดล้อมอีกด้วย

นอกจากนี้ยังมีโรคใบข้าวที่มีลักษณะที่คล้ายกันอีกมากโดยเฉพาะในระยะแรกเริ่มของโรค เช่น โรคไหม้ โรคใบจุดสีน้ำตาล และโรคใบขีดสีน้ำตาล เป็นต้น ซึ่งโรคใบข้าวเหล่านี้หากไม่สังเกตุอย่างละเอียดก็จะแยกแยะออกได้ยากหรือไม่สามารถแยกแยะออกได้

ปัจจุบันเทคโนโลยีการประมวลผลภาพได้มีการพัฒนาอย่างก้าวกระโดด และ Convolutional Neural Networks (CNNs) เป็นเทคนิคที่ดีที่สุดสำหรับ Computer vision ในด้าน Deep learning ดังนั้น CNN จึงมีการประยุกต์ใช้ในการจำแนกพืชต่างๆ เช่น การจำแนกประเภทของสมุนไพร (Carranza-Rojas, Goeau et al. 2017) การจำแนกพืชที่เป็นโรคและไม่เป็นโรค

(Mohanty, Hughes et al. 2016, Sladojevic, Arsenovic et al. 2016, Ferentinos 2018) การระบุและประเมินความรุนแรงของความเครียดทางชีวภาพบนใบกาแฟ (Esgario, Krohling et al. 2020) สำหรับโรคข้าว ได้มีการใช้วิธีการจำแนกโรคข้าวโดยใช้วิธีการ Transfer learning ด้วยแบบจำลอง AlexNet ร่วมกับ SVM โดย (Shrivastava, Pradhan et al. 2019) และการตรวจหาโรคข้าวด้วย Deep transfer learning โดยใช้ฐานข้อมูลภาพสาธารณะและชุดข้อมูลจากสภาพแวดล้อมจริง (Chen, Zhang et al. 2020)

เนื่องจาก CNN จำเป็นต้องใช้รูปภาพจำนวนมากในการเรียนรู้แต่ภาพใบข้าวที่เป็นโรค และไม่เป็นโรคนั้นหาได้ยาก และไม่เป็นสาธารณะรวมถึงภาพเป็นภาพถ่ายจากห้องปฏิบัติการ ซึ่งมีพื้นหลังเป็นสีเดียว (สีขาว) ซึ่งสภาพแวดล้อมไม่ได้เหมือนกับสภาพแวดล้อมจริงรวมถึงยังไม่ครอบคลุมโรคใบข้าวที่มักพบในประเทศไทย ในงานวิจัยนี้จึงได้ทำการเก็บชุดข้อมูลรูปภาพเพิ่มเติมในพื้นที่จังหวัดนครปฐมระหว่างปี พ.ศ. 2,561 - 2,563 แต่ก็ถือว่าข้อมูลภาพที่ได้ทั้งหมดนั้นยังคงไม่เพียงพอ

ดังนั้น Image Data Augmentation Technique เป็นเทคนิคการประมวลผลภาพที่น่าสนใจที่สามารถเพิ่มจำนวนของภาพได้เป็นอย่างดีได้ถูกนำมาปรับใช้กับงานวิจัยนี้

งานวิจัยนี้แบ่งออกเป็น 3 ส่วนสำคัญด้วยกัน ได้แก่ ส่วนที่ 1 การสร้างฐานข้อมูลใหม่สำหรับภาพใบข้าวที่เป็นโรครวมถึงภาพใบข้าวที่ไม่เป็นโรคที่พบจากนาข้าวจริงในประเทศไทย ส่วนที่ 2 การทำการทดลองโดยการใช้ Pre-trained CNN ทั้งหมด 9 โมเดล ที่เป็นโมเดลเวอร์ชันใหม่ที่ได้รับความนิยมในปัจจุบัน ในการจำแนกโรคใบข้าวทั้งหมด 5 โรครวมถึงใบข้าวที่ไม่เป็นโรคที่พบได้ทั่วไปในประเทศไทย ซึ่งแสดงให้เห็นว่าโมเดลมีความสามารถในการจำแนกโรคใบข้าวได้ ส่วนที่ 3 นำเสนอการเพิ่มจำนวนของภาพด้วยเทคนิค Image Data Augmentation Technique เพื่อให้โมเดลมีประสิทธิภาพในการเรียนรู้และจำแนกได้ดียิ่งขึ้น

1.2 วัตถุประสงค์ของงานวิจัย

1) เพื่อศึกษาวิธีการ และประสิทธิภาพในการใช้ Pre-trained CNN ในการจำแนกโรคใบข้าว
2) เพื่อพัฒนาวิธีการสำหรับวิเคราะห์โรคใบข้าว ที่เกิดจาก เชื้อรา แบคทีเรีย และ เชื้อไวรัส ทั้งหมด 5 โรค ได้แก่ โรคไหม้ (Blast disease) โรคใบจุดสีน้ำตาล (Brown Spot disease) โรคขอบใบแห้ง (Bacterial Leaf Blight disease) โรคใบสีส้ม (Rice Tungro disease) โรคใบขีดสีน้ำตาล (Narrow Brown Spot disease) รวมถึง ข้าวที่ไม่เป็นโรค (Healthy) ด้วยภาพถ่ายได้อย่างแม่นยำ
3) เพื่อศึกษาแนวทางพื้นฐานของการใช้ Image Data Augmentation Technique ในการเพิ่มจำนวนของภาพเพื่อเพิ่มประสิทธิภาพให้กับโมเดล

1.3 ขอบเขตของงานวิจัย

1) พัฒนาระบบวิเคราะห์โรคใบข้าวที่เกิดจาก
    ก. เชื้อรา Pyricularia grisea Sacc. (โรคไหม้)
    ข. เชื้อรา Helminthosporium oryzae Breda de Haan. (โรคใบจุดสีน้ำตาล)
    ค. เชื้อรา Cercospora oryzae I. Miyake (โรคใบขีดสีน้ำตาล)
    ง. เชื้อแบคทีเรีย Xanthomonas oryzae pv. oyzae (ex-Ishiyama) Swings et al. (โรคขอบใบแห้ง)
    จ. เชื้อไวรัส Rice Tungro Bacilliform Virus (RTBV) Rice Tungro Spherical Virus (RTSV) (โรคใบสีส้ม)
    ฉ. ข้าวที่ไม่เป็นโรค (Healthy)
2) เก็บชุดรูปภาพข้อมูลฝึกสอนจากโรคใบข้าวที่เกิดขึ้นในนาปรัง อ.บางเลน จ.นครปฐม
3) สร้างขั้นตอนวิธี (Algorithm) โดยใช้หลักการของการเรียนรู้แบบเชิงลึกสำหรับคอมพิวเตอร์วิชัน (Deep learning for computer vision) เพื่อวิเคราะห์โรคใบข้าวในข้อ 1) โดยใช้พรีเทรน CNN โมเดล (Pre-trained CNN model) ได้แก่ Xception ResNet50V2 ResNet101V2 ResNet152V2 InceptionV3 InceptionResNetV2 DenseNet121 DenseNet169 และ DenseNet201

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้โมเดล (Model) ที่มีระสิทธิภาพที่ดีที่สุดการจำแนกโรคใบข้าวได้
2) ได้แนวทางในการเพิ่มประสิทธิภาพให้กับโมเดลด้วย Image Data Augmentation Technique
3) ได้วิธีการที่สามารถนำไปต่อยอดเป็นนวัตกรรมสำหรับช่วยเกษตรกรแก้ไขปัญหาเกี่ยวกับการวิเคราะห์โรคใบข้าวได้อย่างมีประสิทธิภาพ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 โรคข้าว

โรคพืช หรือโรคข้าว (ดารา เจตนะจิตร 2550) หมายถึง ความผิดปกติที่พืชแสดงออก สาเหตุของโรคอาจจะเกิดจากสิ่งมีชีวิตหรือไม่มีชีวิต อาจจะเกิดขึ้นจากสาเหตุเดียว ๆ หรือเกิดร่วมกันก็ได้ สิ่งมีชีวิตที่ทำให้เกิดโรคเรียกว่า เชื้อโรค เชื้อสาเหตุของโรคข้าวอาจเกิดจาก เชื้อรา แบคทีเรีย ไวรัส ไฟโตพลาสมา และไส้เดือนฝอย จุลินทรีย์เหล่านี้สามารถทำให้ข้าวแสดงอาการผิดปกติได้ชัดเจนที่ใบ ลำต้น กาบใบ รวงและเมล็ด

ลักษณะอาการเกิดโรค อาจแบ่งเป็นกลุ่มใหญ่ ๆ ได้ดังนี้

1) เตี้ยแคระแกรน
2) ใบมีสีผิดปกติ เช่น เหลือง หรือด่างซีด
3) ตายเป็นจุด ๆ ตามเนื้อเยื่อ เช่นใบจุด ใบขีด หรือใบแห้ง
4) อาการเหี่ยวเนื่องจากการอุดตันของท่อน้ำท่ออาหาร
5) ส่วนของพืชผิดปกติ เช่น โรคดอกกระถิน โรครากปม ฯลฯ

โรคพืชเกิดขึ้นได้เมื่อสภาพแวดล้อมเหมาะสม ต้นพืชอ่อนแอ และเชื้อโรคมีความรุนแรง อาจจำลองความสัมพันธ์ของการเกิดโรคได้ดังรูปที่ 2.1

รูปที่ 2.1 ความสัมพันธ์ของการเกิดโรค

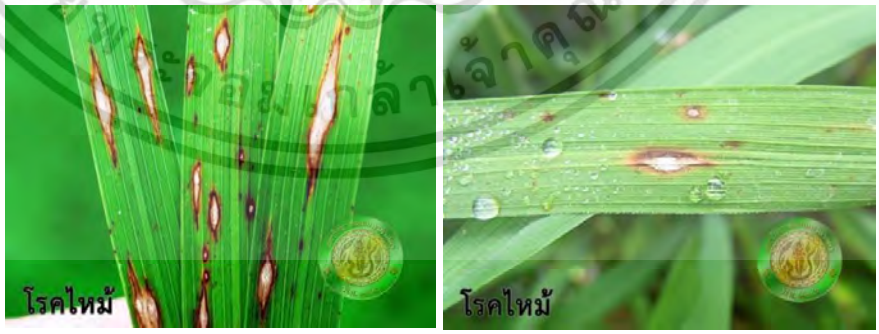
การแพร่ระบาดหรือการระบาดของโรคเกิดได้เมื่อเชื้อสาเหตุของโรคเพิ่มมากขึ้นในสภาพนิเวศของพืช นอกจากนี้เขตเกษตรกรรมก็เป็นปัจจัยหนึ่งที่เอื้อต่อการเกิดโรครุนแรง เช่น การระบาดของโรคไหม้จะรุนแรงในสภาพข้าวไร่มากกว่าข้าวนาสวน และหากใส่ปุ๋ยไนโตรเจนสูงจะทำให้เป็นโรครุนแรงยิ่งขึ้น การระบาดของโรคอาจจะมีปัจจัยของสิ่งมีชีวิตเข้ามาเกี่ยวข้อง เช่น การระบาดของโรคใบหงิกจะเพิ่มตามปริมาณแมลงพาหะเพลี้ยกระโดดสีน้ำตาล โดยตัวเต็มวัยจะมีความสามารถในการถ่ายทอดโรคได้สูงกว่าตัวอ่อน การแพร่ระบาดมักเป็นไปทางเดียวกับการอพยพของแมลง

การใช้พันธุ์ต้านทานในการป้องกันโรคข้าวเป็นวิธีที่ให้ผลดี แต่มักพบว่าข้าวมีความต้านทานลดลงอย่างรวดเร็วการใช้สารป้องกันกำจัดโรคยังมีความจำเป็น เนื่องจากบางโรคยังไม่มีพันธุ์ต้านทานโรค เช่น โรคกาบใบแห้ง โรคลำต้นเน่า และการใช้สารป้องกันกำจัดโรคยังคงมีประสิทธิภาพดีในการควบคุมโรคฉะนั้นการป้องกันกำจัดโรคจึงแตกต่างกันไปตามชนิดของเชื้อสาเหตุ เช่น การใช้พันธุ์ต้านทานในการป้องกันกำจัดโรคขอบใบแห้งและโรคเขียวเตี้ยดีกว่าโรคไหม้ เนื่องจากโรคไหม้เป็นโรคที่เชื้อสาเหตุ มีการเปลี่ยนแปลงอย่างรวดเร็ว

โรคที่แสดงลักษณะการถูกทำลายออกทางใบเช่น โรคไหม้ โรคใบจุดสีน้ำตาล โรคใบขีดสีน้ำตาล โรคขอบใบแห้ง โรคใบขีดโปร่งแสง โรคใบแถบแดง โรคใบสีส้ม และโรคใบสีแสด เป็นต้น

2.1.1 โรคไหม้ (Rice Blast Disease)

โรคไหม้ (Rice Blast Disease) สาเหตุเกิดจาก เชื้อรา Pyricularia grisea Sacc. พบมากในนาน้ำฝน ข้าวพันธุ์พื้นเมืองไวต่อช่วงแสง พบส่วนใหญ่ในภาคเหนือ ภาคตะวันตกเฉียงเหนือ ภาคตะวันตกและภาคใต้ ลักษณะอาการ ระยะกล้า ใบมีแผลจุดสีน้ำตาลรูปตา มีสีเทาอยู่ตรงกลางแผล ความกว้างของแผลประมาณ 2-5 มิลลิเมตรและความยาวประมาณ 10-15 มิลลิเมตร แผลสามารถขยายลุกลามและกระจัดกระจายทั่วบริเวณใบ ถ้าโรครุนแรงกล้าข้าวจะแห้งฟุบตาย อาการคล้ายถูกไฟไหม้ ระยะคอรวง (ระยะออกรวง) ถ้าข้าวกำลังจะเริ่มให้รวงเมื่อถูกเชื้อราทำลายเมล็ดจะลีบหมด แต่ถ้าเป็นโรคตอนรวงข้าวเก็บเกี่ยว จะปรากฏรอยแผลช้ำสีน้ำตาลที่บริเวณคอรวง ทำให้เปราะหักง่ายรวงข้าวร่วงหล่นเสียหายมาก การแพร่ระบาดพบโรคในแปลงที่ต้นข้าวหนาแน่น ทำให้อากาศไม่ถ่ายเทค่อนข้างเย็น อุณภูมิประมาณ 22-25 องศาเซลเซียส ลมแรงจะช่วยให้โรคแพร่กระจายได้ดี



รูปที่ 2.2 โรคไหม้ (Rice blast disease)

2.1.2 โรคใบจุดสีน้ำตาล (Brown Spot Disease)

โรคใบจุดสีน้ำตาล (Brown Spot Disease) สาเหตุเกิดจาก เชื้อรา Helminthosporium oryzae Breda de Haan. (Bipolaris oryzae (Brada de Haan) Shoemaker) พบมากทั้งนาน้ำฝนและนาชลประทาน ในภาคกลาง ภาคเหนือ ภาคตะวันตก ภาคตะวันออกเฉียงเหนือ และภาคใต้ อาการแผลที่ใบข้าว พบมากในระยะแตกกอมีลักษณะเป็นจุดสีน้ำตาล รูปกลมหรือรูปไข่ ขอบนอกสุดของแผลมีสีเหลืองขนาดเส้นผ่านศูนย์กลาง 0.5-1 มิลลิเมตร แผลที่มีการพัฒนาเต็มที่ขนาดประมาณ 1-2 x 4-10 มิลลิเมตร บางครั้งพบแผลไม่เป็นวงกลมหรือรูปไข่ แต่จะเป็นรอยเปื้อนคล้ายสนิมกระจัดกระจายทั่วไปบนใบข้าว แผลยังสามารถเกิดบนเมล็ดข้าวเปลือก ทำให้เมล็ดข้าวเปลือกสกปรกเสื่อมคุณภาพ เมื่อนำไปสีข้าวสารจะหักง่าย

รูปที่ 2.3 โรคใบจุดสีน้ำตาล (Brown spot disease)

2.1.3 โรคใบขีดสีน้ำตาล (Narrow Brown Spot Disease)

โรคใบขีดสีน้ำตาล (Narrow Brown Spot Disease) สาเหตุเกิดจาก เชื้อรา Cercospora oryzae I. Miyake พบมากทั้งนาน้ำฝนและนาชลประทาน ในภาคกลาง ภาคเหนือ ภาคตะวันตก ภาคตะวันออกเฉียงเหนือ และภาคใต้ อาการที่ใบมีสีน้ำตาลเป็นขีด ๆ ขนานไปกับเส้นใบข้าว มักพบในระยะข้าวแตกกอ แผลไม่กว้าง ตรงกลางเล็กและไม่มีรอยช้ำที่แผล ต่อมาแผลจะขยายมาติดกัน แผลจะมีมากตามใบล่างและปลายใบ ใบที่เป็นโรคจะแห้งตายจากปลายใบก่อน ต้นข้าวที่เป็นโรครุนแรงจะมีแผลสีน้ำตาลที่ข้อต่อใบได้เช่นกัน เชื้อนี้สามารถทำลายคอรวง ทำให้คอรวงเน่าและหักพับได้

รูปที่ 2.4 โรคใบขีดสีน้ำตาล (Narrow brown spot disease)

2.1.4 โรคขอบใบแห้ง (Bacterial Leaf Blight or Bacterial Blight Disease)

โรคขอบใบแห้ง (Bacterial Leaf Blight or Bacterial Blight Disease) สาเหตุเกิดจาก เชื้อแบคทีเรีย Xanthomonas oryzae pv. oyzae (ex Ishiyama) Swings et al. ชื่อเดิม X. campestris pv. oryzae (Ishiyama Dye) พบมากในนาน้ำฝน และนาชลประทานภาคเหนือ ภาคตะวันออกเฉียงเหนือ และภาคใต้ อาการโรคนี้เป็นได้ตั้งแต่ระยะกล้า แตกกอจนถึง ออกรวง ต้นกล้าก่อนนำไปปักดำจะมีจุดเล็ก ๆ ลักษณะช้ำที่ขอบใบของใบล่าง ต่อมาประมาณ 7-10 วัน จุดช้ำนี้จะกลายเป็นทางสีเหลืองยาวตามใบข้าว ใบที่เป็นโรคจะแห้งเร็วและสีเขียวจะจางลงเป็นสีเทา ๆ อาการในระยะปักดำจะแสดงหลังปักดำแล้วหนึ่งเดือนถึงเดือนครึ่ง ใบที่เป็นโรคขอบใบมีรอยขีดช้ำ ต่อมาจะเปลี่ยนเป็นสีเหลือง ที่แผลมีหยดน้ำสีครีมคล้ายยางสนกลม ๆ ขนาดเล็กเท่าหัวเข็มหมุด ต่อมาจะกลายเป็นสีน้ำตาลและหลุดไปตามลมหรือน้ำฝน ซึ่งจะทำให้โรคสามารถระบาดต่อไปได้แผลจะขยายไปตามความยาวของใบบางครั้งขยายเข้าไปข้างในตามความกว้างของใบ ขอบแผลมีลักษณะเป็นขอบลายหยักแผลนี้เมื่อนานไปจะเปลี่ยนเป็นสีเทา ใบที่เป็นโรคขอบใบจะแห้งและม้วนตามความยาว ในบางกรณีที่เชื้อมีปริมาณสูงเข้าทำลายทำให้ท่ออาหารอุดตัน ต้นข้าวทั้งต้นจะเหี่ยวเฉาและตายโดยรวดเร็วเรียกอาการของโรคนี้ว่า คริเสก

รูปที่ 2.5 โรคขอบใบแห้ง (Bacterial leaf blight or bacterial blight disease)

2.1.5 โรคใบขีดโปร่งแสง (Bacterial Leaf Streak Disease)

โรคใบขีดโปร่งแสง (Bacterial Leaf Streak Disease) สาเหตุเกิดจาก เชื้อแบคทีเรีย Xanthomonas oryzae pv. oryzicola (Fang et al.) Swings et al. พบมากในนาน้ำฝน และนาชลประทาน ภาคกลาง ภาคตะวันออกเฉียงเหนือ และภาคใต้ อาการโรคนี้เป็นได้ตั้งแต่แตกกอ จนถึงออกรวง อาการปรากฏที่ใบ ขั้นแรกเห็นเป็นขีดช้ำยาวไปตามเส้นใบ ต่อมาค่อย ๆ เปลี่ยนเป็นสีเหลืองหรือส้ม เมื่อแผลขยายรวมกันก็จะเป็นแผลใหญ่ แสงสามารถทะลุผ่านได้ และพบแบคทีเรียในรูปหยดน้ำสีครีมคล้ายยางสนกลม ๆ ขนาดเล็กเท่าหัวเข็มหมุดปรากฏอยู่บนแผล ส่วนความยาวของแผลขึ้นอยู่กับความต้านทานของพันธุ์ข้าว และความรุนแรงของเชื้อแต่ละพื้นที่ ในพันธุ์ที่ไม่มีความต้านทานเลย แผลจะขยายจนใบไหม้ไปถึงกาบใบด้วย ลักษณะอาการของแผลจะคล้ายคลึงกับเกิดบนใบ ส่วนในพันธุ์ต้านทานจำนวนแผลจะน้อย และแผลจะไม่ค่อยขยายตามแนวยาวรอบ ๆ แผลจะมีสีน้ำตาลดำ

รูปที่ 2.6 โรคใบขีดโปร่งแสง (Bacterial leaf streak disease)

2.1.6 โรคใบแถบแดง (Red Stripe Disease)

โรคใบแถบแดง (Red Stripe Disease) สาเหตุเกิดจาก เชื้อแบคทีเรีย Microbacterium sp. พบมากในนาชลประทาน เขตภาคกลาง อาการลักษณะอาการที่สำคัญของโรคเริ่มแรกใบข้าวจะเป็นจุดสีเหลือง แผลเป็นรูปกลมหรือรูปไข่ จากนั้นจะขยายจากที่เริ่มเป็นแถบไปทางปลายใบ สีของแผลจะเข้มขึ้นเป็นสีเหลืองส้ม บางครั้งจุดนี้จะมีสีเข้ม แผลที่เกิดขึ้นเมื่อเป็นรุนแรงจะแห้งทั้งใบ

รูปที่ 2.7 โรคใบแถบแดง (Red stripe disease)

2.1.7 โรคใบสีส้ม (Tungro Disease or Yellow Orange Leaf Disease)

โรคใบสีส้ม (Tungro Disease or Yellow Orange Leaf Disease) สาเหตุเกิดจากเชื้อไวรัส Rice Tungro Bacilliform Virus (RTBV) Rice Tungro Spherical Virus (RTSV) อาการต้นข้าวเป็นโรคได้ทั้งระยะกล้า แตกกอ ตั้งท้อง หากได้รับเชื้อตอนข้าวอายุอ่อน (ระยะกล้า-แตกกอ) ข้าวจะเสียหายมากกว่าได้รับเชื้อตอนข้าวอายุแก่ (ระยะตั้งท้อง-ออกรวง) ข้าวเริ่มแสดงอาการตั้งแต่อายุ 15-20 วัน ทั้งนี้แล้วแต่ว่าข้าวจะได้รับเชื้อระยะใด อาการเริ่มต้นใบข้าวจะเริ่มมีสีเหลืองสลับเขียวต่อมาจะเป็นสีเหลือง เริ่มจากปลายใบเข้าหาโคนใบ ถ้าเป็นรุนแรงในระยะกล้าต้นข้าวอาจถึงตาย ถ้าอาการแสดงหลังปักดำ เริ่มสังเกตได้ที่ใบเช่นกัน ต้นที่เป็นโรคจะเตี้ยแคระแกรนช่วงลำต้นสั้นกว่าปกติมาก ใบใหม่ที่โผล่ออกมามีตำแหน่งต่ำกว่าข้อต่อใบล่าสุด ถ้าเป็นรุนแรงอาจตายทั้งกอ ถ้าไม่ตายเมื่อถึงระยะออกรวงให้รวงเล็ก หรือไม่ออกรวงเลย และออกรวงล่าช้ากว่าปกติ

รูปที่ 2.8 โรคใบสีส้ม (Tungro disease or yellow orange leaf disease)

2.1.8 โรคใบสีแสด (Orange Leaf Disease)

โรคใบสีแสด (Orange Leaf Disease) สาเหตุเกิดจาก เชื้อไฟโตพลาสมา (Phytoplasma) อาการต้นข้าวเป็นโรคได้ ในระยะแตกกอ ตั้งท้อง ต้นข้าวที่เป็นโรคนี้ ใบแสดงอาการสีแสดจากปลายใบที่ใบล่าง และเป็นสีแสดทั่วทั้งใบยกเว้นเส้นกลางใบ ใบที่เป็นโรคทั้งใบจะม้วนจากขอบใบทั้งสองข้างเข้ามาหาเส้นกลางใบ ทำให้ใบแห้งในที่สุดต้นข้าวแตกกอน้อยแต่สูงตามปกติ ไม่มีอาการเตี้ยและตายอย่างรวดเร็ว โรคใบสีแสดนี้เกิดเป็นกอ ๆ ไม่แพร่กระจายเป็นบริเวณกว้าง

รูปที่ 2.9 โรคใบสีแสด (Orange leaf disease)

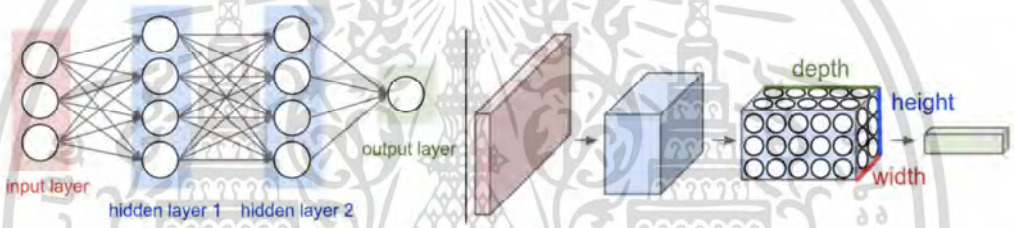
2.2 การเรียนรู้แบบเชิงลึก (Deep Learning)

การเรียนรู้แบบเชิงลึก (Deep Learning) (Dandawate and Kokare 2015) หมายถึง สถาปัตยกรรมเครือข่ายประสาทเทียมที่มีชั้น (Layer) การประมวลผลจำนวนมาก วิธีการเหล่านี้ได้รับการพัฒนาขึ้นมาเป็นอย่างมาก หรือเรียกว่า State-of-the-art ในการรู้จำเสียงพูด การรับรู้วัตถุในภาพ การตรวจจับวัตถุและโดเมนอื่น ๆ อีกมาก เช่น การค้นคว้ายา และ จีโนมิกส์ (Genomics) การเรียนรู้แบบเชิงลึกครอบคลุมถึงโครงสร้างที่ซับซ้อนในชุดข้อมูลขนาดใหญ่โดยใช้ขั้นตอนวิธี Back propagation ที่จะกำหนดว่าเครื่อง (Machine) ใดควรจะได้รับการเปลี่ยนค่าพารามิเตอร์ภายในที่ใช้ในการคำนวณในแต่ละชั้นจากการแทนในชั้นก่อนหน้า

2.3 โครงข่ายประสาทแบบคอนโวลูชันนอล

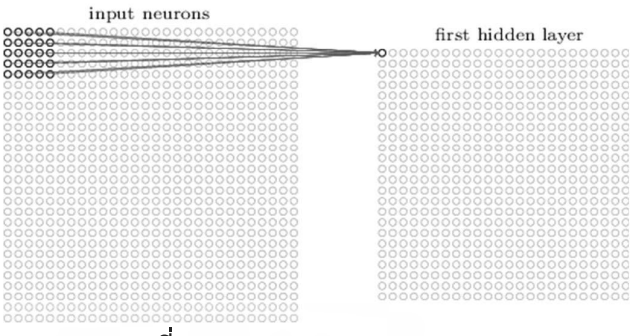
โครงข่ายประสาทแบบคอนโวลูชันนอล (Convolutional neural network) (Nielsen 2018) ปัจจุบันคือโมเดล State-of-the-art สำหรับการจำแนก (Classification) รูปภาพ ซึ่งโครงข่ายประสาทแบบคอนโวลูชันนอลมีความคล้ายคลึงกันกับโครงข่ายประสาทเทียมแบบปกติ ที่ประกอบไปด้วยนิวรอน (Neurons) จำนวนมาก ที่มีการเรียนรู้ด้วยค่าน้ำหนัก (Weights) และค่าไบแอส (Biases) ในแต่ละนิวรอนจะรับค่าข้อมูลนำเข้า (Input) จากนั้นทำการคำนวณ Dot product หรืออาจจะใช้ฟังฟังก์ชันที่ไม่เป็นเส้นตรง (Non-linearity Function) ในการคำนวณ ซึ่งโครงข่ายประสาทเทียมจะทำการรับข้อมูลนำเข้าคือพิกเซล (Pixcels) ของภาพจากนั้นจะคำนวณในแต่ละชั้น และมีการคำนวณฟังก์ชันการสูญเสีย (Loss function) และในชั้นสุดท้ายจะเชื่อมต่อด้วย Fully-connected ซึ่งชั้นนี้จะเป็นชั้นที่คำนวณผลลัพธ์โดยจำแนกว่าข้อมูลภาพที่นำเข้านั้นอยู่ในกลุ่ม (Class) ใด ซึ่งทั้งหมดที่กล่าวมาข้างต้นนี้ทั้งโครงข่ายประสาทแบบคอนโวลูชันนอล และโครงข่ายประสาทเทียมแบบปกติมีการทำงานที่เหมือนกันแต่สิ่งที่ต่างกันต่างกันของโครงข่ายประสาทเทียมทั้งสองแบบมีด้วยกัน ดังนี้

1) ข้อมูลนำเข้า กล่าวคือ โครงข่ายประสาทแบบคอนโวลูชันนอลจะรับข้อมูลนำเข้าที่เป็นรูปภาพโดยที่ไม่ผ่านกระบวนการใดในลักษณะสามมิติ (3D) ในขณะที่โครงข่ายประสาทเทียมแบบปกติข้อมูลนำเข้าจะเป็นเวกเตอร์เดี่ยว (Single vector) หรือ 1 มิติ ตัวอย่างเช่นใน โครงข่ายประสาทเทียมแบบปกติ มีข้อมูลนำเข้าเป็นภาพขนาด 32x32x3 (กว้าง 32, สูง 32, ช่องสี 3 สี) นิวรอนที่เชื่อมต่อกันในชั้นซ่อน (Hidden layer) แรกจะมีค่าน้ำหนักจำนวน 32 * 32 * 3 = 3072 แต่ถ้าภาพมีขนาดใหญ่ขึ้นก็จะทำให้มีค่าพารามิเตอร์จำนวนมากตามไปด้วยซึ่งจะส่งผลให้เกิด Overfitting แต่สำหรับโครงข่ายประสาทแบบคอนโวลูชันนอลจะรับข้อมูลนำเข้าเป็นรูปภาพที่ไม่ต้องผ่านกระบวนการใด โดยแต่ละชั้นนิวรอนจะจัดอยู่ใน 3 มิติ คือมีความกว้าง ความสูง และความลึก ดังรูปที่ 2.10



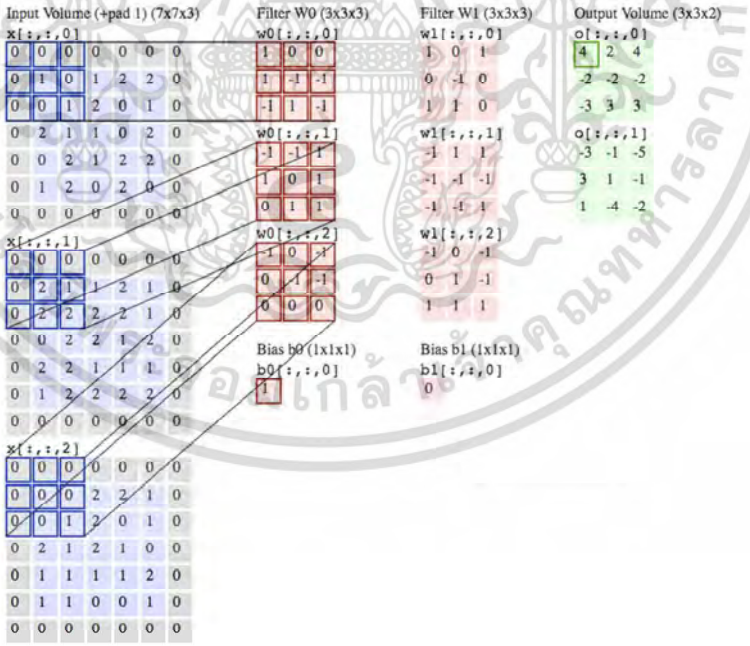
รูปที่ 2.10 สถาปัตยกรรมโครงข่ายประสาทแบบปกติ (ซ้าย) และสถาปัตยกรรมของชั้นแบบ 3 มิติในโครงข่ายประสาทแบบคอนโวลูชันนอล

2) เนื่องจากโครงข่ายประสาทแบบปกติจะทำการรับข้อมูลนำเข้าเป็นพิกเซล และทำการแปลงให้อยู่ในรูปเวกเตอร์เดี่ยว ดังนั้นค่าน้ำหนักของโครงข่ายประสาทแบบปกติจะมีค่าน้ำหนักเท่ากับข้อมูลนำเข้า ส่วนโครงข่ายประสาทแบบคอนโวลูชันนอลข้อมูลนำเข้าและชั้นซ่อนจะไม่ได้เชื่อมต่อกับพิกเซลทั้งหมดแต่จะเชื่อมต่อกันด้วยพื้นที่เล็ก ๆ เรียกว่า Local receptive field ซึ่งเป็นหนึ่งในแนวคิดพื้นฐานของโครงข่ายประสาทแบบคอนโวลูชันนอลดัง รูปที่ 2.11

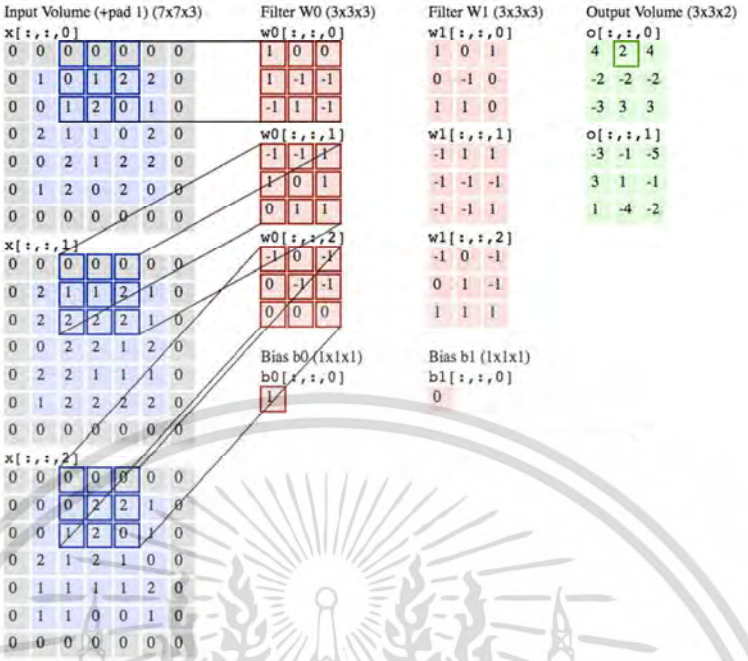


รูปที่ 2.11 Local receptive field

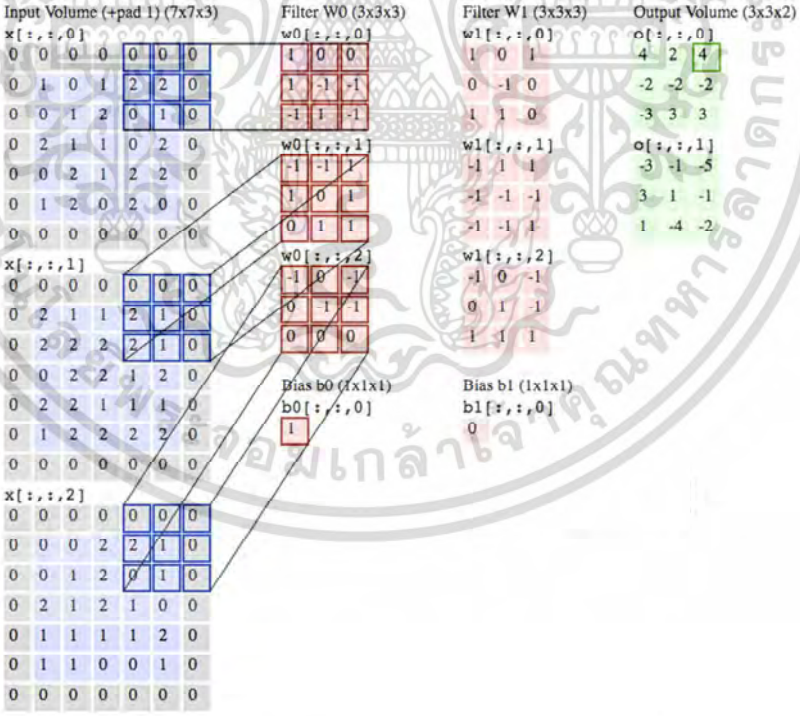
3) โครงข่ายประสาทแบบคอนโวลูชันนอลมีคุณสมบัติที่เรียกว่า Parameter Sharing ซึ่งจะจำกัดจำนวนของ ค่าน้ำหนัก และไบแอส โดยใช้พารามิเตอร์ร่วมกันในแต่ละชั้นความลึกของข้อมูลนำเข้า โดยใช้หลักการที่ว่า ถ้าคุณสมบัติหนึ่ง (Feature) เป็นประโยชน์ในการคำนวณที่ตำแหน่ง (x_1, y_1) แล้ว ดังนั้นจึงเป็นประโยชน์ในการคำนวณที่ตำแหน่งอื่น ๆ (x_2, y_2) ด้วย ดังแสดงตัวอย่างให้เห็นคุณสมบัติ Parameter Sharing ดังรูปที่ 2.12 ถึง 2.15 โดยกำหนดให้ข้อมูลนำเข้ามีขนาดเท่ากับ 7x7 และมีความลึกเท่ากับ 3 ค่าน้ำหนักมีขนาด 3x3 และมีความลึกเท่ากับ 3 และค่าไบแอสมีค่าเท่ากับ 1



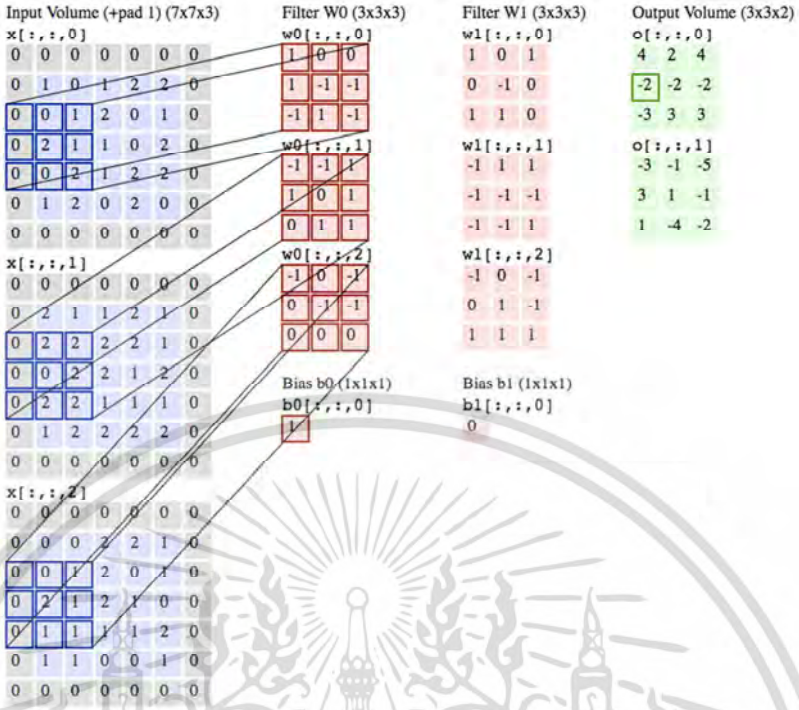
รูปที่ 2.12 ตัวอย่าง Parameters Sharing สำหรับการคำนวณในแต่ละชั้นของโครงข่ายประสาทแบบคอนโวลูชันนอล (1)



รูปที่ 2.13 ตัวอย่าง Parameters Sharing สำหรับการคำนวณในแต่ละชั้นของโครงข่ายประสาทแบบคอนโวลูชันนอล (2)



รูปที่ 2.14 ตัวอย่าง Parameters Sharing สำหรับการคำนวณในแต่ละชั้นของโครงข่ายประสาทแบบคอนโวลูชันนอล (3)



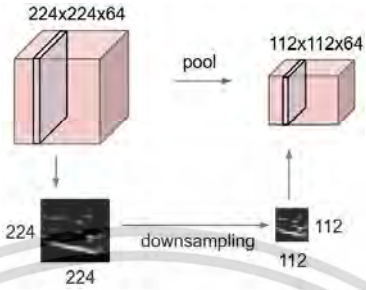
รูปที่ 2.15 ตัวอย่าง Parameters Sharing สำหรับการคำนวณในแต่ละชั้นของโครงข่ายประสาทแบบคอนโวลูชันนอล (4)

2.3.1 ส่วนประกอบของโครงข่ายประสาทแบบคอนโวลูชันนอล

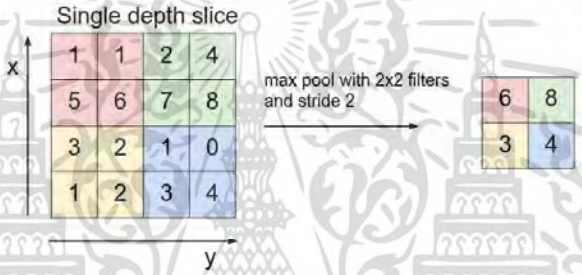
ส่วนประกอบของโครงข่ายประสาทแบบคอนโวลูชันนอล (TensorFlow 2018) มีด้วยกัน 3 ส่วน

1) ชั้นคอนโวลูชันนอล (Convolutional layers) จะถูกกำหนดโดยคอนโวลูชันฟิวเตอร์ (Convolution filters) โดยที่แต่ละพื้นที่ย่อย (Subregion) ของภาพจะถูกดำเนินการทางคณิตศาสตร์ ซึ่งเป็นค่าค่าเดียวใน Output feature map จากนั้นจะใช้ฟังก์ชั้นกระตุ้น (Activation) โดยปกติจะใช้ฟังก์ชั้นกระตุ้น ReLU ไปยังเอาท์พุทเพื่อนำค่าที่ไม่เป็นเชิงเส้นเข้าไปยังโมเดล
2) Pooling layers มีหน้าที่ในการลดขนาดเชิงพื้นที่ หรือเรียกว่า down samples เพื่อลดจำนวนของพารามิเตอร์และการคำนวนในโครงข่าย โดยรูปแบบที่พบมากที่สุดคือ Pooling layers ฟิวเตอร์จะมีขนาด 2x2 โดยใช้การดำเนินการ MAX จะเรียกว่า Max Pooling และจะเคลื่อนที่ไปทีละ 2 ทุก ๆ (เนื่องจากฟิวเตอร์จะมีขนาด 2x2) ความลึกของข้อมูลนำเข้า ส่วนความลึกจะมีขนาดเท่าเดิม ดังรูปที่ 2.16 และ 2.17 นอกจากนี้ยังมี ฟังก์ชันอื่น ๆ เช่น Average pooling และ L2-norm pooling แต่

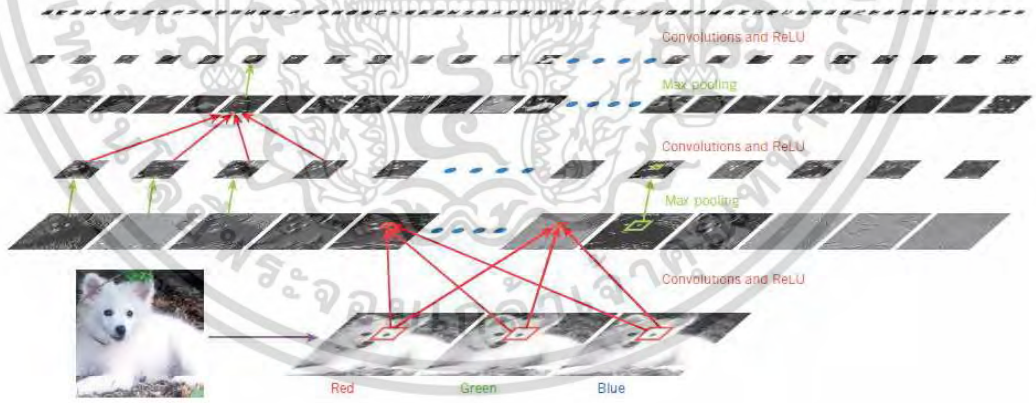
Average pooling มักถูกใช้ในอดีต แต่ได้ลดลงเมื่อไม่นานมานี้เมื่อเทียบกับ Max Pooling เพราะว่า Max Pooling สามารถทำงานได้ดีกว่าในทางปฏิบัติ



รูปที่ 2.16 ตัวอย่างการลดขนาดของ Pooling layers



รูปที่ 2.17 ตัวอย่างการลดขนาดของ Pooling layers ความลึกเดียว



รูปที่ 2.18 ตัวอย่างข้อมูลเข้าและข้อมูลออกของแต่ละชั้นของโครงข่ายประสาทแบบคอนโวลูชันนอล ภาพจาก [24]

3) Dense layers หรือ fully connected จะทำหน้าที่เพื่อจำแนกโดยการสกัดคุณสมบัติจาก Convolutional layers และ Pooling layers และใน Dense layers ทุกโหนด (Node) ในชั้นนี้จะเชื่อมต่อกับทุกโหนดในชั้นก่อนหน้า

โดยปกติโครงข่ายประสาทแบบคอนโวลูชันนอลจะประกอบไปด้วย Stack ของโมดูลที่สลับซับซ้อนซึ่งทำการสกัดคุณลักษณะ แต่ละโมดูลประกอบไปด้วย ชั้นคอนโวลูชันนอลตามด้วย Pooling layers และชั้นสุดท้ายจะตามด้วย Dense layers ในโครงข่ายประสาทแบบคอนโวลูชันนอล Dense layers ในชั้นสุดท้าย จะประกอบไปด้วยหนึ่งโหนดต่อหนึ่งเป้าหมาย ด้วยฟังก์ชัน Softmax ซึ่งจะให้ค่าระหว่าง 0 ถึง 1 ซึ่งจะสามารถตีความได้ว่าภาพจะตกอยู่ในกลุ่มเป้าหมายใด

2.4 การเรียนรู้แบบเชิงลึกสำหรับคอมพิวเตอร์วิชัน (Deep learning for computer vision)

เนื่องจากโครงข่ายประสาทแบบคอนโวลูชันนอล เป็นโครงข่ายที่มีความสามารถในการจำแนกรูปภาพได้ ดังนั้นการเรียนรู้แบบเชิงลึกสำหรับคอมพิวเตอร์วิชัน (Chollet 2021) จึงอาศัยโครงข่ายประสาทแบบคอนโวลูชันนอล ในการสร้างโมเดล ปัจจุบันระบบการเรียนรู้ของเครื่อง (Machine-learning systems) มีโครงสร้างข้อมูลพื้นฐานที่เรียกว่า Tensors ซึ่งถูกกำหนดโดย Google’s TensorFlow ใน Tensors จะบรรจุด้วย ตัวเลข ลักษณะทั่วไปของเมทริกซ์ (Matrix) ตัวเลขของมิติของข้อมูล การดำเนินการทั้งหมดจะดำเนินการผ่าน Tensors

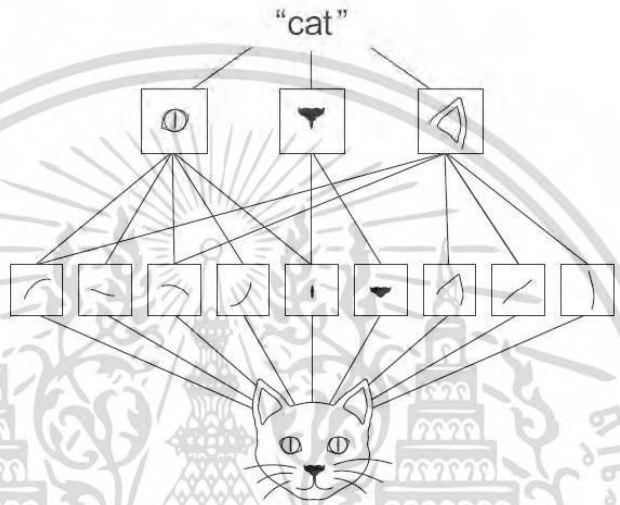
การดำเนินการของคอนโวลูชัน (Convolution operation) การดำเนินการระหว่าง Dense layers และชั้นคอนโวลูชัน (Convolution layers) คือ Dense layers จะเรียนรู้รูปแบบ (Pattern) โดยรวม (Global patterns) ส่วนชั้นคอนโวลูชันจะเรียนรู้รูปแบบพื้นฐาน (Local patterns) ดังรูปที่ 2.19

รูปที่ 2.19 การเรียนรูปของชั้นคอนโวลูชัน

ซึ่งลักษณะเฉพาะดังกล่าวทำให้โครงข่ายประสาทแบบคอนโวลูชันนอล มีคุณสมบัติที่น่าสนใจสองประการดังนี้

1) รูปแบบที่เรียนรู้จะไม่เปลี่ยนเมื่อย้ายวัตถุ คือ รูปแบบที่เรียนรู้ไม่ว่าวัตถุจะอยู่ส่วนใดของภาพเมื่อย้ายตำแหน่งไปอยู่ตำแหน่งใด ๆ

2) สามารถเรียนรู้ลำดับชั้นแบบพิเศษของรูปแบบได้ คือ ชั้นคอนโวลูชันชั้นแรกจะเรียนรู้รูปแบบพื้นฐาน เช่น ขอบภาพ และชั้นคอนโวลูชันชั้นที่ 2 จะเรียนรู้รูปแบบขนานใหญ่ที่ได้จากจากคุณสมบัติของชั้นแรก และอื่น ๆ และคุณสมบัตินี้เองจึงทำให้ให้โครงข่ายประสาทแบบคอนโวลูชันนอลเรียนรู้แนวคิดเกี่ยวกับภาพที่ซับซ้อน และเป็นนามธรรมมากขึ้น ดังรูปที่ 2.20



รูปที่ 2.20 ตัวอย่างการเรียนรู้ลำดับชั้นแบบพิเศษ

2.5 Transfer Learning

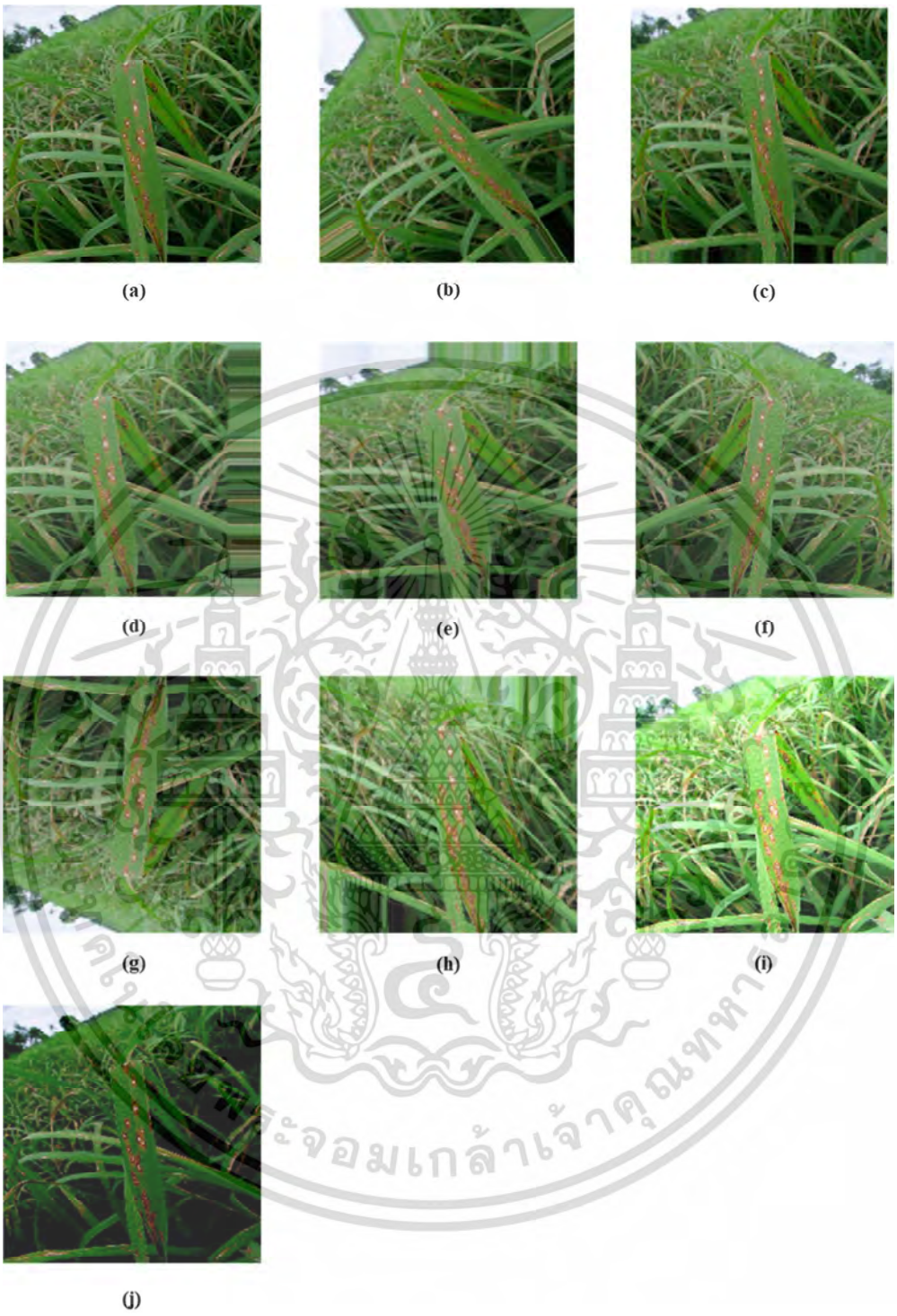
เนื่องจากโมเดลประเภทโครงข่ายประสาทเทียมแบบคอโวลูชันนอล ใช้ระยะเวลาในการฝึกสอนที่ยาวนานมาก และต้องใช้ชุดข้อมูลภาพขนาดใหญ่โดยการเรียนรู้ของโมเดลตัวแปรค่าน้ำหนักเริ่มต้น (Weight initialization) เริ่มต้นจากการสุ่ม และโมเดลจะเรียนรู้ด้วยชุดข้อมูลขนาดใหญ่จึงต้องมีการปรับค่าน้ำหนัก (Weight) จำนวนมาก ซึ่งต้องใช้ทรัพยากรและเวลาในการฝึกสอนที่มากขึ้นด้วย ดังนั้นการใช้วิธีการ Transfer Learning (Tan, Sun et al. 2018) เป็นการถ่ายโอนความรู้จากโมเดลที่ได้รับการฝึกสอนมาแล้วจากชุดข้อมูลขนาดใหญ่ (ใช้ค่าน้ำหนักที่ได้รับการปรับมาแล้ว) ซึ่งชุดข้อมูลขนาดใหญ่ที่โมเดลได้รับการฝึกสอนมานั้นมาจากการแข่งขัน ILSVRC (The ImageNet large scale visual recognition challenge) (Deng, Dong et al. 2009) ที่ Google ได้จัดขึ้น ซึ่งเป็นการแข่งขันการสร้างโมเดลแบบคอนโวลูชันนอลนำมาประมวลผลด้วยข้อมูลภาพขนาดใหญ่ ด้วยชุดข้อมูล ImageNet ประกอบไปด้วยภาพ 1,000 Class ชุดข้อมูลฝึกสอน 1,281,167 ภาพ ชุดข้อมูลตรวจสอบ (Validation) 50,000 ภาพ และชุดข้อมูลทดสอบ 100,000 ภาพ ดังนั้นจะเห็นได้ว่าโมเดลได้รับการ

ฝึกสอนมาแล้วจากชุดข้อมูลขนาดใหญ่การใช้เทคนิค Transfer Learning ทำให้การฝึกสอนโมเดลใช้เวลา และทรัพยากรน้อยกว่าการฝึกสอนโมเดลตั้งแต่ต้น (Sketch) ซึ่งจะมีจำนวนของภาพที่น้อยกว่าหรือต้องใช้ทรัพยากรและเวลาที่มากกว่าในการฝึกสอน (Bengio 2012)

2.6 Image Data Augmentation Technique

การเรียนรู้ของโครงข่ายประสาทแบบคอนโวลูชันนอลจะเรียนรู้ได้ดีเมื่อมีจำนวนภาพที่มากขึ้น แต่ในทางเป็นจริงแล้วการเข้าถึงชุดข้อมูลภาพหรือการเก็บรวบรวมภาพ รวมถึงชุดข้อมูลภาพในสาธารณะนั้นมีอยู่อย่างจำกัด ดังนั้น Image Data Augmentation Technique (Sladojevic, Arsenovic et al. 2016, Alzubaidi, Zhang et al. 2021) เป็นอีกหนึ่งเทคนิคที่เพิ่มจำนวนของรูปภาพโดยการจำลองภาพใหม่ขึ้นมาด้วยวิธีการสร้างรูปภาพใหม่จากรูปภาพต้นฉบับ โดยใช้หลักการในการประมวลผลภาพ (Image processing) โดยรูปภาพใหม่จะถูกสร้างขึ้นจากรูปภาพต้นฉบับโดยการดำเนินการบางอย่าง เช่น การหมุน การซูม การเลื่อน การพลิก และกระบวนการอื่นๆ ที่เกี่ยวข้องกับความสว่างและสี ดังรูปที่ 2.21 แสดงการใช้การประมวลผลภาพเพื่อเพิ่มจำนวนของภาพ นอกจากนี้ยังสามารถใช้การประมวลผลภาพร่วมกันได้ เช่น การใช้การหมุนร่วมกับการซูมจะทำให้เกิดภาพใหม่ได้เช่นกัน ดังนั้นเทคนิคนี้จึงทำให้การจำลองภาพใหม่ขึ้นมาจากภาพเดิมได้เป็นจำนวนมาก

ทั้งนี้การใช้ Image Data Augmentation Technique มีข้อควรระวังในการประมวลผลภาพอยู่ด้วยเช่นกัน คือจะต้องคำนึงถึงวัตถุที่เห็นโดยโมเดลและวัตถุจริงต้องสอดคล้องกัน เช่น ภาพของสุนัขที่ขาของสุนัขจะต้องอยู่ด้านล่างของภาพเป็นต้น แต่หากมีการประมวลผลภาพโดยการหมุนภาพให้ขาสุนัขขึ้นไปอยู่ด้านบนก็จะทำให้โมเดลเรียนรู้ในสิ่งที่ผิดกับความเป็นจริงได้และส่งผลให้โมเดลมีประสิทธิภาพในการจำแนกรูปภาพในความเป็นจริงได้ไม่ถูกต้องเช่นกัน



รูปที่ 2.21 (a) ภาพต้นฉบับ (b) การหมุน (c) การเลื่อนในแนวสูง (d) การเลื่อนในแนวกว้าง (e) การ Zoom (f) การกลับด้านในแนวตั้ง (g) การกลับด้านในแนวนอน (h) การ shear (i) การเพิ่มความสว่าง (j) การเปลี่ยนช่องสี

2.7 InceptionV3

Inception V3 (Szegedy, Vanhoucke et al. 2016) เป็นโมเดลการเรียนรู้เชิงลึกที่ใช้โครงข่ายประสาทเทียมแบบคอนโวลูชันที่ใช้สำหรับการจำแนกภาพ InceptionV3 พัฒนามาจาก InceptionV1 ซึ่งเปิดตัวในชื่อ GoogLeNet (Szegedy, Liu et al. 2015) ในปี 2014 ซึ่งได้รับการพัฒนาโดยทีมงานของ Google

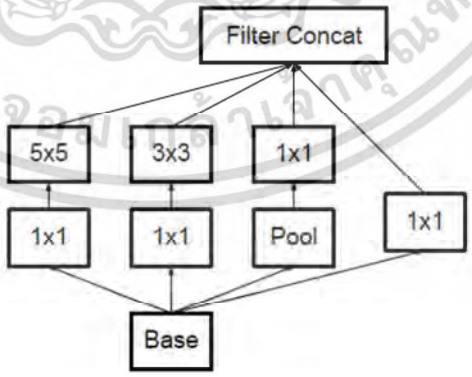
2.7.1 Inception โมดูล

เมื่อโครงสร้างโมเดลโครงข่ายประสาทเทียมแบบคอนโวลูชันที่มีสถาปัตยกรรมที่มีความลึกมากเกินไปจะส่งผลให้โมเดลเกิด Overfitting ได้กล่าวคือ โมเดลเรียนรู้ในชุดข้อมูลฝึกสอนได้ดี แต่ให้ผลไม่ดีในชุดข้อมูลทดสอบ โดย Inception โมดูลได้ถูกนำมาแก้ปัญหานี้ด้วยการนำ Filter หลายตัวที่มีขนาดต่างกันมาวางในระดับ (Level) เดียวกัน เพื่อให้โมเดลขยายไปแนวข้าง ไม่ใช่แนวลึกลงไป

โมเดล Inception ประกอบไปด้วยส่วนสำคัญคือโมดูล Inception จำนวนมาก โดยโมดูลของ Inception ประกอบไปด้วย Layers ที่ทำงานขนานกัน 4 Layers ได้แก่

1) 1×1 convolution
2) 3×3 convolution
3) 5×5 convolution
4) 3×3 max pooling

ชั้นของคอนโวลูชันจะถูกแยกส่วนออกเป็นส่วนเล็ก ๆ ตามรูปที่ 2.22

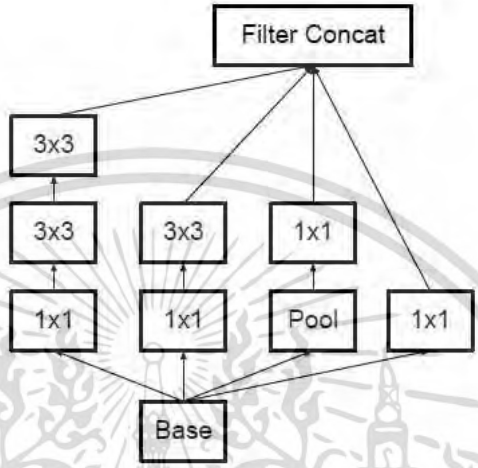


รูปที่ 2.22 โมดูล Inception

2.7.2 สถาปัตยกรรมของ InceptionV3

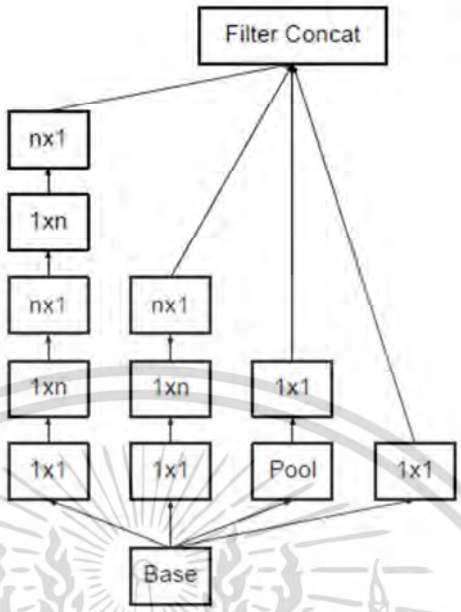
InceptionV3 ถูกพัฒนามาจาก GoogLeNet ดังนี้

1) สถาปัตยกรรมของ Inception ชั้นคอโวลูชันที่แยกส่วนออกมีขนาดลดลง จาก 5 X 5 เป็น 3 X 3 เพื่อลดจำนวนพารามิเตอร์ในการคำนวณ ดังรูปที่ 2.23

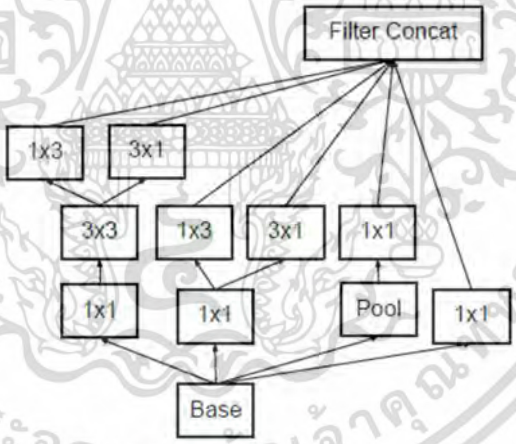


รูปที่ 2.23 ชั้นของคอนโวลูชันที่ถูกแยกส่วนออกเป็นส่วนเล็ก ๆ ที่ลดขนาดลง

2) การแยกตัวประกอบในชั้น Asymmetric Convolutions จะอยู่ในรูปแบบ n×1 โดยจะมีการแทนที่ของชั้น 3 X 3 Convolutions ด้วย 3 X 1 Convolutions เหมือนกันทั้ง 2 Layer ตามรูปที่ 2.24 และจากการแยกตัวประกอบในชั้นนี้ สถาปัตยกรรมของ Inception จะมีลักษณะดังรูปที่ 2.25

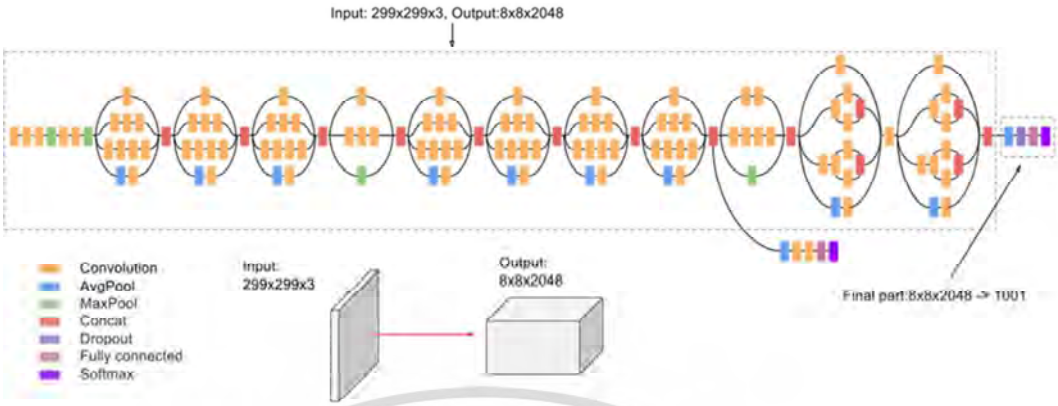


รูปที่ 2.24 การแยกตัวประกอบในชั้น Asymmetric Convolutions



รูปที่ 2.25 การแยกตัวประกอบในชั้น Asymmetric Convolutions (1 X 3)

หลังจากโมเดลได้รับการพัฒนาสถาปัตยกรรมของ Inception แล้ว InceptionV2 จะมีสถาปัตยกรรมทั้งหมดตามรูปที่ 2.26 มีจำนวน Layer เท่ากับ 42 Layer ซึ่งมากกว่าเวอร์ชันก่อนหน้าเล็กน้อย



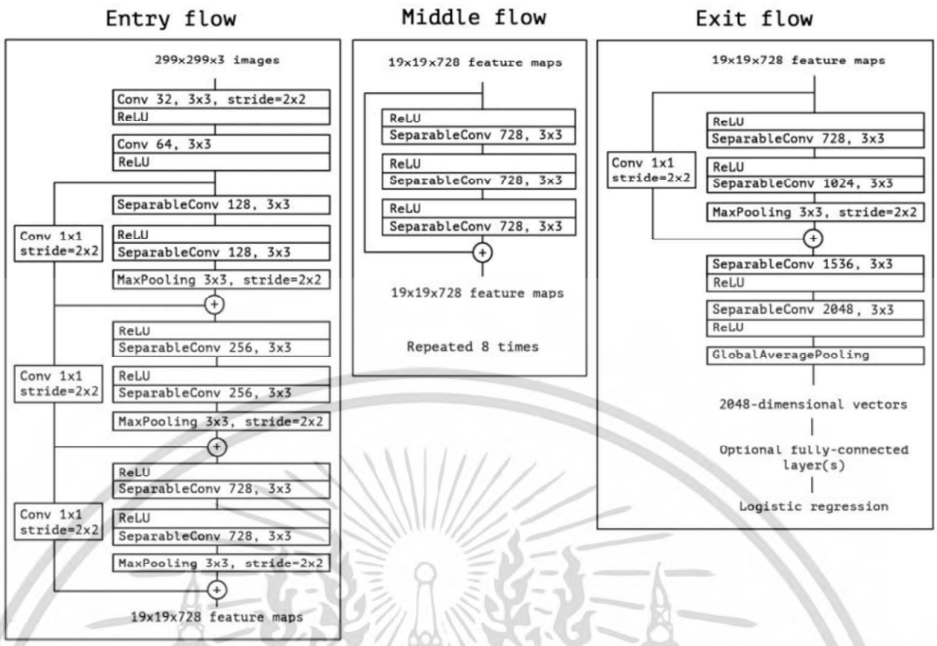
รูปที่ 2.26 สถาปัตยกรรม InceptionV2

2.8 Xception

Xception (Chollet 2017) ย่อมาจาก “Extreme Inception” ซึ่งใช้หลักการของ Inception โดยใช้ 1 x 1 Convolutions เพื่อบีบอัด Input ต้นฉบับ และจากแต่ละช่อง Input space จะใช้ Filter ที่ต่างกันในแต่ละความลึก ใน Xception โมเดลมีการทำขั้นตอนย้อยกลับ คือ แทนที่จะใช้ Filter ในแต่ละความลึกก่อนแล้วจึงบีบอัด Input โดยการใช้ 1 x 1 Convolutions แต่จะเป็นการแยกส่วนของ Convolutions ในเชิงลึก ซึ่งเป็นสถาปัตยกรรมการออกแบบโครงขายประสาทเทียมตั้งแต่ช่วงปี 2014 ได้ถูกนำมาใช้ อีกหนึ่งประการที่ Inception และ Xception ต่างกันคือ ในโมเดล Inception การมีความเป็นเชิงเส้นและไม่เป็นเชิงเส้น (Linearity and Non-linearity) หลังจากการดำเนินการแรกใน Inception โมเดลจะใช้ ReLU ฟังก์ชัน ที่เป็นเชิงเส้น แต่ใน Xception จะใช้ ReLU ฟังก์ชันที่ไม่เป็นเชิงเส้น

2.8.1 สถาปัตยกรรมของ Xception

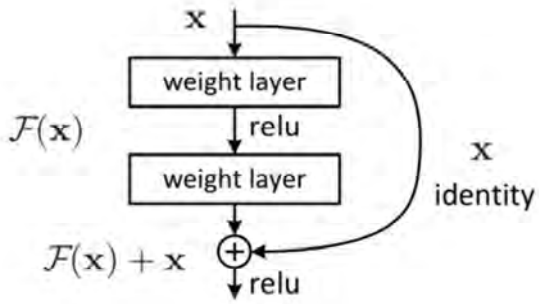
สำหรับสถาปัตยกรรมระบบของ Xception จะแบ่งออกเป็น 3 ส่วนหลัก โดยที่ข้อมูลแรกจะผ่าน Entry flow แล้วจึงผ่านเข้า Middle flow (ทำซ้ำใน Middle flow 8 รอบ) และสุดท้ายจะไปที่ Exit flow ดังรูปที่ 2.27



รูปที่ 2.27 สถาปัตยกรรมของโมเดล Xception

2.9 ResNetV2

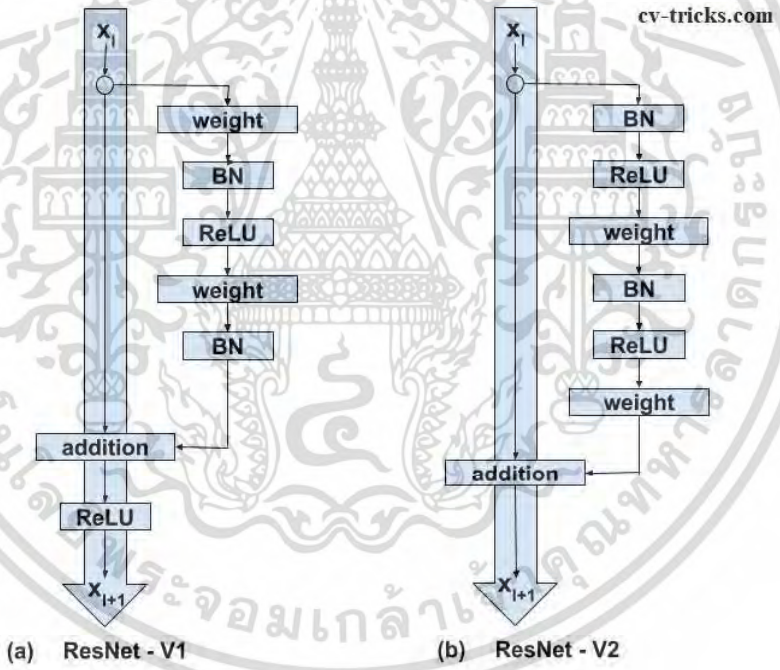
ResNet หรือ Deep Residual Network ถูกนำเสนอในงานวิจัยเรื่อง Deep Residual Learning for Image Recognition (He, Zhang et al. 2016) แก้ปัญหาเรื่อง Vanishing gradient ในโมเดลที่มีสถาปัตยกรรมที่ลึกมากๆ เช่นกันด้วยการสร้าง Shortcut สำหรับโครงข่าย กล่าวคือ ปัญหา Vanishing gradient จะเกิดขึ้นในกรณีดังนี้ เมื่อ Input X และ X ถูกส่งจากชั้นของคอนโวลูชันชั้นแรก ต่อมาถูกส่งต่อไปในชั้นถัด ๆ ไป และหากถูกส่งต่อไปเรื่อย ๆ ค่าของ X ที่ Input จะถูกกลืนเลือนหายไป ดังนั้นสถาปัตยกรรมของ ResNet ได้ถูกออกแบบมาโดยการเพิ่ม Shortcut เพื่อส่งผ่าน X ไปยังบล็อกถัดไป ดังรูปที่ 2.28



รูปที่ 2.28 ResNet บล็อก

2.9.1 สถาปัตยกรรมของ ResNetV2

จากรูปที่ 2.29 แสดงสถาปัตยกรรมพื้นฐานของ ResNetV1 และ ResNetV2 (He, Zhang et al. 2016)



รูปที่ 2.29 สถาปัตยกรรม ResNetV1 (ซ้าย) และ ResNetV2 (ขวา)

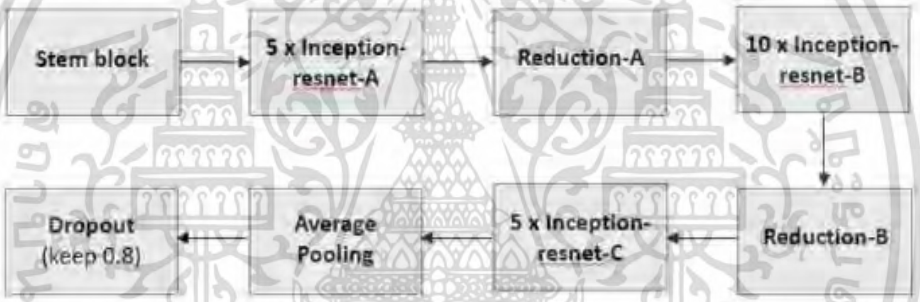
ความแตกต่างที่สำคัญระหว่าง ResNetV1 และ ResNetV2 มีดังนี้

1) ResNetV1 เพิ่ม non-linearity (ReLU) ที่สองหลังจากดำเนินการเพิ่มระหว่าง x และ F(x) ResNetV2 ได้ลบ non-linearity (ReLU) สุดท้ายออก ดังนั้นการล้างเส้นทางของ Input ไปยัง Output ในรูปแบบของ Identity connection

2) ResNetV2 ใช้ Batch Normalization และ ReLU กับ Input ก่อนการคูณด้วยเมทริกซ์น้ำหนัก (Convolution operation) ResNetV1 จะทำ Convolution ตามด้วย Batch Normalization และ ReLU

2.10 InceptionResNetV2

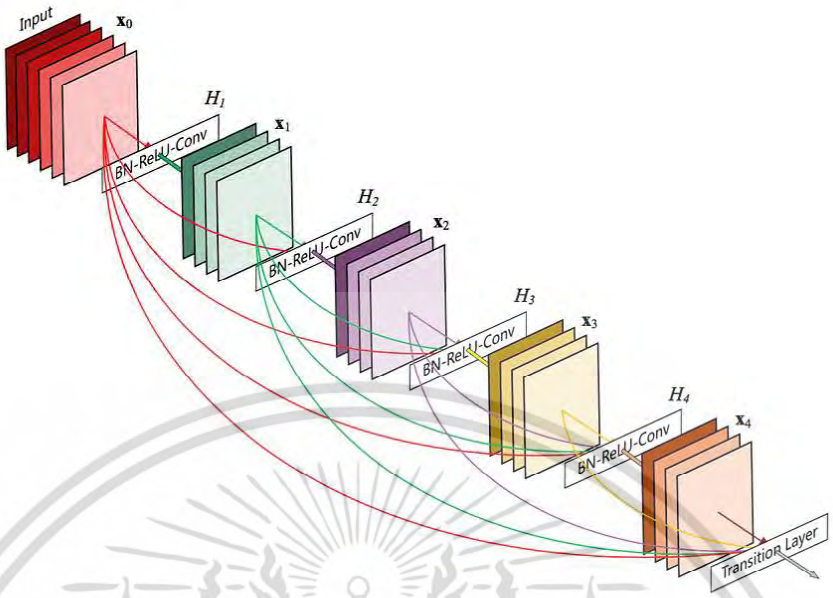
InceptionResNetV2 (Szegedy, Ioffe et al. 2017) เป็นการรวมกันของโครงสร้าง Inception และ Residual (ResNet) ในบล็อก InceptionResNet Convolutional Filter หลายขนาดจะรวมกับ Residual connections การใช้ Residual connections ไม่เพียงแต่หลีกเลี่ยงปัญหาการ Degradation problem ที่เกิดจากโครงสร้างสถาปัตยกรรมระบบที่อยู่ลึกเท่านั้น แต่ยังช่วยลดเวลาการฝึกอบรมอีกด้วย รูปที่ 2.30 แสดงสถาปัตยกรรมเครือข่ายพื้นฐานของ InceptionResnetV2



รูปที่ 2.30 สถาปัตยกรรม InceptionResNetV2

2.11 DenseNet

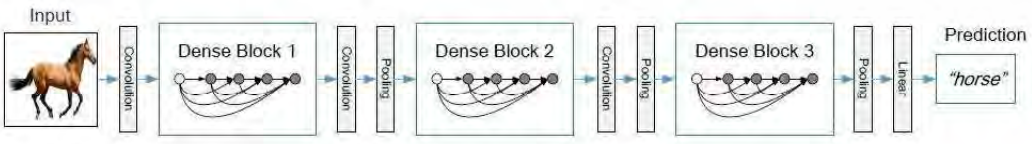
DenseNet (Huang, Liu et al. 2017) เป็นอีกหนึ่งสถาปัตยกรรมของการออกแบบระบบของโครงข่ายประสาทแบบคอนโวลูชัน ที่ออกแบบมาแก้ปัญหา Vanishing gradient เช่นกัน โดยมีหลักการโดย Output จะถูกส่งต่อไปยังบล็อกถัดไปทุกบล็อก (ยกเว้น Input layer) ดังรูปที่ 2.31



รูปที่ 2.31 สถาปัตยกรรม DenseNet

DenseNets จะแบ่งออกเป็น DenseBlocks โดยที่ขนาดของ Feature maps จะคงที่ภายในบล็อก จะจำนวน Filter ระหว่างบล็อกจะเปลี่ยนไป Layer ระหว่างบล็อกจะเรียกว่า Transition Layers ที่จำทำการลดจำนวนของ Chanel ลงเหลือครึ่งหนึ่งจาก Chanel ที่มีอยู่ และจะมีการดำเนินการสามอย่างติดต่อกัน คือ Batch normalization (BN), Rectified linear unit (ReLU) และ convolution (Conv)

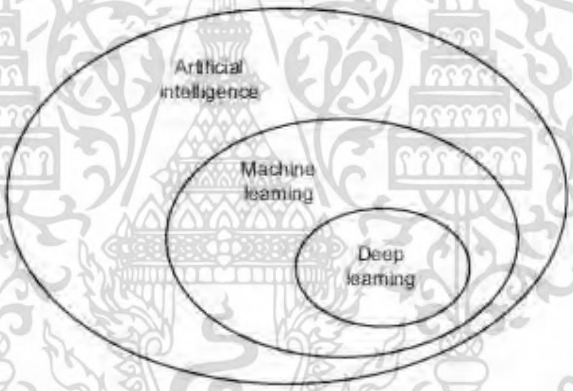
จากรูปที่ 2.32 แสดงสถาปัตยกรรมของระบบที่ประกอบไปด้วย DenseBlocks จำนวน 3 บล็อก Layer ระหว่างบล็อกที่อยู่ติดกันสองบล็อกคือ Transition Layers ซึ่งจะดำเนินการเปลี่ยนแปลง Down sampling (ตัวอย่างเช่น เปลี่ยนขนาดของ Feature-maps) โดยการใช้การดำเนินการ Convolution และ Pooling ในขณะที่ภายใน DenseBlocks ขนาดของ Feature maps จะเท่ากันเพื่อให้สามารถใช้ Feature concatenation ได้



รูปที่ 2.32 สถาปัตยกรรมของระบบที่ประกอบไปด้วย DenseBlocks จำนวน 3 บล็อก

2.12 งานวิจัยที่เกี่ยวข้อง

ปัจจุบันการพัฒนาเทคโนโลยีแบบก้าวกระโดดจึงส่งผลให้ความสามรถในการคำนวณของเครื่องเพิ่มมากขึ้น โดยเฉพาะอย่างยิ่งการพัฒนาระบบการคำนวณโดยใช้ หน่วยประมวลผลภาพ (Graphical Processing Units (GPU)) จึงส่งผลให้เกิดการพัฒนาโมเดล และกระบวนการวิธีใหม่ในการคำนวน อย่างเช่นโมเดลการเรียนรู้แบบเชิงลึก (LeCun, Bengio et al. 2015) ซึ่งถือได้ว่าเป็นหมวดหมู่ของการเรียนรู้ของเครื่อง (Machine learning) (Haykin 2009) แสดงให้เห็นในรูปที่ 2.33 เนื่องจากโมเดลมีความเป็นไปได้ในการคำนวณ จึงเกิดการปฏิวัติในการคำนวณด้านต่าง ๆ เช่น การรู้จำภาพ การรับรู้เสียง และการประมวลผลอื่น ๆ ที่มีความซับซ้อน หรือโมเดลที่ต้องการวิเคราะห์ข้อมูลจำนวนมาก ๆ สำหรับเครื่องมือพื้นฐานของการเรียนรู้แบบเชิงลึกที่ใช้กันคือโครงข่ายประสาทแบบคอนโวลูชันนอล ซึ่งเป็นหนึ่งในเทคนิคที่มีประสิทธิภาพที่ดีที่สุดสำหรับการสร้างโมเดลที่มีการวิเคราะห์ข้อมูลความซับซ้อนและมีการรู้จำรูปแบบของภาพ



รูปที่ 2.33 แผนภาพแผนภาพเวนน์-ออยเลอร์ ความสัมพันธ์ของศาสตร์ปัญญาประดิษฐ์

วิธีการตรวจจับหรือจำแนกโรคพืชด้วยภาพที่ผ่านมาสามารถแยกออกได้เป็นสองกลุ่มหลักด้วยกัน ได้แก่ การตรวจจับหรือจำแนกโรคพืชด้วยกระบวนการประมวลผลภาพร่วมกับการจำแนกด้วยการเรียนรู้ของเครื่องแบบเดิม ที่ใช้การคำนวนโดยที่ไม่ได้ใช้ทรัพยากรของเครื่องมากนัก (ไม่ได้มีการนำ Deep learning มาใช้) และการจำแนกด้วยการใช้ Deep learning ซึ่งการประยุกต์ใช้ Deep learning ได้รับความนิยมเมื่อไม่กี่ปีที่ผ่านมา

เนื่องจากปัจจุบันความสามรถในการคำนวณของเครื่องเพิ่มมากขึ้นดังที่กล่าวไว้ข้างต้นจึงทำให้ Deep learning ถูกเข้ามาใช้ในการรู้จำรูปแบบของภาพมากขึ้น และได้ถูกนำเข้ามาใช้ในการวิเคราะห์ในด้านการเกษตรเมื่อไม่กี่ปีที่ผ่านมาและให้ผลลัพธ์ได้เป็นที่น่าพอใจ ตัวอย่างเช่น การนำ

สถาปัตยกรรมต่าง ๆ ของโครข่ายประสาทแบบคอนโวลูชันนอลมาประยุกต์ใช้ในการจำแนกโรคพืช ดังนี้

1.) Using deep learning for image-based plant disease detection (Mohanty, Hughes et al. 2016)
2.) Deep neural networks based recognition of plant diseases by leaf image classification (Sladojevic, Arsenovic et al. 2016)
3.) Deep learning models for plant disease detection and diagnosis (Ferentinos 2018)

นอกจากนี้ยังมีการประยุกต์ใช้โครข่ายประสาทแบบคอนโวลูชันนอลในพืชที่เฉพาะเจาะจงมากขึ้น เช่น การตรวจจับโรคมะเขือเทศและศัตรูพืชของต้นมะเขือเทศ (Fuentes, Yoon et al. 2017) และการตรวจจับโรคมันสำปะหลัง (Ramcharan, Baranowski et al. 2017) ซึ่งให้ผลลัพธ์เป็นที่น่าพอใจเป็นอย่างมาก

และจากงานวิจัยของ (Pawara, Okafor et al. 2017) ได้มีการเปรียบเทียบประสิทธิภาพระหว่างเทคนิคการรู้จำรูปแบบแบบเดิม และโมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล ในการระบุชนิดของพืช (ซึ่งจะอธิบายในหัวข้อที่ 2.5.3) และงานวิจัยนี้ได้แสดงให้เห็นว่าโมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล มีประสิทธิภาพกว่าการเรียนรู้แบบเดิม

2.12.1 Using deep learning for image-based plant disease detection

บทความวิจัยเรื่อง Using deep learning for image-based plant disease detection (Mohanty, Hughes et al. 2016) ได้ทำการทดลองเกี่ยวกับการใช้โมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล ในการจำแนกพืชที่เป็นโรคและไม่เป็นโรคโดยจำแนกจาก 26 สายพันธุ์พืช 14 โรคพืช โดยจำแนกเป็น 38 คลาส และใช้ภาพใบพืชทั้งหมด 54,306 ภาพ จากฐานข้อมูล PlantVillage โดยได้ทำการทดลอง 60 รูปแบบโดยมีการกำหนดค่าที่ต่างกันตามพารามิเตอร์ดังต่อไปนี้

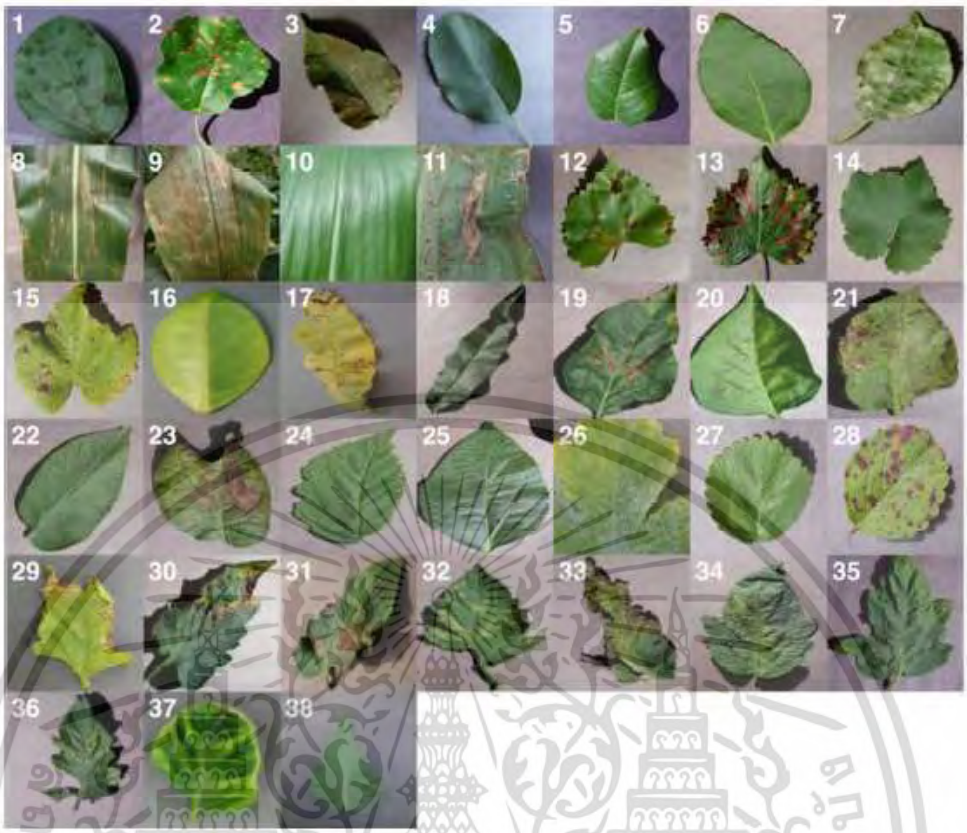
1) สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอล ได้แก่ AlexNet และ GoogLeNet
2) วิธีการฝึกสอน ได้แก่ ถ่ายโอนการเรียนรู้ (Transfer learning) และ การฝึกสอนจาก Scratch (Training from Scratch)
3) ประเภทของชุดข้อมูลฝึกสอน ได้แก่ ภาพสี ภาพโทนสีเทา (Grayscale) และภาพแบ่งส่วนของใบ (Leaf Segmented)

4) อัตราส่วนของชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบ

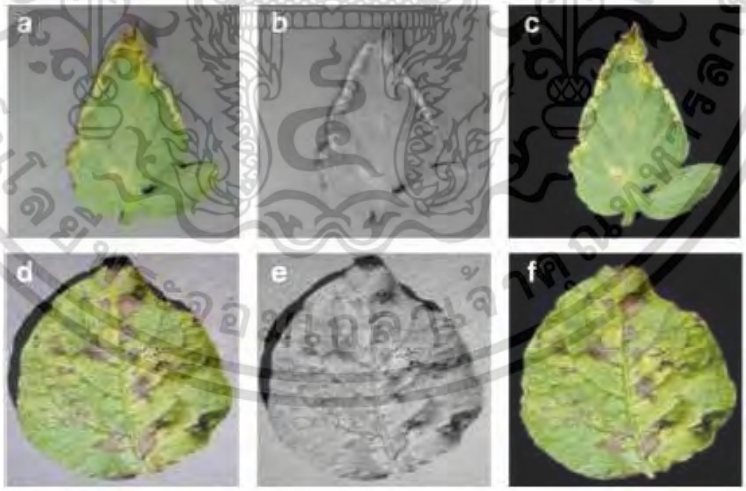
- ฝึกสอน 80% ทดสอบ 20%
- ฝึกสอน 60% ทดสอบ 40%
- ฝึกสอน 50% ทดสอบ 50%
- ฝึกสอน 40% ทดสอบ 60%
- ฝึกสอน 20% ทดสอบ 80%

โดยการทดลองนี้จะใช้ทั้งหมด 30 epoch สำหรับทั้ง 60 กรณี ใช้ Stochastic Gradient Descent ในการปรับค่าน้ำหนัก อัตราการเรียนรู้พื้นฐาน (Learning rate) เท่ากับ 0.005 นโยบายอัตราการเรียนรู้ (Learning rate policy) เท่ากับ ทุก ๆ 10 epoch โม-เมนตัม (Momentum) เท่ากับ 0.9 ค่าของ Weight decay เท่ากับ 0.0005 ค่าแกรมมา (Gamma) เท่ากับ 0.1 และขนาดแบทช์ (Batch size) 24 (ในกรณีของ GoogLeNet) และ 100 (ในกรณีของ AlexNet) และการทดลองนี้จะใช้ Caffe framework ในการพัฒนา

5) ดังนั้นชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบจะมีภาพทั้งหมด 54,306 ภาพ จำแนกได้เป็น 38 คลาส ตัวอย่างดังรูปที่ 2.34 ในการทดลองนี้ผู้วิจัยทำการปรับขนาดภาพให้เป็น 256 × 256 พิกเซล และในการทดลองนี้จะใช้ชุดข้อมูลที่ต่างกันได้แก่ ภาพสี ภาพโทนสีเทา (Grayscale) และภาพแบ่งส่วนของใบ (Leaf Segmented) ดังแสดงในรูปที่ 2.35



รูปที่ 2.34 ตัวอย่างภาพในชุดข้อมูลทั้ง 38 คลาส



รูปที่ 2.35 ตัวอย่างชุดข้อมูลภาพสี (a, d) ภาพภาพโทนสีเทา (b, e) และ ภาพแบ่งส่วนของใบ (c, f)

โดยผลลัพธ์ของการทดลองแสดงในรูปที่ 2.36 โดยค่าในตารางที่จะแสดงค่าของ F1-score ซึ่งมีรูปแบบดังนี้ $F_1 score_{\{mean\ precision,\ \ mean\ recall,\ \ Overall\ accuracy\}}$ ซึ่งประสิทธิภาพที่ดีที่สุดคือ F1 score of 0.9934 (Overall accuracy of 99.35%)

	AlexNet		GoogleNet	
	Transfer learning	Training from scratch	Transfer learning	Training from scratch
TRAIN: 200%, TEST: 80%				
Color	0.9736 (0.9742, 0.9737, 0.9738)	0.9118 (0.9137, 0.9132, 0.9130)	**0.9820** (0.9824, 0.9821, 0.9821)	0.9430 (0.9440, 0.9431, 0.9429)
Grayscale	0.9361 (0.9368, 0.9369, 0.9371)	0.8524 (0.8539, 0.8555, 0.8553)	**0.9563** (0.9570, 0.9564, 0.9564)	0.8828 (0.8842, 0.8835, 0.8841)
Segmented	0.9724 (0.9727, 0.9727, 0.9726)	0.8945 (0.8956, 0.8963, 0.8969)	**0.9808** (0.9810, 0.9808, 0.9808)	0.9377 (0.9388, 0.9380, 0.9380)
TRAIN: 400%, TEST: 60%				
Color	0.9860 (0.9861, 0.9861, 0.9860)	0.9555 (0.9557, 0.9558, 0.9558)	**0.9914** (0.9914, 0.9914, 0.9914)	0.9729 (0.9731, 0.9729, 0.9729)
Grayscale	0.9584 (0.9588, 0.9589, 0.9588)	0.9088 (0.9090, 0.9101, 0.9100)	**0.9714** (0.9717, 0.9716, 0.9716)	0.9361 (0.9364, 0.9363, 0.9364)
Segmented	0.9812 (0.9814, 0.9813, 0.9813)	0.9404 (0.9408, 0.9408, 0.9408)	**0.9896** (0.9896, 0.9896, 0.9896)	0.9643 (0.9647, 0.9642, 0.9642)
TRAIN: 50%, TEST: 50%				
Color	0.9896 (0.9897, 0.9896, 0.9897)	0.9644 (0.9647, 0.9647, 0.9647)	**0.9916** (0.9916, 0.9916, 0.9916)	0.9772 (0.9774, 0.9773, 0.9773)
Grayscale	0.9661 (0.9663, 0.9663, 0.9663)	0.9312 (0.9315, 0.9318, 0.9319)	**0.9788** (0.9789, 0.9788, 0.9788)	0.9507 (0.9510, 0.9507, 0.9509)
Segmented	0.9867 (0.9868, 0.9868, 0.9869)	0.9551 (0.9552, 0.9555, 0.9556)	**0.9909** (0.9910, 0.9910, 0.9910)	0.9720 (0.9721, 0.9721, 0.9722)
TRAIN: 600%, TEST: 40%				
Color	0.9907 (0.9908, 0.9908, 0.9907)	0.9724 (0.9725, 0.9725, 0.9725)	**0.9924** (0.9924, 0.9924, 0.9924)	0.9824 (0.9825, 0.9824, 0.9824)
Grayscale	0.9686 (0.9689, 0.9688, 0.9688)	0.9388 (0.9396, 0.9395, 0.9391)	**0.9785** (0.9789, 0.9786, 0.9787)	0.9547 (0.9554, 0.9548, 0.9551)
Segmented	0.9855 (0.9856, 0.9856, 0.9858)	0.9595 (0.9597, 0.9597, 0.9596)	**0.9905** (0.9906, 0.9906, 0.9906)	0.9740 (0.9743, 0.9740, 0.9745)
TRAIN: 80%, TEST: 20%				
Color	**0.9927** (0.9928, 0.9927, 0.9928)	**0.9782** (0.9786, 0.9782, 0.9782)	**0.9934** (0.9935, 0.9935, 0.9935)	**0.9836** (0.9839, 0.9837, 0.9837)
Grayscale	**0.9726** (0.9728.0.9727, 0.9725)	**0.9449** (0.9451, 0.9454, 0.9452)	**0.9800** (0.9804, 0.9801, 0.9798)	**0.9621** (0.9624, 0.9621, 0.9621)
Segmented	**0.9891** (0.9893, 0.9891, 0.9892)	**0.9722** (0.9725, 0.9724, 0.9723)	**0.9925** (0.9925, 0.9925, 0.9924)	**0.9824** (0.9827, 0.9824, 0.9822)

Each cell in the table represents the mean F_1 score (mean precision, mean recall, overall accuracy) for the corresponding experimental configuration.
The bold values are the F_1 scores of the best performing models in the respective row/column.

รูปที่ 2.36 ผลลัพธ์ของการทดลองทั้ง 60 กรณี

แต่ทั้งหมดทั้งมวลการทดลองนี้ยังมีข้อจำกัดอยู่ที่ว่าเมื่อทดสอบภาพที่ถ่ายภายใต้เงื่อนไขที่ต่างจากภาพที่ใช้ในการฝึกสอนความถูกต้องจะลดลงอย่างมาก และอีกข้อจำกัดหนึ่งคือภาพใบที่ใช้มีพื้นหลังที่เป็นสีพื้นเดียวกัน

2.12.2 Deep neural networks-based recognition of plant diseases by leaf image classification

บทความวิจัยเรื่อง Deep neural networks based recognition of plant diseases by leaf image classification (Sladojevic, Arsenovic et al. 2016) โดยงานวิจัยนี้มีมีวัตถุประสงค์เพื่อศึกษาแนวทางการพัฒนาโมเดลสำหรับการรู้จำโรคใบพืช โดยการใช้โครงข่ายประสาทแบบคอนโวลูชันนอล ในการจำแนกโรคใบพืช 13 โรคออกจากใบที่ไม่เป็นโรค โดยเริ่มจากการรวบรวมภาพเพื่อสร้างฐานข้อมูลโดยอาศัยการประเมินโดยผู้เชี่ยวชาญด้านการเกษตร สำหรับโมเดลจะใช้ CaffeNet และในการฝึกสอนโมเดลใช้ Caffe framework ในการพัฒนา และได้ผลการทดลองสำหรับโมเดลที่ได้รับการพัฒนามีความแม่นยำระหว่าง 91% ถึง 98% หรือโดยเฉลี่ย 96.3% ซึ่งมีรายละเอียดในงานวิจัยบางส่วนดังนี้

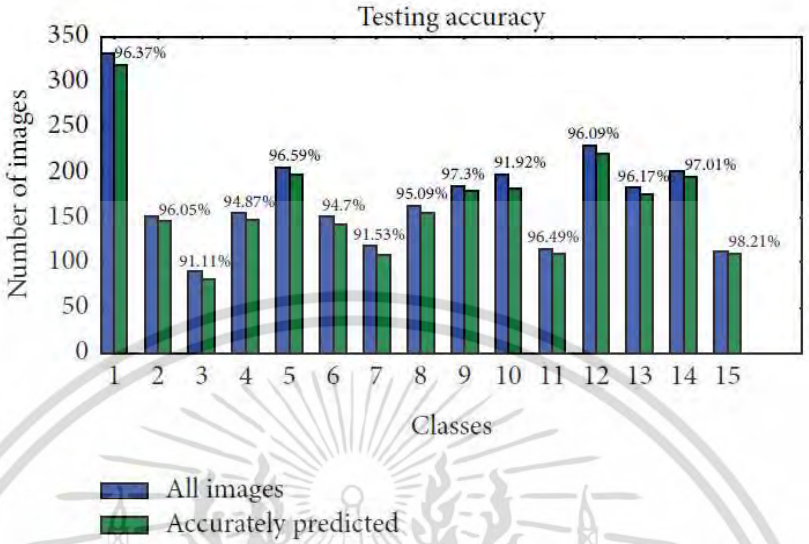
ชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบจะได้จากการค้นหาจากอินเทอร์เน็ตจากแหล่งต่าง ๆ โดยใช้ภาษาต่าง ๆ ในการค้นหา โดยภาพในชุดข้อมูลจะจัดกลุ่มได้เป็น 15 คลาส และ 13 คลาสเป็นโรคพืชที่เห็นได้จากใบ ในขั้นตอนนี้ภาพที่ซ้ำกันจะถูกเอาออกด้วย Python script โดยดูจาก Mata data ได้แก่ ชื่อ ขนาด และวันที่ หลังจากนั้นภาพที่ได้จะได้รับการประเมินโดยผู้เชี่ยวชาญ ขั้นตอนต่อไปคือการเพิ่มจำนวนรูปภาพโดยการประมวลผลภาพ แสดงในรูปที่ 2.37 (ก) การประมวลผลภาพโดย Affine transformation (ข) การประมวลผลภาพโดย Perspective transformation (ค) การประมวลผลภาพโดยการหมุนภาพ (Rotation) สุดท้ายจะได้ชุดข้อมูลฝึกสอนจำนวน 30,880 ภาพ และชุดข้อมูลสำหรับทดสอบ 2,589 ภาพ ซึ่งแสดงในรูปที่ 2.38 นอกจากนี้ภาพที่มีความละเอียดและขนาดน้อยกว่า 500 พิกเซล จะไม่ถูกใช้ และมีการปรับขนาดภาพเป็น 256 × 256 พิกเซล เพื่อลดเวลาในการฝึกสอน สำหรับโมเดลที่ใช้คือ CaffeNet เป็นโมเดลสำหรับการฝึกสอนโดยจะทำการปรับแต่งและไม่ปรับแต่งโมเดล สำหรับโมเดลที่ได้รับการปรับแต่งใน CaffeNet นั้นชั้นของฟังก์ชันกระตุ้น Softmax เดิม CaffeNet จะจำแนกประเภทออกเป็น 1,000 คลาส จะถูกปรับเปลี่ยนให้เป็น 15 คลาสและขั้นตอนนี้ก็ทำในโมเดลที่ไม่ปรับแต่งเช่นกัน และเนื่องจากชุดข้อมูลที่ใช้มีน้อยกว่า เมื่อเปรียบเทียบกับ ImageNet ดังนั้นการกำจัดปัญหา Overfitting จะถูกจัดการด้วยการปรับลดค่าเริ่มต้นของอัตราการเรียนรู้ (Learning rates) สำหรับชั้นซ่อน โดยอัตราการเรียนรู้ของชั้นบนสุดถูกกำหนดไว้ที่ 10 ในขณะที่อัตราการเรียนรู้ของทุกชั้นกำหนดไว้ที่ 0.1 และการปรับค่าน้ำหนักโดยขั้นตอนวิธี Back-propagation สำหรับ 100,000 รอบ สำหรับการปรับค่าพารามิเตอร์ของชั้นซ่อนและ Hyperparameters จะทำการทดลองซ้ำเพื่อหาค่าที่ดีที่สุด

ซึ่งผลการทดสอบได้ความแม่นยำ 96.3% หลังจากฝึกอบรมจากโมเดลที่ได้รับการปรับแต่ง 100 ครั้ง รูปที่ 2.39 แสดงผลการทดลองความถูกต้องของการทำนายสำหรับแต่ละคลาส จะเห็นได้ว่าภาพที่มีจำนวนภาพต่ำกว่าในชุดข้อมูลการฝึกอบรมจะมีความถูกต้องต่ำลงด้วย ในทางตรงกันข้ามโมเดลมีความแม่นยำในการทำนายภาพพื้นหลังได้ดี ซึ่งเป็นประโยชน์อย่างยิ่งในการช่วยให้แยกใบพืชออกจากสภาพแวดล้อมได้

Class	Number of original images	Total number of images: original and augmented	Number of images from the dataset used for validation
(1) Healthy leaf	565	4523	331
(2) Pear, cherry, and peach, porosity	265	2124	152
(3) Peach, powdery mildew	108	1296	90
(4) Peach, *Taphrina deformans*	152	1552	156
(5) Apple, pear, *Erwinia amylovora*	232	2368	205
(6) Apple, pear, *Venturia*	183	2200	151
(7) Apple, powdery mildew	120	1440	118
(8) Apple, Rust	163	1960	163
(9) Pair, *Gymnosporangium sabinae*	267	2142	185
(10) Pair, gray leaf spot	122	1464	198
(11) Grapevine, wilt	287	2300	114
(12) Grapevine, mites	250	2000	230
(13) Grapevine, powdery mildew	237	1900	183
(14) Grapevine, downy mildew	297	2376	201
(15) Background images	1235	1235	112
	4483	30880	2589

รูปที่ 2.37 ชุดข้อมูลของใบพืช

รูปที่ 2.38 ภาพที่ผ่านการประมวลผลภาพ



รูปที่ 2.39 ผลการทดลองความถูกต้องของการทำนายสำหรับแต่ละคลาส

สำหรับวิจัยนี้จะเห็นได้ว่าเทคนิคการใช้การประมวลผลภาพในการเพิ่มจำนวนของภาพถือเป็นอีกหนึ่งเทคนิคที่น่าสนใจในการเพิ่มจำนวนของภาพ

2.12.3 Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition

บทความวิจัยเรื่อง Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition (Pawara, Okafor et al. 2017) ทำการใช้การเรียนรู้ของเครื่องและคอมพิวเตอร์วิชัน เพื่อทำการรู้จำภาพพืชชนิดต่าง ๆ โดยมีวัตถุประสงค์เพื่อทำการเปรียบเทียบ ตัวอธิบายคุณลักษณะของภาพแบบโลคอล (Local feature descriptor) และ Bag of visual word กับโครงข่ายประสาทแบบคอนโวลูชัน โดยใช้ชุดข้อมูลฝึกสอน 3 ชุดข้อมูลได้แก่ ชุดข้อมูล AgrilPlant ชุดข้อมูล LeafSnap และ ชุดข้อมูล Folio และใช้สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอล GoogLeNet และ AlexNet ทั้งแบบปรับแต่ง (Fine-tune) และไม่ปรับแต่ง (Training from Scratch) และทำการเปรียบเทียบกับ ตัวอธิบายคุณลักษณะของภาพแบบโคคอลที่ใช้ k-nearest neighbors และ Bag of visual words กับ Histogram of oriented gradients ร่วมกับ Support vector machines (SVM) และ Multi-layer perceptrons ซึ่งผลลัพธ์ที่ได้ วิธีที่ใช้สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลที่ปรับแต่งของ GoogLeNet

ให้ผลลัพธ์ที่ดีที่สุดที่มีความแม่นยำถึงร้อยละ 98.33 และ 97.66 ในชุดข้อมูล AgrilPlant และ LeafSnap ตามลำดับ สำหรับชุดข้อมูล Folio สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลที่ปรับแต่งของ AlexNet ให้ผลลัพธ์ที่มีความแม่นยำถึงร้อยละประมาณ 97.6 ซึ่งมีรายละเอียดการทดลองมี ดังนี้

1) โครงสร้างสถาปัตยกรรมของโครงข่ายประสาทแบบคอนโวลูชันนอล

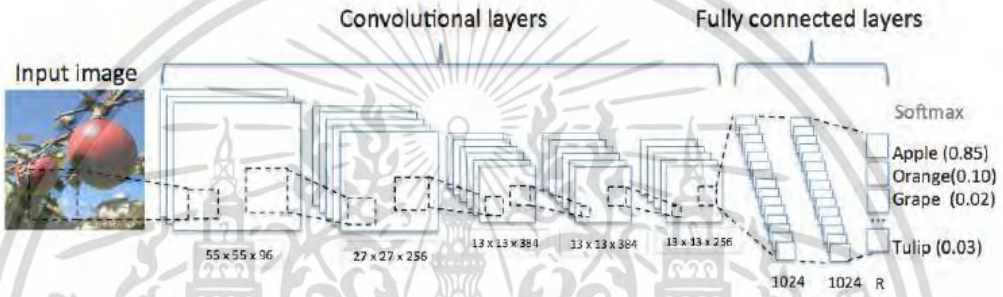
1.1) สถาปัตยกรรมของ AlexNet

AlexNet แบบเดิมประกอบไปด้วย 8 ชั้นของค่าน้ำหนักโดยมี 5 ชั้นของ คอโวลูชันนอล และ 3 ชั้นของ Fully connected โดยสองชั้นแรก (conv(Alsayed, Alsabei et al. 2021)) จะทำการ Normalization และ Max pooling ชั้นที่ 6 และ 7 จะเป็นชั้นของ Fully connected (fc(Alsayed, Alsabei et al. 2021)) ซึ่งประกอบไปด้วย 4,096 นิวรอน และชั้นสุดท้าย (fc8) จะประกอบไปด้วย 1,000 นิวรอนเพราะว่า ชุดข้อมูลของ ImageNet มี 1,000 คลาส และมีการใช้ฟังก์ชันกระตุ้น ReLU ฟังก์ชันใน 7 ชั้นแรก อัตราการดรอปเอาท์ (Droupout ratio) เท่ากับ 0.5 ในชั้นที่ 6 และ 7 ผลลัพธ์จากชั้นสุดท้าย fc8 จะถูกป้อนเข้าไปที่ softmax ฟังก์ชัน

ในการทดลองนี้มีการปรับแต่ง AlexNet โดยลดจำนวนนิวรอนจาก 4,096 นิวรอน เป็น 256 512 และ 1,024 นิวรอน ตามลำดับในชั้น fc6 และ fc7 เพื่อเพิ่มประสิทธิภาพของการคำนวณ และลดความเสี่ยงจาก Overfitting ซึ่งได้มีการทดลองกับชุดข้อมูล AgriPlant (ตารางที่ 2.3) โดยได้จำนวนนิวรอนที่เหมาะสมคือ 1,024 นิวรอนที่ให้ค่าความถูกต้อง (Accuracy) มากที่สุด และได้ลดเวลาของการกระบวนการฝึกสอนลงร้อยละ 34 เมื่อเทียบกับ 4,096 นิวรอน ดังนั้นในการทดลองนี้ทุกชุดข้อมูลจะใช้จำนวนนิวรอนในชั้น fc6 และ fc7 เท่ากับ 1,024 ในสถาปัตยกรรมของ AlexNet ดังแสดงในรูปที่ 2.40

ตารางที่ 2.1 ผลการทดลองการหาจำนวนนิวรอนที่เหมาะสมและเวลาที่ลดลงเมื่อเทียบกับ 1,024 ในสถาปัตยกรรมของ AlexNet

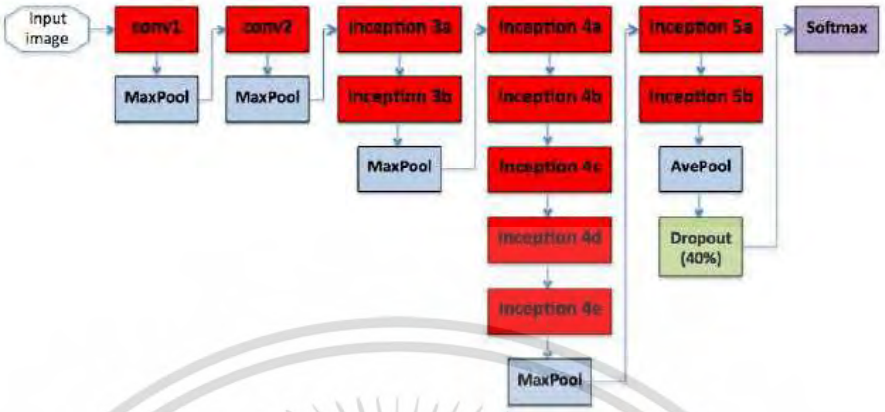
จำนวนของนิวรอน	ค่าความถูกต้อง	ร้อยละของเวลาที่ลดลง
4,096	-	-
1,024	**88.30 ± 1.34**	34.06
512	89.53 ± 0.61	39.09
256	88.90 ± 1.35	41.08



รูปที่ 2.40 สถาปัตยกรรมของ AlexNet ที่ใช้ในการทดลอง

1.2) สถาปัตยกรรมของ GoogLeNet

การทดลองนี้ได้ใช้โมเดลต้นแบบของ GoogLeNet ที่ได้ถูกนำเสนอโดย Szegdy และคณะ ในปี ค.ศ.2015 ซึ่งได้นำเสนอสถาปัตยกรรมของ Inception โมดูล ที่สามารถลดพารามิเตอร์ในการฝึกสอนได้จำนวนมาก โดย Inception โมดูล จะใช้การรวมกันแบบขนานของ 1x1 3x3 และ 5x5 คอนโวลูชัน พร้อมกับชั้น Pooling นอกจากนี้ตัวกรองคอนโวลูชันนอล 1x1 จะถูกเพิ่มในโครงข่ายก่อน 3x3 และ 5x5 คอนโวลูชั้น สำหรับลดขนาดมิติ ซึ่งสถาปัตยกรรมของ GoogLeNet จะประกอบไปด้วย 22 ชั้น พร้อมกับ Max Pooling 4 ชั้น Average Pooling 1 ชั้น และใช้ และใช้ ReLU ในชั้นคอโวลูชันนอล ทุกชั้น รวมถึงข้างใน Inception โมดูลด้วย เพื่อจัดการกับปัญหา Vanishing gradients ในโครงข่าย นอกจากนี้มีการเพิ่มตัวช่วยในการจำแนกประเภท (Auxiliary classifiers) 2 ตัว เข้าไปตรงกลางระหว่างกระบวนการฝึกสอน และอัตราการดรอปเอาท์เท่ากับ 0.4 ดังรูปที่ 2.41 แสดงชั้นคอนโวลูชันและ Inception โมดูลที่ถูกออกแบบไว้ใน GoogLeNet



รูปที่ 2.41 ชั้นคอนโวลูชันและ Inception โมดูลที่ถูกออกแบบไว้ใน GoogLeNet

2) ตัวอธิบายโลคอล (Local descriptor) แบบดั้งเดิม

2.1) Histogram of oriented gradients

ตัวสกัดคุณลักษณะ Histogram of oriented gradients (HOG) เป็นขั้นตอนวิธีที่การแสดงวัตถุที่สนใจโดยการนับ Occurrences ของการไล่ระดับสี ความเข้มและทิศทางของการไล่ระดับสี และแบ่งส่วนของภาพ ตัวอธิบาย HOG (HOG descriptor) จะจำเวกเตอร์คุณลักษณะ (Feature vectors) โดยมีขั้นตอนดังนี้

1. แบ่งภาพออกเป็นออกเป็นบล็อกเล็ก ๆ
2. คำนวณการไล่ระดับสีในแนวนอน H_x และแนวตั้ง H_y ของพิกเซล โดยใช้เคอเนล (Kernel) [-1, 0, 1] เป็นตัวตรวจจับการไล่ระดับสี
3. คำนวณ ขนาด M และทิศทาง θ ของการไล่ระดับสี

$$M_{(x,y)} = \sqrt{H_x^2 + H_y^2} \quad (1)$$

$$\theta = arctan\frac{H_x}{H_y} \quad (2)$$

4. สร้างฮิสโทแกรม (Histogram) โดยใช้น้ำหนักของการไล่ระดับทิศทางของแต่ละเซลล์ใส่ไปใน Orientation bin เฉพาะ
5. ใช้ L2 normalization กับ Orientation bin เพื่อลดความแปรปรวนของความสว่าง และจะได้เวกเตอร์คุณลักษณะ

ในการทดลองนี้ใช้ขนาดของบล็อก 5x5 และ 8 Orientation bin ซึ่งจะให้ เวกเตอร์คุณลักษณะ 200 มิติ จากนั้นนำเวกเตอร์คุณลักษณะเข้าตัวจำแนกประเภท KNN

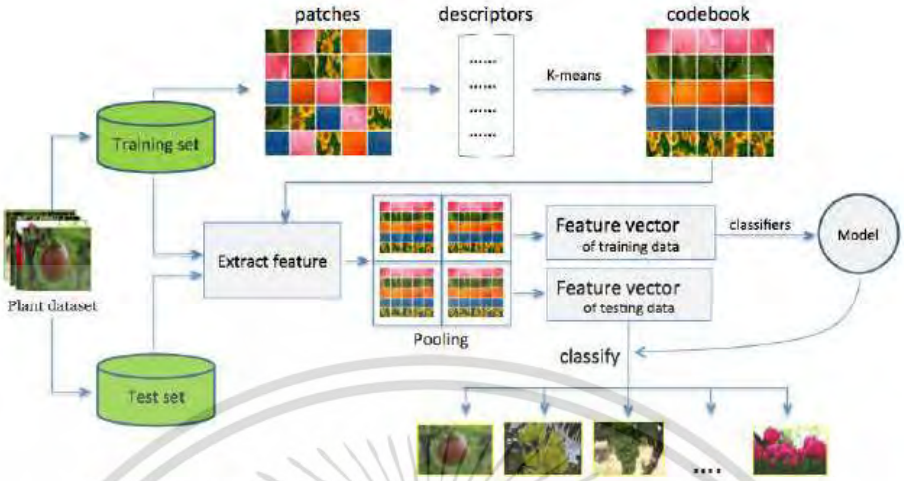
2.2) Bag of visual words กับ Histogram of oriented gradients

โมเดล Bag of visual words (BOW) เป็นแนวคิดของคอมพิวเตอร์-วิชันในการอธิบายภาพที่สามารถหาได้จากการจัดกลุ่มของคุณลักษณะของพื้นที่ในภาพที่ประกอบด้วยข้อมูลของภาพจำนวนมาก เช่น สีหรือพื้นผิว

ในการทดลองนี้ได้ทำการรวม BOW กับ HOG เรียกว่า BOW- HOG ซึ่งมีวิธีการสร้างเวกเตอร์คุณลักษณะดังนี้

1) คำนวณเซตของแพทช์ (Patches) P = {p1, p2, ..., pn} โดยที่ n คือจำนวนของแพทช์ ขนาดของแพทช์จะมีขนากเป็น w x w พิกเซล สำหรับแต่ละแพทช์สามารถคำนวณได้จาก ตัวอธิบายโลคอล (Local descriptors) และใช้เป็นข้อมูลเข้าเพื่อสร้าง Codebook
2) Codebook C ใช้ขั้นตอนวิธีการจัดกลุ่ม K-means clustering เพื่อสกัดเวกเตอร์คุณลักษณะ ซึ่งจำนวนของเวกเตอร์คุณลักษณะจะขึ้นอยู่กับจำนวนของ Centroids
3) สร้างคุณลักษณะ BOW โดยการตรวจจับ Occurrences ของภาพในแต่ละกลุ่ม ภาพแต่ละภาพจะแบ่งออกเป็น 4 Quadrants และทำการคำนวณ Feature activation โดยใช้ Sum-pooling

ในการทดลองนี้ตัวอธิบาย HOG จะถูกใช้เป็นตัวอธิบายโมเดล จำนวนของแพทช์เท่ากับ 400,000 ขนาดของแต่ละแพทช์เท่ากับ 15x15 พิกเซลและจำนวนของ Centroids เท่ากับ 600 เนื่องจากภาพถูกแบ่งออกเป็น 4 Quadrants ดังนั้น HOG-BOW จะสร้างเวกเตอร์คุณลักษณะ 2,400 มิติ จากนั้นนำข้อมูลป้อนเข้าสู่ตัวจำแนกประเภท L2-SVM และ Multi-Layer Perceptorn (MLP) ซึ่ง HOG-BOW ที่ใช้ในการทดลองนี้แสดงในรูปที่ 2.42



รูปที่ 2.42 ขั้นตอนการทำงานของ HOG-BOW

3) การทดลอง

3.1) ชุดข้อมูลพืช

โดยงานวิจัยนี้ได้ใช้โดยใช้ชุดข้อมูลฝึกสอน 3 ชุดข้อมูลได้แก่ ชุดข้อมูล AgrilPlant ชุดข้อมูล LeafSnap และ ชุดข้อมูล Folio AgrilPlant แต่ละชุดข้อมูลประกอบไปด้วยดังนี้

- ชุดข้อมูล AgrilPlant

  ชุดข้อมูล AgrilPlant ประกอบด้วยภาพจำนวน 3,000 ภาพ จาก www.flicker.com ประกอบด้วย 10 คลาส มีพืชดังนี้ แอปเปิล กล้วย องุ่น ขนุน ส้ม มะละกอ พลับ สับปะรด ทานตะวัน และทิวลิป แต่ละคลาสมีทั้งหมด 300 ภาพ ตัวอย่างภาพในชุดข้อมูล AgrilPlant ดังรูปที่ 2.43 ซึ่งความท้าทายคือ ชุดข้อมูล 1. ภาพมีความคล้ายคลึงกัน 2. ความหลากหลายของพืชชนิดเดียวกันเช่น แอปเปิลสีเขียวและสีแดง 3. ความซับซ้อนของพื้นหลัง

- ชุดข้อมูล LeafSnap

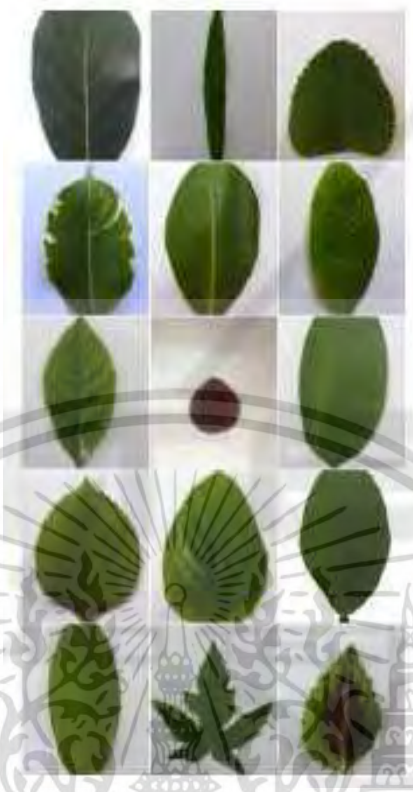
  ชุดข้อมูล LeafSnap ประกอบด้วยภาพใบพืช 7,719 ภาพ และ 184 สายพันธุ์ภาพทั้งหมดเป็นภาพถ่าย ในสภาพแวดล้อมกลางแจ้งด้วยโทรศัพท์มือถือและอาจมี Noise ไม่ชัด และเงา จำนวนภาพในแต่ละคลาสไม่เท่ากันมีตั้งแต่ 10 ถึง 183 ตัวอย่างชุดข้อมูลดังรูปที่ 2.44

- ชุดข้อมูล Folio

ชุดข้อมูล Folio ประกอบด้วยภาพใบพืช 3 ชนิดแต่ละชนิดมีประมาณ 20 ภาพ รูปภาพทั้งหมดถูกถ่ายภาพใต้แสงอาทิตย์บนพื้นหลังสีขาว ตัวอย่างชุดข้อมูลดังรูปที่ 2.45

รูปที่ 2.43 ตัวอย่างชุดข้อมูล AgrilPlant

รูปที่ 2.44 ตัวอย่างชุดข้อมูล LeafSnap



รูปที่ 2.45 ตัวอย่างชุดข้อมูล Folio

3.2) การทดลอง

ในการทดลองจะทำการใช้อัตราส่วนของชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบ 80:20 และ 5 ครั้งของการทำ Cross validation ความละเอียดของภาพจะถูกปรับเป็น 256x256 พิกเซล พารามิเตอร์ส่วนใหญ่ในสถาปัตยกรรมของโครงข่ายประสาทแบบคอนโวลูชันนอลสำหรับ AlexNet และ GoogLeNet ได้กำหนดไว้เหมือนกันทั้งที่แบบปรับแต่งและไม่ปรับแต่ง ยกเว้นค่า Max iteration และ Step size ซึ่งแสดงในตารางที่ 2.4

สำหรับ HOG กับตัวจำแนกประเภท KNN และ HOG-BOW กับตัวจำแนกประเภท MLP และ SVM จะกำหนดค่า k ที่เหมาะสมที่สุดสำหรับตัวจำแนกประเภท KNN ในช่วงของ k = 3, 5, 7, 9 ในชุดข้อมูลแต่ละชุดจะใช้การค้นหาแบบกริด (Grid search) และกำหนดพารามิเตอร์ C สำหรับการแบ่งกลุ่มของ SVM ในช่วงของ $C = 2^1, 2^2, \ldots, 2^8$ และเลือกพารามิเตอร์ C ที่ดีที่สุด โดยการทดสอบในการทำ Cross validation 5

ครั้ง และสำหรับ MLP ใช้ Scaled conjugate gradient เป็นขั้นตอนวิธีในการฝึกสอน จำนวนของนิวรอนเท่ากับ 512 และมีอัตราการเรียนรู้เท่ากับ 0.001 ตามลำดับซึ่งเป็นค่าที่ให้ผลดีที่สุดในการทดลองเบื้องต้น

ตารางที่ 2.2 การกำหนดพารามิเตอร์ในสถาปัตยกรรมของโครงข่ายประสาทแบบคอนโวลูชันนอลสำหรับ AlexNet และ GoogLeNet ใน 3 ชุดข้อมูล

พารามิเตอร์	AgrilPlant	LeafSnap	Folio
Learning rate	0.001	0.001	0.001
Weight decay	0.0005	0.0005	0.0005
Train batch size	20	20	20
Validation batch size	10	10	10
Max iteration (scratch)	50000	50000	50000
Step size (scratch)	25000	25000	25000
Max iteration (fine-tuned)	20000	20000	20000
Step size (fine-tuned)	10000	10000	10000
Test iterations of solver	30	77	6
Test iterations evaluation	60	154	12

4) ผลการทดลอง และการอภิปรายผลการทดลอง

ซึ่งผลการทดลองในตารางที่ 2.5 แสดงความถูกต้องของแต่ละเทคนิคและแต่ละชุดข้อมูล

ตารางที่ 2.3 ผลการทดลองในการเปรียบเทียบความถูกต้องของแต่ละเทคนิค

วิธีการ	AgrilPlant	LeafSnap	Folio
HOG with KNN	38.13 ± 0.53	58.51 ± 2.47	84.30 ± 1.62
HOG-BOW with MLP	74.63 ± 2.16	79.27±3.36	92.37±1.78
HOG-BOW with SVM	79.43±1.68	72.63±0.38	92.78±2.17
AlexNet scratch	89.53±0.61	76.67±0.56	84.83±2.85
AlexNet fine-tuned	96.37±0.83	89.51±0.75	**97.67±1.60**
GoogLeNet scratch	93.33±1.24	89.62±0.50	89.75±1.74
GoogLeNet fine-tuned	**98.33±0.51**	**97.66±0.34**	**97.63±1.84**

5) สรุปการทดลอง

จากผลการทดลองเห็นได้ชัดว่าสถาปัตยกรรมของโครงข่ายประสาทแบบคอนโวลูชันนอลที่ได้รับการปรับแต่งให้ประสิทธิภาพในการจำแนกที่สูงกว่าเทคนิคตัวอธิบายคุณลักษณะแบบดั้งเดิม ซึ่ง GoogLeNet ที่ปรับแต่ง ให้ผลลัพธ์ที่ดีที่สุด โดยมีความถูกต้องถึงร้อยละ 98.33 และ 97.66 ในชุดข้อมูล AgrilPlant และ LeafSnap ตามลำดับ ส่วนในชุดข้อมูล Folio ซึ่งข้อมูลมีขนาดค่อนข้างเล็ก AlexNet ที่ปรับแต่งจึงให้ผลลัพธ์ที่ดีที่สุดประมาณร้อยละ 97.6

ดังนั้นสำหรับงานวิจัยนี้ได้พิสูจน์ให้เห็นว่าโมเดลที่ใช้โครงข่ายประสาทแบบคอนโวลูชันจำแนกข้อมูลภาพชุดข้อมูลที่เกี่ยวกับพืชได้ดีกว่า โมเดลแบบดั้งเดิมอีกทั้งโมเดลแบบดั้งเดิมยังมีการออกแบบและการพัฒนาที่มีขั้นตอนที่มากกว่าด้วย

2.12.4 Deep learning models for plant disease detection and diagnosis

บทความวิจัยเรื่อง Deep learning models for plant disease detection and diagnosis (Ferentinos 2018) ได้นำโครงข่ายประสาทแบบคอนโวลูชันนอลเพื่อตรวจจับและวินิจฉัยโรคพืช โดยใช้ภาพใบพืชของพืชที่เป็นโรคและไม่เป็นโรค จากฐานข้อมูลแบบเปิด ซึ่งประกอบไปด้วยชุดข้อมูลภาพจำนวน 87,848 ภาพซึ่งมี 25 ชนิดพืช จำแนกได้เป็น 58 คลาสของ [พืช, โรค] รวมถึงคลาสใบพืชที่ไม่เป็นโรค ซึ่งสถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลที่ได้ความถูกต้องสูงสุดถึงร้อยละ 99.53 ซึ่งมีรายละเอียดของงานวิจัยดังนี้

1) โมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล

งานวิจัยนี้ได้ทำการใช้สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลพื้นฐาน ได้แก่ 1. AlexNet 2. AlexNetOWTBn 3. GoogLeNet 4. Overfeat และ 5. VGG โมเดลเหล่านี้ใช้ Torch71 machine learning computational framework ในการการสร้างโมเดล ซึ่งใช้ภาษาโปรแกรม LuaJIT2 ขั้นตอนวิธีการฝึกสอนจะทำบน GPU ของการ์ด NVIDIA® GTX1080 โดยใช้ CUDA® (แพลทฟอร์มโปรแกรมแบบขนาน) บนระบบปฏิบัติการ Ubuntu 16.04 LTS

2) ชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบ

ชุดข้อมูลประกอบไปด้วยภาพถ่ายใบพืชจำนวน 87,848 ภาพ มีทั้งภาพใบที่เป็นโรคและไม่เป็นโรค ซึ่งจะแบ่งได้เป็น 58 คลาสของคู่พืชและโรคพืช [พืช,โรค] ดังรูปที่ 2.46 และ รูปที่ 2.47 แสดงคู่ของพืชและโรค จำนวนขอภาพในแต่ละคลาส และ

ร้อยละของภาพที่ถ่ายจากห้องทดลองหรือถ่ายจาสภาพแวดล้อมจริง ดังรูปที่ 2.48 ซึ่งภาพถ่ายที่ถ่ายจากสภาพแวดล้อมจริงจะเห็นว่ามีความซับซ้อนของพื้นหลัง และวัตถุต่าง ๆ ที่ไม่เกี่ยวข้องกับพืช (เช่น รองเท้า) พื้นผิว แสงเงา และปัจจัยอื่น ๆ ที่ต่างกัน

การแบ่งชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบจะแบ่งออกเป็น ชุดข้อมูลฝึกสอนร้อยละ 80 และชุดข้อมูลทดสอบร้อยละ 20 ซึ่งเป็นอัตราส่วนที่นิยมใช้กัน ดังนั้นจะได้ชุดข้อมูลฝึกสอนสำหรับโครงข่ายประสาทแบบคอนโวลูชันนอลจำนาน 70,300 ภาพ นั่นแสดงว่าส่วนที่เหลือ 7,548 ภาพจะเป็นชุดข้อมูลทดสอบ และใช้ Pseudorandom ใน Python ที่พัฒนาขึ้นเพื่อทำการสุ่มเลือกภาพอย่างสม่ำเสมอ ดังนั้นร้อยละของภาพที่ถ่ายจากห้องปฏิบัติการ และภาพที่ถ่ายจากสภาพแวดล้อมจริง จะได้รับการสุ่มอย่างเท่าเทียมกัน นอกจากนี้ภาพของชุดข้อมูลทดสอบและชุดข้อมูลฝึกสอนยังได้รับการประมวลผลภาพเบื้องต้นก่อน เพื่อทำการลดขนาด ตัดภาพออกมาขนาด 256 × 256 พิกเซล ในขณะที่อัตราส่วนของจำนวนชุดข้อมูลทดสอบและชุดข้อมูลฝึกสอนยังเท่ากับ 80/20 เท่าเดิม ส่วนการใช้การประมวลผลภาพเช่น การปรับภาพให้เป็นโทนสีเทา (Grayscale) หรือการแบ่งส่วนภาพ ๖ (Segmentation) ไม่จำเป็นที่จะต้องทำเพราะว่าการเรียนรู้แบบเชิงลึกในโครงข่ายประสาทแบบคอนโวลูชันนั้นมีความสามารถในการระบุคุณลักษณะที่สำคัญและไม่สำคัญของชุดรูปภาพได้ เนื่องจากชุดข้อมูลมีชุดข้อมูลภาพที่ถ่ายจากสภาพแวดล้อมจริง 12 คลาส ผู้วิจัยจึงได้ทำการทดลอง 12 คลาส สองครั้งดังนี้ 1. ให้โมเดลเรียนรู้จากชุดข้อมูลฝึกสอนที่ได้มาจากห้องปฏิบัติการอย่างเดียวและทดสอบด้วยชุดข้อมูลภาพที่ถ่ายจากสภาพแวดล้อมจริง 2. ให้โมเดลเรียนรู้จากชุดข้อมูลภาพที่ถ่ายจากสภาพแวดล้อมจริงและทดสอบด้วยชุดข้อมูลที่ได้มาจากห้องปฏิบัติการ ทั้งสองการทดสอบนี้ในการแบ่งอัตราส่วนของชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบเป็นแนวคิดที่ไม่ดีนัก เพราะว่าทั้ง 12 คลาส มีร้อยละของภาพที่ถ่ายในห้องปฏิบัติการเท่ากับ 55.8 และภาพที่ถ่ายจากสภาพแวดล้อมจริงร้อยละ 44.2 ทำให้สัดส่วนของชุดข้อมูลฝึกสอนกับชุดข้อมูลทดสอบอยู่ในระดับที่ต่ำกว่า 80/20 ที่ใช้ในการพัฒนาโมเดลพื้นฐาน ในกรณีที่ชุดข้อมูลฝึกสอนน้อยกว่าชุดข้อมูลทดสอบซึ่งโดยทั่วไปแล้วไม่สามารถยอมรับได้ แต่อย่างไรก็ตามผู้วิจัยได้แสงผลลัพธ์ของการทดสอบ

Class	Plant common name	Plant scientific name	Disease common name	Disease scientific name	Images (number)	Laboratory conditions (%)	Field conditions (%)
c_0	Apple	Malus domestica	–	–	1835	89.7	10.3
c_1	Apple	Malus domestica	Apple scab	Venturia inaequalis	630	100.0	0.0
c_2	Apple	Malus domestica	Cedar apple rust	Gymnosporangium juniperi-virginianae	276	100.0	0.0
c_3	Apple	Malus domestica	Black rot	Botryosphaeria obtusa	712	87.2	12.8
c_4	Banana	Musa paradisiaca	–	–	1643	0.0	100.0
c_5	Banana	Musa paradisiaca	Black sigatoka	Mycosphaerella fijensis	240	0.0	100.0
c_6	Banana	Musa paradisiaca	Banana speckle	Mycosphaerella musae	3284	0.0	100.0
c_7	Blueberry	Vaccinium spp.	–	–	1735	86.7	13.3
c_8	Cabbage	Brassica oleracea	–	–	420	0.0	100.0
c_9	Cabbage	Brassica oleracea	Black rot	Xanthomonas campestris	64	0.0	100.0
c_10	Cantaloupe	Cucumis melo	–	–	1055	0.0	100.0
c_11	Cassava (manioc)	Manihot esculenta	Brown leaf spot	Cercosporidium henningsii	43	100.0	0.0
c_12	Cassava (manioc)	Manihot esculenta	Cassava green spider mite	Mononychellus tanajoa & progresivus	892	100.0	0.0
c_13	Celery	Apium graveolens	Early blight, Cercospora	Cercospora apii	1204	0.0	100.0
c_14	Cherry (& sour)	Prunus spp.	–	–	854	100.0	0.0
c_15	Cherry (& sour)	Prunus spp.	Powdery mildew	Podosphaera spp.	1052	100.0	0.0
c_16	Corn (maize)	Zea mays	–	–	4450	26.1	73.9
c_17	Corn (maize)	Zea mays	Cercospora leaf spot	Cercospora zeae-maydis	1457	35.2	64.8
c_18	Corn (maize)	Zea mays	Common rust	Puccinia sorghi	1614	73.9	26.1
c_19	Corn (maize)	Zea mays	Northern Leaf Blight	Exserohilum turcicum	985	100.0	0.0
c_20	Cucumber	Cucumis sativus	–	–	267	0.0	100.0
c_21	Cucumber	Cucumis sativus	Downy mildew	Pseudoperonospora cubensis	1318	0.0	100.0
c_22	Eggplant	Solanum melongena	–	–	515	0.0	100.0
c_23	Gourd	Langenaria spp.	Downy mildew	Pseudoperonospora cubensis	114	0.0	100.0
c_24	Grape	Vitis vinifera	–	–	613	69.0	31.0
c_25	Grape	Vitis vinifera	Black rot	Guignardia bidwellii	1180	100.0	0.0
c_26	Grape	Vitis vinifera	Esca (Black measles)	Phaeomoniella chlamydospora	1384	100.0	0.0
c_27	Grape	Vitis vinifera	Leaf blight	Pseudocercospora vitis	1076	100.0	0.0
c_28	Onion	Allium cepa	–	–	154	0.0	100.0
c_29	Orange	Citrus sinensis	Huanglongbing	Candidatus Liberibacter	5507	100.0	0.0

รูปที่ 2.46 ข้อมูลของชุดข้อมูลภาพ

Class	Plant common name	Plant scientific name	Disease common name	Disease scientific name	Images (number)	Laboratory conditions (%)	Field conditions (%)
c_30	Peach	Prunus persica	–	–	360	100.0	0.0
c_31	Peach	Prunus persica	Bacterial sport	Xanthomonas campestris	2297	100.0	0.0
c_32	Pepper, bell	Capsicum annuum	–	–	2029	72.8	27.2
c_33	Pepper, bell	Capsicum annuum	Bacterial spot	Xanthomonas campestris	997	100.0	0.0
c_34	Potato	Solanum tuberosum	–	–	152	100.0	0.0
c_35	Potato	Solanum tuberosum	Late blight	Phytophthora infestans	1000	100.0	0.0
c_36	Potato	Solanum tuberosum	Early blight	Alternaria solani	3167	31.6	68.4
c_37	Pumpkin	Cucurbita spp.	Cucumber mosaic	Cucumber mosaic virus (CMV)	2387	0.0	100.0
c_38	Raspberry	Rubus spp.	–	–	371	100.0	0.0
c_39	Soybean	Glycine max	–	–	6235	81.6	18.4
c_40	Soybean	Glycine max	Downy mildew	Peronospora manshurica	851	0.0	100.0
c_41	Soybean	Glycine max	Frogeye leaf spot	Cercospora sojina	2023	0.0	100.0
c_42	Soybean	Glycine max	Septoria Leaf Blight	Septoria glycines	3565	0.0	100.0
c_43	Squash	Cucurbita spp.	–	–	264	0.0	100.0
c_44	Squash	Cucurbita spp.	Powdery mildew	Erysiphe cichoracearum, Sphaerotheca fuliginea	1835	100.0	0.0
c_45	Strawberry	Fragaria spp.	–	–	456	100.0	0.0
c_46	Strawberry	Fragaria spp.	Leaf scorch	Diplocarpon earlianum	3396	29.7	70.3
c_47	Tomato	Lycopersicum esculentum	–	–	1592	100.0	0.0
c_48	Tomato	Lycopersicum esculentum	Bacterial spot	Xanthomonas campestris pv. Vesicatoria	2127	100.0	0.0
c_49	Tomato	Lycopersicum esculentum	Early blight	Alternaria solani	2579	38.8	61.2
c_50	Tomato	Lycopersicum esculentum	Late blight	Phytophthora infestans	1910	100.0	0.0
c_51	Tomato	Lycopersicum esculentum	Septoria leaf spot	Septoria lycopersici	1771	100.0	0.0
c_52	Tomato	Lycopersicum esculentum	Spider mites	Tetranychus urticae	1653	100.0	0.0
c_53	Tomato	Lycopersicum esculentum	Tomato mosaic virus	Tomato mosaic virus (ToMV)	373	100.0	0.0
c_54	Tomato	Lycopersicum esculentum	Leaf Mold	Fulvia fulva	952	100.0	0.0
c_55	Tomato	Lycopersicum esculentum	Target spot	Corynespora cassiicola	1404	100.0	0.0
c_56	Tomato	Lycopersicum esculentum	TYLCV	Begomovirus (Fam. Geminiviridae)	5357	100.0	0.0
c_57	Watermelon	Citrullus lanatus	–	–	172	0.0	100.0
TOTAL:					***87,848***	***62.7***	***37.3***

รูปที่ 2.47 ข้อมูลของชุดข้อมูลภาพ (ต่อ)

รูปที่ 2.48 ตัวอย่างภาพที่ถ่ายจากสภาพแวดล้อมจริง

3) การทดลองและผลลัพธ์

3.1) โมเดลการเรียนรู้แบบเชิงลึกที่ดีที่สุด

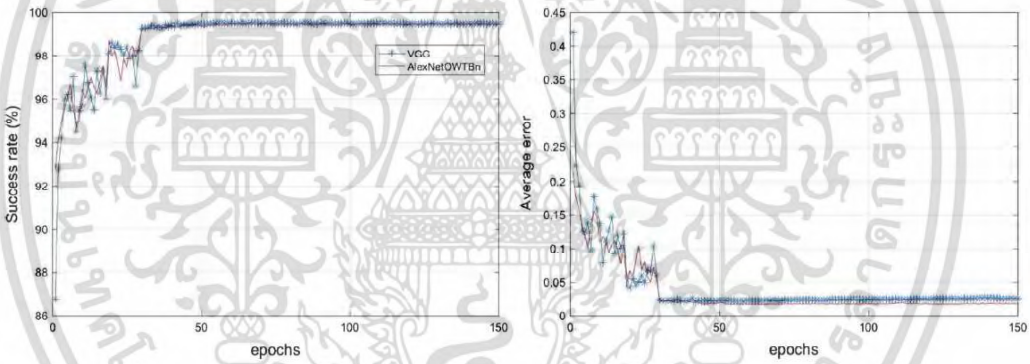
โมเดลโครงข่ายประสาทแบบคอนโวลูชันนอลทั้งหมดในข้อ 2) ได้ถูกกำหนดค่าของพารามิเตอร์ตามรูปที่ 2.49 หลังจากที่ได้ทดลองหาค่าที่เหมาะสมแล้ว โดยค่าของอัตราการเรียนรู้ Learning rate เริ่มตั้งแต่ 0.01 และลดลงทุก ๆ 20 Epoch โดยลดลง 1/2 หรือ 1/5 สลับกันไปจนถึง 0.0001 การเปรียบเทียบประสิทธิภาพของโมเดลขึ้นอยู่กับในชุดข้อมูลทดสอบ (ทุกโมเดลได้รับความแม่นยำร้อยละ 100 ในชุดข้อมูลฝึกสอน) รูปที่ 2.50 แสดงเปอร์เซ็นต์ ของความถูกต้องในการจำแนกประเภทในระหว่างการทดสอบของโมเดลต่าง ๆ โดยใช้ภาพต้นฉบับและภาพที่มีการปรับลดขนาด

Parameter	Value
Batches/epoch	10,000
Batch size	32
Momentum	0.9
Weight decay	0.0005
Learning rate	0.01–0.0001

รูปที่ 2.49 พารามิเตอร์ของโครงข่ายประสาทแบบคอนโวลูชันนอล

	Original images				Pre-processed images			
Model	Success rate	Average error	Epoch	Time (s/epoch)	Success rate	Average error	Epoch	Time (s/epoch)
AlexNet	99.06%	0.0354	47	7034	98.64%	0.0658	50	1022
AlexNetOWTBn	99.44%	0.0192	46	7520	99.07%	0.0332	45	1125
GoogLeNet	97.27%	0.0957	45	7845	97.06%	0.0984	40	2670
Overfeat	98.96%	0.0412	45	6204	98.26%	0.0848	49	1570
VGG	99.48%	0.0223	48	7294	98.87%	0.0542	49	4208

รูปที่ 2.50 ผลการทดลองของโมเดลโครงข่ายประสาทแบบคอนโวลูชันนอล



รูปที่ 2.51 ประสิทธิภาพในชุดข้อมูลทดสอบของโมเดล VGG และ AlexNetOWTBn

ผลการวิจัยในรูปที่ 2.50 แสดงให้เห็นว่าทุกโมเดลมีประสิทธิภาพที่ดีขึ้นเมื่อใช้ภาพใบพืชต้นฉบับ ซึ่งก็ใช้เวลาในการฝึกสอนมากขึ้นเช่นกัน สำหรับโมเดลที่ได้รับ อัตราการประสบความสำเร็จ (Success rate) สูงสุด คือ VGG (ร้อยละ 99.48) และโมเดลที่ได้รับอัตราการทดสอบข้อผิดพลาด (Average error 0.0192) ที่น้อยที่สุดคือ AlexNetOWTBn และโมเดล VGG ใช้เวลาในการสร้างโมเดลประมาณ 5.5 วัน ในรูปที่ 2.51 แสดงประสิทธิภาพในชุดข้อมูลทดสอบสำหรับทั้งสองโมเดลระหว่างกระบวนการฝึกสอน และรูปที่ 3.52 แสดงการจำแนกประเภทของการสุ่มตัวอย่างภาพที่เลือกตลอดทั้งชุดข้อมูลทดสอบ การจำแนกประเภทภาพใช้เวลาเฉลี่ยประมาณ 2

มิลลิวินาทีใน GPU เดียวกันกับที่ใช้ในการฝึกสอน รูปที่ 3.53 จะแสดงถึงลำดับของความน่าจะเป็นที่โมเดลทำนายลำดับแรกคือคำตอบสุดท้ายที่โมเดลทำนาย

รูปที่ 3.52 ตัวอย่างผลการทดสอบ

3.2) ความสำคัญชนิดของภาพฝึกสอน

โมเดลที่ได้อัตราความสำเร็จสูงสุด (VGG และ AlexNetOWTBn) ได้ได้รับการทดสอบเพิ่มเติมเพื่อตรวจสอบความสำคัญของการฝึกสอนของภาพถ่ายชนิดที่ต่างกัน (ภาพที่ถ่ายจากห้องทดลอง และภาพที่ถ่ายจากสภาพแวดล้อมจริง) ผลที่ได้แสดงไว้ในรูปที่ 3.53 ซึ่งอัตราความสำเร็จจะต่ำกว่า เนื่องจากภาพที่ถ่ายจากสภาพแวดล้อมจริงมีจำนวนน้อยกว่าภาพที่ถ่ายจากห้องทดลอง ซึ่งภาพที่ได้ถ่ายจากสภาพแวดล้อมจริงมีอยู่ 12 คลาสจากทั้งหมด 58 คลาส ซึ่งผลการทดลองแสดงให้เห็นว่าแบบโมเดลจะมีประสิทธิภาพที่ดีขึ้นเมื่อได้รับการฝึกสอนจากภาพที่ถ่ายจากสภาพแวดล้อมจริงและระบุภาพที่ถ่ายจากห้องทดลอง (อัตราความสำเร็จสูงถึงร้อยละ 68) ในทางกลับกันเมื่อฝึกอบรมด้วยภาพที่ถ่ายจากห้องทดลองและระบุภาพที่ถ่ายจากสภาพแวดล้อมจริงมีอัตราความสำเร็จที่ต่ำกว่า (ประมาณร้อยละ 33) ซึ่งแสดงให้เห็นถึงความจริงที่ว่าการระบุภาพที่ถ่ายจากสภาพแวดล้อมจริงเป็นงานที่ยากและซับซ้อนกว่า และยังเป็นงานที่ยากสำหรับการพัฒนาระบบตรวจจับอัตโนมัติและวินิจฉัยโรคพืช

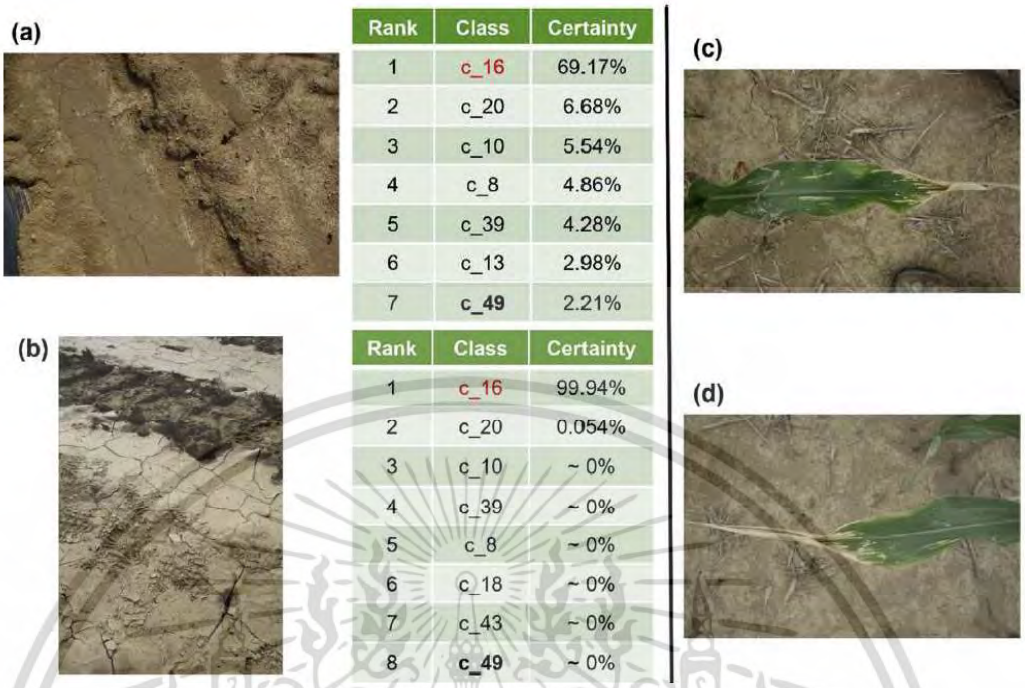
	Training: Laboratory - Testing: Field				Training: Field - Testing: Laboratory			
Model	Success rate	Average error	Epoch	Time (s/epoch)	Success rate	Average error	Epoch	Time (s/epoch)
AlexNetOWTBn	32.23%	3.5484	53	4375	62.57%	1.9369	104	–
VGG	33.27%	7.8541	54	4901	65.69%	2.6786	134	–

รูปที่ 3.53 ผลการทดสอบประสิทธิภาพในชุดข้อมูลที่เปรียบเทียบระหว่างภาพที่ถ่ายจากห้องทดลอง และภาพที่ถ่ายจากสภาพแวดล้อมจริง

3.3) สถานการณ์ที่ยังไม่แน่นอนและกรณีบ่งชี้

จากอัตราความสำเร็จร้อยละ 99.53 ของโมเดลแสดงให้เห็นว่าจากชุดข้อมูลทดสอบ 17,548 มี 82 ภาพเท่านั้นที่มีความผิดพลาดอยู่คือในภาพไม่มีใบพืชใด ๆ ซึ่งเป็นภาพในคลาส c_49 แต่โมเดลจำแนกให้อยู่ในคลาส c_16 (ข้าวโพดที่มีสุขภาพดี) ตารางในรูปที่ 3.54 ซึ่งกรณีนี้ถือว่าโมเดลจำแนกประเภทไม่ถูกต้องอาจเนเพราะว่าลักษณะของภาพพื้นดินปรากฏอยู่ในภาพของ c_16 ด้วยเช่นกัน ดังนั้นความถูกต้องของโมเดลจะมีประสิทธิภาพสูงกว่าร้อยละ 99.53

ปัญหาอื่น ๆ ของภาพที่ถ่ายจากสภาพแวดล้อมจริง ได้แก่ ภาพที่มีการบังแสงบางส่วนบนใบไม้ ภาพที่มีวัตถุหลายอย่างนอกเหนือจากภาพใบ เช่น นิ้วมือ รองเท้า ส่วนของเสื้อ ฯลฯ ภาพที่มีส่วนที่เหลือน้อยและไม่มีศูนย์กลางของเฟรม แสดงให้เห็นในรูปที่ 3.55 ซึ่งปัญหาเหล่านี้จัดการได้ด้วยการเรียนรู้ตัวอย่างที่มากขึ้น



รูปที่ 3.54 ปัจจัยที่ทำให้ข้อผิดพลาดในการจำแนกประเภท (ภาพที่ไม่มีใบพืช)

รูปที่ 3.55 ปัจจัยที่ทำให้ข้อผิดพลาดในการจำแนกประเภท

4) ข้อสรุป

จากผลการทดลองพบว่าสถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันของ VGG มีอัตราการประสบความสำเร็จร้อยละ 99.53 (+ error 0.47%) ซึ่งในการทดลองแสดงให้เห็นว่า โครงข่ายประสาทแบบคอนโวลูชันเหมาะสำหรับการตรวจจับและวินิจฉัยโรคพืชโดยอัตโนมัติผ่านการวิเคราะห์ภาพใบไม้ง่าย ๆ สำหรับการใช้

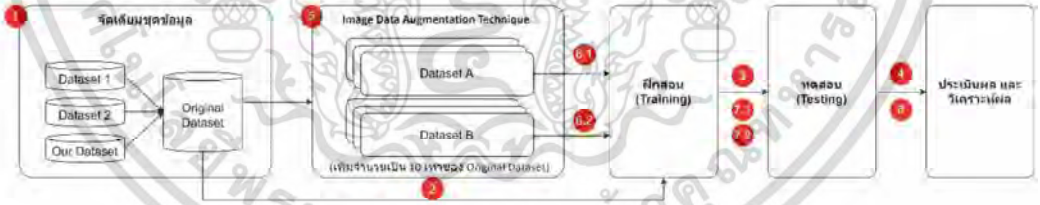
ภาพที่ถ่ายจากสภาพแวดล้อมจริงยังคงต้องได้รับการพัฒนาให้เพิ่มประสิทธิภาพในการจำแนกเพิ่มขึ้น สำหรับการตรวจจับและวินิจฉัยโรคพืชนั้นยังมีความเป็นไปได้สำหรับการนำไปใช้งานเนื่องจากการจำแนกภาพในโมเดลที่ได้รับการฝึกสอนแล้วนั้นใช้เวลาประมาณ 2 มิลลิวินาที สำหรับการคำนวณบน GPU หนึ่งตัว ซึ่งช่วยให้สามารถนำไปรวมเข้ากับแอปพลิเคชันบนมือถือได้ อย่างเช่น สมาร์ทโฟน (Smartphones) แต่สำหรับการใช้งานอาจจะใช้งานเป็นผู้ช่วยตัดสินใจหรือใช้ในการเฝ้าระวัง เนื่องจากประสิทธิภาพที่ได้ยังต้องมีการปรับปรุงอยู่สำหรับภาพที่ถ่ายจากสภาพแวดล้อมจริง

ดังนั้นงานวิจัยนี้จึงแสดงให้เห็นแล้วว่าโครงข่ายประสาทแบบคอนโวลูชันสามารถนำไปใช้ในการจำแนกโรคพืชได้

บทที่ 3
วิธีการดำเนินงานวิจัย

ในงานวิจัยนี้แบ่งการดำเนินงานออกเป็น 3 ส่วนหลัก ได้แก่

1) การจัดทำชุดข้อมูลใบข้าว ได้แก่ โรคไหม้ โรคใบจุดสีน้ำตาล โรคใบขีดสีน้ำตาล โรคขอบใบแห้ง โรคใบสีส้ม ซึ่งเป็นโรคข้าวที่มักพบได้ในประเทศไทย (Rice Department 2016) และใบข้าวที่ไม่เป็นโรค รวม 6 Class ประกอบไปด้วยชุดข้อมูลจาก 2 ฐานข้อมูลสาธารณะ และชุดข้อมูลที่จัดทำเอง เพื่อให้ชุดข้อมูลมีความครอบคลุมโรคข้าวในประเทศไทย รวมถึงสภาพแวดล้อม พื้นหลังของภาพ ในการถ่ายภาพด้วย
2) การฝึกสอนโมเดล ด้วยการถ่ายทอดองค์ความรู้จากโมเดลที่ได้รับการฝึกสอนมาแล้วด้วยชุดข้อมูล ImageNet ได้แก่โมเดล ResNet50V2 ResNet101V2 ResNet152V2 InceptionV3 InceptionResNetV2 DenseNet121 DenseNet169 และ DenseNet201
3) การวิเคราะห์ผล และการปรับปรุงประสิทธิภาพ ด้วยการเพิ่มจำนวนของภาพด้วย Image Data Augmentation Technique กัน 2 วิธี เพื่อทดลองการตอบสนองของโมเดลกับชุดข้อมูลที่มีจำนวนและความซับซ้อนของภาพเพิ่มขึ้น



รูปที่ 3.1 ขั้นตอนการทดลอง

สำหรับการทดลองทั้งหมดมีขั้นตอน ดังนี้

1) รวบรวมรูปภาพจาก 3 ฐานข้อมูลรูปภาพจาก 3 แหล่งเข้าด้วยกัน เรียกว่า Original Dataset ได้แก่
   1.1) Dataset 1: UCI database (Newman 1998)
   1.2) Dataset 2: Rice Leaf Disease Image Samples dataset (Sethy 2020)
   1.3) Our Dataset: ชุดข้อมูลรูปภาพที่ผู้วิจัยได้จัดทำขึ้น

2) ฝึกสอนชุดข้อมูล Original Dataset
3) ทดสอบชุดข้อมูล Original Dataset
4) วิเคราะห์ผลชุดข้อมูล Original Dataset
5) ทดลองเพิ่มประสิทธิภาพด้วยการเพิ่มจำนวนของภาพด้วย Image Data Augmentation Technique ด้วยวิธีการสุ่มค่าและการกำหนดพารามิเตอร์ที่ต่างกัน 2 ชุดข้อมูล (Dataset A และ Dataset B)
6) ฝึกสอนชุดข้อมูล
   6.1) ฝึกสอนชุดข้อมูล Dataset A
   6.2) ฝึกสอนชุดข้อมูล Dataset B
7) ทดสอบชุดข้อมูล Dataset A และ Dataset B
   7.1) ทดสอบชุดข้อมูล Dataset A
   7.2) ทดสอบชุดข้อมูล Dataset B
8) เปรียบเทียบและวิเคราะห์ผล

3.1 ชุดข้อมูล

ชุดข้อมูลฝึกสอนมีความสำคัญเป็นอย่างมากกับการเรียนรู้ของโครงข่ายประสาทแบบคอนโวลูชัน ซึ่งต้องประกอบไปด้วยภาพที่มีจำนวนมากและหลากหลายเพียงพอที่จะให้โมเดลจดจำ และเรียนรู้ได้อย่างมีประสิทธิภาพ เปรียบเสมือนการเรียนรู้ของมนุษย์ยิ่งเราเห็นมากเท่าไหร่เรายิ่งจำได้มากขึ้นเท่านั้น

ในงานวิจัยนี้ได้ทำการรวบรวมชุดข้อมูลจาก ฐานข้อมูลสาธารณะที่สามารถเข้าถึงได้ 2 แหล่งด้วยกันได้แก่ UCI database (Newman 1998) และ Rice Leaf Disease Image Samples dataset (Sethy 2020) รวมเข้ากับฐานข้อมูลชุดรูปภาพที่ผู้วิจัยได้จัดทำขึ้นเอง เพื่อให้ชุดข้อมูลมีความครอบคลุมโรคข้าวในประเทศไทย รวมถึงสภาพแวดล้อม พื้นหลังของภาพที่หลากหลายมากขึ้น

ชุดข้อมูลทั้งหมดที่รวบรวมครอบคลุมโรคข้าวที่มักพบในประเทศไทย 5 โรค ได้แก่ โรคไหม้ โรคใบจุดสีน้ำตาล โรคใบขีดสีน้ำตาล โรคขอบใบแห้ง และ โรคใบสีส้ม รวมถึงใบข้าวที่ไม่เป็นโรคด้วย รวมทั้งหมด 6 Class มีจำนวนตามตารางที่ 3.1

ตารางที่ 3.1 ชุดข้อมูลทั้งหมด

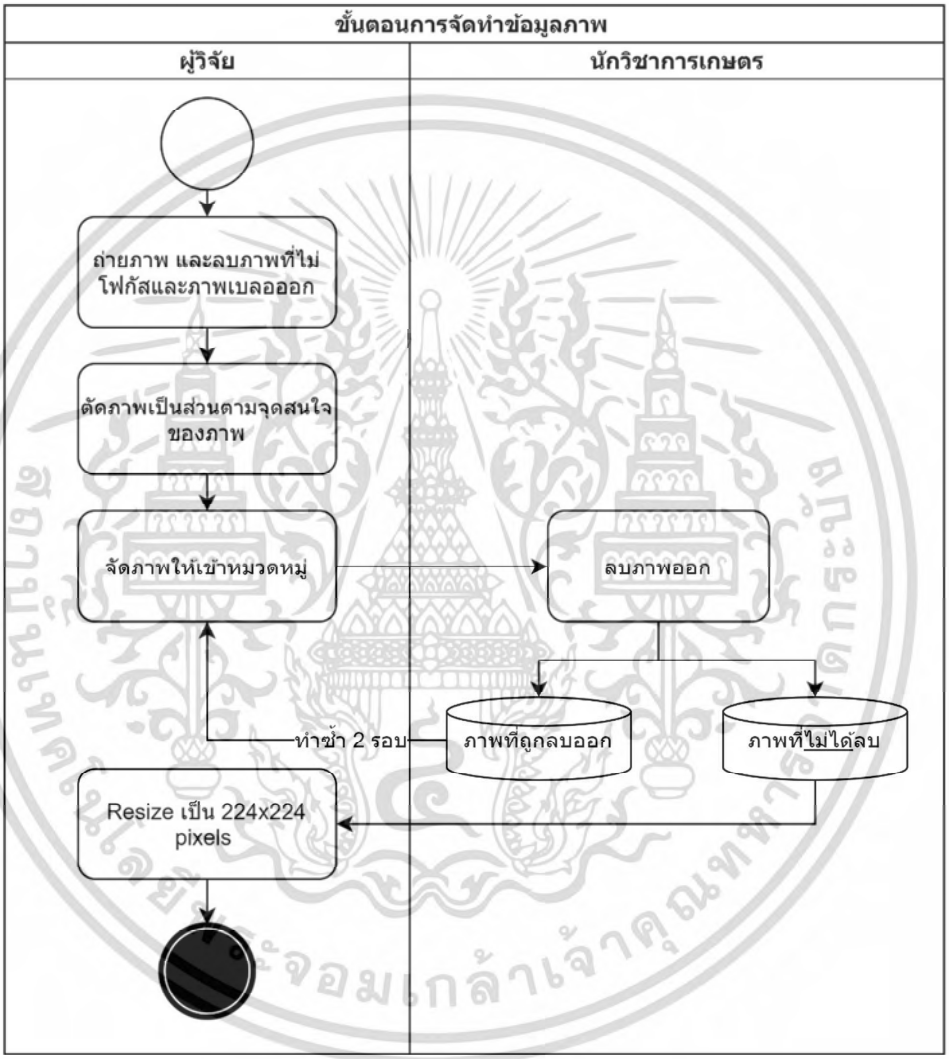
โรคข้าว	Dataset 1	Dataset 2	Our Dataset	รวม (Original Dataset)
ข้าวที่ไม่เป็นโรค (Healthy)	-	-	899	**899**
โรคไหม้ (Blast disease)	-	1440	315	**1,755**
โรคใบจุดสีน้ำตาล (Brown Spot disease)	40	1600	240	**1,880**
โรคขอบใบแห้ง (Bacterial Leaf Blight disease)	40	1584	155	**1,779**
โรคใบสีส้ม (Rice Tungro disease)	-	1308	-	**1,308**
โรคใบขีดสีน้ำตาล (Narrow Brown Spot disease)	-	-	279	**279**
สภาพแวดล้อมการถ่ายภาพ	ห้องปฏิบัติ-การ	สภาพแวด-ล้อมจริง	สภาพแวด-ล้อมจริง	-

3.1.1 การจัดทำชุดข้อมูลโรคข้าว

สำหรับขั้นตอนการจัดทำชุดข้อมูลได้ทำการจัดเก็บในพื้นที่ อำเภอบางเลน จังหวัดนครปฐม โดยการถ่ายภาพด้วยสมาร์ทโฟน โดยมีสภาพแวดล้อม แสง และพื้นหลังของภาพที่ต่างกัน ในช่วงปี พ.ศ. 2561-2563 โดยการดำเนินการ ดังนี้ (รูปที่ 3.2)

1) ผู้วิจัยถ่ายภาพด้วยสมาร์ทโฟน ภาพที่ไม่โฟกัส (Focus) หรือภาพที่เบลอ (Blur) จะถูกลบออกทันทีในขั้นตอนนี้ และได้ภาพทั้งหมด 1,732 ภาพ
2) ภาพจะถูกตัดส่วนที่สนใจของภาพ เฉพาะภาพที่ถูกถ่ายในมุมกว้าง (ดังรูปที่ 3.3) จำนวนของภาพทั้งหมดเป็น 2,712 ภาพ
3) จากนั้นภาพทั้งหมดจะถูกจัดให้เข้าหมวดหมู่ของโรคข้าวโดยดูจากลักษณะอาการร่องรอยในเบื้องต้นก่อนโดยการจำแนกโดยผู้วิจัย
4) ภาพทั้งหมดจะถูกส่งไปให้นักวิชาการเกษตรทำการลบภาพที่ไม่ได้อยู่ในหมวดหมู่ที่ถูกต้องออก
5) ภาพที่ถูกลบจะถูกทำซ้ำในขั้นตอนที่ 3 และ 4 จำนวน 2 รอบ

6) จากนั้นภาพที่ไม่สามารถจำแนกได้ภาพนั้นจะถูกคัดออกไป และเหลือภาพที่นำไปประมวลผลต่อไปได้ทั้งหมด 1,888 ภาพ
7) จากนั้นภาพในชุดข้อมูลจะถูก Resize เป็น 224x224 pixels ตาม Input ในทุกๆ โมเดล



รูปที่ 3.2 ขั้นตอนการจัดทำข้อมูลภาพ

รูปที่ 3.3 ภาพที่ถูกตัดส่วนที่สนใจของภาพเฉพาะภาพที่ถูกถ่ายในมุมกว้าง

3.2 การตั้งค่าการทดลอง

โครงข่ายประสาทแบบคอนโวลูชันนอล งานวิจัยนี้จะใช้สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลที่ได้รับการฝึกสอนแล้วไว้แล้วจากชุดข้อมูล ImageNet จาก Keras ได้แก่ Xception ResNet50V2 ResNet101V2 ResNet152V2 InceptionV3 InceptionResNetV2 DenseNet121 DenseNet169 และ DenseNet201 โดยมี ขนาด พารามิเตอร์ และความลึก ตามตารางที่ 3.2 ใช้การ Transfer Learning ค่าน้ำหนัก (Weights) จาก ImageNet กำหนดให้ Batch size เท่ากับ 64 จำนวน Epoch เท่ากับ 20 และ Learning เท่ากับ 0.0001 ชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบจะถูกแบ่งเป็น 80:20 อ้างอิงจาก (Mohanty, Hughes et al. 2016) เพื่อความแม่นยำของ Accuracy จะทำการทดสอบ Run โมเดลทั้งหมด 10 ครั้งโดยในแต่ละครั้งจะมีการสุ่มภาพที่ใช้เป็นชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบใหม่ในแต่ละรอบที่ Run ใหม่ จากนั้นนำ Accuracy ทั้ง 10 รอบหาค่าเฉลี่ย โมเดลจะถูกรันบน Google Colab (Bisong 2019) Intel Xeon CPU @2.20 GHz, 13 GB RAM, Tesla K80 accelerator และ GDDR5 VRAM 12 GB

ตารางที่ 3.2 สถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชันนอลที่ใช้ในการทดลอง

Model	Size	Parameters	Depth
Xception	88 MB	22,910,480	126
ResNet50V2	98 MB	25,613,800	-
ResNet101V2	171 MB	44,675,560	-
ResNet152V2	232 MB	60,380,648	-
InceptionV3	92 MB	23,851,784	159
InceptionResNetV2	215 MB	55,873,736	572
DenseNet121	33 MB	8,062,504	121
DenseNet169	57 MB	14,307,880	169
DenseNet201	80 MB	20,242,984	201

3.3 การปรับปรุงประสิทธิภาพของโมเดล

จากผลการทดลองของโมเดลในตารางที่ 3.2 จะเห็นได้ว่า Accuracy อยู่ที่ระหว่าง 0.49766 ถึง 0.55430 ซึ่งถือได้ว่าโมเดลยังมีประสิทธิภาพในการเรียนรู้ยังไม่เป็นที่ประทับใจ

ตารางที่ 3.3 ผลการทดลองชุดข้อมูล Original Dataset

Model	Accuracy
Xception	0.54570
ResNet50V2	0.55156
ResNet101V2	0.55430
ResNet152V2	0.53125
InceptionV3	0.54766
InceptionResNetV2	0.54297
DenseNet121	0.49766
DenseNet169	0.52219
DenseNet201	0.60586

สำหรับแนวทางในการปรับปรุงประสิทธิภาพนั้นผู้วิจัยได้ตั้งสมมติฐานไว้ว่า ข้อมูลภาพที่ใช้เรียนรู้นั้นสำหรับ Pre-trained โมเดลนั้นอาจไม่เพียงพอ จึงต้องเพิ่มจำนวนของภาพให้มากขึ้นเพื่อทดลองปรับปรุงประสิทธิภาพในขั้นต่อไป แต่เนื่องจากสถานการณ์ โควิด-19 แพร่ระบาดจึงทำให้การ

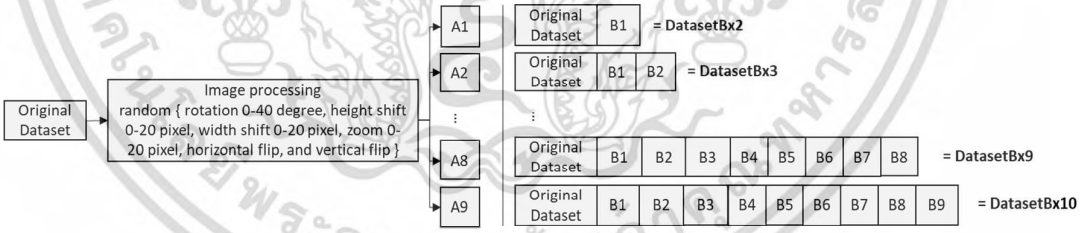
เก็บข้อมูลภาพเพิ่มเติมนั้นทำได้ยาก และต้องใช้เวลาที่ยาวนานในการตรวจสอบชุดข้อมูลภาพ ดังนั้นแนวทางที่จะปรับปรุงประสิทธิภาพของโมเดลได้อีกแนวทางหนึ่งคือการใช้ Image Data Augmentation Technique ซึ่งเป็นอีกหนึ่งเทคนิคในการเพิ่มจำนวนของภาพที่น่าสนใจ ในงานวิจัยนี้ได้ทำการทดลองสร้างชุดข้อมูลใหม่ 2 วิธีด้วยกันด้วยใช้พารามิเตอร์ และช่วงที่ใช้ในการสุ่ม ที่ต่างกันและได้ชุดข้อมูลใหม่ 2 ชุดได้แก่ DatasetA และ DatasetB ที่เพิ่มขึ้นจาก Original Dataset เป็น 10 เท่า ซึ่งจะอธิบายในข้อ 3.3.1 และ 3.3.2 ต่อไป

3.3.1 การประมวลผล DatasetA

การใช้ Image Data Augmentation Technique เป็นเทคนิคในการเพิ่มจำนวนของภาพ โดยในการทดลองนี้จะใช้ Keras (imageDataGenerator) ในการประมวลผลภาพ

สำหรับการสร้าง DatasetA จะใช้ช่วงของการสุ่มพารามิเตอร์ในการสร้างภาพใหม่ให้อยู่ในช่วงที่ภาพไม่ได้มีการแต่ต่างจากภาพต้นฉบับมากนัก เพื่อไม่ให้คุณลักษณะจุดที่สนใจของภาพเสียหาย เช่นลักษณะของร่องรอยบิดเบี้ยวไป โดยมีพารามิเตอร์ในการสุ่มดังนี้ Rotation 0-40 องศา Height Shift 0-20 pixel. Width Shift 0-20 pixel. Zoom 0-20 pixel. Horizontal Flip และ Vertical Flip

ใน DatasetA จะใช้พารามิเตอร์ในการประมวลผลภาพเหมือนกันทั้งหมดในการเพิ่มชุดข้อมูลทั้งหมด 10 เท่า จากชุดข้อมูล Original ดังภาพที่ 3.3

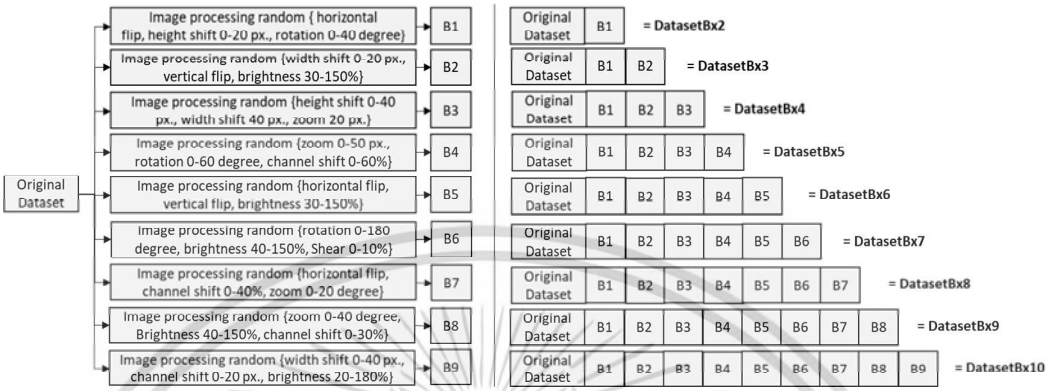


รูปที่ 3.4 การเพิ่มจำนวนของภาพของชุดข้อมูล DatasetA

3.3.2 การประมวลผล DatasetB

สำหรับการสร้าง DatasetB จะเพิ่มพารามิเตอร์ที่เกี่ยวกับสี และแสง เข้าไป และทำการเพิ่มช่วงของการสุ่มค่าของพารามิเตอร์ให้มากขึ้น โดยในแต่ละเท่าที่เพิ่มขึ้นของชุดข้อมูลเราจะใช้พารามิเตอร์ที่ต่างกัน เพื่อเพิ่มความซับซ้อนให้กับภาพ ทั้งนี้ช่วงที่ใช้ในการสุ่ม

พารามิเตอร์ที่เพิ่มขึ้นนั้นยังคงรักษาคุณลักษณะของภาพไว้อยู่เช่นกัน โดยพารามิเตอร์ที่ถูกกำหนดในแต่ละเท่า ดังแสดงในภาพที่ 3.4



รูปที่ 3.4 การเพิ่มจำนวนของภาพของชุดข้อมูล DatasetB

บทที่ 4
ผลการวิจัยและการอภิปรายผล

จากผลการทดลองโดยการใช้ข้อมูลชุดรูปภาพทั้งหมดที่ได้รวมรวมมาจาก 2 แหล่งข้อมูลสาธารณะได้แก่ UCI database (Newman 1998) และ Rice Leaf Disease Image Samples dataset (Sethy 2020) และ ชุดข้อมูลภาพที่ผู้วิจัยได้ทำการรวมรวมขึ้นเองในระยะเวลาและทรัพยากรที่จำกัด ซึ่งชุดข้อมูลทั้ง 3 แหล่งนี้ที่รวมเข้าไว้ด้วยกันเรียกว่า Original Dataset จากนั้นนำชุดข้อมูล Original Dataset เข้าสู่กระบวนการปรับขนาดเพื่อให้สอดคล้องกับ Input ของพรีเทรนโมเดล และนำชุดข้อมูลทั้งหมดแบ่งออกเป็นชุดข้อมูลฝึกสอนและชุดข้อมูลทดสอบ 80:20 จากนั้นทำการฝึกสอนและทดสอบโมเดลตามผลการทดลองที่ได้ในตารางที่ 4.3 (Accuracy Original Dataset) ผลปรากฏว่าค่าของ Accuracy อยู่แค่เพียง 0.49766 ถึง 0.60586 เท่านั้นซึ่งถือว่ายังไม่เป็นที่ประทับใจและยังคงต้องปรับปรุงต่อไป

ดังนั้นผู้วิจัยจึงได้ตั้งสมมติฐานไว้ว่า ข้อมูลภาพที่ใช้เรียนรู้นั้นสำหรับ Pre-trained โมเดลนั้นอาจไม่เพียงพอ จึงต้องเพิ่มจำนวนของภาพให้มากขึ้นเพื่อทดลองปรับปรุงประสิทธิภาพในขั้นต่อไป แต่เนื่องจากสถานการณ์ โควิด-19 แพร่ระบาดจึงทำให้การเก็บข้อมูลภาพเพิ่มเติมนั้นทำได้ยาก และต้องใช้เวลาที่ยาวนานในการตรวจสอบชุดข้อมูลภาพ ดังนั้นแนวทางที่จะปรับปรุงประสิทธิภาพของโมเดลได้อีกแนวทางหนึ่งคือการใช้ Image Data Augmentation Technique ในการเพิ่มจำนวนของภาพ โดยที่ได้เพิ่มจำนวนของภาพทั้ง 2 วิธี เป็นสองกลุ่มได้แก่ DatasetA และ DatasetB ซึ่งใช้พารามิเตอร์และช่วงของการสุ่มที่ต่างกัน (DatasetA (ผลการทดลองแสดงในตารางที่ 4.1) และ DatasetB (ผลการทดลอลแสดงในตารางที่ 4.2)) ซึ่งสามารถเพิ่มประสิทธิภาพให้กับโมเดลได้

ตารางที่ 4.1 ผลการทดลอง Accuracy ของ DatasetA

Model	Original Dataset	x2	x3	x4	x5	x6
Xception	0.54570	0.65230	0.66223	0.67194	0.67031	0.67617
ResNet50V2	0.55156	0.67152	0.67500	0.68748	0.68867	0.68359
ResNet101V2	0.55430	0.66992	0.67247	0.68248	0.67109	0.69464
ResNet152V2	0.53125	0.65039	0.66636	0.67044	0.65769	0.66797
InceptionV3	0.54766	0.64189	0.63984	0.63633	0.68750	0.66563
InceptionResNetV2	0.54297	0.65742	0.65595	0.65039	0.68847	0.66179
DenseNet121	0.49766	0.57656	0.59125	0.61244	0.59609	0.60078
DenseNet169	0.52219	0.63034	0.62473	0.64727	0.63828	0.64922
DenseNet201	0.60586	0.64730	0.65805	0.66758	0.65117	0.67948

Model	x7	x8	x9	x10
Xception	0.67383	**0.68867**	0.68681	0.66316
ResNet50V2	**0.69102**	0.69102	0.69786	0.68867
ResNet101V2	**0.69766**	0.69142	0.69628	0.66720
ResNet152V2	0.68055	**0.68398**	0.67263	0.67031
InceptionV3	0.66953	**0.68984**	0.68164	0.64063
InceptionResNetV2	**0.69450**	0.67813	0.66563	0.68945
DenseNet121	**0.62914**	0.58683	0.58398	0.59042
DenseNet169	**0.66938**	0.66914	0.64336	0.66655
DenseNet201	**0.68083**	0.68016	0.65898	0.67250

ตารางที่ 4.2 ผลการทดลอง Accuracy ของ DatasetB

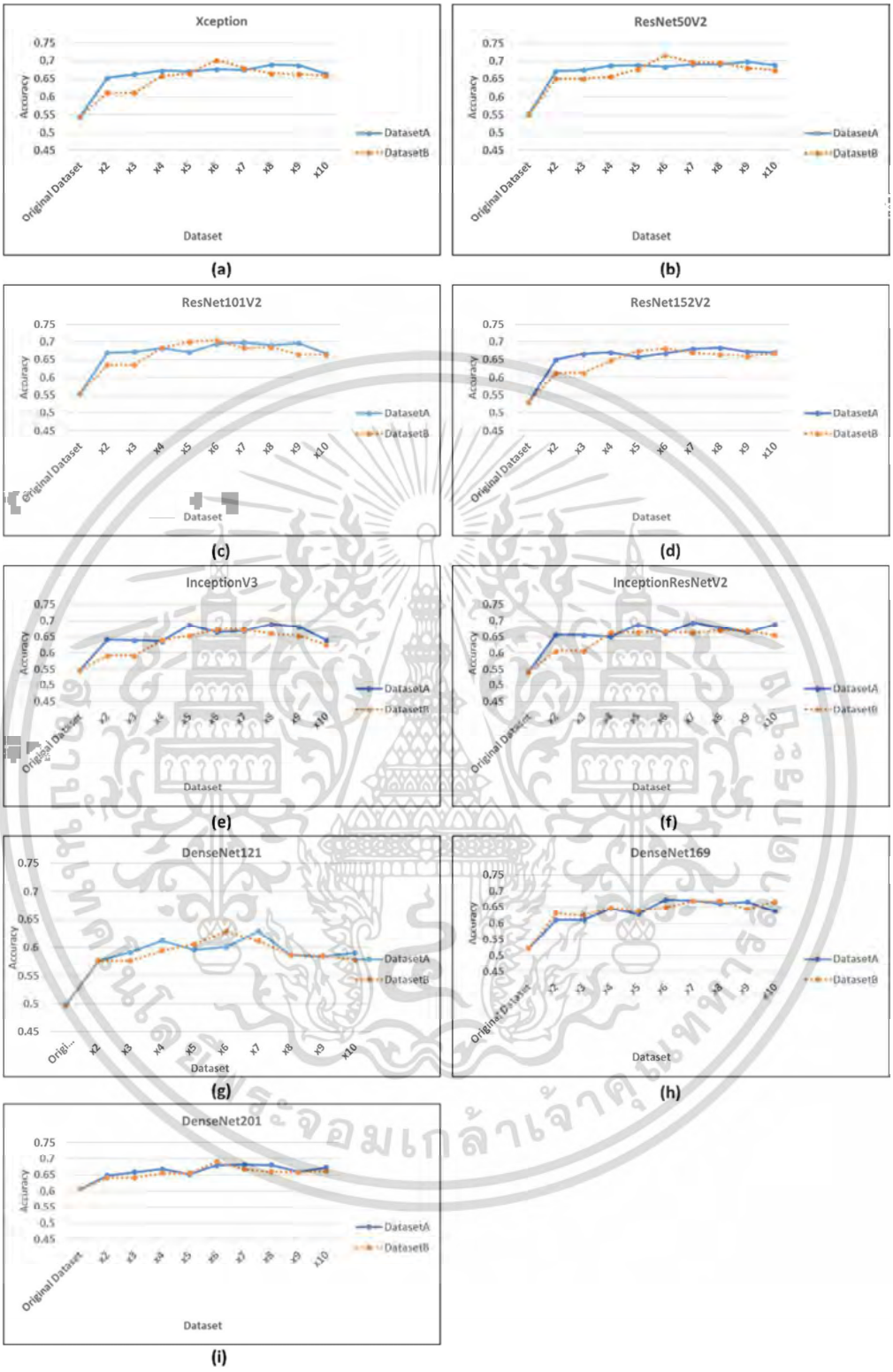
Model	Original Dataset	x2	x3	x4	x5	x6
Xception	0.54570	0.61133	0.61133	0.65775	0.66391	**0.70183**
ResNet50V2	0.55156	0.65117	0.65117	0.65669	0.67695	**0.71689**
ResNet101V2	0.55430	0.63594	0.63594	0.68422	0.69992	**0.70338**
ResNet152V2	0.53125	0.61250	0.61250	0.64831	0.67422	**0.68183**
InceptionV3	0.54766	0.59180	0.59180	0.64023	0.65451	0.67266
InceptionResNetV2	0.54297	0.60575	0.60575	0.66367	0.66511	0.66742
DenseNet121	0.49766	0.57644	0.57644	0.59438	0.60594	**0.62944**
DenseNet169	0.52219	0.61016	0.61016	0.64506	0.62813	**0.67273**
DenseNet201	0.60586	0.64117	0.64117	0.65508	0.65469	**0.69089**

Model	x7	x8	x9	x10
Xception	0.67906	0.66417	0.66186	0.65853
ResNet50V2	0.69688	0.69520	0.68086	0.67422
ResNet101V2	0.68281	0.68516	0.66445	0.66367
ResNet152V2	0.66953	0.66445	0.65938	0.66703
InceptionV3	**0.67305**	0.66133	0.65313	0.62539
InceptionResNetV2	0.66250	**0.67070**	0.67031	0.65492
DenseNet121	0.61133	0.58711	0.58516	0.57773
DenseNet169	0.66992	0.66094	0.66602	0.63672
DenseNet201	0.66680	0.65938	0.65703	0.66195

ตารางที่ 4.3 สรุปผลการทดลอง เปรียบเทียบระหว่างชุดข้อมูล Original Dataset, DatasetA และ DatasetB

Model	Accuracy Original Dataset	Accuracy DatasetA	Accuracy DatasetB	Increase from Original Dataset %
Xception	0.54570	0.68867	**0.70183**	15.613
ResNet50V2	0.55156	0.69102	**0.71689**	**16.533**
ResNet101V2	0.55430	0.69766	**0.70338**	14.908
ResNet152V2	0.53125	**0.68398**	0.68183	15.273
InceptionV3	0.54766	**0.68984**	0.67305	14.219
InceptionResNetV2	0.54297	**0.69450**	0.67070	15.153
DenseNet121	0.49766	0.62914	**0.62944**	13.178
DenseNet169	0.52219	0.66938	**0.67273**	15.055
DenseNet201	0.60586	0.68083	**0.69089**	8.503

จากผลการทดลองประสิทธิภาพของโมเดลในชุดข้อมูล Original Dataset โมเดลที่ให้ประสิทธิภาพสูงสุดคือ DenseNet201 (0.60586) และโมเดลที่ให้ค่าต่ำสุดใน Original Dataset คือ DenseNet121 (0.49766) จากนั้นได้ปรับปรุงประสิทธิภาพของโมเดลด้วยการเพิ่มจำนวนภาพด้วย Image Data Augmentation Technique ทั้ง 2 วิธี ผลปรากฏว่า ชุดข้อมูล DatasetB ในโมเดล ResNet50V2 ได้ทำให้ประสิทธิภาพของโมเดลเพิ่มได้สูงถึง 16.533 เปอร์เซ็นต์ เมื่อเทียบกับ Original Dataset จากผลการทดลองแสดงให้เห็นว่าการเพิ่มจำนวนของภาพทำให้ประสิทธิภาพของโมเดลเพิ่มขึ้นทุกในโมเดล ตั้งแต่ 13.178 ถึง 16.533 เปอร์เซ็นต์ ยกเว้น DenseNet201 มีประสิทธิภาพเพิ่มขึ้นเพียง 8.503 เปอร์เซ็นต์ เมื่อตั้งข้อสังเกตจากผลการทดลอง และโครงสร้างของโมเดลจะเห็นได้ว่า Accuracy ของ DenseNet201 ให้ค่าสูงที่สุดตั้งแต่ Original Dataset แล้ว ดังนั้นอาจหมายความได้ว่าโมเดลได้เรียนรู้คุณลักษณะที่สำคัญของภาพมาตั้งแต่ก่อนที่จะมีการเพิ่มจำนวนของภาพแล้ว ดังนั้นการเพิ่มของภาพจึงไม่ส่งผลให้โมเดลได้เรียนรู้เพิ่มมากขึ้นเท่าใด กล่าวคือยิ่งเมื่อเพิ่มจำนวนภาพมากขึ้นเท่าใดโมเดลภาพจะเริ่มมีความคล้ายกันมากขึ้นเรื่อย ๆ ดังนั้นความสามารถในการเรียนรู้ของโมเดลก็จะไม่เพิ่มมากขึ้น และภาพที่โมเดลไม่สามารถแยกแยะได้หรือ Outline Data ตั้งแต่แรกก็ถูกสร้างขึ้นมาเพิ่มด้วยเช่นเดียวกัน ดังนั้นนอกจากประสิทธิภาพในการเรียนรู้จะคงที่แล้วยังส่งผลให้ประสิทธิภาพของโมเดลลดลงได้อีกด้วยเนื่องจากการเพิ่ม Data ที่เป็นตัวอย่างที่ทำให้โมเดลเกิดความสับสนมากขึ้น



รูปที่ 4.1 กราฟแสดงการเปรียบเทียบ Accuracy ของแต่ละโมเดลสำหรับชุดข้อมูล Original Dataset, DatasetA และ DatasetB

จากรูปที่ 4.1 แสดงการเพิ่มขึ้นของ Accuracy ในชุดข้อมูลที่เพิ่มขึ้นในแต่ละเท่าตัวจาก Original Dataset จากกราฟในทุกโมเดลเมื่อเพิ่มจำนวนของภาพด้วย Image Data Augmentation Technique แล้ว จากนั้นกราฟจะเริ่มคงที่ หรือเพิ่มขึ้นและลดลงไม่มากนักเนื่องมาจากโมเดลได้เรียนรู้คุณลักษณะสำคัญของภาพจนครบถ้วนสุดความสามารถแล้วจึงทำให้องค์ความรู้ของโมเดลไม่ได้เพิ่มขึ้นตามจำนวนของภาพที่เพิ่มขึ้นเสมอไป และจากการทดลองจุดที่โมเดลให้ค่า Accuracy สูงสุดจะอยู่ที่จำนวนของภาพที่เพิ่มจาก Original Dataset 6-8 เท่า สำหรับ DatasetA ส่วนใหญ่จะให้ Accuracy สูงสุดอยู่ที่การเพิ่มขึ้นของจำนวนของภาพ 7-8 เท่า ในขณะเดียวกันใน DatasetB จะอยู่ที่ 6 เท่า เนื่องจาก DatasetB ได้ใช้การประมวลผลภาพที่มากกว่า เช่น การใช้พรามิเตอร์ และช่วงของการสุ่มในการเพิ่มจำนวนของภาพในแต่ละเท่าที่ไม่เหมือนกัน และการเพิ่มพารามิเตอร์ในการประมวลที่เกี่ยวข้องกับ แสงและสีเพิ่มเข้ามาด้วย จึงส่งผลให้โมเดลเรียนรู้ได้เร็วกว่า

ดังนั้นข้อมูลภาพที่มีความหลากหลายและซับซ้อนตามธรรมชาติยังคงเป็นหนึ่งปัจจัยที่จำเป็นสำหรับการเรียนรู้ของโมเดลอยู่ และการใช้ Image Data Augmentation Technique เข้ามาช่วยเพิ่มจำนวนของข้อมูลของภาพนั้นก็เทคนิคหนึ่งที่ช่วยเพิ่มประสิทธิภาพของโมเดลได้ในระดับหนึ่งด้วยเช่นกัน

อีกประการหนึ่งที่สังเกตได้คือ การเพิ่มขึ้นของ Accuracy จาก Original Dataset ขึ้นมาเป็น 1 (x2) เท่าตัวด้วยการใช้ Image Data Augmentation Technique นั้นจะเห็นได้ว่า Accuracy จะสูงขึ้นมากกว่าในการเพิ่มขึ้นในจำนวนเท่าต่อ ๆ ไป อาจสรุปได้ว่าภาพที่ได้ทำการเพิ่มไปนั้นยังไม่ได้มีความคล้ายภาพก่อนหน้ามากนัก จึงทำให้โมเดลเรียนรู้ได้ดีด้วยเช่นกัน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้ทำการเก็บรวบรวมข้อมูลจาก 3 แหล่งเข้าไว้ด้วยกันได้แก่ข้อมูลจากฐานข้อมูลที่เปิดให้เข้าถึงแบบสาธารณะ และฐานข้อมูลที่เราได้จัดทำขึ้นใหม่ เรียกว่า Original Dataset จากนั้นทำการทดลองกับ Original Dataset ผลการทดลองปรากฏว่า Accuracy อยู่ระหว่าง 0.49766 ถึง 0.60586 ซึ่งผลลัพธ์ยังไม่เป็นที่น่าประทับใจ ต่อมาจึงใช้ Image Data Augmentation Technique ในการเพิ่มจำนวนภาพในวิธีที่แตกต่างกัน 2 วิธี (DatasetA และ DatasetB) จึงสามารถเพิ่มประสิทธิภาพให้กับโมเดลได้ถึง 16.533 เปอร์เซ็นต์ และจากการทดลองจุดที่โมเดลให้ค่า Accuracy สูงสุดจะอยู่ที่จำนวนของภาพที่เพิ่มจาก Original Dataset 6-8 เท่า อย่างไรก็ตามโมเดลยังคงต้องได้รับการพัฒนาต่อไปเพื่อให้ประสิทธิภาพของโมเดลมากขึ้น และสามารถนำไปประยุกต์ใช้ได้จริงในภาคการเกษตรต่อไป

5.2 ข้อเสนอแนะ

จากผลการทดลองทั้งหมดที่กล่าวมาจะเห็นได้ว่าโมเดลยังคงต้องได้รับการพัฒนาต่อไป โดยมีแนวทางที่น่าสนใจในหลายทิศทาง ได้แก่

1) เพิ่มข้อมูลรูปภาพด้วยการถ่ายภาพจากแปลงปลูกจริง หรือ รวบรวมชุดข้อมูลภาพจากแหล่งที่เปิดให้เข้าถึงแบบสาธารณะให้มากขึ้น เพราะการใช้ Image Data Augmentation Technique เป็นเพียงการจำลองข้อมูลเพิ่มขึ้น ซึ่งในความเป็นจริงตามธรรมชาติแล้วร่องรอยของการเกิดโรคมีความหลากหลาย รวมถึงสภาพแวดล้อมด้วย เช่น แสง สี เป็นต้น จากนั้นทำการปรับแต่ง (Fine-tune) โมเดล Densenet201 ที่ให้ประสิทธิภาพที่ดีที่สุดตั้งแต่แรก (ในชุดข้อมูล Original Dataset)
2) การใช้ Data Augmentation Technique ในการเพิ่มจำนวนภาพ ประกอบกับทำการปรับแต่ง (Fine-tune) Pre-train CNN โมเดล เพื่อให้สามารถเรียนรู้คุณลักษณะที่เฉพาะได้แก่โมเดล ResNet50V2 ResNet101V2 และ Xception ที่เป็นโมเดลที่ให้ประสิทธิภาพที่ดีที่สุดสำหรับชุดข้อมูลที่ใช้ Image Data Augmentation Technique ในการเพิ่มจำนวนของภาพ

3) การใช้ Data Augmentation Technique ขั้นสูง เช่น การประยุกต์ใช้ Generative adversarial networks (GANs) (Pan, Yu et al. 2019) ในการ Pre-process data ก่อนที่จะส่งไปประมวลผลที่โมเดล Densenet201 ResNet50V2 ResNet101V2 และ Xception เป็นต้น

4) ทำการศึกษา Data Augmentation Technique ด้วยวิธีการต่าง ๆ ที่เหมาะสมกับการนำไปใช้กับโรคข้าวได้อย่างมีประสิทธิภาพและทำการทดลองปรับแต่ง (Fine-tune) โมเดลที่น่าสนใจของเรา ได้แก่ Densenet201 ที่ให้ประสิทธิภาพที่ดีที่สุดตั้งแต่แรก (ในชุดข้อมูล Original Dataset) และ ResNet50V2 ResNet101V2 Xception ที่เป็นโมเดลที่ให้ประสิทธิภาพที่ดีที่สุดสำหรับชุดข้อมูลที่ใช้ Image Data Augmentation Technique ในการเพิ่มจำนวนของภาพ

นอกจากนี้รูปภาพในธรรมชาติยังคงมีความหลากหลาย ซึ่งร่องรอยที่เกิดบนใบข้าวอาจไม่ใช่โรคข้าวก็เป็นได้ เช่น ร่องรอยของการถูกแมลง หรือหนอน ทำลาย เป็นต้น

สำหรับในทุกแนวทางที่กล่าวมานั้นถือเป็นสิ่งที่ท้าทายและน่าสนใจเป็นอย่างยิ่ง

อย่างไรก็ตามในงานวิจัยนี้ยังคงมีข้อจำกัดที่ได้รับการปรับปรุงเพิ่มเติมด้วยอีกหลายประการด้วยเช่นกัน ได้แก่

1) ข้อจำกัดด้านการเก็บข้อมูล และจัดทำชุดข้อมูล
   - ฤดูการทำนานั้นจะแบ่งออกเป็นช่วง ไม่คงที่ในแต่ละปี
   - การเกิดโรคข้าวสำหรับแปลงที่ได้รับการดูแลเป็นอย่างดีจะไม่สามารถเก็บตัวอย่างของภาพโรคข้าวได้
   - ทรัพยากรบุคล และงบประมาณมีอยู่อย่างจำกัด
2) ข้อจำกัดทางด้านเทคนิค
   - การประมวลผลภาพใช้เวลาที่ยาวนานในการจัดเตรียม
   - ทรัพยากร และงบประมาณมีอยู่อย่างจำกัด
3) ข้อจำกัดในด้านการวางแผน
   - เนื่องจากงานวิจัยนี้เป็นการศึกษาเชิงวิจัยและพัฒนา (Research and Develop) โดยมีการลองผิดลองถูกโดยยึดตามกระบวนการวิธีวิจัย จึงทำให้เกิดจุดบกพร่องในการพิจารณาบ้างเล็กน้อย

ทั้งนี้ในการพัฒนาต่อไปทางผู้วิจัยจำเป็นต้องกำจัดข้อจำกัดทั้งหมดนี้ต่อไปเพื่อให้การทดลองประสบความสำเร็จ และมีประสิทธิภาพที่ดีขึ้นสำหรับนำไปใช้ในภาคการเกษตรของประเทศไทยต่อไป

เอกสารอ้างอิง

ดารา เจตนะจิตร, น. น., พากเพียร อรัญนารถ, วิชิต ศิริสันธนะ, วิชชุดา รัตนากาญจน์, รัศมี ฐิติเกียรติพงศ์, วันชัย โรจนหัสดิน และธัญลักษณ์ อารยาพันธ์ (2550). **โรคข้าวและการป้องกัน**. กรุงเทพฯ, โรงพิมพ์ชุมนุมสหกรณ์การเกษตร.

รัฐบาลไทย. (2565, 3 สิงหาคม 2565). **จุรินทร์มั่นใจ ส่งออกข้าวตามเป้าสิ้นปี 7 ล้านตัน 7 เดือน เพิ่ม 58% เดินหน้ายุทธศาสตร์ข้าว ได้ข้าวพันธุ์ใหม่คุณภาพเยี่ยมแล้ว 6 สายพันธุ์ เตรียมขยายสู่มือเกษตรกร**. [Online]. เข้าถึงได้จาก : https://www.thaigov.go.th/news/contents/details/57561?fbclid=IwAR0kjyOMsxWMRQi7yJ3xN5zZWpiaCiJu9lC9E3391d0Iym1Ov87KIW4bJ-M.

Alsayed, A., A. Alsabei and M. Arif (2021). "Classification of apple tree leaves diseases using deep learning methods." International Journal of Computer Science & Network Security **21**(7): 324-330.

Alzubaidi, L., J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie and L. Farhan (2021). "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions." Journal of big Data **8**(1): 1-74.

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. Proceedings of ICML workshop on unsupervised and transfer learning, JMLR Workshop and Conference Proceedings.

Berno, T., G. Dentice and J. J. Wisansing (2019). Kin kao laew reu young ('have you eaten rice yet')?: A new perspective on food and tourism in Thailand. Food tourism in Asia, Springer: 17-30.

Bisong, E. (2019). Google Colaboratory. Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. Berkeley, CA, Apress: 59-64.

Carranza-Rojas, J., H. Goeau, P. Bonnet, E. Mata-Montero and A. Joly (2017). "Going deeper in the automated identification of Herbarium specimens." BMC evolutionary biology **17**(1): 1-14.

Chen, J., D. Zhang, Y. A. Nanehkaran and D. Li (2020). "Detection of rice plant diseases based on deep transfer learning." Journal of the Science of Food and Agriculture **100**(7): 3246-3256.

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

Chollet, F. (2021). Deep learning with Python, Simon and Schuster.

Dandawate, Y. and R. Kokare (2015). An automated approach for classification of plant diseases towards development of futuristic Decision Support System in Indian perspective. 2015 International conference on advances in computing, communications and informatics (ICACCI), IEEE.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei (2009). Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition, Ieee.

Esgario, J. G., R. A. Krohling and J. A. Ventura (2020). "Deep learning for classification and severity estimation of coffee leaf biotic stress." Computers and Electronics in Agriculture **169**: 105162.

Ferentinos, K. P. (2018). "Deep learning models for plant disease detection and diagnosis." Computers and Electronics in Agriculture **145**: 311-318.

Fuentes, A., S. Yoon, S. C. Kim and D. S. Park (2017). "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition." Sensors **17**(9): 2022.

Haykin, S. (2009). Neural networks and learning machines, 3/E, Pearson Education India.

He, K., X. Zhang, S. Ren and J. Sun (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition.

He, K., X. Zhang, S. Ren and J. Sun (2016). Identity mappings in deep residual networks. European conference on computer vision, Springer.

Huang, G., Z. Liu, L. Van Der Maaten and K. Q. Weinberger (2017). Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition.

LeCun, Y., Y. Bengio and G. Hinton (2015). "Deep learning. nature, 521 (7553), 436-444." Google Scholar Google Scholar Cross Ref Cross Ref: 25.

Mohanty, S. P., D. P. Hughes and M. Salathé (2016). "Using deep learning for image-based plant disease detection." Frontiers in plant science **7**: 1419.

Newman, D. J. (1998). **UCI repository of machine learning database**. [Online]. Available : http://www. ics. uci. edu/~mlearn/MLRepository. html.

Nielsen, M. (2018). **Deep learning**. [Online]. Available : http://neuralnetworksanddeep learning.com/chap6.html.

Pan, Z., W. Yu, X. Yi, A. Khan, F. Yuan and Y. Zheng (2019). "Recent progress on generative adversarial networks (GANs): A survey." IEEE access **7**: 36322-36333.

Pawara, P., E. Okafor, O. Surinta, L. Schomaker and M. A. Wiering (2017). "Comparing Local Descriptors and Bags of Visual Words to Deep Convolutional Neural Networks for Plant Recognition." ICPRAM **479**(2017): 486.

Ramcharan, A., K. Baranowski, P. McCloskey, B. Ahmed, J. Legg and D. P. Hughes (2017). "Deep learning for image-based cassava disease detection." Frontiers in plant science **8**: 1852.

Rice Department, T. (2016). "Rice Knowledge Bank." [Online]. Available : https://www. ricethailand.go.th/rkb3/index.htm.

Sethy, P. K. (2020). "Rice Leaf Disease Image Samples." Mendeley Data.

Shrivastava, V. K., M. K. Pradhan, S. Minz and M. P. Thakur (2019). "Rice plant disease classification using transfer learning of deep convolution neural network." International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences.

Sladojevic, S., M. Arsenovic, A. Anderla, D. Culibrk and D. Stefanovic (2016). "Deep neural networks based recognition of plant diseases by leaf image classification." Computational intelligence and neuroscience **2016**.

Szegedy, C., S. Ioffe, V. Vanhoucke and A. A. Alemi (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. Thirty-first AAAI conference on artificial intelligence.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich (2015). Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna (2016). Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition.

Tan, C., F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu (2018). A survey on deep transfer learning. International conference on artificial neural networks, Springer.

TensorFlow. (2018). **Build a Convolutional Neural Network using Estimators**. [Online]. Available : https://www.tensorflow.org/tutorials/estimators/cnn#intro_to_convolutional_neural_networks.

ภาคผนวก

ภาคผนวก ก

Source Code สำหรับการประมวลผลโมเดล

```
# -*- coding: utf-8 -*-
"""Model.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1nxbDxiEAQ9Vcr2N1dQprR86n_i__0tdN
"""

import pandas as pd

from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout,
MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt

!pip install -U -q PyDrive
```

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

"""#Download Dataset"""

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

fid =
drive.ListFile({'q':"title='Dataset2fromRiceLeafDiseaseImageSamples.zip'"}).GetList()[0]['id
']
f = drive.CreateFile({'id': fid})
f.GetContentFile('Dataset2fromRiceLeafDiseaseImageSamples.zip')

f.keys()

PATH = '/content/Dataset2fromRiceLeafDiseaseImageSamples/Rice Leaf
Disease Images'

!unzip Dataset2fromRiceLeafDiseaseImageSamples.zip

os.listdir(PATH)

train_dir = os.path.join(PATH)

train_HL_dir = os.path.join(train_dir,'HL')
```

```
train_BD_dir = os.path.join(train_dir,'BD')
train_BLBD_dir = os.path.join(train_dir,'BLBD')
train_BSD_dir = os.path.join(train_dir,'BSD')
train_RTD_dir = os.path.join(train_dir,'RTD')
train_NBSD_dir = os.path.join(train_dir,'NBSD')


 S=[train_HL_dir
 ,train_BD_dir
 ,train_BLBD_dir
 ,train_BSD_dir
 ,train_RTD_dir
 ,train_NBSD_dir
  ]
 j=1
 Sum=0
 for i in S:
   Sum=Sum+len(os.listdir(i))
   print('t',j,':',len(os.listdir(i)))
    j=j+1

"""#Preparing the data"""

train_BD_dir

#https://machinelearningmastery.com/image-augmentation-deep-learning-
keras/

IMG_HEIGHT = 224
IMG_WIDTH = 224
```

```
batch_size = 256
epochs = 20

image_gen_train = ImageDataGenerator(
                    rescale=1./255
                    ,validation_split=0.2
                    )

train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                     subset="training",
                                                     class_mode='categorical')
train_x_scaled, train_images_y_encoded = next(train_data_gen)

validation_data_gen =
image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                     subset="validation",
                                                     class_mode='categorical')
val_x_scaled, val_images_y_encoded = next(validation_data_gen)

sample_training_images, _ = next(train_data_gen)

# # This function will plot images in the form of a grid with 1 row and 5
columns where images are placed in each column.
def plotImages(images_arr):
```

```
    fig, axes = plt.subplots(1, 5, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

plotImages(sample_training_images[:5])

"""#Training the model

#Classify ImageNet classes with Model
* https://keras.io/api/applications/
"""

# !pip install efficientnet

from keras.preprocessing import image

# import model
from tensorflow.keras.applications import Xception
from tensorflow.keras.applications import VGG16, VGG19
from tensorflow.keras.applications import ResNet50,ResNet101, ResNet152
from tensorflow.keras.applications import ResNet50V2,ResNet101V2,
ResNet152V2
from tensorflow.keras.applications import InceptionV3, InceptionResNetV2
from tensorflow.keras.applications import MobileNet, MobileNetV2
from tensorflow.keras.applications import DenseNet121, DenseNet169,
DenseNet201
```

```
        from tensorflow.keras.applications import NASNetLarge, NASNetMobile
        from tensorflow.keras.applications import EfficientNetB0, EfficientNetB2,
EfficientNetB3
        from tensorflow.keras.applications import EfficientNetB4, EfficientNetB5,
EfficientNetB6
        from tensorflow.keras.applications import EfficientNetB7,EfficientNetB1

        #Checkpoint
        from tensorflow.keras.callbacks import ModelCheckpoint

        from tensorflow.keras.models import Model
        from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Input
        from tensorflow.keras.optimizers import Adam
        import numpy as np
        import plotly.graph_objects as go
        from sklearn.model_selection import train_test_split

        def Train_Model():
          NameModel=[Xception,VGG16,VGG19,ResNet50,ResNet50V2,ResNet101,
            ResNet101V2,ResNet152,ResNet152V2,InceptionV3,InceptionResNetV2,
            MobileNet,MobileNetV2,DenseNet121,DenseNet169,DenseNet201,
            NASNetMobile,NASNetLarge ,EfficientNetB0 ,EfficientNetB1 ,
            EfficientNetB1 ,EfficientNetB2 ,EfficientNetB3 ,EfficientNetB4 ,
            EfficientNetB5,EfficientNetB6 ,EfficientNetB7]

TextModel=["Xception","VGG16","VGG19","ResNet50","ResNet50V2","ResNet101",

"ResNet101V2","ResNet152","ResNet152V2","InceptionV3","InceptionResNetV2",
            "MobileNet","MobileNetV2","DenseNet121","DenseNet169","DenseNet201",
            "NASNetMobile","NASNetLarge" ,"EfficientNetB0" ,"EfficientNetB1" ,
```

```
    "EfficientNetB1" ,"EfficientNetB2" ,"EfficientNetB3" ,"EfficientNetB4" ,
    "EfficientNetB5","EfficientNetB6" ,"EfficientNetB7"]

Text="\
  Please enter the model you want to run \n\
    1.) Xception please enter 1 \n\
    2.) VGG16 please enter 2 \n\
    3.) VGG19 please enter 3 \n\
    4.) ResNet50 please enter 4 \n\
    5.) ResNet50V2 please enter 5 \n\
    6.) ResNet101 please enter 6 \n\
    7.) ResNet101V2 please enter 7 \n\
    8.) ResNet152 please enter 8 \n\
    9.) ResNet152V2 please enter 9 \n\
    10.) InceptionV3 please enter 10 \n\
    11.) InceptionResNetV2 please enter 11 \n\
    12.) MobileNet please enter 12 \n\
    13.) MobileNetV2 please enter 13 \n\
    14.) DenseNet121 please enter 14 \n\
    15.) DenseNet169 please enter 15 \n\
    16.) DenseNet201 please enter 16 \n\
    17.) NASNetMobile please enter 17 \n\
    18.) NASNetLarge please enter 18 \n\
    19.) EfficientNetB0 please enter 19 \n\
    20.) EfficientNetB1 please enter 20 \n\
    21.) EfficientNetB1 please enter 21 \n\
    22.) EfficientNetB2 please enter 22 \n\
    23.) EfficientNetB3 please enter 23 \n\
    24.) EfficientNetB4 please enter 24 \n\
    25.) EfficientNetB5 please enter 25 \n\
```

```
    26.) EfficientNetB6 please enter 26 \n\
    27.) EfficientNetB7 please enter 27 \n\
"
choice = int(input(Text))

# while (choice not in(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
#                      19,20,21,22,23,24,25,26,27)):
#     print("Invalid input, please try again")
#     choice = input(Text)

# Checkpoint In the /output folder
filepath ='/content/drive/My Drive/Project_Pound/'+TextModel[choice-
1]+'.hdf5'
# Keep only a single checkpoint, the best over test accuracy.
checkpoint =ModelCheckpoint(filepath, monitor = 'val_loss', verbose = 1,
  save_weights_only=True,
  save_best_only=True,
  mode='max'
 )

#Model
num_classes = 4
base_model = NameModel[choice-1]

base_model = base_model(weights="imagenet", include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
predictions = Dense(num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

```
    for layer in base_model.layers:
        layer.trainable = False

    model.compile(loss='categorical_crossentropy',
                  optimizer=Adam(lr=0.0001),
                  metrics=['accuracy', tf.keras.metrics.TopKCategoricalAccuracy( k=5,
name='top_5_accuracy', dtype=None)])
    return model,checkpoint

model,checkpoint=Train_Model()

batch_size = 256
epochs=20

# model.summary()

history = model.fit(train_x_scaled, train_images_y_encoded,
batch_size=batch_size, epochs=epochs,
                    shuffle=True
                    ,validation_split=0.1
                    ,verbose=1
                    ,callbacks=[checkpoint],
                    )

import plotly.graph_objects as go
fig = go.Figure()

fig.add_trace(go.Scatter(x=history.epoch,
                         y=history.history['accuracy'],
                         mode='lines+markers',
```

```
                    name='Training accuracy'))
        fig.add_trace(go.Scatter(x=history.epoch,
                    y=history.history['val_accuracy'],
                    mode='lines+markers',
                    name='Validation accuracy'))
        fig.update_layout(title='Accuracy',
                    xaxis=dict(title='Epoch'),
                    yaxis=dict(title='Percentage'))
        fig.show()

        results = model.evaluate(val_x_scaled,
val_images_y_encoded,batch_size=256)
        print("test loss, test acc:", results)

        results_train = model.evaluate(train_x_scaled,
train_images_y_encoded,batch_size=256)
        print("train loss, train acc:", results_train)

"""# END"""
```

ภาคผนวก ข

Source Code สำหรับการเพิ่มจำนวนของภาพด้วยเทคนิค Image Data Augmentation Technique

DatasetA

```
# -*- coding: utf-8 -*-
"""Preprocess_DatasetA.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1Dfvh3koQMNgc4Tzs5xBTOpOoTpInZo-A
"""

import pandas as pd
import tensorflow as tf

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt

from google.colab import drive
```

```
drive.mount('/content/drive')

!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

"""#Download Dataset"""

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

classN = 'RTD'

fid = drive.ListFile({'q':"title='RTD.zip'"}).GetList()[0]['id']
f = drive.CreateFile({'id': fid})
f.GetContentFile(classN+'.zip')

f.keys()

!unzip RTD.zip

"""#Data Preparation"""

from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
```

```
IMG_HEIGHT = 224
IMG_WIDTH = 224
batch_size = 15000
epochs = 15

PATH = '/content/Data'+classN
train_dir = os.path.join(PATH)

train_dir

# load data
image_gen_train = ImageDataGenerator(
                    rescale=1./255
                    # ,validation_split=0.2
                    )
train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                     class_mode='categorical')
train_x_scaled, train_images_y_encoded = next(train_data_gen)

# define data preparation
datagen = ImageDataGenerator(
                    horizontal_flip=True,
                    height_shift_range=0.2,
                    zoom_range=0.2,
                    rotation_range=40,
                    width_shift_range=0.2,
                    vertical_flip=True,
```

```
                    )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_15')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_15',
            save_prefix=classN+'_x_1_15', save_format='png'):
  break
```

DatasetB

```
        # -*- coding: utf-8 -*-
        """Preprocess_DatasetB.ipynb

        Automatically generated by Colaboratory.

        Original file is located at

https://colab.research.google.com/drive/1nbo7e6DTV5kO8bGZQlBOl0rBXazT-sP8
        """

        import pandas as pd
        import tensorflow as tf

        from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout,
MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
import numpy as np
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

"""#Download Dataset"""

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

classN = 'show'

fid = drive.ListFile({'q':"title='show.zip'"}).GetList()[0]['id']
f = drive.CreateFile({'id': fid})
f.GetContentFile(classN+'.zip')
```

```
f.keys()

!unzip show.zip

"""#Data Preparation"""

from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot

IMG_HEIGHT = 224
IMG_WIDTH = 224
batch_size = 15000
epochs = 15

PATH = '/content/'+classN
train_dir = os.path.join(PATH)

train_dir

# load data
image_gen_train = ImageDataGenerator(
                    rescale=1./255
                    # ,validation_split=0.2
                    )
train_data_gen = image_gen_train.flow_from_directory(batch_size=batch_size,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_HEIGHT, IMG_WIDTH),
                                                     class_mode='categorical')
train_x_scaled, train_images_y_encoded = next(train_data_gen)
```

```
"""# **Preprocess**"""

# define data preparation
datagen = ImageDataGenerator(
                horizontal_flip=True,
                height_shift_range=0.2,
                rotation_range=40,
               )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_1')
for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
    save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_1',
    save_prefix=classN+'_x_1_1_', save_format='png'):
  break

# define data preparation
datagen = ImageDataGenerator(
                width_shift_range=0.2,
                vertical_flip=True,
                brightness_range=[0.3,1.5]
               )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
```

```
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_2')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_2',
             save_prefix=classN+'_x_1_2_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    height_shift_range=0.4,
                    width_shift_range=0.4,
                    zoom_range=0.2,
                   )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_3')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_3',
             save_prefix=classN+'_x_1_3_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    zoom_range=0.5,
```

```
                    rotation_range=60,
                    channel_shift_range=0.6,
                )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_4')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_4',
            save_prefix=classN+'_x_1_4_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    horizontal_flip=True,
                    vertical_flip=True,
                    brightness_range=[0.3,1.5],
                )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_5')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_5',
```

```
        save_prefix=classN+'_x_1_5_', save_format='png'):
    break

# define data preparation
datagen = ImageDataGenerator(
            rotation_range=180,
            brightness_range=[0.4,1.5],
            shear_range=20,
           )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6')
for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
    save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6',
    save_prefix=classN+'_x_1_6_', save_format='png'):
  break

# define data preparation
datagen = ImageDataGenerator(
            rotation_range=180,
            brightness_range=[0.4,1.5],
            shear_range=10,
           )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
```

```
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_A')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_A',
             save_prefix=classN+'_x_1_6_A_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    rotation_range=180,
                    brightness_range=[0.4,1.5],
                    shear_range=5,
                   )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_B')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_B',
             save_prefix=classN+'_x_1_6_B_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    rotation_range=180,
```

```
                    brightness_range=[0.4,1.5],
                    zoom_range=0.1,
                )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_C')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_6_C',
            save_prefix=classN+'_x_1_6_C_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    horizontal_flip=False,
                    channel_shift_range=0.4,
                    zoom_range=0.2,
                )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_7')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_7',
```

```
        save_prefix=className+'_x_1_7_', save_format='png'):
    break

# define data preparation
datagen = ImageDataGenerator(
            zoom_range=0.4,
            rotation_range=45,
            width_shift_range=0.4,
           )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+className+'_x_1_8')
for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
    save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+className+'_x_1_8',
    save_prefix=className+'_x_1_8_', save_format='png'):
  break

# define data preparation
datagen = ImageDataGenerator(
        zoom_range=0.4,
            brightness_range=[0.4,1.5],
            channel_shift_range=0.3,
           )
# fit parameters from data
datagen.fit(train_x_scaled)
# configure batch size and retrieve one batch of images
```

```
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_9')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_9',
             save_prefix=classN+'_x_1_9_', save_format='png'):
          break

        # define data preparation
        datagen = ImageDataGenerator(
                    width_shift_range=0.4,
                    channel_shift_range=0.2,
                    brightness_range=[0.2,1.8]
                   )
        # fit parameters from data
        datagen.fit(train_x_scaled)
        # configure batch size and retrieve one batch of images
        os.makedirs('/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_10')
        for X_batch, y_batch in datagen.flow(train_x_scaled,
train_images_y_encoded,batch_size=batch_size,
            save_to_dir='/content/drive/My
Drive/Project_Pound/Dataset/'+classN+'_x_1_10',
             save_prefix=classN+'_x_1_10_', save_format='png'):
  break
```

ประวัติผู้เขียน

ชื่อ	นายวิทวัส เหมหงษา
วัน เดือน ปี เกิด	27 สิงหาคม พ.ศ. 2537
ที่อยู่ปัจจุบัน	14 หมู่ 9 ตำบลบางไทรป่า อำเภอบางเลน จังหวัดนครปฐม 73130
ประวัติการศึกษา	วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ เกรดเฉลี่ย 3.15 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ทุนการศึกษาที่ได้รับ	ทุนอุดหนุนการศึกษา ปี 2560 คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ผลงานวิชาการ	Wittawat Hamhongsa, Rungrat Wiangsripanawan, and Pairat Thorncharoensri. "Rice Diseases Recognition Using Transfer Learning from Pre-trained CNN Model." Proceedings of the 19th International Conference on Computing and Information Technology (IC2IT 2023). Cham: Springer Nature Switzerland, 2023.