การแบ่งส่วนรูปภาพดอกไม้ด้วยการใช้ชาเลียนซีแมปร่วมกับการประยุกต์ใช้ปริภูมิสีเอชเอสวีและหน้ากากสี

น.ส.ธนณัฏฐ์ หงษ์ทอง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาสถิติ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

Flower Image Segmentation using Saliency Map with the Application of HSV Color Space and Color Mask

Miss Thananath Hongthong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Statistics
Department of Statistics
FACULTY OF COMMERCE AND ACCOUNTANCY
Chulalongkorn University
Academic Year 2021

หัวข้อวิทยานิพนธ์	การแบ่งส่วนรูปภาพดอกไม้ด้วยการใช้ซาเลียนซีแมปร่วมกับการประยุกต์ใช้ปริภูมิสีเอชเอสวีและหน้ากากสี
โดย	น.ส.ธนณัฏฐ์ หงษ์ทอง
สาขาวิชา	สถิติ
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	อาจารย์ ดร.สุรณพีร์ ภูมิวุฒิสาร

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะพาณิชยศาสตร์และการบัญชี

(รองศาสตราจารย์ ดร.วิเลิศ ภูริวัชร)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร.เสกสรร เกียรติสุไพบูลย์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(อาจารย์ ดร.สุรณพีร์ ภูมิวุฒิสาร)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วิฐรา พึ่งพาพงศ์)

..... กรรมการภายนอกมหาวิทยาลัย

(รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช)

ธนณัฏฐ์ หงษ์ทอง : การแบ่งส่วนรูปภาพดอกไม้ด้วยการใช้ซาเลียนซีแมปร่วมกับการประยุกต์ใช้ปริภูมิสีเอชเอสวีและหน้ากากสี. (Flower Image Segmentation using Saliency Map with the Application of HSV Color Space and Color Mask) อ. ที่ปรึกษาหลัก : อ. ดร.สุรณพีร์ ภูมิวุฒิสาร

การจำแนกประเภทรูปภาพดอกไม้เป็นสิ่งที่ท้าทาย เนื่องจากความคล้ายคลึงกันทางกายภาพของดอกไม้ เทคนิคการแบ่งส่วนรูปภาพ (Image segmentation) สามารถลดความซับซ้อนขององค์ประกอบภายในพื้นหลังภาพ ทำให้การจำแนกประเภทรูปภาพดอกไม้มีประสิทธิภาพมากขึ้น งานวิจัยชิ้นนี้ได้นำเสนอแนวคิดการแบ่งส่วนรูปภาพ โดยอิงการใช้ประโยชน์จากซาเลียนซีแมป (Saliency map) ในการเลือกบริเวณที่สนใจภายในภาพ และการใช้ปริภูมิสีเอชเอสวี (HSV) ผนวกกับการใช้หน้ากากสี (Color mask) ในการช่วยลดรายละเอียดที่ไม่สำคัญภายในพื้นหลังของรูปภาพ ผลการทดลองแสดงให้เห็นว่าวิธีการที่นำเสนอให้ผลลัพธ์การแบ่งส่วนรูปภาพโดยวัดจากค่าเฉลี่ย IoU เท่ากับ 54% (ซึ่งมากกว่างานวิจัยก่อนหน้า 13 %) ในขณะที่ค่าความถูกต้อง ความแม่นยำ ค่าความครบถ้วน และค่า F1 เมื่อจำแนกประเภทดอกไม้ด้วยแบบจำลอง VGG16 ที่ผ่านการปรับโครงสร้างเท่ากับ 87 %

สาขาวิชา สถิติ ลายมือชื่อนิสิต .....

ปีการศึกษา 2564 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6280122826 : MAJOR STATISTICS

KEYWORD: Saliency map, HSV, Color Mask

Thananath Hongthong : Flower Image Segmentation using Saliency Map with the Application of HSV Color Space and Color Mask. Advisor: SURONAPEE PHOOMVUTHISARN, Ph.D.

The classification of flowers is a challenging task, due to the similarity of flowers' physical characteristics. Image segmentation techniques can simplify such details within the image background, making it possible to classify flowers efficiently. In this paper, we propose a technique for image segmentation based on saliency map to select interested region within the image. The use of saliency map combining with the HSV color space with color mask can reduce insignificant details within the image background. Experimental results have shown that our method can select interested region and can reduce the background detail considerably. Our method can achieve 54% mean IoU (up to 13% higher than previous works), while achieving accuracy, precision, recall and F1 values at 87% when it integrates efficiency with the VGG-16 pre-trained model.

Field of Study: Statistics Student's Signature .....

Academic Year: 2021 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดีได้ความกรุณา และความกรุณาอนุเคราะห์ช่วยเหลืออย่างดีจากอ.ดร. สุรณพีร์ ภูมิวุฒิสาร อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้มอบความรู้ คำแนะนำและชี้ให้เห็นข้อบกพร่องที่เกิดขึ้นเพื่อนำไปทบทวนและแก้ไขปรับปรุงงาน ตลอดจนให้คำแนะนำเมื่อพบกับปัญหาต่างๆ ที่เกิดขึ้นระหว่างการดำเนินการวิจัย จนกระทั่งวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยจึงขอกราบพระคุณอาจารย์เป็นอย่างสูง ณ ที่นี้

ขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. เสกสรรค์ เกียรติไพบูลย์ ประธานการสอบวิทยานิพนธ์ และ ผู้ช่วยศาสตราจารย์ ดร. วิฐรา พึ่งพาพงศ์ กรรมการสอบวิทยานิพนธ์ และ รองศาสตราจารย์ ดร. วรพจน์ กรีสุระเดช กรรมการผู้ทรงคุณวุฒิภายนอก ที่ได้สละเวลาเพื่อให้คำแนะนำและตรวจสอบข้อบกพร่องที่เกิดขึ้นภายในงาน เพื่อให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์มากยิ่งขึ้น อีกทั้งคณาจารย์ประจำ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัยทุกท่านที่ได้ถ่ายถอดความรู้ ทั้งทางด้านสถิติและการนำความรู้ที่ได้รับไปประยุกต์ใช้ในสายงานที่สนใจให้แก่ผู้วิจัยจนสำเร็จลุล่วงไปได้ด้วยดี

ธนณัฏฐ์ หงษ์ทอง

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของงานวิจัย

การจำแนกรูปภาพดอกไม้เป็นสิ่งที่ท้าทาย เนื่องจากดอกไม้บางชนิดมีความคล้ายคลึงกันมาก อีกทั้งดอกไม้บางชนิดแม้จะมีสายพันธุ์ต่างกันแต่ก็มีลักษณะทางกายภาพที่คล้ายคลึงกันทั้ง สี รูปร่าง และลักษณะภายนอก มากกว่านั้นภายในรูปภาพมักจะมีองค์ประกอบอื่นรวมอยู่ด้วย เช่น ใบไม้ หญ้า ท้องฟ้า ฯลฯ ซึ่งโดยทั่วไปการจำแนกพืชมักจะทำโดยผู้เชี่ยวชาญด้านพฤกษศาสตร์ โดยจะทำการวัดคุณลักษณะของพืชด้วยตนเอง แต่ข้อเสียของวิธีการนี้คือค่อนข้างใช้เวลานาน ยิ่งไปกว่านั้นถ้าหากรูปภาพที่ใช้มีจำนวนมาก อาจเกิดข้อผิดพลาดได้เนื่องจากพื้นหลังของรูปภาพมีความซับซ้อน เพื่อลดความซับซ้อนของภาพ เทคนิคการแบ่งส่วนรูปภาพสามารถแบ่งรูปภาพออกเป็นพื้นที่ที่มีรายละเอียดต่างกันได้อย่างชัดเจน ทำให้สามารถเลือกพื้นที่ที่สนใจภายในภาพสำหรับนำไปจำแนกประเภทดอกไม้ได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

ในปัจจุบันได้มีวิธีการมากมาย (X. Wang et al., 2017) (Ren et al., 2016) ที่ใช้ในการระบุบริเวณของวัตถุที่สนใจภายในภาพ ก่อนจะนำบริเวณนั้นไปจำแนกประเภทของบริเวณนั้นภายในภาพ ซึ่งวิธีการนี้จะแบ่งกระบวนการทำงานออกเป็น 3 ขั้นตอน คือ การเสนอพื้นที่ภายในภาพที่อาจมีวัตถุที่สนใจ จากนั้นจึงทำการสกัดคุณลักษณะของบริเวณนั้นด้วยแบบจำลอง CNN แล้วจึงค่อยจำแนกประเภทของของวัตถุบริเวณนั้น ซึ่งวิธีการนี้ทำให้การจำแนกประเภทรูปภาพจำนวนมากมีความสะดวกและรวดเร็วมากยิ่งขึ้น เพราะเป็นการรวมทั้งขั้นตอนการระบุบริเวณที่สนใจ และขั้นตอนการจำแนกประเภทเข้าด้วยกัน ในขณะที่การใช้วิธีการนี้ในการทำงานกับชุดข้อมูลรูปภาพขนาดเล็กนั้นให้ผลลัพธ์ที่ดีเช่นกัน แต่วิธีการนี้อาจขาดความยืดหยุ่นในกระบวนการทำงาน เนื่องจากกระบวนการทำงานของระบบที่มีความต่อเนื่องกัน ทำให้เมื่อต้องการปรับปรุงเพียงขั้นตอนการระบุบริเวณที่สนใจภายในภาพนั้นทำได้ยาก ด้วยเหตุนี้จึงมีงานวิจัย (Vu et al., 2019) (Pan et al., 2014) ที่ทำการแบ่งส่วนรูปภาพก่อนเพื่อระบุบริเวณที่สนใจ ก่อนจะนำบริเวณนั้นไปจำแนกประเภทวัตถุ เพื่อช่วยให้สามารถทำการเตรียมข้อมูลรูปภาพด้วยวิธีการที่หลากหลายและเหมาะสมกับลักษณะข้อมูลที่ใช้ได้มากขึ้น

โดยทั่วไปแนวคิดเกี่ยวกับการแบ่งส่วนรูปภาพจะประกอบไปด้วยวิธีพื้นฐาน 3 แบบ ได้แก่ (1) การปรับค่าเทรดโฮลด์ (Threshold) (2) การสร้างโครงร่าง (Structure) ของดอกไม้เพื่อเก็บบริเวณที่สนใจ และ (3) การเปลี่ยนปริภูมิสี (Color space) ของรูปภาพ ซึ่งการเลือกใช้เพียงวิธีการใดวิธีการ

หนึ่งในการแยกพื้นหน้า (Foreground) และพื้นหลัง (Background) ออกจากกันจะทำให้แบ่งส่วนรูปภาพได้ไม่ดีนัก ถ้าหากพื้นหลังของภาพมีความซับซ้อนมาก จะทำให้แยกบริเวณที่สนใจออกจากพื้นหลังรูปภาพได้ยากมากยิ่งขึ้น ด้วยเหตุนี้จึงมีงานวิจัยที่นำเอาข้อดีของวิธีการทั้ง 3 วิธีมาผนวกกัน ด้วยการนำข้อดีของการใช้การระบุตำแหน่งวัตถุมาใช้ระบุตำแหน่งสำคัญของวัตถุภายในภาพ จากนั้นทำการลบภาพพื้นหลังที่ไม่เกี่ยวข้องด้วยการปรับความเข้มของสีภายในภาพ (Panwar et al., 2016) หรือการเปลี่ยนปริภูมิสีของภาพ (Chen et al., 2008) ซึ่งงานวิจัยเหล่านี้แสดงให้เห็นว่าการที่จะระบุตำแหน่งของวัตถุให้แม่นยำจำเป็นต้องมีโครงร่างของดอกไม้ที่ชัดเจน จึงมีงานวิจัยที่เสนอวิธีการหาโครงร่างของดอกไม้เพื่อหาบริเวณที่สนใจในรูปภาพ ซึ่งมีทั้งการใช้ K-means clustering (Sabri et al., 2016) และการใช้ Saliency map โดยเฉพาะการใช้ Saliency map เป็นสิ่งที่น่าสนใจเป็นอย่างมาก เนื่องจาก Saliency map สามารถแสดงรายละเอียดของบริเวณที่สนใจภายในรูปภาพได้ค่อนข้างชัดเจน อีกทั้งยังช่วยลดรายละเอียดที่ไม่สำคัญภายในภาพลงได้ (Liu et al., 2016) แต่ถึงอย่างไรการใช้เพียง Saliency map เพียงอย่างเดียวยังไม่สามารถแบ่งรูปภาพได้ดีเท่าที่ควร เนื่องจากภายในรูปภาพที่ถูกครอบตัดจะยังคงเหลือบริเวณที่ไม่เกี่ยวข้องหลงเหลืออยู่ค่อนข้างมาก หากต้องการครอบตัดรูปภาพให้เหลือเพียงบริเวณที่สนใจเพื่อนำภาพไปใช้ในการจำแนกประเภทของดอกไม้จำเป็นที่จะต้องลดรายละเอียดพื้นหลังของภาพให้น้อยลง

เพื่อที่จะแก้ไขปัญหานี้ ในงานวิจัยชิ้นนี้จึงได้นำเสนอแนวคิดการแบ่งส่วนรูปภาพ โดยอิงการใช้ประโยชน์จาก Saliency map ในการเก็บรายละเอียดของบริเวณที่สนใจภายในภาพ และการใช้ประโยชน์จากปริภูมิสี HSV ผนวกกับการประยุกต์ใช้ Color mask ในการช่วยลดรายละเอียดที่ไม่สำคัญภายในพื้นหลังของรูปภาพออกไป โดยงานวิจัยชิ้นนี้จะแบ่งออกเป็น 2 ขั้นตอน ขั้นตอนแรกจะเป็นขั้นตอนการแบ่งส่วนรูปภาพของดอกไม้ และในขั้นตอนที่สองจะเป็นการนำรูปภาพที่ได้มาจากขั้นตอนแรกมาใช้จำแนกประเภทของดอกไม้

1.2 วัตถุประสงค์ของงานวิจัย

งานวิจัยนี้มีเป้าหมายเพื่อศึกษาวิธีการแบ่งส่วนรูปภาพสำหรับนำไปใช้ในการจำแนกประเภทของดอกไม้ จึงได้กำหนดวัตถุประสงค์ในงานวิจัย ดังนี้

1. เพื่อศึกษาและนำเสนอแนวคิดในการแบ่งส่วนรูปภาพ
2. เพื่อศึกษาวิธีการที่เกี่ยวข้องกับการแบ่งส่วนรูปภาพ เพื่อนำวิธีการที่เหมาะสมมาทำการประยุกต์ใช้แบ่งส่วนรูปภาพให้ดียิ่งขึ้น

3. เพื่อเปรียบเทียบประสิทธิภาพวิธีการที่นำเสนอในงานวิจัยก่อนหน้า และวิธีการที่นำเสนอในงานวิจัยนี้ โดยเปรียบเทียบทั้งประสิทธิภาพในการแบ่งส่วนรูปภาพ และประสิทธิภาพในการจำแนกประเภทดอกไม้

4. เพื่อแสดงให้เห็นประโยชน์ของการเตรียมข้อมูล (Preprocessing) ด้วยการแบ่งส่วนรูปภาพก่อนนำไปจำแนกประเภทดอกไม้

1.3 ขอบเขตของงานวิจัย

1. ขอบเขตด้านเนื้อหา

เนื่องจากในงานวิจัยก่อนหน้าชุดข้อมูลรูปภาพจำนวน 1 ชุดชื่อว่า Oxford 17 ซึ่งเป็นชุดข้อมูลรูปภาพที่มีความซับซ้อนของพื้นหลังรูปภาพน้อยจนถึงปานกลาง และดอกไม้ยังเป็นบริเวณที่ใหญ่ที่สุดภายในภาพ แต่ในงานวิจัยชิ้นนี้ศึกษารูปภาพดอกไม้ที่มีความหลากหลายทางกายภาพ เช่น สี ลักษณะของกลีบดอก ฯลฯ โดยรูปภาพที่ใช้มีความซับซ้อนทั้งลักษณะของดอกไม้ที่ไม่ได้มีเพียงดอกเดียวในภาพ แต่ยังรวมถึงดอกไม้ที่มีลักษณะอยู่รวมกันเป็นกลุ่ม เช่น ช่อ หรือกอ อีกทั้งพื้นหลังของรูปภาพยังมีองค์ประกอบอื่นรวมอยู่ด้วย เช่น ต้น ไม้ ใบหญ้า ท้องฟ้า คน ฯลฯ เพื่อให้รูปภาพมีความสอดคล้องกับรูปภาพโดยทั่วไปมากที่สุด

2. ขอบเขตข้อมูล

งานวิจัยชิ้นนี้เลือกใช้ชุดข้อมูลจำนวน 1 ชุดชื่อว่า Flowers ซึ่งเป็นชุดข้อมูลรูปภาพดอกไม้จาก Kaggle ประกอบไปด้วยรูปภาพจำนวน 4,931 รูป ชุดข้อมูลประกอบไปด้วยรูปภาพดอกไม้ 5 สายพันธุ์ ได้แก่ ดอกเดซี่จำนวน 944 รูป ดอกเดนดิไลอ่อนจำนวน 1,101 รูป ดอกกุหลาบจำนวน 955 รูป ดอกทานตะวันจำนวน 950 รูป และดอกทิวลิปจำนวน 984 รูป

3. ขอบเขตวิธีการวัดประสิทธิภาพ

วิธีการวัดประสิทธิภาพจะประกอบไปด้วยการวัดประสิทธิภาพของการแบ่งส่วนรูปภาพด้วยวิธี Intersection over Union (IoU) และการวัดประสิทธิภาพการจำแนกประเภทดอกไม้โดยการวัดค่าความถูกต้อง (Accuracy) ความแม่นยำ (Precision) ความครบถ้วน (Recall) และ F1

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้วิธีการสำหรับแบ่งส่วนรูปภาพที่สามารถช่วยลดรายละเอียดของพื้นหลังรูปภาพ ที่มีความซับซ้อนระดับปานกลางจนถึงมากได้ดียิ่งขึ้น สำหรับครอบตัดรูปภาพให้เหลือเพียงบริเวณที่สนใจเพื่อนำภาพไปใช้ในการจำแนกประเภทของดอกไม้

2. การเตรียมข้อมูลด้วยการแบ่งส่วนรูปภาพด้วยวิธีการที่นำเสนอสามารถช่วยให้แบบจำลองสามารถจำแนกประเภทดอกไม้ได้ดีมากยิ่งขึ้น

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ทฤษฎีและงานวิจัยที่เกี่ยวข้องประกอบด้วยความรู้พื้นฐานเกี่ยวกับการประมวลผลภาพดิจิตอล (Digital Image Processing) ทฤษฎีพื้นฐานเกี่ยวกับการแบ่งส่วนรูปภาพ (Image Segmentation) การจำแนกประเภทด้วยการใช้แบบจำลองโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) และวิธีการประเมินประสิทธิภาพของการแบ่งส่วนรูปภาพและการจำแนกประเภท ในส่วนงานวิจัยที่เกี่ยวข้อง จะเป็นการทบบวนวรรณกรรมเกี่ยวกับการแบ่งส่วนรูปภาพ ซึ่งจะแสดงรายละเอียดดังนี้

2.1 ความรู้พื้นฐานเกี่บวกับการประมวลผลภาพดิจิตอล (Digital Image Processing)

เนื่องจากงานวิจัยชิ้นนี้มีการประมวลผลบนรูปภาพดิจิตอล ดังนั้นการศึกษาความรู้พื้นฐานเกี่ยวกับการประมวลผลภาพดิจิตอลจึงเป็นสิ่งสำคัญ ซึ่งประกอบไปด้วย รูปภาพดิจิตอล (Digital Image) ภาพระดับสีเทา (Grayscale Image) และ ภาพสี (Color Image) โดยจะแสดงรายละเอียดแยกตามหัวข้อ ดังนี้

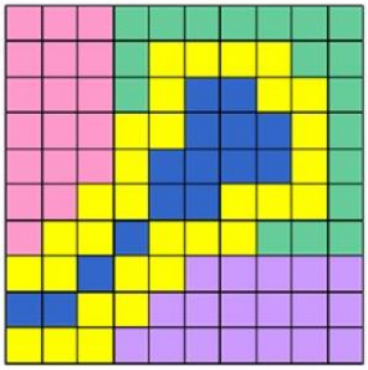
2.1.1 รูปภาพดิจิตอล (Digital Image)

ภาพดิจิทัลคือ การแสดงภาพจริงเป็นชุดตัวเลขที่คอมพิวเตอร์ดิจิทัลสามารถจัดเก็บและจัดการได้ เพื่อที่จะแปลงรูปภาพเป็นตัวเลข จึงได้มีการแบ่งพื้นที่ออกเป็นส่วนย่อย ๆ เรียกว่า พิกเซล (Pixel) โดยตัวเลขที่แสดงในแต่ละพิกเซลจะอธิบายคุณสมบัติบางประการของพิกเซล เช่น ความสว่าง ความเข้ม หรือเฉดสี โดยตัวเลขจะถูกแปลงออกเป็นอาร์เรย์ (Array) ตัวเลข ซึ่งมีการจัดเรียงเป็นแถวและคอลัมน์ที่สอดคล้องกับตำแหน่งแนวตั้งและแนวนอนของพิกเซลในภาพ โดยภาพที่ใช้ในระบบคอมพิวเตอร์ส่วนใหญ่จะแบ่งออกเป็น 2 ประเภท ได้แก่ รูปภาพบิตแมป (Bitmap Image) และรูปภาพแบบเวกเตอร์ (Vector Image) (Linda & George 2001)

1. ภาพแบบบิตแมป (Bitmap Image)

ภาพบิตแมป หรือ ภาพแบบราสเตอร์ (Raster image) จะมีลักษณะเป็นตารางสี่เหลี่ยมผืนผ้าที่ในแต่ละพิกเซลจะประกอบไปด้วยค่าสี โดยภาพบิตแมปจะมีช่องตารางเป็นจำนวนมากทำให้สายตาของมนุษย์ไม่สามารถมองเห็นรายละเอียดของช่องสีภายในภาพได้ ด้วยเหตุนี้

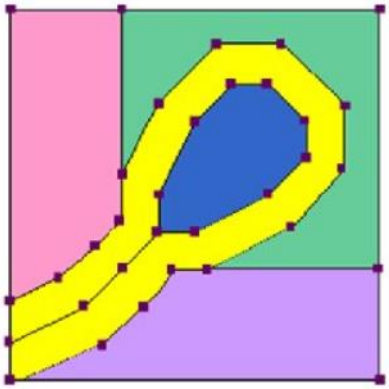
จึงทำให้ไม่สามารถตอบได้ว่าในแต่ละพิกเซลมีค่าสีเป็นอะไร โดยภาพบิตแมปนิยมใช้ในงานภาพถ่าย เนื่องจากภาพลักษณะนี้สามารถแสดงโทนสีของแสงและเงาได้สมจริงมากที่สุด (อรวรรณ, 2559)



ภาพที่ 1 แสดงลักษณะตัวอย่างภาพแบบบิตแมป
(ที่มา : Visualization of Structural Shape Information based on Octree using Terrestrial Laser Scanning, 2016 : 11)

2. ภาพแบบเวกเตอร์ (Vector Image)

ภาพเวกเตอร์เป็นภาพที่สามารถคงรายละเอียดของภาพไว้ได้ แม้ภาพจะถูกขยายให้มีขนาดใหญ่มากเพียงใด ซึ่งคุณสมบัติของภาพชนิดนี้จะมีความแตกต่างจากภาพแบบบิตแมปเป็นอย่างมากที่ไม่สามารถขยายให้ใหญ่ได้ เนื่องจากภาพแบบเวกเตอร์ เป็นไฟล์ที่ประกอบไปด้วยจุดและเส้นที่สร้างขึ้นมาจากรูปทรงที่ถูกกำหนดโดยสมการคณิตศาสตร์ที่มองเห็นได้โดยตรงจากรูปทรงเรขาคณิตที่กำหนดไว้บนระนาบคาร์ทีเซียน เช่น จุด เส้น เส้นโค้ง และรูปหลายเหลี่ยม (อรวรรณ, 2559)



ภาพที่ 2 แสดงลักษณะตัวอย่างภาพแบบเวกเตอร์

(ที่มา : Visualization of Structural Shape Information based on Octree using Terrestrial Laser Scanning, 2016 : 11)

2.1.2 ประเภทของภาพ (Image Types)

ในงานวิจัยชิ้นนี้ได้มีการศึกษากระบวนการแบ่งรูปภาพซึ่งมีการดำเนินงานบนรูปภาพหลายแบบ ดังนั้นความรู้เกี่ยวกับประแภทของภาพจะช่วยให้เข้าใจกระบวนการแบ่งส่วนรูปภาพได้ดียิ่งขึ้น ซึ่งจะแสดงรายละเอียดดังนี้

1. ภาพเฉดสีเทา (Gray Scale Image)

ภาพเฉดสีเทา คือ อาเรย์ 2 มิติที่แต่ละพิกเซลในภาพจะถูกกำหนดตัวเลข 1 ค่า ซึ่งเป็นค่าที่แสดงความเข้มของจุดนั้น เนื่องจากช่วงของค่าพิกเซลจะถูกกำหนดด้วยความละเอียดของบิตของรูปภาพ และรูปภาพดังกล่าวจะถูกจัดเก็บเป็นรูปภาพจำนวนเต็ม N บิตตามรูปแบบที่กำหนด (Chris & Toby, 2011)

ภาพที่ 3 แสดงตัวอย่างลักษณะภาพเฉดสีเทา

2. ภาพไบนารี (Binary image)

ภาพไบนารี่ คือ อาร์เรย์ 2 มิติที่กำหนดให้ทุกพิกเซลในภาพมีค่าตัวเลขเป็น 0 หรือ 1 ในบางครั้งภาพในลักษณะนี้จะถูกเรียกว่าเป็นภาพตรรกะ (Logical Image) โดยสีดำมีค่าเป็น 0 ใช้แทนพื้นหลังของภาพ และบริเวณที่เป็นสีขาวจะมีค่าเป็น 1 ใช้แทนพื้นหน้าของภาพ ภายในภาพจะไม่มีองค์ประกอบอื่นปรากฏ (Chris & Toby, 2011)

ภาพที่ 4 แสดงตัวอย่างลักษณะภาพไบนารี

3. ภาพสี (Color Image)

ภาพสีประกอบด้วยพิกเซลซึ่งแต่ละภาพมีตัวเลขสามตัวที่แสดงถึงระดับสีแดง สีเขียว และสีน้ำเงินของรูปภาพ ณ ตำแหน่งใดตำแหน่งหนึ่ง สีแดง สีเขียว และสีน้ำเงิน (บางครั้งหมายถึง RGB) เป็นสีหลักที่ใช้ในการผสม เมื่อนำสีเหล่านี้มาผสมกันจะได้เฉดสีเพิ่มขึ้นมา เช่น ฟ้า ม่วง แดง และเหลือง โดยสีใหม่ที่ได้จะเกิดจากการผสมปริมาณสีแดง สีเขียว และสีน้ำเงินในปริมาณที่เหมาะสม ซึ่งระดับเฉดสีจะมีทั้งหมด 256 ระดับ (Chris & Toby, 2011)

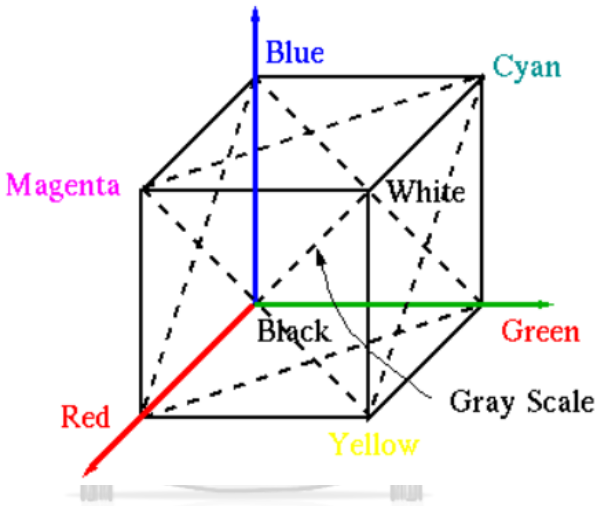
ภาพที่ 5 แสดงตัวอย่างลักษณะภาพสี

2.1.3 ปริภูมิสี (Color Spaces)

เนื่องจากงานวิจัยชิ้นนี้มีการประยุกต์ใช้ประโยชน์ของปริภูมิสีในการลดรายละเอียดของพื้นหลังรูปภาพ ดังนั้นการศึกษาเกี่ยวกับปริภูมิสีจึงเป็นสิ่งสำคัญ ซึ่งความรู้เรื่องปริภูมิสีที่เกี่ยวข้องกับงานวิจัยชิ้นนี้ประกอบไปด้วย ปริภูมิสี RGB ปริภูมิสี HSV และปริภูมิสี CIELAB โดยแสดงตามหัวข้อ ดังนี้

1. ปริภูมิสี RGB

จอภาพคอมพิวเตอร์ส่วนใหญ่ทำงานโดยการระบุสีตามองค์ประกอบสีแดง สีเขียว และสีน้ำเงิน โดยค่าทั้ง 3 จะระบุปริภูมิสี 3 มิติเรียกว่าปริภูมิสี RGB ซึ่งปริภูมิสี RGB จะถูกแสดงอยู่ในรูปลูกบาศก์โดยสีแดงจะแปรผันตามแกนที่หนึ่ง สีเขียวจะแปรผันตามแกนที่สอง และสีน้ำเงินจะแปรผันตามแกนที่สาม ทุกสีสามารถนำมาผสมกันได้โดยสีที่ผสมจะอยู่ภายในลูกบาศก์ ดังแสดงตามภาพที่ 6 ซึ่งแสดงให้เห็นทิศทางของลูกบาศก์ RGB ที่ต่างกัน มุมทั้ง 8 ของลูกบาศก์จะแทนสีหลักทั้ง 3 สี (สีแดง, สีเขียว, สีน้ำเงิน) สามสีรอง (สีฟ้า, สีม่วงแดง, สีเหลือง) สีดำ และสีขาว สีเทาเฉดต่าง ๆ จะอยู่บนเส้นทแยงมุมของลูกบาศก์ซึ่งเชื่อมกับจุดสีดำและสีขาว (Sachs, 1996-1999)



ภาพที่ 6 แสดงแบบจำลองของปริภูมิสี RGB

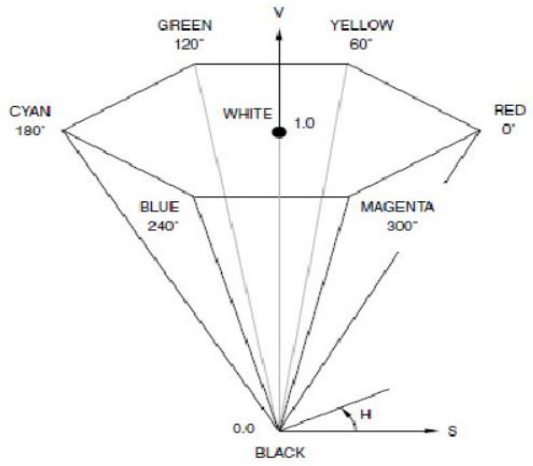
(ที่มา : https://biology.stackexchange.com/questions/72984/trying-to-understand-reduction-of-color-dimension-in-colorblind-case)

2. ปริภูมิสี HSV

ปริภูมิสี HSV เป็นปริภูมิสีทางเลือกที่ได้รับความนิยมสำหรับนำไปประยุกต์ใช้ในการวิเคราะห์รูปภาพเป็นอย่างมาก เนื่องจากการเปลี่ยนแปลงภายในปริภูมิสีนี้เป็นไปตามการไล่ระดับเกรเดียนท์สี (Color Gradient) ด้วยระดับความสว่างที่มากขึ้นจึงช่วยให้แยกสีออกจากกันได้ ด้วยเหตุนี้จึงทำให้ปริภูมิสี HSV สามารถแสดงสีของรูปภาพจริงได้ตรงกับธรรมชาติการมองเห็นของมนุษย์ได้มากกว่าปริภูมิสี RGB (Chris & Toby, 2011) โดยพารามิเตอร์ที่เกี่ยวข้องสามารถอธิบายได้ ดังนี้

- H (Hue) ค่าสี หมายถึง ความยาวคลื่นที่โดดเด่นของสี เช่น สีแดง สีน้ำเงิน และสีเขียว
- S (Saturation) ค่าความอิ่มสี หมายถึง ความบริสุทธิ์ของสี (ปริมาณของแสงสีขาวที่รวมอยู่ในสี)
- V (Value) ค่าความสว่าง หมายถึง ความสว่างของสี

ปริภูมิสี HSV จะกำหนดคุณลักษณะของสีตามค่าสี (Hue) ค่าความอิ่มตัว (Saturation) และค่าความสว่าง (Value) ปริภูมิสีนี้อิงจากโมเดลหกเหลี่ยม (Hexacone Model) ซึ่งสามารถมองเห็นเป็นปริซึมที่มีรูปหกเหลี่ยมที่ปลายด้านหนึ่งที่เรียวลงไปที่จุดเดียวที่ปลายอีกด้านหนึ่ง โดยหน้าหกเหลี่ยมของปริซึมได้มาจากการดูที่ลูกบาศก์ RGB ที่มีศูนย์กลางอยู่ที่มุมสีขาว ลูกบาศก์จะดูเหมือนหกเหลี่ยมที่มีจุดศูนย์กลางเป็นสีขาว ซึ่งมีสีหลักและสีรองรวมกันเป็นหกจุดยอดของหกเหลี่ยม จากภาพที่ 7 แสดงให้เห็นความสว่างที่สุดเท่าที่จะเป็นไปได้ของสีทั้งหมดตามสีและความอิ่มตัวของสี ภาพตัดขวางที่ต่อเนื่องกันของรูปหกเหลี่ยม HSV ที่แคบลงจนถึงจุดยอดนั้นแสดงให้เห็นว่าสีต่าง ๆ จะเริ่มเข้มขึ้นและเข้มขึ้นจนกลายเป็นสีดำในที่สุด (Sachs, 1996-1999)



ภาพที่ 7 แสดงแบบจำลองของปริภูมิสี HSV

(ที่มา : A New No-reference Method for Color Image Quality Assessment, 2012 : 26)

การแสดงภาพ HSV ของภาพ 2 มิติยังเป็นอาร์เรย์ 3 มิติที่ประกอบด้วยสามช่องคือ (h, s, v) และแต่ละตำแหน่งพิกเซลภายในภาพ $I(m, n)$ ประกอบด้วย (h, s, v) ที่สามารภแปลงกลับมาเป็นสี RGB เมื่อต้องการแสดงภาพจริง (Chris & Toby, 2011)

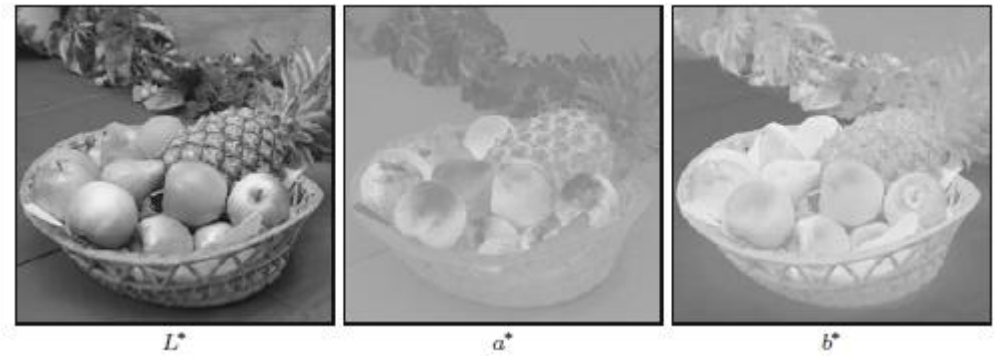
จากภาพที่ 8 แสดงให้เห็นว่าในแต่ละช่องสีของภาพในปริภูมิสี HSV สามารถช่วยให้มองห็นวัตถุที่อยู่ในรูปภาพได้ดีกว่าการแสดงด้วยปริภูมิสี RGB แม้จะมีสภาพแสงที่แตกต่างกัน

H S V

ภาพที่ 8 แสดงตัวอย่างรูปภาพแสดงองค์ประกอบของปริภูมิสี HSV

3. ปริภูมิสี CIELAB

ปริภูมิสี CIELAB หรือที่เรียกกันโดยทั่วไปว่าปริภูมิสี Lab เป็นปริภูมิสีที่ประกอบไปด้วยองค์ประกอบทั้งหมด 3 ค่า ได้แก่ L^*ความสว่าง (Luminosity) และอีก 2 องค์ประกอบได้แก่ a^* และ b^* ซึ่งใช้ในการระบุค่าความเข้ม และความอิ่มตัวของสีเขียว สีแดง สีน้ำเงิน และสีเหลือง โดยองค์ประกอบทั้ง 3 มีค่าความสัมพันธ์ซึ่งจะอ้างถึงจุดอ้างอิงสีขาว คือ $C_{ref} = (X_{ref}, Y_{ref}, Z_{ref})$ (Wilhelm & Mark J., 2016)



ภาพที่ 9 แสดงตัวอย่างรูปภาพแสดงองค์ประกอบของปริภูมิสี Lab

(ที่มา : Digital Image An Algorithmic Introduction, 2016 : 347)

2.1.4 การแปลงปริภูมิสี (Color Conversion)

เนื่องจากในงานวิจัยชิ้นนี้มีการใช้ประโยชน์ของการเปลี่ยนปริภูมิสี เพื่อช่วยในการแสดงรายละเอียดของบริเวณที่สนใจให้ชัดยิ่งขึ้น และลดรายละเอียดที่ไม่เกี่ยวข้องภายในรูปภาพให้น้อยลง ดังนั้นความรู้เกี่ยวกับการแปลงปริภูมิสีจึงเป็นสิ่งสำคัญ เพื่อช่วยให้เข้าใจขั้นตอนของกระบวนการทำงานได้ดีมากขึ้น ในที่นี้จะนำเสนอการแปลงปริภูมิสีที่มีความเกี่ยวข้องกับงานวิจัย ซึ่งจะแสดงรายละเอียดเรียงตามหัวข้อ ดังนี้

1. การแปลงปริภูมิสีจาก RGB เป็นภาพเฉดสีเทา

การแปลงภาพสี RGB เป็นภาพระดับสีเทาดำเนินการโดยการคำนวณค่าสีเทาหรือความสว่างเทียบเท่า Y สำหรับแต่ละ RGB โดย Y สามารถคำนวณจากค่าเฉลี่ยขององค์ประกอบสีทั้งสามของ RGB (Wilhelm & Mark J. , 2016) ดังสมการที่ (1)

$$Y = AVg(R, G, B) = \frac{R+G+B}{3} \tag{1}$$

เนื่องจากสายตามนุษย์จะมองเห็นสีแดงและสีเขียวสว่างกว่าสีน้ำเงิน จึงทำให้ภายในรูปภาพบริเวณที่เป็นสีเขียวและสีแดงจะมืดกว่าบริเวณที่เป็นสีน้ำเงิน ดังนั้นค่าผลรวมน้ำหนักขององค์ประกอบสีที่นำไปใช้คำนวณสามารถคำนวณได้จากค่าความสว่าง (Luminance) ดังแสดงในสมการที่ (2)

$$Y = Lum(R, G, B) = w_R \cdot R + w_G \cdot G + w_B \cdot B \tag{2}$$

โดยที่ $w_R = 0.229$ $w_G = 0.587$ และ $w_B = 0.114$

2. การแปลงปริภูมิสีจาก RGB เป็น HSV

การแปลงปริภูมิสีจาก RGB เป็น HSV เริ่มจากสมมติ R, G และ B เป็น 3 ช่องของพิกเซล และมีค่าอยู่ในช่วง [0, 255] จากนั้นทำการหารด้วย 255 เพื่อให้มีค่าอยู่ในช่วง [0, 1] (Deswal & Sharma, 2014) ดังแสดงในสมการที่ (3) ถึง (11)

จะได้ว่า $$R' = {}^{R}/_{255} \tag{3}$$

$$G' = {}^{G}/_{255} \tag{4}$$

$$B' = {}^{B}/_{255} \tag{5}$$

โดยที่ $$Cmax = \max(\mathrm{R}', \mathrm{G}', \mathrm{B}') \tag{6}$$

$$Cmin = \min(R', G, B') \tag{7}$$

และ $$\Delta = Cmax - Cmin \tag{8}$$

จะได้ ค่าสี (Hue) คือ

$$H = \begin{cases} 60° \times \left(\frac{G'-B'}{\Delta} mod 6\right), & Cmax = R' \\ 60° \times \left(\frac{B'-R'}{\Delta} + 2\right), & Cmax = G' \\ 60° \times \left(\frac{R'-G'}{\Delta} + 4\right), & Cmax = B' \end{cases} \tag{9}$$

ค่าความอิ่มของสี (Saturation) คือ

$$S = \begin{cases} 0, & Cmax = 0 \\ \frac{\Delta}{Cmax}, & Cmax \neq 0 \end{cases} \tag{10}$$

และ ค่าความสว่าของแสง (Value)

$$V = Cmax \tag{11}$$

3. การแปลงปริภูมิสีจาก CIELAB เป็นปริภูมิสี RGB

เริ่มจากการแปลงปริภูมิสี CIELAB เป็นพิกัด CIEXYZ ซึ่งแสดงรายละอียดขั้นตอนดังสมการที่ (12) ถึง (14)

จะได้ว่า $$X = X_{ref} \cdot f_2\left(L' + \frac{a^*}{500}\right) \tag{12}$$

$$Y = Y_{ref} \cdot f_2(L') \tag{13}$$

$$Z = Z_{ref} \cdot f_2\left(L' - \frac{b^*}{200}\right) \tag{14}$$

โดยที่ $L' = \frac{L^*+16}{116}$ และ $f_2(c) = \begin{cases} c^3, c^3 > \epsilon \\ \frac{c-16}{\kappa}, c^3 \leq \varepsilon \end{cases}$

จากนั้นทำการแปลงจากพิกัด CIEXYZ เป็นปริภูมิสี RGB ด้วยการเชื่อมโยงเชิงเส้น (Linear Mapping) ซึ่งสามารถแปลงค่ากลับไปมาได้ ดังแสดงในสมการที่ (15)

จะได้ $$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = M_{RGB} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \text{ และ } \begin{pmatrix} R \\ G \\ B \end{pmatrix} = M_{RGB}^{-1} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{15}$$

โดยที่ $$M_{RGB} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.87592 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix}$$

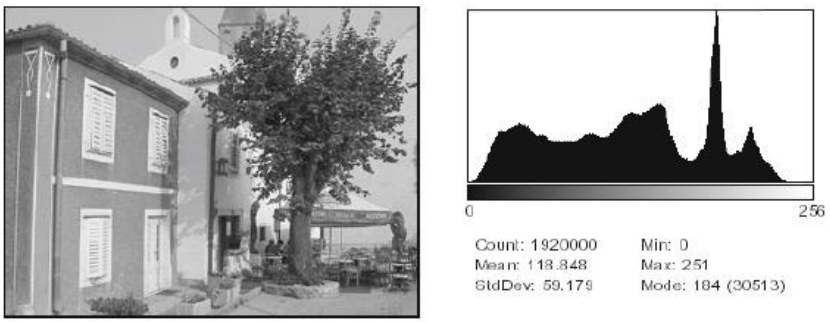
$$M_{RGB}^{-1} = \begin{pmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119913 & 0.950227 \end{pmatrix}$$

ซึ่งทั้ง 3 หลักของเวกเตอร์ M_{RGB}^{-1} เป็นพิกัดพื้นฐานของสี R G และ B ในปริภูมิ XYZ จากสมการที่ (12) ถึง (15)

จึงได้ว่า $R = M_{RGB}^{-1} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ $G = M_{RGB}^{-1} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ และ $B = M_{RGB}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ (16)

2.15 ฮิสโตแกรมของภาพสี (Histograms of Color Images)

โดยทั่วไปแล้วฮิสโตแกรมมีหน้าที่ในการแจกแจงความถี่ เมื่อกล่าวถึงในบริบทการทำงานบนรูปภาพ ฮิสโตแกรมมีหน้าที่ในการแจกแจงความถี่ของความเข้มสีที่ปรากฏภายในภาพ ดังแสดงในภาพที่ 10



ภาพที่ 10 ตัวอย่างการใช้ฮิสโตแกรมในการแจกแจงความถี่ของความเข้มสีภายในภาพ
(ที่มา : Digital Image An Algorithmic Introduction, 2016 : 38)

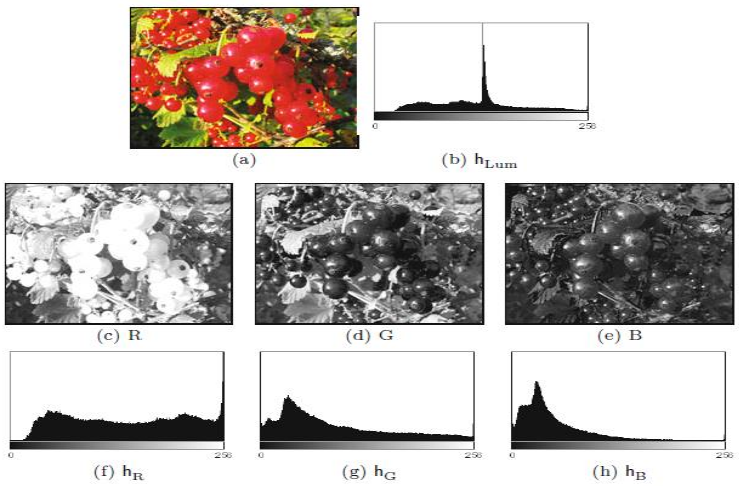
ฮิสโตรแกรมของรูปภาพสีจะมีหน้าที่ซึ่งจำแนกออกได้เป็น 2 ประเภท ได้แก่ ฮิสโตแกรมที่แสดงความสว่าง (Intensity Histogram) และ ฮิสโตแกรมที่แสดงค่าความถี่ในแต่ละช่องสี (Individual Color Channel Histograms) ซึ่งจะแสดงรายละเอียดได้ดังนี้

1. ฮิสโตแกรมที่แสดงความสว่าง (Intensity Histogram)

ฮิสโตรแกรมความสว่างจะคำนวณหาผลรวมความสว่างในแต่ละพิกเซลของรูปภาพ ซึ่งฮิสโตแกรมจะสามารถแสดงได้เพียงค่าความสว่างโดยรวมภายในภาพ แต่ไม่สามารถที่จะหาค่าเฉลี่ยของแต่ละช่องสีได้ (Wilhelm & Mark J. , 2016) ดังแสดงในภาพที่ 11

2. ฮิสโตแกรมที่แสดงค่าความถี่ในแต่ละช่องสี (Individual Color Channel Histograms)

เนื่องจากการใช้ฮิสโตแกรมที่แสดงความสว่างเป็นการแสดงผลรวมของสีในทุกพิกเซลในภาพ อาจทำให้เกิดข้อผิดพลาดได้ เช่น ฮิสโตแกรมแสดงความสว่างของสีภายในภาพ แต่มีบางพิกเซลภายในภาพมีความอิ่มของสีมากจนเกินไป โดยองค์ประกอบของฮิสโตแกรมจะแสดงให้เห็นรายละเอียดการแจกแจงภายในช่องสี เมื่อทำการคำนวณหาองค์ประกอบของฮิสโตแกรมในแต่ละช่องสี จะทำให้ฮิสโตแกรมสามารถแยกความสว่างของภาพในแต่ละช่องสีได้ชัดเจนมากขึ้น ดังแสดงในภาพที่ 2.11 จากภาพจะแสดงให้เห็นการแสดงค่าความสว่างภายในภาพด้วยฮิสโตแกรมที่แสดงความสว่าง h_{Lum} และองค์ประกอบของฮิสโตแกรมซึ่งแยกตามช่องสี R G และ B นั่น คือ h_R h_G และ h_B ตามลำดับ ด้วยวิธีการนี้จะทำให้เห็นการกระจายขของเฉดสีภายในภาพได้ชัดเจนมากยิ่งขึ้น (Wilhelm & Mark J. , 2016)



ภาพที่ 11 แสดงตัวอย่างฮิสโตแกรมที่แสดงความสว่าง และ ฮิสโตแกรมที่แสดงค่าความถี่ในแต่ละช่องสี

(ที่มา : Digital Image An Algorithmic Introduction, 2016 : 46)

2.2 ความรู้พื้นฐานเกี่ยวกับการแบ่งส่วนรูปภาพ (Segmentation Basic Knowledge)

การแบ่งส่วนรูปภาพเป็นกระบวนการแบ่งรูปภาพดิจิตอลออกเป็นหลายส่วน โดยจุดมุ่งหมายของวิธีการนี้คือ การนำเสนอรูปภาพให้อยู่ในรูปแบบอย่างง่ายเพื่อช่วยให้สามารถวิเคราะห์องค์ประกอบภายในภาพได้ง่ายมากขึ้นโดยการใช้วัตถุและขอบเขต เช่น เส้นตรง เส้นโค้ง ฯลฯ ภายในภาพ นอกจากนี้การแบ่งส่วนรูปภาพยังเป็นการระบุประเภทของแต่ละพิกเซล (Pixel) ภายในภาพโดยที่พิกเซลที่เป็นประเภทเดียวกันจะมีลักษณะบางอย่างร่วมกัน (Linda & George 2001) (Milan et al., 2015) ในที่นี้กำหนดให้รูปภาพ R เป็นเป็นเซตจำกัดของพื้นที่ $R_1, \dots, R_s$

จะได้ว่า $$R = \bigcup_{i=1}^{s} R_i, \;\; R_i \cap R_j = \emptyset, \;\; i \neq j \qquad (17)$$

ในปัจจุบันมีเทคนิคการแบ่งส่วนรูปภาพมากมาย โดยวิธีการเหล่านี้มาจากขั้นตอนพื้นฐาน 2 วิธี ได้แก่ การแบ่งส่วนรูปภาพตามพื้นที่ และเส้นขอบ (Edge) โดยสามารถแบ่งประเภทวิธีการแบ่งส่วนรูปภาพได้ทั้งหมด 3 กลุ่ม ดังนี้

2.2.1 เทคนิคการแบ่งส่วนด้วยเค้าโครง (Structural Segmentation Techniques)

เทคเทคนี้จะแบ่งส่วนรูปภาพโดยอาศัยรายละเอียดของโครงสร้างของส่วนที่ต้องการของรูปภาพ เช่น พื้นที่ที่ต้องการจะแบ่งส่วน (Kaur & Kaur, 2014)

2.2.2 เทคนิคการแบ่งส่วนสุ่ม (Stochastic Segmentation Techniques)

วิธีการนี้จะแบ่งส่วนรูปภาพโดยทำงานบนค่าพิกเซลที่ไม่ต่อเนื่องของภาพแทนการสนใจรายละเอียดโครงสร้างของพื้นที่ภายในภาพ (Kaur & Kaur, 2014)

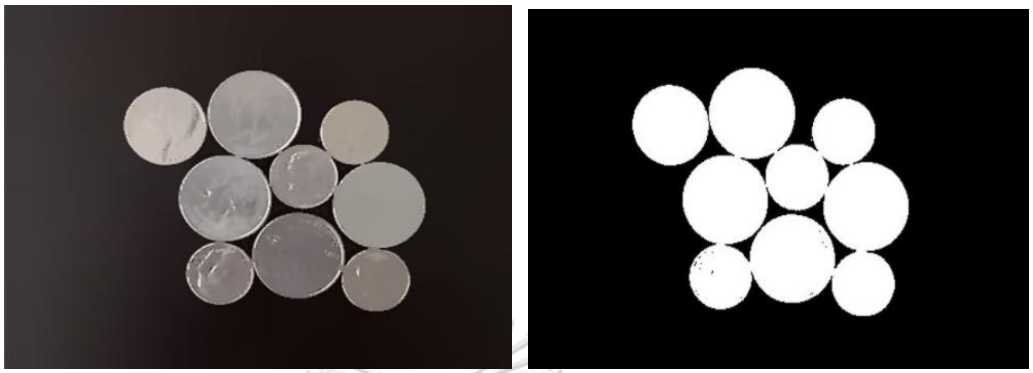
2.2.3 เทคนิคแบบผสม (Hybrid Techniques)

วิธีการนี้เป็นการผสมวิธีการแบ่งส่วนรูปภาพด้วยวิธีการที่กล่าวไว้ในหัวข้อที่ 2.2.1 และ 2.2.2 เข้าด้วยกันซึ่งมีวิธีการที่หลากหลายมาก ในงานวิจัยชิ้นนี้จะเสนอเพียงบางเทคนิคที่เป็นที่นิยม และเป็นวิธีพื้นฐานที่นิยมนำไปต่อยอดในหลายงานวิจัย โดยจะแสดงรายละเอียดได้ดังนี้

2.2.3.1. วิธีปรับค่าเทรชโฮลด์ (Thresholding Method)

วิธีการนี้เป็นการแบ่งพิกเซลภายในรูปภาพตามระดับความเข้มของพิกเซล โดยจะทำให้พื้นหน้าของภาพมีสีสว่างกว่าพื้นหลังของภาพ วิธีการนี้ทำได้โดยการใช้ค่าเทรชโฮลด์ที่เหมาะสมในที่นี้ให้แทนค่าด้วย T โดยค่านี้เป็นค่าที่ใช้ทั้งหมดภายในภาพ ในที่นี้กำหนดให้ภาพตั้งต้นเป็น $p(x, y)$ และภาพที่ได้จากการปรับค่าเทรชโฮลด์ คือ $q(x, y)$ (Kaur & Kaur, 2014)

จะได้ว่า $$q(x,y) = \begin{cases} 1, & \text{ถ้า } p(x,y) > T \\ 0, & \text{ถ้า } p(x,y) \leq T \end{cases} \quad (18)$$



(ก) ภาพตั้งต้น (ข) ภาพที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีปรับค่าเทรดโฮลด์

ภาพที่ 12 แสดงตัวอย่างการแบ่งส่วนรูปภาพด้วยวิธีปรับค่าเทรดโฮลด์
(ที่มา : https://www.pyimagesearch.com/2015/11/02/watershed-opencv/)

2.2.3.2 วิธีการแบ่งส่วนตามขอบ (Edge Based Segmentation Method)

การแบ่งส่วนตามขอบจะใช้เทคนิคการระบุเส้นขอบของภาพ ด้วยการคำนวณค่าอนุพันธ์อันดับหนึ่งของความเข้มภายในภาพที่มากกว่าอนุพันธ์อันดับสองซึ่งจุดตัดเท่ากับศูนย์ โดยเส้นขอบที่ได้จะมีความเชื่อมโยงกันละทำการแบ่งส่วนพื้นที่ออกจากกัน ตัวอย่างของวิธีการแบ่งส่วนตามขอบพื้นฐาน เช่น ตัวดำเนินการโซเบล (Sobel Operator) ตัวดำเนินการแคนนี (Canny Operator) ฯลฯ (Kaur & Kaur, 2014)

(ก) ภาพตั้งต้น (ข) ภาพที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีแบ่งส่วนตามขอบ

ภาพที่ 13 แสดงตัวอย่างการแบ่งส่วนรูปภาพด้วยตัวดำเนินการโซเบล

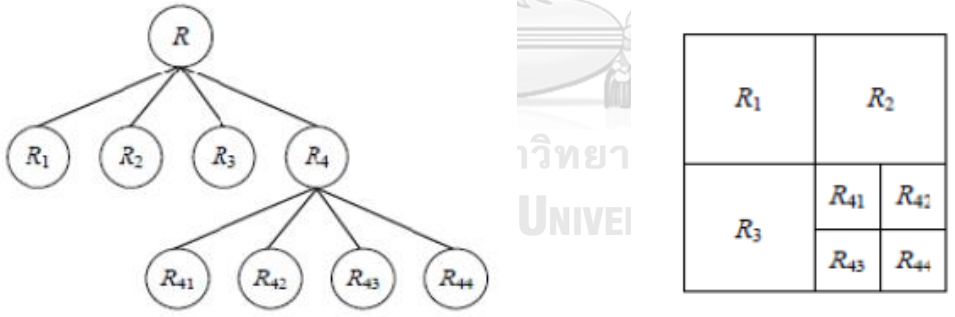
(ที่มา : https://www.mathworks.com/discovery/edge-detection.html)

2.2.3.3 วิธีการแบ่งส่วนตามพื้นที่ (Region Based Segmentation Method)

วิธีการนี้จะทำการแบ่งส่วนรูปภาพออกเป็นหลายพื้นที่ตามความคล้ายคลึงกันของลักษณะพื้นที่ ซึ่งจะมีวิธีการพื้นฐานทั้งหมด 2 วิธี ได้แก่

1. กระบวนการขยายกลุ่มของพิกเซล (Region Growing Methods) คือ กระบวนการขยายกลุ่มของพิกเซลให้มีขนาดใหญ่ขึ้นตามเกณฑ์ที่กำหนด โดยอาจจะเริ่มจากกลุ่มของพิกเซลที่เรียกว่า Seed พิกเซลและขยายพื้นที่ไปยังพื้นที่ใกล้เคียงที่มีคุณสมบัติคล้ายคลึงกัน (Kaur & Kaur, 2014)

2. กระบวนการแยกพื้นที่และการรวมกลับพื้นที่ (Region Splitting and Merging) คือ กระบวนการรวมหรือแยกจุดรูปภาพ โดยกระบวนการรวมกลับพื้นที่จะเริ่มจากพื้นที่เริ่มต้น (Initial Region) ซึ่งอาจเป็นจุดภาพเพียงหนึ่งจุดภาพแล้วทำการรวมจุดภาพที่เหมือนกันเข้าด้วยกันจนได้เป็นบริเวณของวัตถุ ส่วนกระบวนการแยกพื้นที่จะเริ่มจากบริเวณใหญ่ที่ถูกแบ่งส่วน แล้วแบ่งส่วนย่อยให้มีขนาดเล็กลงจนกว่าจะได้บริเวณที่สนใจที่ไม่เหมือนกัน (Kaur & Kaur, 2014)



(ก) กระบวนการขยายกลุ่มของพิกเซล (ข) กระบวนการแยกพื้นที่และการรวมกลับพื้นที่

ภาพที่ 14 แสดงกระบวนการแยกพื้นที่และรวมกลับพื้นที่

(ที่มา : Various Image Segmentation Techniques: A Review, 2014 : 812)

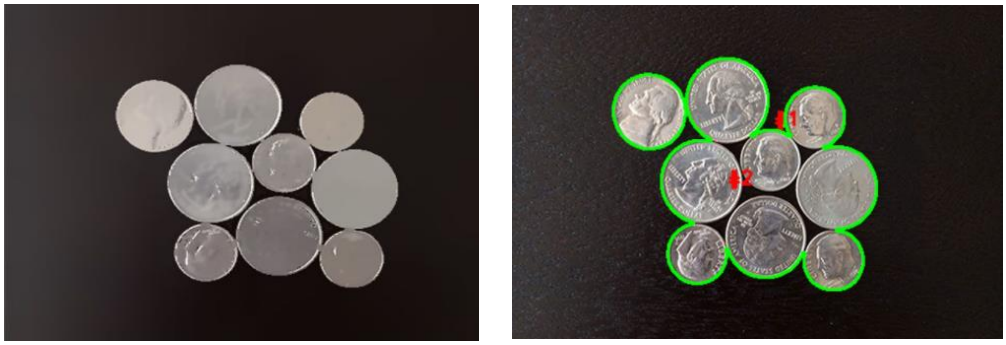
2.2.3.4 วิธีการแบ่งส่วนด้วยการจัดกลุ่ม(Clustering Based Segmentation Method)

วิธีการนี้เป็นการจักลุ่มให้พิกเซลที่มีลักษณะเดียวกันให้มาอยู่ด้วยกัน การจัดกลุ่มข้อมูลจะทำการแบ่งข้อมูลออกเป็นสมาชิกของแต่ละกลุ่ม โดยที่สมาชิกที่อยู่กลุ่มเดียวกันจะมีคุณลักษณะที่คล้ายคลึงกัน ซึ่งการจัดกลุ่มจะถูกแบ่งออกเป็น 2 ประเภทคือ Hard Clustering หรือการกำหนดให้แต่ละชุดข้อมูลแบ่งออกเป็นกลุ่มที่แยกออกจากกันโดยสิ้นเชิง นั่นคือ 1 พิกเซลจะสามารถอยู่ได้เพียงกลุ่มเดียวเท่านั้น และ Soft Clustering หรือการที่ข้อมูลมีโอกาสที่จะอยู่ในหลายๆ กลุ่มได้ ขึ้นอยู่กับความน่าจะเป็นของตัวข้อมูล (Kaur & Kaur, 2014)

ภาพที่ 15 แสดงตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีการจัดกลุ่ม
(ที่มา : https://cierra-andaur.medium.com/using-k-means-clustering-for-image-segmentation-fe86c3b39bf4)

2.2.3.5 วิธีการแบ่งส่วนด้วยขั้นตอนวิธีสันปันน้ำ (Watershed Based Methods)

วิธีการนี้จะให้ความเข้มแทนแอ่งน้ำที่มีก้นบ่อเป็นจุดต่ำสุด เมื่อน้ำเริ่มปริ่มที่ขอบแอ่งน้ำจะทำให้เกิดการเชื่อมกันระหว่างแอ่งน้ำ เพื่อที่จะแยกแอ่งน้ำออกจากกันจำเป็นที่จะต้องมีเส้นขอบที่แบ่งส่วนของพื้นที่ออกจากกัน วิธีการนี้เป็นการไล่ระดับเฉดสีของพื้นที่ภายในภาพ โดยพิกเซลที่มีการไล่ระดับเฉดสีจะแสดงให้เห็นพื้นที่ที่ต่อเนื่องกัน (Kaur & Kaur, 2014)

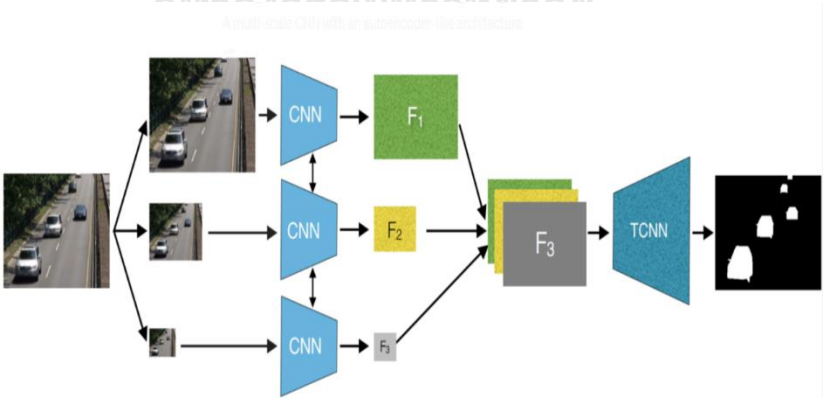


(ก) ภาพตั้งต้น (ข) ภาพที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีแบ่งสันปันน้ำ

ภาพที่ 16 แสดงตัวอย่างการแบ่งส่วนรูปภาพด้วยวิธีการแบ่งส่วนรูปภาพด้วยวิธีแบ่งสันปันน้ำ (ที่มา : https://www.pyimagesearch.com/2015/11/02/watershed-opencv/)

2.2.3.6 วิธีการแบ่งส่วนรูปภาพด้วยโครงข่ายประสาทเทียม (Artificial Neural Network Based Segmentation Method)

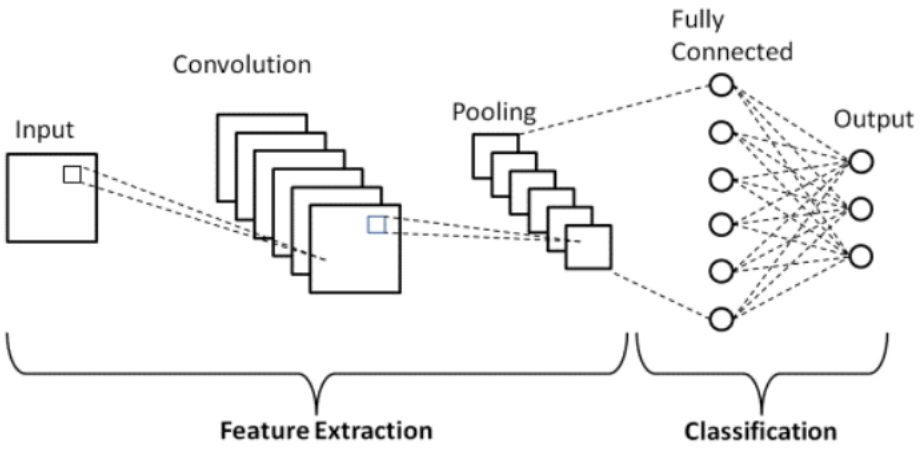
วิธีการนี้เป็นการเลียนแบบวิธีการตัดสินใจของสมองของมนุษย์ โดยจะใช้แยกพื้นหลังรูปภาพออกจากพื้นหน้าของรูปภาพ โดยโครงข่ายประสาทเทียมจะมีขนาดใหญ่และมีโหนด (Nodes) เชื่อมโยงกัน และมีค่าน้ำหนัก (Weight) ในทุกชั้นที่เชื่อมโยงกัน โดยจะมีขั้นตอนการทำงาน 2 ขั้นตอน คือ สกัดคุณลักษณะและแบ่งส่วนรูปภาพ (Kaur & Kaur, 2014)



ภาพที่ 17 แสดงตัวอย่างการแบ่งส่วนรูปภาพด้วยโครข่ายประสาทเทียม (ที่มา : https://towardsdatascience.com/foreground-image-segmentation-with-fgsegnet-9ecbe3d194ab)

2.3 การจำแนกประเภทด้วยการใช้แบบจำลองโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network)

โครงข่ายประสาทเทียม (CNN) หรือที่เรียกว่า ConNet เป็นโครงข่ายประสาทเทียม (Artificial Neuron Network : ANN) ประเภทหนึ่งที่มีปัตยกรรมโครงสร้างแบบฟีดฟอร์เวิร์ด (Feed-forward) ซึ่งมีความสามารถที่พิเศษกว่าโครงข่ายอื่นๆที่มีชั้นเชื่อมโยงสมบูรณ์ (Fully connected) โดยแบบจำลอง CNN สามารถเรียนรู้คุณลักษณะของวัตถุโดยเฉพาะข้อมูลเชิงพื้นที่ (Spatial data) และยังสามารถระบุประเภทของข้อมูลเหล่านี้ได้อย่างชาญฉลาด แบบจำลอง CNN ประกอบด้วยชุดชั้นประมวลผลที่สามารถเรียนรู้คุณลักษณะที่หลากหลายของข้อมูล เช่น ข้อมูลรูปภาพ ด้วยการจำลองการมองเห็นของมนุษย์ในพื้นที่ย่อยๆ จากการแยกแยะคุณลักษณะทางกายภาพ เช่น สี ลายเส้น และอื่นๆ จากนั้นนำมาผสานกันเพื่อทำนายประเภทของภาพ (Ghosh et al., 2020)



ภาพที่ 18 แสดงแนวคิดของแบบจำลอง CNN
(ที่มา : https://www.upgrad.com/blog/basic-cnn-architecture/)

ปัจจุบันแบบจำลอง CNN ได้ถูกนำไปประยุกต์ใช้ในงานหลายประเภท เช่น การจำแนกภาพ (Image classification) การจดจำใบหน้า (Face detection) การจดขำข้อความ (Speech recognition) ฯลฯ โดยองค์ประกอบหรือโครงสร้างพื้นฐานต่าง ๆ ของแบบจำลอง CNN จะแสดงรายละเอียดแยกตามหัวข้อ ดังนี้

2.3.1 โครงสร้างแบบจำลอง CNN

ดังที่ได้นำเสนอเอาไว้ข้างต้นว่าแบบจำลอง CNN ประกอบไปด้วยบล็อก (Blocks) หรือเรียกว่าโครงสร้างชั้น ในหัวข้อนี้จะอธิบายรายละเอียดการสร้างบล็อกต่าง ๆ ในแบบจำลอง CNN ซึ่งจะมีทั้งหมด 3 ชั้น ดังนี้

2.3.1.1 ชั้นคอนโวลูชัน (Convolution)

ชั้นคอนโวลูชันนั้นถือว่าเป็นชั้นที่สำคัญที่สุด เนื่องจากในชั้นนี้จะประกอบไปด้วยเซตของเคอร์นอล (Kernels) ที่จะทำการคอนโวไปบนรูปภาพด้วยวิธีการคอนโวลูชัน ซึ่งเป็นการคำนวณแบบ Dot product กับเคอร์นอลแบบจัตุรัสขนาดเล็ก เช่น 3x3 5x5 หรือ 7x7 เป็นต้น โดยผลลัพธ์ที่ได้จากการคำนวณเรียกว่าฟีเจอร์แมป (Feature map) (Ghosh et al., 2020)

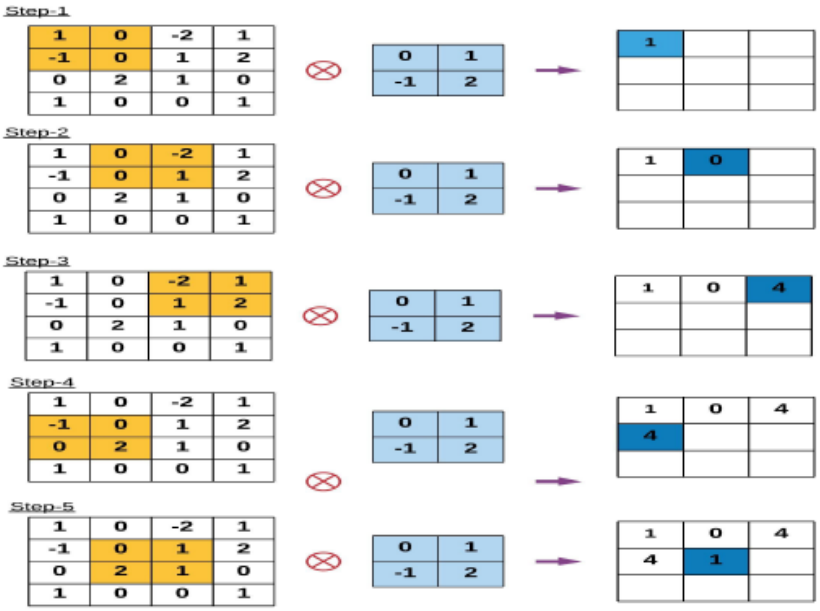
1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

0	1
-1	2

(ก) แสดงตัวอย่างภาพเฉดสีเทาขนาด 4 x 4 (ข) แสดงตัวอย่างเคอร์นอลขนาด 2 x 2

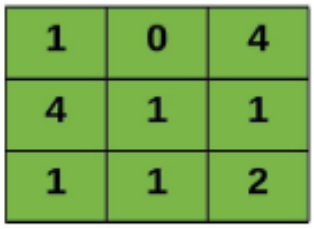
ภาพที่ 19 แสดงตัวอย่างภาพเฉดสีเทาขนาด 4 x 4
และแสดงตัวอย่างเคอร์นอลขนาด 2 x 2
(ที่มา : Fundamental Concepts of Convolutional Neural Network, 2020 : 7)

โดยกระบวนการคอนโวลเคอร์นอลขนาด 2 x 2 บนรูปภาพเฉดสีเทาขนาด 4 x4 จะแสดงรายละเอียดขั้นตอนของกระบวนการเอาไว้ ดังแสดงในภาพที่ 20



ภาพที่ 20 แสดงตัวอย่างขั้นตอนการคอนโวลเคอร์นอลขนาด 2 x 2 บนรูปภาพเฉดสีเทขนาด 4 x 4

(ที่มา : Fundamental Concepts of Convolutional Neural Network, 2020 : 7)



ภาพที่ 21 แสดงตัวอย่างผลลัพธ์ของฟีเจอร์แมปที่ได้จากขั้นตอนการคอนโวลูชัน

(ที่มา : Fundamental Concepts of Convolutional Neural Network, 2020 : 8)

โดยฟีเจอร์แมปที่ได้จากขั้นตอนคอนโวลูชันจะมีขนาดความกว้างและความยาว ดังสมการที่ (19) และ (20)

$$h' = \frac{h-f+p}{s} + 1 \quad (19)$$

$$w' = \frac{w-f+p}{s} + 1 \quad (20)$$

โดยที่ h' แทน ความยาวของฟีเจอร์แมป

w' แทน ความกว้างของฟีเจอร์แมป

w แทน ความกว้างของรูปภาพตั้งต้น

h แทน ความยาวของรูปภาพตั้งต้น

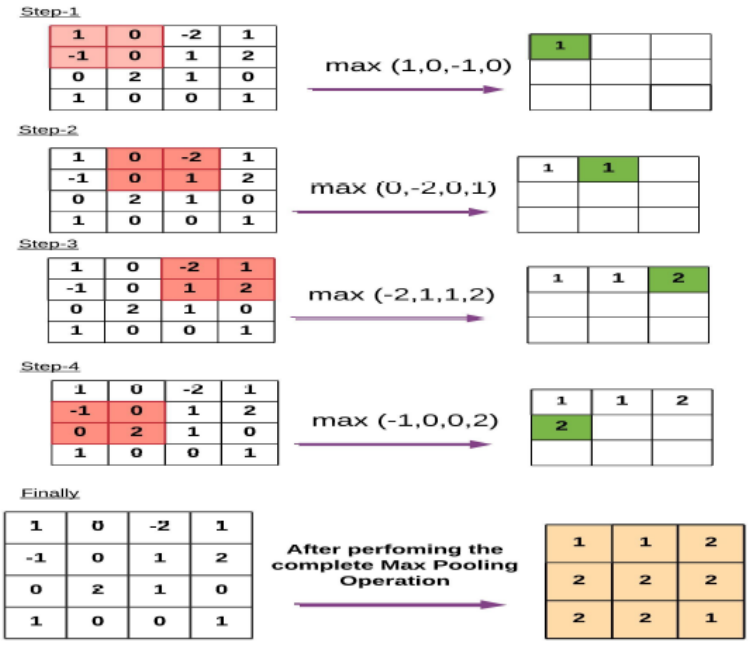
f แทน ขนาดของเคอร์นอล

s แทน จำนวนที่เลื่อนต่อก้าว(Stride) ของตัวดำเนินการคอนโวลูชัน

p แทน จำนวนที่เพิ่มขอบ (padding) ของตัวดำเนินการคอนโวลูชัน

2.3.1.2 ชั้นพูลลิง (Pooling Layer)

ชั้นพูลลิงเป็นชั้นที่อยู่ค่นกลางระหว่างชั้นคอนโวลูชัน โดยวัตถุประสงค์เพื่อลดขนาด (Down Sampling) ของฟีเจอร์แมปให้มีขนาดเล็กลง ซึ่งสามารถใช้ฟังก์ชันค่าเฉลี่ย ฟังก์ชันค่าต่ำสุด และฟังก์ชันค่าสูงสุดในการคำนวณ ดังนั้น หากเลือกใช้ฟังก์ชันค่าสูงสุดในการคำนวณ จะเรียกว่า พูลด้วยค่าสูงสุด (Max Pooling) (Ghosh et al., 2020)



ภาพที่ 22 แสดงตัวอย่างพูลลิ่งด้วยค่าสูงสุด

(ที่มา : Fundamental Concepts of Convolutional Neural Network, 2020 : 10)

โดยฟีเจอร์แมปที่ได้จากการพูลด้วยค่าสูงสุดจะมีความกว้างและความยาว ดังแสดงในสมการที่ (21) ถึง (22)

จะได้ว่า $$h' = \left[\frac{h-f}{s}\right] \quad (21)$$

\ $$w' = \left[\frac{w-f}{s}\right] \quad (22)$$

โดยที่ h' แทน ความยาวของฟีเจอร์แมป

w' แทน ความกว้างของฟีเจอร์แมป

w แทน ความกว้างของฟีเจอร์แมป

h แทน ความยาวของฟีเจอร์แมป

f แทน ขนาดของพื้นที่พูลลิง

s แทน จำนวนที่เลื่อนต่อก้าว(Stride) ของตัวดำเนิการพูลลิง

2.3.1.3 ชั้นเชื่อมโยงสมบูรณ์ (Fully-connected Layer)

โดยทั่วไปชั้นสุดท้ายของแบบจำลอง CNN จะประกอบด้วยชั้นเชื่อมโยงสมบูรณ์ โดยที่แต่ละนิวรอล (Neuron) ภายในชั้นจะถูกเชื่อมโยงกับนิลรอนของตัวเองในชั้นก่อนหน้า โดยทุกนิวรอลที่อยู่ในชั้นสุดท้ายของฟีเจอร์แมปจะถูกนำไปเปลี่ยนรูป หรือเรียกว่าแฟลตเทน (Flatten) เพื่อส่งไปคำนวณในชั้นถัดไปในชั้นโครงข่ายประสาทเทียมแบบ MLP (Multilayer Perceptron) เพื่อจำแนกประเภทของภาพ (Multi-class Classification) ซึ่งชั้นสุดท้ายของชั้นเชื่อมโยงสมบูรณ์จะเป็นชั้นแสดงผล (Output layer) (Ghosh et al., 2020)

2.3.1.4 ฟังก์ชันกระตุ้น (Activation Functions)

หน้าที่หลักของฟังก์ชันกระตุ้นคือ การแมปอินพุต (Input) ไปยังเอาต์พุต (Output) โดยค่าของอินพุตจะได้มาจากการคำนวณผลรวมน้ำหนักของนิวรอลอินพุตและไบแอส (Bias) ที่เกิดขึ้น กล่าวคือฟังก์ชันกระตุ้นจะเป็นตัวพิจารณาว่าจะส่งค่าใดออกมาเป็นเอาต์พุต ฟังก์ชันกระตุ้นที่ใช้ในแบบจำลอง CNN จะแสดงในลักษณะไม่เชิงเส้น (Non-linear) ของชั้นนั้น ๆ เพื่อช่วยในการเรียนรู้ความซับซ้อน โดยฟังก์ชันกระตุ้นที่ใช้ใน Deep Neuron Network มีหลายแบบ เช่น ฟังก์ชันซิกมอยด์ (Sigmoid function) ฟังก์ชันแทนเอช (Tanh function) และฟังก์ชันเรลู (ReLu) (Ghosh et al., 2020) ซึ่งจะแสดงรายละเอียดได้ดังนี้

1. ฟังก์ชันซิกมอยด์ (Sigmoid function)

ฟังก์ชันกระตุ้นซิกมอยด์เป็นฟังก์ชันที่ใช้จำนวนจริงเป็นอินพุต ซึ่งผลลัพธ์ที่ได้จะแสดงผลเป็นตัวเลขมีค่าอยู่ในช่วง [0, 1] ลักษณะกราฟของฟังก์ชันจะเป็นเส้นโค้งคล้ายตัวเอส ดังแสดงในภาพที่ 23 (ก) โดยสามารถอธิบายได้ด้วยสมการที่ (23)

$$f(x)_{sigm} = \frac{1}{1+e^{-x}} \tag{23}$$

2. ฟังก์ชันแทนเอช (Tanh function)

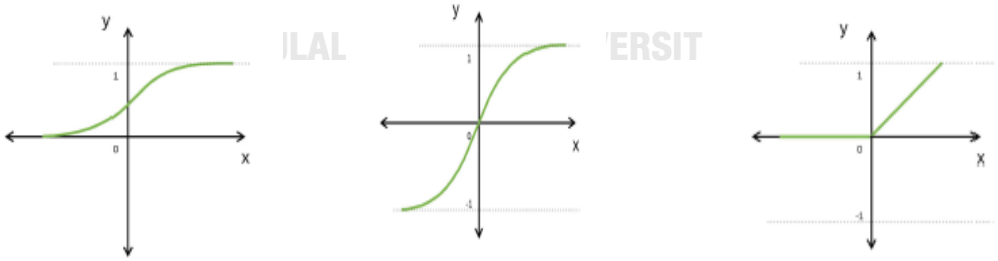
ฟังก์ชันกระตุ้นแทนเอชจะรับค่าเป็นจำนวนจริง โดยผลลัพธ์ที่ได้จะมีค่าอยู่ในช่วง [-1, 1] ดังแสดงในภาพที่ 23 (ข) โดยสามารถอธิบายได้ด้วยสมการที่ (24)

$$f(x)_{tanh} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{24}$$

3. ฟังก์ชันเรลู (ReLu)

ฟังก์ชันกระตุ้น ReLu (Rectifier Linear Unit) เป็นฟังก์ชันกระตุ้นที่นิยมใช้มากที่สุดในแบบจำลอง CNN เนื่องจากฟังก์ชันนี้จะทำการแปลงค่าอินพุตให้มีค่าเป็นบวก โดยประโยชน์ของฟังก์ชันกระตุ้น ReLu คือใช้ปริมาณการประมวลผลที่น้อยเมื่อเทียบกับวิธีอื่น ดังแสดงในภาพที่ 23 (ค) โดยสามารถอธิบายได้ด้วยสมการที่ (25)

$$f(x)_{ReLu} = \max(0, x) \tag{25}$$



(ก) กราฟฟังก์ชันซิกมอยด์ (ข) กราฟฟังก์ชันแทนเอช (ค) กราฟฟังก์ชันเรลู

ภาพที่ 23 แสดงลักษณะกราฟของฟังก์ชันกระตุ้น

(ที่มา : Fundamental Concepts of Convolutional Neural Network, 2020 : 11)

2.3.1.5 ฟังก์ชันเสมือนเป้าหมาย (Loss Functions)

ในส่วนนี้จะเป็นการยกตัวอย่างฟังก์ชันเสมือนเป้าหมายที่ใช้ในการจำแนกประเภทรูปภาพในงานวิจัยชิ้นนี้ ซึ่งจะมีรายละเอียดดังนี้

1. ซอร์ฟแมกซ์ฟังก์ชัน (Soft-Max Loss Function)

ซอร์ฟแมกซ์ฟังก์ชันเป็นฟังก์ชันที่นิยมใช้อย่างแพร่หลายในการจำแนกประเภทรูปภาพ ซึ่งผลลัพธ์ที่ได้จะเป็นค่าความน่าจะเป็น $p \in \{0,1\}$ โดยซอร์ฟแมกซ์ฟังก์ชันจะใช้ในชั้นแสดงผลลัพธ์ของแบบจำลอง CNN ซึ่งการกระจายความน่าจะเป็น $p, y \in R^N$ โดย p แทน ความน่าจะเป็นของแต่ละประเภทผลลัพธ์ และ y แทน ผลลัพธ์ที่ต้องการและความน่าจะเป็นของแต่ละประเภทผลลัพธ์ (Ghosh et al., 2020)

จะได้ $$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} \quad (26)$$

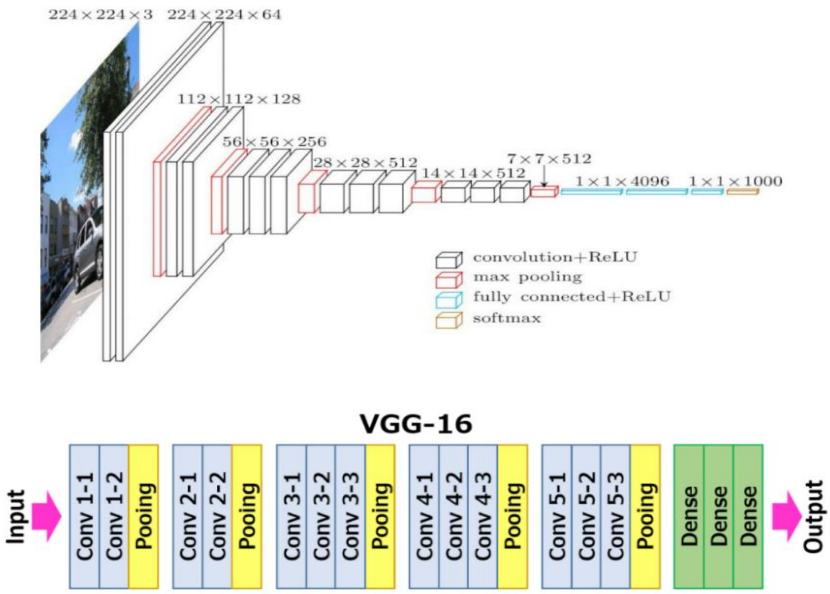
และ $$H(p,y) = \sum_i y_i \log(p_i) \quad \text{ซึ่ง } i \in [1,N] \quad (27)$$

โดย N แทน จำนวนนิวรอลในชั้นผลลัพธ์

e^{a_i} แทน ผลลัพธ์ที่ไม่ได้มอมัลไลซ์ (Unnormalize) จากชั้นโครงข่ายก่อนหน้า

2.3.2 แบบจำลอง VGGNet

VGGNet เป็นแบบจำลอง CNN ที่ได้รับความนิยมเป็นอย่างมาก ซึ่งนำเสนอขึ้นโดย Simonya และ Zisserman ในปี 2014 โดยเฉพาะ VGGNet-16 และ VGGNet-19 หรือเรียกกันโดยทั่วไปว่า VGG16 และ VGG19 ตามลำดับ เป็นแบบจำลองที่ได้รับความนิยมมากที่สุด เนื่องจากในงานวิจัยชิ้นนี้ได้มีการนำแบบจำลอง VGG16 มาทำ Fine-tuning ซึ่งเป็นกระบวนการที่ใช้เมื่อมีการนำแบบจำลอง Pre-trained มาใช้สอนชุดข้อมูลของตัวเอง ดังนั้นการทำความเข้าใจเกี่ยวกับโครงสร้างของแบบจำลอง VGG16 สามารถช่วยให้เข้าใจแนวทางการนำแบบจำลองมาใช้จำแนกประเภทดอกไม้ได้ดียิ่งขึ้น โดยโครงสร้างของแบบจำลอง VGG16 (Ghosh et al., 2020) ได้แสดงไว้ในภาพที่ 24



ภาพที่ 24 แสดงโครงสร้างแบบจำลอง VGG16

(ที่มา : https://neurohive.io/en/popular-networks/vgg16/)

แบบจำลอง VGG16 มีทั้งหมด 5 บล็อก จำนวน 16 ชั้น โดยไม่นับชั้นพูลลิง ในชั้นแรกและชั้นที่ 2 ของแบบจำลองจะประกอบไปด้วยเคอร์นอลขนาด 3 x 3 จำนวน 64 ฟิลเตอร์ (Filter) โดยจะรับค่าเป็นรูปภาพขนาด 224 x224 x3 ในชั้นที่ 3 และ 4 ประกอบไปด้วยเคอร์นอลขนาด 3 x 3 จำนวน 124 ฟิลเตอร์ของผลลัพธ์ที่ได้จะลดลงเป็น 56 x 56 x128 ในชั้นที่ 5 ถึง 7 ประกอบด้วยฟิลแตอร์จำนวน 256 ของเคอร์นอลขนาด 3 x3 ในชั้นที่ 8 ถึง 13 ประกอบด้วยฟิลแตอร์จำนวน 512 ของเคอร์นอลขนาด 3 x3 และในชั้นที่ 14 ถึง 16 จะเป็นชั้นเชื่อมโยงสมบูรณ์โดยที่แต่ละชั้นจะประกอบด้วยโหนดชั้นละ 4,096 โหนดและมีฟังก์ชันกระตุ้นเป็นฟังก์ชันซอร์ฟแมกซ์ (Tammina, 2019)

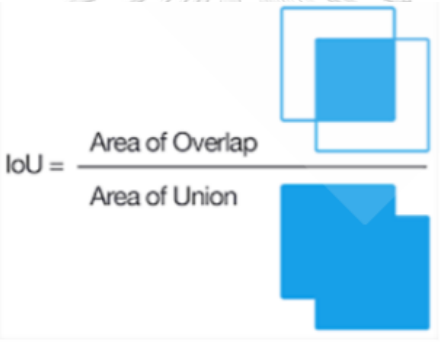
2.4 วิธีการประเมินประสิทธิภาพของการแบ่งส่วนรูปภาพและการจำแนกประเภท

เนื่องจากงานวิจัยมุ่งเน้นศึกษาการแบ่งส่วนรูปภาพดอกไม้เพื่อนำไปใช้ในการจำแนกประเภทดอกไม้ ดังนั้นวิธีการประเมินประสิทธิภาพของการแบ่งส่วนรูปภาพและการจำแนกประเภทของภาพจึงเป็นสิ่งสำคัญ ที่จะแสดงให้เห็นถึงประสิทธิภาพของวิธีการ โดยวิธีการประเมินจะแบ่งตามรายละเอียดหัวข้อดังนี้

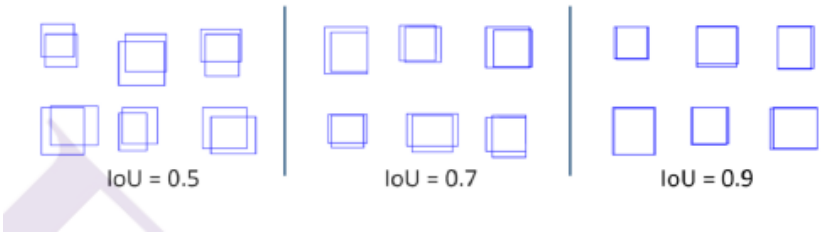
2.4.1 การประเมินประสิทธิภาพของการแบ่งส่วนรูปภาพ

IoU (Intersection over Union) เป็นวิธีการวัดประสิทธิภาพของการแบ่งส่วนรูปภาพที่นิยมเป็นอย่างมาก โดยค่า IoU สามารถคำนวณได้จากสัดส่วนของพื้นที่ที่ซ้อนทับกันระหว่างพื้นที่ที่ถูกต้อง (Ground truth) กับพื้นที่ที่ได้จากการแบ่งส่วนรูปภาพหารกับพื้นที่ทั้งสองแบบรวมกัน ดังแสดงในสมการที่ (28) ซึ่งถ้าค่า IoU มากกว่า 0.5 จึงจะสามารถยอมรับได้ (Najjar & Zagrouba, 2012) ดังแสดงในภาพที่ 25 และ 26

$$IoU = \frac{true\ foreground \cap segmented\ foreground}{true\ foreground \cup segmented\ foreground} \quad (28)$$



ภาพที่ 25 แสดงให้เห็นตัวอย่างวิธีการคำนวณค่า IoU
(ที่มา : การตรวจหาต้นไม้เป็นโรคโดยอัตโนมัติด้วยภาพถ่ายมุมสูงจากโครน และวิธีการเรียนรู้เชิงลึก, 2562 : 19)



ภาพที่ 26 แสดงพื้นที่ของค่า IoU
(ที่มา : การตรวจหาต้นไม้เป็นโรคโดยอัตโนมัติด้วยภาพถ่ายมุมสูงจากโครน และวิธีการเรียนรู้เชิงลึก, 2562 : 20)

2.4.2 วิธีประเมินประสิทธิภาพของการจำแนกประเภทรูปภาพ

ในการวัดประสิทธิภาพในการสกัดคุณลักษณะจะวัดประสิทธิภาพด้วยค่าความถูกต้อง (Accuracy) ความแม่นยำ (Precision) ความครบถ้วน (Recall) และ F1 (Ornek & Ceylan, 2019) ดังแสดงในสมการที่ (29) ถึง (32)

1. ค่าความถูกต้อง (Accuracy) คือ ความถูกต้องที่สิ่งที่ทำนายทำนายได้ตรงกับสิ่งที่เกิดขึ้นจริง

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (29)$$

2. ความแม่นยำ (Precision) คือ การเปรียบเทียบการทำนายที่ถูกต้องว่าจริงและเกิดขึ้นจริง (TP) กับการทำนายว่าจริงแต่สิ่งที่เกิดขึ้นคือไม่จริง (FP)

$$Precision = \frac{TP}{TP+FP} \quad (30)$$

3. ความครบถ้วน (Recall) คือ ความถูกต้องของการทำนายว่าเป็นจริงเทียบกับจำนวนครั้งของเหตุการณ์ที่ทำนายและเกิดขึ้นว่าเป็นจริง

$$Recall = \frac{TP}{TP+FN} \quad (31)$$

4. F1 คือ ค่าเฉลี่ยแบบฮาร์มอนิก (Harmonic) ระหว่างค่าความถูกต้องและความครบถ้วน จุดประสงค์ของ F1คือการสร้างเมทริกซ์เดี่ยว (Single Metric) ที่ใช้ในการวัดความสามารถของแบบจำลอง

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (32)$$

โดยที่ TP คือ ข้อมูลที่ทำนายแล้วถูกต้องเมื่อเทียบกับข้อมูลจริง

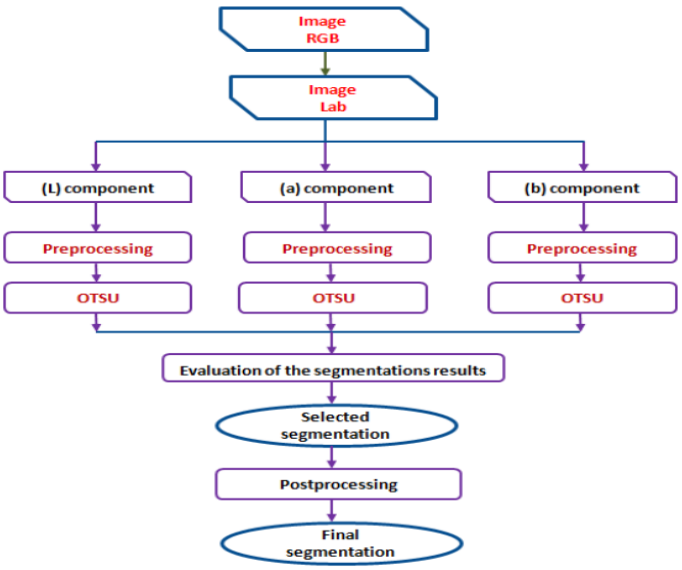
FP คือ ข้อมูลที่อยู่ในข้อมูลจริงแต่ไม่มีในการทำนาย

FN คือ ข้อมูลที่ทำนายแล้วไม่ถูกต้องเมื่อเทียบกับข้อมูลจริง

2.5 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง (Theory and Related Research)

งานวิจัยชิ้นนี้ศึกษาเกี่ยวกับวิธีการการแบ่งส่วนรูปภาพ จึงได้ศึกษางานวิจัยที่เกี่ยวข้องเพื่อนำมาใช้เป็นแนวทางในการพัฒนาวิธีการที่จะนำเสนอต่อไป จากการศึกษาพบว่างานวิจัยที่เคยศึกษาเกี่ยวกับการแบ่งส่วนรูปภาพโดยทั่วไปจะใช้วิธีพื้นฐานทั้งหมด 3 แบบ ได้แก่ การปรับค่าเทรดโฮลด์ การสร้างโครงร่างของดอกไม้เพื่อเก็บบริเวณที่สนใจ และการเปลี่ยนปริภูมิสี แต่พบว่างานวิจัยเหล่านี้มักจะใช้วิธีเหล่านี้ร่วมกันกับวิธีการอื่นๆ เพื่อช่วยให้วิธีการแบ่งส่วนรูปภาพมีประสิทธิภาพมากยิ่งขึ้น

การแบ่งส่วนรูปภาพด้วยวิธีการปรับค่า Threshold เป็นวิธีที่ได้รับความนิยมเป็นอย่ามาก (Sathya & Kayalvizhi, 2011) (Bhargavi & Jyothi, 2014) (W. Wang et al., 2017) เนื่องจากเป็นวิธีที่ไม่ซับซ้อนและใช้เวลาน้อย แต่วิธีนี้ไม่เหมาะกับการแบ่งส่วนชุดข้อมูลรูปภาพขนาดใหญ่ เพราะการปรับค่า Threshold ให้มีค่าพอดีกับชุดข้อมูลรูปภาพเป็นเรื่องที่ท้าทายมาก จึงมีโอกาสเกิดข้อผิดพลาดได้ ด้วยเหตุนี้งานวิจัยที่ใช้วิธีการนี้ (Thanh & Thanh, 2020) จึงนิยมใช้วิธีการนี้ร่วมกับการเปลี่ยนปริภูมิสี เช่น Lab เพื่อลดข้อผิดพลาดที่จะเกิดขึ้น ซึ่งการนำเอาประโยชน์เหล่านี้ไปใช้สามารถศึกษาได้จากงานวิจัยของ Najjar (2012) และผู้ร่วมวิจัย ได้มีการปรับปรุงวิธีการแบ่งส่วนรูปภาพดอกไม้ ซึ่งแสดงขั้นตอนการทำงานไว้ในภาพที่ 27 โดยเริ่มจากการแปลงรูปภาพจากปริภูมิสี RGB ให้เป็นปริภูมิสี Lab เพื่อช่วยให้รูปภาพสามารถแสดงองค์ประกอบที่สนใจในภาพได้ชัดเจนมากยิ่งขึ้น จากนั้นจึงใช้ขั้นตอนการเตรียมรูปภาพกับทุกองค์ประกอบของปริภูมิสี Lab ดังแสดงในภาพที่ 28 ซึ่งองค์ประกอบ L สามารถช่วยให้แบ่งส่วนรูปภาพได้ดีมากขึ้น แต่ยังเหลือองค์ประกอบที่ไม่สนใจอยู่ภายในภาพจึงได้มีการลบชั้นแสงออกไป เพื่อช่วยลดปัญหาสีเพี้ยนจากพื้นผิวดอกไม้ จากนั้นจึงทำการปรับค่า Threshold ที่องค์ประกอบ a และ b ด้วย OTSU เทรดโฮลด์ ด้วยวิธีการนี้จะช่วยให้สามารถแบ่งส่วนรูปภาพได้ดีขึ้น และง่ายต่อการนำไปใช้งานต่อ



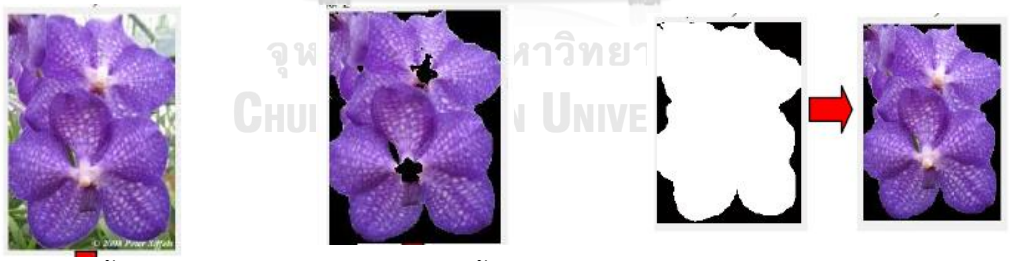
ภาพที่ 27 แสดงกระบวนการทำงานของวิธีการที่นำเสนอโดย Najjar
(ที่มา : Flower image segmentation based on color analysis and a supervised evaluation, 2012 : 398)

(ก) รูปภาพตั้งต้น

(ข) รูปภาพที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Najjar

ภาพที่ 28 แสดงผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีการของ Najjar
(ที่มา : Flower image segmentation based on color analysis and a supervised evaluation, 2012 : 400)

นอกจากการปรับค่าเทรดโฮลด์ การแบ่งส่วนรูปภาพโดยการใช้ K-mean clustering ยังเป็นอีกหนึ่งวิธีที่ได้รับความนิยมเช่นกัน (Yadav & Sharma, 2013) (Garg & Kaur, 2016) (Sharma & Sehgal, 2016) เนื่องจากมีอัลกอริทึมที่เข้าใจง่ายและทำงานได้เร็ว แต่การใช้วิธีนี้เพียงอย่างเดียวไม่สามารถลดรายละเอียดของพื้นหลังรูปภาพได้ งานวิจัยที่ใช้วิธีนี้ (Zheng et al., 2018) (Hassan et al., 2017) จึงใช้วิธีการอื่นควบคู่ไปด้วย อย่างการเปลี่ยนปริภูมิสีของรูปภาพเป็น HSV หรือ Lab โดย Sabri (2016) และผู้ร่วมวิจัยได้มีการนำเสนอวิธีการแบ่งรูปภาพดอกกล้วยไม้ เพื่อเปรียบเทียบประสิทธิภาพที่ได้ระหว่างวิธีที่นำเสนอกับวิธี FCM ซึ่งวิธีที่นำเสนอมีการใช้ K-mean clustering ในการแบ่งส่วนรูปภาพโดยกำหนดให้ค่า K = 2 จากนั้นทำการเตรียมสัณฐานดอกไม้โดยการนำรูปภาพตั้งต้นมาผ่านตัวดำเนินการ (Operation) ด้วยวิธี Filling และ Closing ในการแสดงให้เห็นโครงร่างของดอกไม้ และทำการลบรายละเอียดจุดรบกวน (Noise) ภายในภาพออกไป ในขั้นตอนสุดท้ายจะเป็นการนำรูปภาพตั้งต้นมาซ้อนทับกับรูปสัณฐานดอกไม้ จะได้รูปภาพที่สามารถแสดงรายละเอียดของบริเวณที่สนใจได้อย่างชัดเจน ดังแสดงในภาพที่ 29 และ 30 โดยเมื่อทำการประเมินการแบ่งส่วนรูปภาพด้วด้วยวิธี FCM ด้วยการวัดค่า AO (Area Overlap) FPR (False Positive Rate) และ FNR (False Negative Rate) กับรูปภาพที่ถูกต้อง (Ground truth image) พบว่าวิธีที่นำเสนอในงานวิจัยนี้ให้ผลลัพธ์ค่าเฉลี่ยที่สูงกว่าวิธี FCM



(ก) รูปภาพตั้งต้น (ข) รูปภาพที่ผ่านขั้นตอนการแบ่งส่วนรูปภาพด้วย K-mean clustering (ค) รูปภาพการนำสัณฐานของดอกไม้ที่ผ่านขั้นตอนการแบ่งส่วนรูปภาพมาซ้อนทับกับรูปภาพตั้งต้น

ภาพที่ 29 แสดงขั้นตอนการแบ่งส่วนรูปภาพด้วยวิธีของ Sabri
(ที่มา : K-Means vs. Fuzzy C-Means for Segmentation of Orchid Flowers, 2016 : 84)

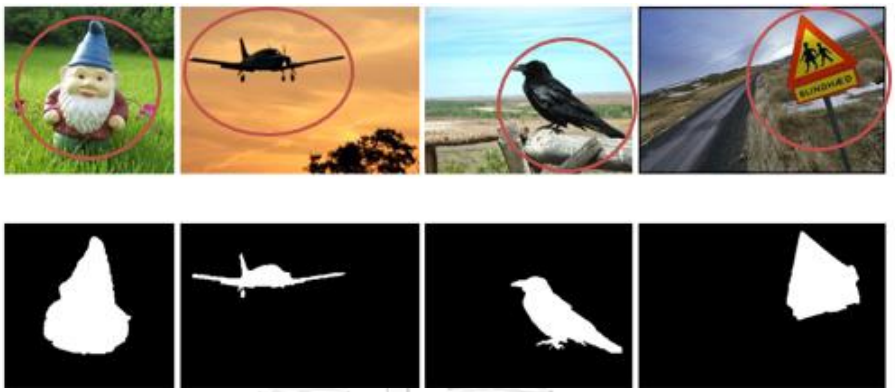
Flower Species	Original Image	K-Means algorithm	FCM algorithm
Avanda			
Ascocentrum			

ภาพที่ 30 แสดงผลลัพธ์การแบ่งส่วนด้วยวิธีของ กับวิธี FCM แสดงผลลัพธ์ที่ได้จากขั้นตอนการแบ่งส่วนรูปภาพด้วยวิธีของ Sabri (ที่มา : : K-Means vs. Fuzzy C-Means for Segmentation of Orchid Flowers, 2016 : 85)

จากวิธีการนี้จะเห็นได้ว่าการใช้ K-mean clustering สามารถช่วยลดรายละเอียดภายในภาพลงได้ และการใช้สัณฐานของดอกไม้ที่เป็นภาพไบนารีมาซ้อนทับกับภาพที่ได้จากการแบ่งส่วนสามารถช่วยลดรายละเอียดของภาพได้ดีมากขึ้น แต่การใช้ K-mean clustering กับรูปภาพที่มีพื้นหลังซับซ้อน และมีจำนวนรูปภาพที่มากขึ้นอาจจะทำให้การกำหนดค่า K = 2 ไม่สามารถแบ่งส่วนรูปภาพได้ดีเท่าการใช้ชุดข้อมูลรูปภาพขนาดเล็ก

เนื่องจากวิธีการที่มีการนำเสนอในงานวิจัยส่วนมาก (Chen et al., 2013) (Cooley et al., 2017) มักจะพยายามหาบริเวณของวัตถุที่สนใจภายในรูปภาพ Yangyang (2019) และผู้ร่วมวิจัยจึงได้นำเสนอวิธีการแบ่งส่วนรูปภาพด้วยการใช้ Saliency map โดยงานวิจัยชิ้นนี้จะนำเสนอวิธีการแบ่งส่วนรูปภาพดอกไม้ ด้วยการการใช้ Saliency map ที่สร้างมาจากวิธี RC โดยวิธีการสร้าง Saliency map มาจากแนวคิดที่ว่าภายในรูปภาพจะมีส่วนที่เด่นปรากฏอยู่ ซึ่งความเด่น (Saliency) หมายถึงคุณสมบัติที่เป็นเอกลักษณ์ของรูปภาพในบริบทของการประมวลผลภาพ คุณลักษณะเฉพาะนี้จะแสดงให้เห็นถึงตำแหน่งที่ดึงดูดสายตาภายในรูปภาพ โดยซาเลียนซีแมป คือรูปภาพที่ความสว่างของพิกเซล

จะแสดงความเด่นของพิกเซล โดยความสว่างของพิกเซลจะส่งผลโดยตรงต่อความเด่นของรูปภาพ (Sharma & Sehgal, 2016)



ภาพที่ 31 รูปภาพแถวบนในวงกลมสีแดงแสดงให้เห็นวัตถุที่เด่น (Salient object) รูปภาพแถวล่างแสดงความเด่น

(ที่มา : Global Contrast based Salient Region Detection, 2014 : 1)

คนส่วนใหญ่มักจะสนใจรูปภาพที่มีความแตกต่างของพื้นหน้าและพื้นหลังอย่างชัดเจน นอกจากความแตกต่างที่ชัดเจนระหว่างพื้นหน้ากับพื้นหลังภาพแล้ว ความสำพันธ์เชิงพื้นที่ (Spatial Relationship) ยังมีบทบาทสำคัญต่อความสนใจของมนุษย์มากเช่นกัน ซึ่งในงานวิจัยของ Yangyang ได้มีการใช้วิธีจำแนกความต่างในรูปภาพที่ชื่อว่า RC (Regional Contrast) ซึ่งเป็นการผนวกทั้งการเปรียบเทียบความต่างทั้งหมดภายในรูปภาพ (Global contrast) และความสอดคล้องเชิงพื้นที่ (Spatial coherence) โดย Histogram based contrast หรือ HC จะเป็นตัวกำหนดค่าความโดดเด่นของพิกเซลโดยพิจารณาจากความต่างของสีของแต่ละพิกเซลในภาพ (Cheng et al., 2014) ซึ่งจะแสดงรายละเอียดของวิธีการได้ดังนี้

ภาพที่ 32 แสดงองค์ประกอบของ RC

ในที่นี้ค่าความเด่นของพิกเซลในรูปภาพ (Saliency value) สามารถคำนวณได้จากสมการที่ (33) และ (34) เนื่องจากพิกเซลมีสีใกล้เคียงกันจะมีค่านัยสำคัญเหมือนกัน จึงมีการจัดรูปสมการใหม่โดยการจัดกลุ่มให้สีที่คล้ายกันมาอยู่ด้วยกัน ด้วยวิธีนี้จะทำให้มองเห็นค่าความเด่นของแต่ละสีได้ดียิ่งขึ้น ดังแสดงในสมการที่ (35)

$$S(I_k) = \sum_{\forall I_i \epsilon I} D(I_k, I_i) \tag{33}$$

$$S(I_k) = D(I_k, I_1) + D(I_k, I_2) + \cdots + D(I_k, I_N) \tag{34}$$

$$S(I_k) = S(c_l) = \sum_{j=1}^{n} f_j D(c_l, c_j) \tag{35}$$

โดยที่ I แทน รูปภาพ

I_k แทน ค่าความเด่นของพิกเซลภายในรูปภาพ

D แทน เมทริกซ์ระยะสีระหว่างพิกเซลในรูปภาพ (Color Distance Matrix)

c_l แทน สีของพิกเซลที่ I_k

n แทน จำนวนสีของพิกเซลที่แตกต่างกัน

N แทน จำนวนพิกเซลในรูปภาพที่ I

f_j แทน ความน่าจะเป็นที่ c_j จะปรากฏอยู่ในรูปภาพ

$S(I_k)$ แทน ค่าความเด่นของพิกเซลในรูปภาพ (Salient Value)

ถึงแม้ว่าจะมีการจัดกลุ่มสีที่คล้ายกันแล้ว แต่ยังมีโอกาสที่บางเฉดสียังไม่ถูกจัดกลุ่ม จึงทำให้ยังมีค่าความเด่นสุ่มกระจายภายในภาพ ด้วยเหตุนี้จึงมีการนิยามค่าความเด่นของสีด้วยค่าเฉลี่ยน้ำหนักของค่าความเด่นของสีที่คล้ายกัน เรียกวิธีการนี้ว่าการปรับสีพื้นที่ให้เรียบ (Color space smoothing) จากนั้นให้ค่าของน้ำหนักที่มากกว่าเป็นตัวแทนของค่าเฉดสีในแต่ละกลุ่ม ดังแสดงใสมการที่ (36)

จะได้ $$S'(c) = \frac{1}{(m-1)T} \sum_{i=1}^{m} \big(T - D(c, c_i)\big) S(c_i) \tag{36}$$

โดยที่ $S'(c)$ แทน ค่าความเด่นของพิกเซลในรูปภาพที่มีการปรับสีพื้นให้เรียบ

T แทน $\sum_{i=1}^{m} D(c, c_i)$

$D(c, c_i)$ แทน ระยะห่างของสีภายในรูปภาพ

m แทน $\frac{n}{4}$

ในขั้นตอนนี้จะเป็นการอธิบายวิธีการคำนวณหาค่าความเด่นในแต่ละพื้นที่ สำหรับพื้นที่ r_k ค่าความเด่นสามารถคำนวณได้จากการวัดความต่างของสีจากพื้นที่ r_k ไปยังพื้นที่ใดๆในรูปภาพ ดังแสดงในสมการที่ (37) และ (38)

$$S(r_k) = \sum_{r_k \neq r_i} w(r_i) D_r(r_k, r_i) \tag{37}$$

$$D_r(r_1, r_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f(c_{1,i}) f(c_{2,j}) D(c_{1,i}, c_{2,j}) \tag{38}$$

โดยที่ r_k แทน พื้นที่

$w(r_i)$ แทน น้ำหนักของพื้นที่ r_i

$f(c_{k,j})$ แทน ความน่าจะเป็นของสีที่ $c_{k,j}$ ท่ามกลางสีทั้งหมด n_k ในพื้นที่ r_k

$S(r_k)$ แทน ค่าความเด่นของพื้นที่ในรูปภาพ

$D_r(r_1, r_2)$ แทน ระยะห่างระหว่างพื้นที่ r_1 และ r_2

เพื่อที่จะเพิ่มผลกระทบเชิงพื้นที่ จึงได้มีการเพิ่มพจน์การถ่วงน้ำหนักเชิงพื้นที่ ซึ่งในงานวิจัยของ Chen ได้กำหนดค่า σ_s^2 เท่ากับ 0.4 เพื่อใช้ในการคำนวณค่าความเด่นเชิงพื้นที่ โดยพจน์น้ำหนักจะเกี่ยวข้องกับค่าเฉลี่ยระยะทางระหว่างพิกเซลในพื้นที่ r_kกับจุดศูนย์กลางของรูปภาพ ดังนั้นพื้นที่ที่มีระยะทางใกล้กับจุดศูนย์กลางของรูปภาพจะให้ค่าความเด่นที่สูง ดังแสดงในสมการที่ (39)

$$S(r_k) = w_s(r_k) \sum_{r_k \neq r_i} e^{\frac{D_s(r_k, r_i)}{-\sigma^2 s}} w(r_i) D_r(r_k, r_i) \tag{39}$$

โดยที่ $w_s(r_k)$ แทน พจน์การถ่วงน้ำหนักเชิงพื้นที่

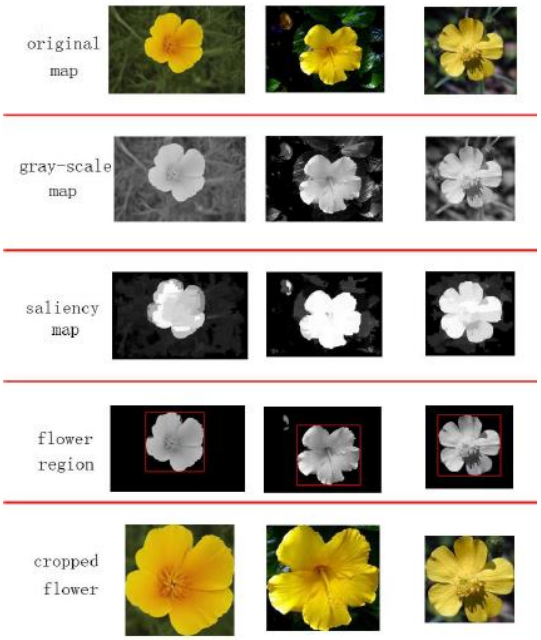
$D_s(r_k, r_i)$ แทน ระยะทางเชิงพื้นที่ระหว่างพื้นที่ r_k และ r_i

σ_s แทน การควบคุมจุดเด่นของระยะทางเชิงพื้นที่ถ่วงน้ำหนัก

$w(r_i)$ แทน น้ำหนักของพื้นที่ r_i

เมื่อได้ Saliency map จากวิธีการ RC ขั้นตอนต่อมาคือ ทำการแปลงรูปภาพของ Saliency map ให้เป็นรูปภาพเฉดสีเทา จากนั้นทำการนำรูปภาพดอกไม้ตั้งต้นมาเปลี่ยนเฉดสีจาก RGB ให้เป็นรูปภาพเฉดสีเทา และทำการซ้อนทับกันระหว่าง Saliency map ที่แปลงสีให้เป็นสีเทากับรูปภาพดอกไม้ตั้งต้นที่ถูกแปลงเฉดสีให้เป็นสีเทา เนื่องจากภายในภาพยังคงเหลือรายละเอียดของพื้นหลัง งานวิจัยชิ้นนี้

จึงได้พยายามปรับพื้นหลังรูปภาพให้เป็นสีดำ เพื่อลดรายละเอียดพื้นหลังจากนั้นจึงทำการหาบริเวณพื้นที่เชื่อมต่อกันหากที่สุดภายในภาพ และทำการสร้าง Bounding box เพื่อล้อมรอบบริเวณนั้นแล้วครอบตัดรูปภาพเพื่อนำไปใช้จำแนกประเภทดอกไม้ด้วยแบบจำลอง PCANet



ภาพที่ 33 แสดงขั้นตอนการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang (ที่มา : A Flower Image Classification Algorithm Based on Saliency Map and PCANet, 2019 : 18)

สิ่งที่น่าสนใจในงานชิ้นนี้คือการนำ Saliency map มาใช้สร้างโครงร่างของดอกไม้เพื่อเก็บบริเวณที่สนใจ นอกจากจะช่วยแสดงบริเวณที่สนใจภายในภาพให้เด่นชัดขึ้นได้แล้ว ยังช่วยลดรายละเอียดในบริเวณพื้นหลังของรูปภาพลงได้ จึงช่วยให้ลดระยะเวลาในขั้นตอนการเตรียมรูปภาพลงได้ แต่ข้อจำกัดของวิธีการนี้คือ หากลดรายละเอียดภายในบริเวณพื้นหลังงรูปภาพได้ไม่ดีมากพอ จะทำให้เมื่อสร้าง Bounding box ล้อมรอบบริเวณที่สนใจ ภายในบริเวณที่ถูกล้อมรอบจะเหลือองค์ประกอบของภาพพื้นหลัง ทำให้เมื่อทำการครอบตัดบริเวณนั้นพื่อนำไปใช้ต่อ จะได้รูปภาพดอกไม้ที่เหลือบริเวณที่ไม่สนใจภายในรูปภาพหลงเหลืออยู่ค่อนข้างมาก

จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการแบ่งส่วนรูปภาพ ทำให้ผู้วิจัยเห็นจุดร่วมกันของวิธีการเหล่านี้ นั่นคือวิธีการแบ่งส่วนรูปภาพที่นำเสนอในงานวิจัยเหล่านี้จะมีทั้งหมด 2

ขั้นตอน โดยขั้นตอนแรกจะเป็นการแบ่งส่วนรูปภาพดอกไม้ และในขั้นตอนที่ 2 จะเป็นขั้นตอนการลดรายละเอียดของพื้นหลังรูปภาพ ผู้วิจัยมีความสนใจเทคนิคการแบ่งส่วนรูปภาพด้วยการใช้ Saliency map ในการเก็บบริเวณที่สนใจภายในรูปภาพ และเทคนิคการสร้างสัณฐานของดอกไม้เพื่อช่วยลดรายละเอียดของพื้นหลังดอกไม้ เนื่องจาก Saliency map สามารถช่วยให้มองเห็นองค์ประกอบสำคัญภายในภาพได้ชัดเจนมากขึ้น อีกทั้งยังช่วยลดรายละเอียดพื้นหลังของรูปภาพทำให้ช่วยให้ลดรายละเอียดของพื้นหลังภาพง่ายมากยิ่งขึ้น ในขณะที่การสร้างสัณฐานของดอกไม้ช่วยให้สามารถลดรายละเอียดของพื้นหลังภาพที่ความซับซ้อนมากได้อย่างมีประสิทธิภาพ ด้วยเหตุนี้ผู้วิจัยจึงสนใจที่จะต่อยอดแนวคิดวิธีการแบ่งส่วนรูปภาพโดยใช้เทคนิคของทั้ง 2 นี้ เพื่อช่วยให้การแบ่งส่วนรูปภาพที่มีความซับซ้อนมากได้มีประสิทธิภาพมากยิ่งขึ้น ซึ่งผู้วิจัยจะกล่าวถึงขั้นตอนการดำเนินงานวิจัยในบทถัดไป

บทที่ 3
วิธีการดำเนินการวิจัย

ในงานวิจัยชิ้นนี้ได้ศึกษาเกี่ยวกับวิธีการแบ่งส่วนรูปภาพเพื่อนำมาใช้ปรับปรุงขั้นตอนการแบ่งส่วนรูปภาพดอกไม้ เพื่อทำให้สามารถเก็บรายละอียดของบริเวณที่สนใจได้ดีขึ้น และสามารถลดรายละเอียดของพื้นหลังรูปภาพที่มีความซับซ้อนมากให้ดียิ่งขึ้น สำหรับทำการครอบตัดรูปภาพเพื่อนำไปใช้ต่อในการจำแนกประเภทรูปภาพ โดยเนื้อหาในส่วนนี้จะแสดงให้เห็นวิธีการและขั้นตอนของวิธีการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ รวมถึงขั้นตอนการนำรูปภาพที่ได้จากการแบ่งส่วนรูปภาพไปใช้ในการจำแนกประเภทดอกไม้

3.1 ระเบียบวิธีในการดำเนินการวิจัย

ขั้นตอนการวิจัยเพื่อนำไปพัฒนา และต่อยอดองค์ความรู้แนวคิดการแบ่งส่วนรูปภาพประกอบไปด้วยขั้นตอนการดำเนินการทั้งหมด 4 ขั้นตอน แสดงรายละเอียดได้ดังนี้

ขั้นตอนที่ 1 ทำการศึกษาความรู้พื้นฐานเกี่ยวกับทฤษฎีและงานวิจัยที่เกี่ยวข้อง ดังแสดงรายละเอียดขั้นตอนดังนี้

1.1 ความรู้พื้นฐานเกี่บวกับการประมวลผลภาพดิจิตอล (Digital Image Processing)

1.2 ความรู้พื้นฐานเกี่ยวกับการแบ่งส่วนรูปภาพ (Segmentation Basic Knowledge)

1.3 การจำแนกข้อมูลด้วยแบบจำลองโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network)

1.4 วิธีประเมินประสิทธิภาพของการแบ่งส่วนรูปภาพและจำแนกประเภทรูปภาพ

1.5 ทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการแบ่งส่วนรูปภาพ

1.6 ชุดข้อมูลคำสั่งและไลบรารี (Library) ที่เกี่ยวข้องสำหรับใช้จัดการกับข้อมูลรูปภาพ และใช้ในการจำแนกประเภทรูปภาพ

ขั้นตอนที่ 2 ศึกษาวิธีการแบ่งส่วนรูปภาพด้วยการทดลอง

2.1 เตรียมชุดข้อมูลดอกไม้สำหรับนำไปใช้ศึกษาการแบ่งส่วนรูปภาพ

2.2 ทำการศึกษาวิธีการแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang ด้วยการทดลองกับข้อมูลชุดรูปภาพที่เตรียมไว้

2.3 ทำการศึกษาต่อยอดจากวิธีการของ Yangyang ด้วยการบูรการความรู้ที่ได้ศึกษามาจากขั้นตอนที่ 1 และทำการทดลองกับชุดข้อมูลที่เตรียมไว้

2.4 ทำการทดลองด้วยการนำชุดข้อมูลรูปภาพตั้งต้นไปจำแนกประเภทดอกไม้

2.5 ทำการทดลองด้วยการนำชุดข้อมูลรูปภาพที่ได้จากวิธีการแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีการของ Yangyang ไปจำแนกประเภทดอกไม้

2.6 ทำการทดลองด้วยการนำชุดข้อมูลรูปภาพที่ได้จากวิธีการแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีการที่นำเสนอไปจำแนกประเภทดอกไม้

ขั้นตอนที่ 3 การประเมินประสิทธิภาพของวิธีการที่ทดลอง

3.1 ประเมินประสิทธิภาพการแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang และวิธีการที่นำเสนอ

3.2 ประเมินประสิทธิภาพการจำแนกรูปภาพของชุดข้อมูลตั้งต้น และชุดข้อมูลที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang และวิธีการที่นำเสนอ

3.3 ทำการวิเคราะห์ผลลัพธ์ที่ได้จากการทดลองและสรุปผลการทดลอง

ขั้นตอนที่ 4 จัดทำเอกสารวิทยานิพนธ์ เพื่อเป็นผลงานตีพิมพ์ เพื่อรวบรวมองค์ความรู้ที่ได้จากการศึกษา และเพื่อนำเสนอผลงานที่ได้จากการพัฒนาและต่อยอดจนเป็นองค์ความรู้ใหม่

3.2 ขั้นตอนการดำเนินการวิจัย

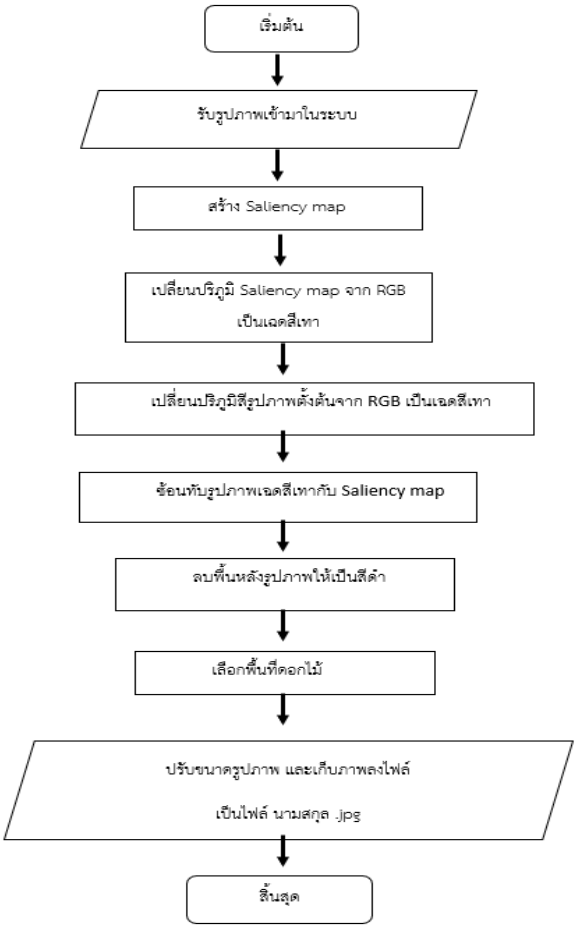
งานวิจัยชิ้นนี้จะแบ่งออกเป็น 2 ขั้นตอน ได้แก่ (1) ขั้นตอนแรกจะเป็นขั้นตอนการแบ่งส่วนรูปภาพของดอกไม้ โดยจะแบ่งเป็นการทดลองแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang และการทดลองแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ (2) ในขั้นตอนที่สองจะเป็นการนำรูปภาพที่ได้มาจากขั้นตอนแรกมาใช้จำแนกประเภทของดอกไม้ ซึ่งจะอธิบายรายละเอียดแยกตามหัวข้อ ดังนี้

3.2.1 ขั้นตอนการแบ่งส่วนรูปภาพของดอกไม้

เนื่องจากรูปภาพดอกไม้มักจะมีองค์ประกอบอื่นที่ไม่เกี่ยวข้องอยู่ภายในรูปภาพ การจะนำรูปภาพไปใช้ในการจำแนกประเภทดอกไม้ในทันทีอาจจะทำให้แบบจำลองไม่สามารถเรียนรู้คุณลักษณะ (Feature) สำคัญของดอกไม้ได้ดี ด้วยเหตุนี้จึงจำเป็นที่จะต้องลบรายละเอียดบางส่วนภายในพื้นหลังของภาพเพื่อลดความซับซ้อนก่อนนำไปใช้ในการจำแนกประเภท โดยในงานชิ้นนี้ได้นำเสนอขั้นตอนการสกัดเพื่อหาพื้นที่สำคัญภายในรูปภาพดอกไม้ โดยการทดลองแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีการของ Yangyang และทำการทดลองแบ่งส่วนดอกไม้ด้วยวิธีการที่นำเสนอ ซึ่งจะนำเสนอรายละเอียดเรียงตามหัวข้อต่อไปนี้

3.2.1.1 การแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีการของ Yangyang

การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang มีขั้นตอนการแบ่งส่วนรูปภาพเรียงตามลำดับวิธีการดังแสดงในภาพที่ 34



ภาพที่ 34 แสดงขั้นตอนการทดลองแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang

1. การสร้าง Saliency map

นำชุดข้อมูลรูปภาพดอกไม้ตั้งต้นจำนวน 4,931 รูปมาสร้าง Saliency map โดยใช้วิธีการจำแนกความต่างในรูปภาพที่ชื่อว่า RC ซึ่งได้นำเสนอเอาไว้ในสมการที่ (33) ถึง (39) โดยกำหนดค่าพจน์การถ่วงน้ำหนักเชิงพื้นที่ σ_s^2 เท่ากับ 0.4 เพื่อทำการสร้างโครงร่างของดอกไม้และแสดงรายละเอียดของบริเวณที่สนใจภายในรูปภาพให้เด่นชัดยิ่งขึ้น

2. การเปลี่ยนปริภูมิสีของรูปภาพตั้งต้นและ Saliency map

เนื่องจากต้องการให้พื้นหลังของรูปภาพซึ่งเกิดจากเค้าโครงที่ได้ของ Saliency map ถูกเปลี่ยนเป็นเฉดสีเทา จึงเปลี่ยน Saliency map จากปริภูมิสี RGB ให้เป็นเฉดสีเทา เพื่อให้ง่ายต่อการลดรายละเอียดพื้นหลัง และทำการเปลี่ยนสีปริภูมิของรูปภาพดอกไม้ตั้งต้นให้เป็นเฉดสีเทา

3. การลดรายละเอียดพื้นหลังรูปภาพ

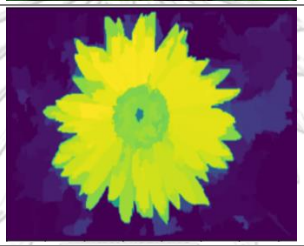
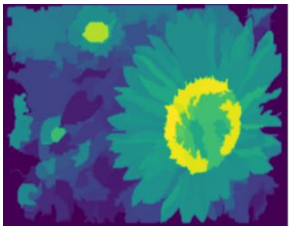
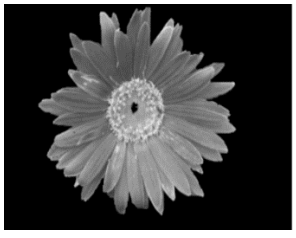
นำ Saliency map และรูปภาพตั้งต้นที่ได้จากกระบวนการในขั้นตอนที่ 2 มาทำการซ้อนทับกัน ตำแหน่งของดอกไม้จากภาพตั้งต้นเฉดสีเทาจะซ้อนทับกับตำแหน่งของดอกไม้ใน Saliency map จากนั้นจึงทำการลดรายละเอียดพื้นหลังรูปภาพ เนื่องจากในงานวิจัยของ Yangyang ไม่ได้นำเสนอขั้นตอนการลดรายละเอียดของพื้นหลังโดยละเอียด ในขั้นตอนการทดลองจึงทำการปรับค่าเทรดโฮลด์ของภาพเป็น TOZERO ซึ่งการปรับค่าเทรดโฮลด์ด้วยวิธีนี้จะเป็นการทำให้พิกเซลในภาพถูกตั้งค่าเป็น 0 ถ้าหากพิกเซลทั้งหมดมีค่าน้อยกว่าค่าเทรดโฮลด์ ซึ่งกำหนดค่าเทรดโฮลด์เท่ากับ 120

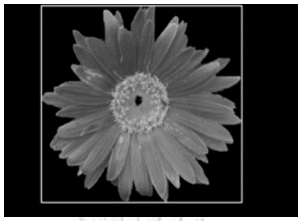
4. การเลือกพื้นที่ดอกไม้

นำภาพที่ลดรายละเอียดพื้นหลังรูปภาพจากขั้นตอนที่ 3 มาทำการคำนวณหาพื้นที่ที่ติดกันมากที่สุด ด้วยการเปรียบเทียบรูปร่าง (Contour) ภายในภาพ ถ้าหากขนาดของวัตถุใดในภาพมีขนาดใหญ่ที่สุดจะถูกสร้าง Bounding box เพื่อล้อมรอบบริเวณนั้น และทำการเก็บค่าพิกัดของ Bounding box เอาไว้ โดยจะเก็บค่าพิกัดมุมซ้ายบนและมุมขวาล่างเอาไว้ สำหรับนำไปใช้คำนวณในการวัดประสิทธิภาพการแบ่งส่วนรูปภาพ จากนั้นทำการครอบตัดบริเวณที่เลือกและเปลี่ยนขนาดรูปภาพให้มีขนาด 224x224 พิกเซล เพื่อนำไปใช้ในขั้นตอนการจำแนกรูปภาพ

ซึ่งรายละเอียดตัวอย่างของผลลัพธ์ที่ได้จากขั้นตอนที่ 1 ถึง 3 ได้จัดแสดงเอาไว้ในตารางที่ 1

ตารางที่ 1 แสดงตัวอย่างการทดลองแบ่งส่วนรูปภาพrด้วยวิธีการของ Yangyang ด้วยภาพที่มีความซับซ้อนของพื้นหลังน้อย และภาพที่มีความซับซ้อนของพื้นหลังมาก

ขั้นตอนการแบ่งส่วนรูปภาพ	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนน้อย	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนมาก
รูปภาพตั้งต้น		
Saliency map		
Saliency map ที่ผ่านการเปลี่ยนปริภูมิสีจาก RGB เป็นเฉดสีเทา		
รูปภาพดอกไม้ตั้งต้นที่ผ่านการเปลี่ยนปริภูมิสีจาก RGB เป็นเฉดสีเทา		
รูปภาพดอกไม้หลังจากผ่านขั้นตอนการแบ่งรูปภาพ		

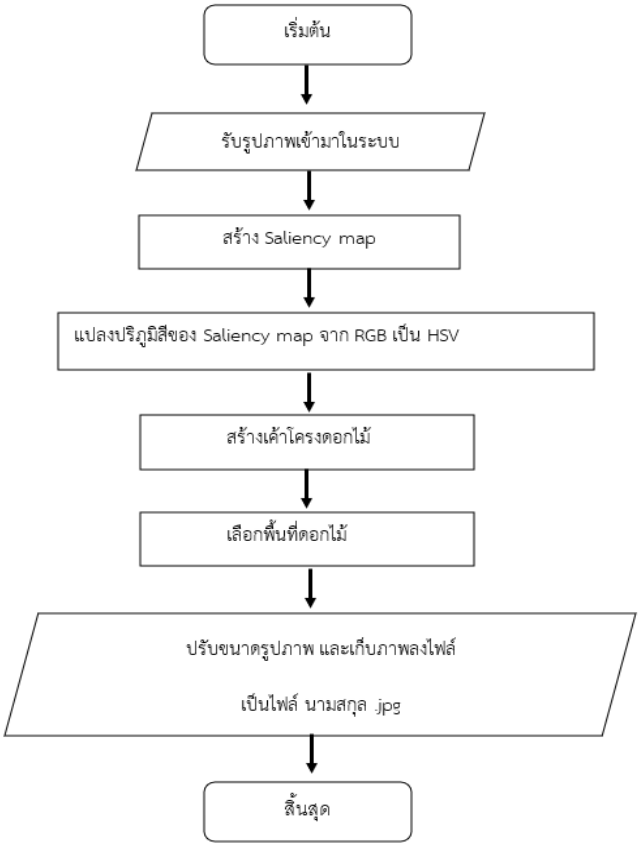
ขั้นตอนการแบ่งส่วนรูปภาพ	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนน้อย	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนมาก
พื้นที่ของดอกไม้		
ภาพดอกไม้ที่ถูกครอบตัด		

จากการศึกษาการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang พบว่าวิธีการนี้สามารถแบ่งส่วนรูปภาพได้ดีทั้งรูปภาพที่มีความซับซ้อนภายในพื้นหลังภาพตั้งแต่เล็กน้อยจนถึงปานกลาง แต่เมื่อใช้แบ่งส่วนรูปภาพที่มีความซับซ้อนมากขึ้น รูปภาพที่ได้จากการแบ่งส่วนยังคงเหลือรายละเอียดภายในพื้นหลังค่อนข้างมาก ทำให้บริเวณของรูปภาพที่ถูกครอบตัดมีรายละเอียดของพื้นหลังหลงเหลืออยู่ค่อนข้างมาก ซึ่งเป็นผลมาจากการเลือกใช้วิธีในการลดรายละเอียดของพื้นหลังรูปภาพ พบว่าจากการทดลองซึ่งทำการปรับค่าเทรดโฮลด์ของภาพเป็น TOZERO นั้นไม่สามารถใช้ได้กับรูปภาพที่มีความซับซ้อนภายในพื้นหลังภาพที่หลากหลาย

ด้วยเหตุผลที่ได้กล่าวมาในข้างต้น ในงานวิจัยชิ้นนี้จะเสนอวิธีการลดรายละเอียดพื้นหลังรูปภาพด้วยการแปลงปริภูมิสีของ Saliency map เพื่อให้องค์ประกอบของพื้นหน้าและพื้นหลังสามารถแยกออกจากกันได้ชัดเจนมากยิ่งขึ้น จากนั้นจึงใช้ Color mask มาวางซ้อนทับลงไปเพื่อทำให้เกิดเป็นสัณฐานของดอกไม้ โดยรายละเอียดของขั้นตอนการทดลองได้อธิบายรายละเอียดไว้ในหัวข้อที่ 3.2.1.2

3.2.1.2 การแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีการที่นำเสนอ

จากการศึกษาการแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang ปัญหาที่เกิดขึ้นจากกรณีที่ Saliency map ไม่สามารถลดรายละเอียดพื้นหลังของภาพที่มีความซับซ้อนมาก จึงนำมาสู่การศึกษา เพื่อหาวิธีการที่สามารถช่วยลดรายละเอียดของบริเวณของพื้นหลังรูปภาพที่มีความซับซ้อนมากได้ดียิ่งขึ้น ซึ่งการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอมีขั้นตอนการแบ่งส่วนรูปภาพดังแสดงในภาพที่ 35 รายละเอียดขั้นตอนการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ ได้อธิบายรายละเอียดขั้นตอนเอาไว้ดังนี้



ภาพที่ 35 ผังแสดงขั้นตอนการแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีที่นำเสนอ

1. การสร้าง Saliency map

ในงานชิ้นนี้ Saliency map จะได้มาจากวิธีการจำแนกความต่างในรูปภาพที่ชื่อว่า RC ซึ่งเป็นวิธีการเดียวกับที่ใช้ในงานวิจัยของ Yangyang ดังที่นำเสนอไว้ในสมการที่ (33) ถึง (39) โดยกำหนดค่าพจน์การถ่วงน้ำหนักเชิงพื้นที่ σ_s^2 เท่ากับ 0.4 ซึ่ง Saliency map จะ

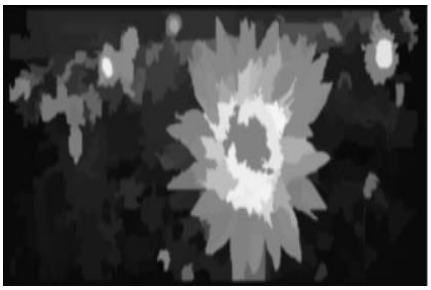
ช่วยแยกวัตถุที่สนใจออกจากพื้นหลังได้และช่วยลดรายละเอียดของพื้นหลังบางส่วนภายในภาพได้ จึงทำให้ง่ายต่อการหาบริเวณที่สนใจภายในภาพได้ดีขึ้น

2. การแปลงปริภูมิสีของ Saliency map จาก RGB เป็น HSV

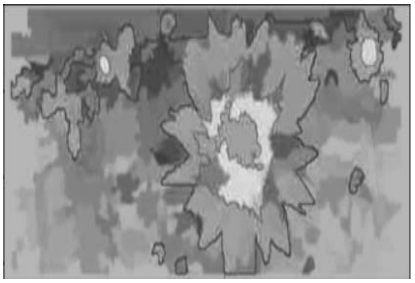
เนื่องจากความหลากหลายที่เกิดขึ้นภายในรูปภาพ ทั้งความหลากหลายของแสง เงา และจุดรบกวน (Noise) ที่เกิดขึ้นในภาพถ่าย เพื่อลดปัญหาเหล่านี้ จึงทำการเปลี่ยนปริภูมิสีของ Saliency map จาก RGB เป็นปริภูมิสี HSV ด้วยวิธีการที่นำเสนอไว้ในหัวข้อที่ 2.1.4 ข้อที่ 2 ด้วยปริภูมิสี HSV มีพารามิเตอร์หลายค่าได้แก่ ค่าสี ค่าความอิ่มของสี และ ค่าความสว่างของแสงของแต่ละพิกเซลภายในภาพ จึงทำให้สามารถเห็นเฉดสีที่แท้จริงของพิกเซลภายในรูปภาพได้ โดยปราศจากข้อกังวลเกี่ยวกับความเข้มที่มีระดับแตกต่างกัน

3. การสร้างเค้าโครงดอกไม้

จากวิธีการของ Yangyang ที่ได้ทำการปรับปริภูมิสีของ Saliency map ให้เป็นเฉดสีเทา เพื่อแสดงรายละเอียดของบริเวณที่สนใจดังแสดงในภาพที่ 36 (ก) ในที่นี้จะใช้รูปภาพดอกไม้จำนวน 1 ภาพเพื่อเป็นตัวอย่างสำหรับแสดงให้เปรียบเทียบความต่างของสีของภาพที่ได้มาจากวิธีการปรับเฉดสีที่แตกต่างกัน และแสดงวิธีการคำนวณค่าความต่างของระดับสีภายในภาพ (Contrast) ซึ่งคำนวณได้จากค่าเบี่ยงเบนมาตรฐานของสีภายในภาพซึ่งได้ค่าเท่ากับ 66 เมื่อนำภาพที่ได้จากขั้นตอนที่ 2 มาทำการแปลงเฉดสีให้เป็นสีเทาดังแสดงในภาพที่ 36 (ข) และคำนวณหาค่าความต่างของสีด้วยการคำนวณค่าเบี่ยงเบนมาตรฐานของสีภายในภาพซึ่งได้ค่าเท่ากับ 27 ซึ่งแสดงให้เห็นว่าการปรับปริภูมิสีของ Saliency map ให้เป็นเฉดสีเทาช่วยให้เห็นความต่างของพื้นหน้าและพื้นหลังภาพได้ดีกว่า Saliency map ที่เปลี่ยนปริภูมิสีเป็น HSV เป็นภาพเฉดสีเทา



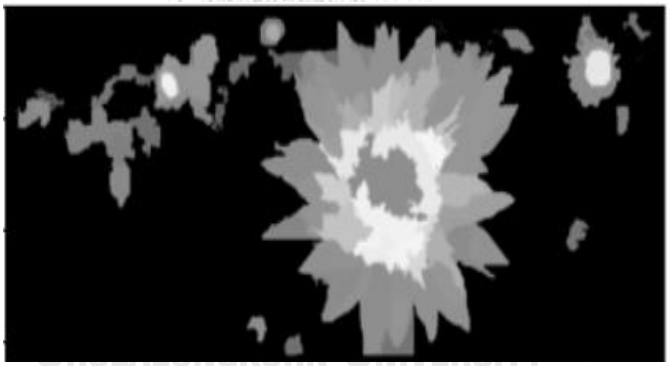
(ก) Saliency map เฉดสีเทา



(ข) Saliency map ที่เปลี่ยนปริภูมิสีเป็น HSV เป็นภาพเฉดสีเทา

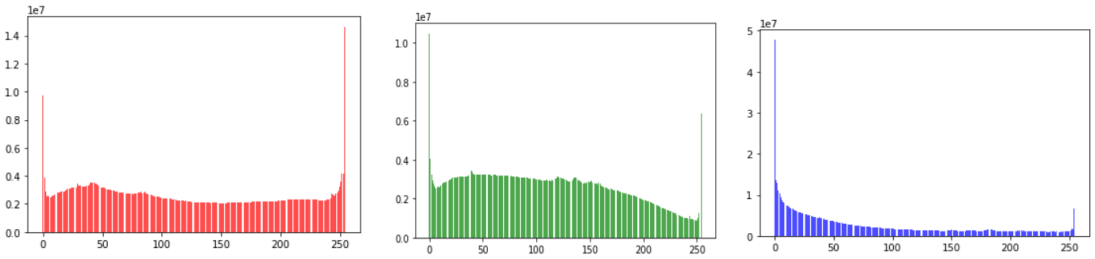
ภาพที่ 36 แสดงตัวอย่างความต่างของสีของภาพที่ถูกปรับเฉดสีให้แตกต่างกัน

เมื่อนำ Saliency map เฉดสีเทาไปลดรายละเอียดพื้นหลังรูปภาพด้วยวิธีการในหัวข้อ 3.2.1.1 ข้อที่ 3 ซึ่งได้ผลลัพธ์ดังแสดงในภาพที่ 37 เมื่อนำตัวอย่างรูปภาพตัวอย่างมาคำนวณความต่างของระดับสีภายในภาพอีกครั้งพบว่าได้ค่าเท่ากับ 94



ภาพที่ 37 แสดงความต่างของสีภายใน Saliency map เฉดสีเทาเมื่อมีการลดรายละเอียดพื้นหลังรูปภาพ

ในวิธีการที่นำเสนอ เพื่อที่จะทำให้พื้นหน้าและพื้นหลังภาพมีความแตกต่างกันอย่างชัดเจนมากยิ่งขึ้น จึงทำการสร้าง Color mask ซึ่งได้มาจากการดูการกระจายฮิสโตแกรมของเฉดสีของปริภูมิ RGB ในชุดข้อมูล ดังแสดงในภาพที่ 38



R G B

ภาพที่ 38 แสดงฮิสโตแกรมขององค์ประกอบสีของรูปภาพในชุดข้อมูลทั้งหมด

จากภาพที่ 38 แสดงให้เห็นว่าชุดข้อมูลรูปภาพมีองค์ประกอบสีส่วนใหญ่ภายในภาพเป็นสีเขียวและสีแดง ในขณะที่สีน้ำเงินเป็นองค์ประกอบสีที่มีน้อยที่สุดภายในภาพ ในงานวิจัยชิ้นนี้ทำการสร้าง Color mask โดยการกำหนดให้ความเข้มสีที่มากที่สุดและความเข้มที่น้อยที่สุดของสี HSV มีค่าเป็นสีเขียว โดยในงานวิจัยชิ้นนี้จะทำการกำหนดค่าความเข้มสูงสุดของสี HSV เป็นอาเรย์ [90, 225, 225] ในขณะที่ความเข้มที่น้อยที่สุดของสี HSV เป็นอาเรย์ [15, 0, 0] ดังแสดงในภาพที่ 39

(ก) ความเข้มน้อยสุดของสี HSV (ข) ความเข้มสูงสุดของสี HSV

ภาพที่ 39 แสดงระดับความเข้มของสี HSV

จากนั้นจึงใช้ฟังก์ชัน cv2.inRange(src, lowerbound, upperbound) ในการสร้างสัณฐานดอกไม้ โดยสามารถอธิบายพารามิเตอร์ในแต่ละอาร์กิวเมนต์ (Argument) ได้ดังนี้

- src แทน Saliency map ที่เปลี่ยนปริภูมิสีเป็น HSV
- lowerbound แทน อาร์เรย์ความเข้มน้อยสุดของสี HSV
- upperbound แทน อาร์เรย์ความเข้มสูงสุดของสี HSV

โดยฟังก์ชัน cv2.inRange() จะทำให้ได้ภาพไบนารี โดยที่พิกเซลสีขาวซึ่งมีค่าพิกเซลเท่ากับ 255 จะแทนพิกเซลที่อยู่ในช่วงความเข้มสูงสุดของสี HSV และความเข้มน้อยสุดของสี HSV ในขณะที่พิกเซลของสีที่ไม่ได้อยู่ในช่วงที่กำหนดจะกลายเป็นสีดำซึ่งมีค่าพิกเซลเป็น 0 ซึ่งวิธีนี้จะแสดงเค้าโครงของดอกไม้และลดรายละเอียดที่ซับซ้อนภายในพื้นหลังของภาพ และทำให้เห็นความต่างของสีภายในภาพซึ่งแยกพื้นหน้าและพื้นหลังของภาพได้ดีมากยิ่งขึ้น เมื่อนำตัวอย่างรูปภาพตัวอย่างไปคำนวณหาค่าความต่างของสีด้วยการคำนวณค่าเบี่ยงเบนมาตรฐานของสีภายในภาพจะได้ค่าเท่ากับ 112 ซึ่งมากกว่าค่าความต่างของสีภายใน Saliency map เฉดสีเทาที่ผ่านการลดรายละเอียดพื้นหลังรูปภาพดังแสดงในภาพที่ 40

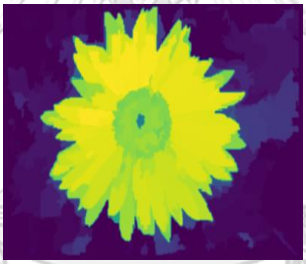
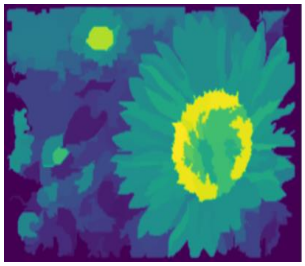
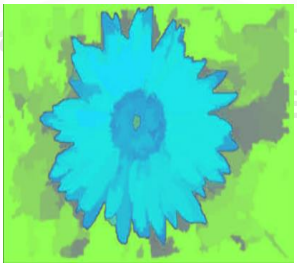
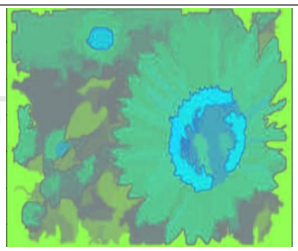
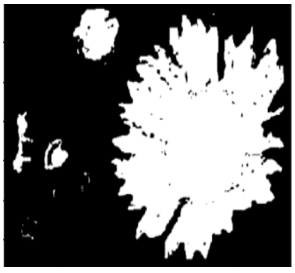
ภาพที่ 40 แสดงตัวอย่างรูปภาพสัณฐานดอกไม้ที่ได้จากการเปลี่ยนปริภูมิสีของ Saliency map เป็น HSV และทำการสร้าง Color mask เพื่อลดรายละเอียดพื้นหลังภาพ

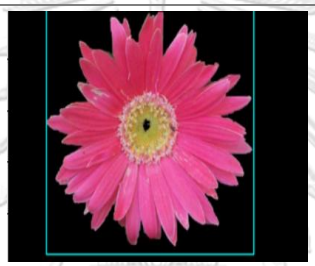
4. การเลือกพื้นที่ดอกไม้

นำภาพที่ลดรายละเอียดพื้นหลังรูปภาพจากขั้นตอนที่ 3 มาซ้อนทับกับรูปภาพตั้งต้น จากนั้นคำนวณหาพื้นที่ที่ติดกันมากที่สุด ด้วยการเปรียบเทียบขนาดของวัตถุภายในภาพ ถ้าหากขนาดของวัตถุใดในภาพมีขนาดใหญ่ที่สุดจะถูกสร้าง Bounding box เพื่อล้อมรอบบริเวณนั้น และทำการเก็บค่าพิกัดของ Bounding box เอาไว้ โดยจะเก็บค่าพิกัดมุมซ้ายบนและมุมขวาล่างเอาไว้ สำหรับนำไปใช้คำนวณในการวัดประสิทธิภาพการแบ่งส่วนรูปภาพ จากนั้นทำการครอบตัดบริเวณที่เลือกและเปลี่ยนขนาดรูปภาพให้มีขนาด 224x224 พิกเซล เพื่อนำไปใช้ในขั้นตอนการจำแนกรูปภาพ

ซึ่งรายละเอียดตัวอย่างของผลลัพธ์ที่ได้จากขั้นตอนที่ 1 ถึง 4 ได้จัดแสดงเอาไว้ในตารางที่ 2

ตารางที่ 2 แสดงตัวอย่างการทดลองแบ่งส่วนรูปภาพrด้วยวิธีการที่นำเสนอ ด้วยภาพที่มีความซับซ้อนของพื้นหลังน้อย และภาพที่มีความซับซ้อนของพื้นหลังมาก

ขั้นตอนการแบ่งส่วนรูปภาพ	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนน้อย	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของYangyang กรณีพื้นหลังมีความซับซ้อนมาก
รูปภาพตั้งต้น		
Saliency map		
Saliency map ที่ผ่านการเปลี่ยนปริภูมิสีจาก RGB เป็น HSV		
เค้าโครงของดอกไม้		

ขั้นตอนการแบ่งส่วนรูปภาพ	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang กรณีพื้นหลังมีความซับซ้อนน้อย	ตัวอย่างผลลัพธ์การแบ่งส่วนรูปภาพด้วยวิธีของYangyang กรณีพื้นหลังมีความซับซ้อนมาก
รูปภาพดอกไม้หลังจากผ่านขั้นตอนการแบ่งรูปภาพ		
พื้นที่ของดอกไม้		
ภาพดอกไม้ที่ถูกครอบตัด		

จากการทดลองพบว่าวิธีการที่นำเสนอสามารถช่วยลดรายละเอียดของพื้นหลังรูปภาพที่มีความซับซ้อนน้อยจนถึงมากได้ค่อนข้างดี ซึ่งช่วยให้ Bounding box สามารถล้อมรอบบริเวณที่สนใจได้ดียิ่งขึ้น ส่งผลให้รูปภาพที่ถูกครอบตัดสำหรับนำไปใช้จำแนกประเภทดอกไม้ มีรายละเอียดของสิ่งที่สนใจภายในภาพมากขึ้น

3.2.2 ขั้นตอนการจำแนกประเภทดอกไม้

ในงานวิจัยชิ้นนี้ขั้นตอนการจำแนกประเภทดอกไม้จะใช้ข้อมูลจำนวน 3 ชุด ได้แก่ ชุดข้อมูลดอกไม้ตั้งต้น และชุดข้อมูลที่ผ่านการแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีของ Yangyang และวิธีการที่นำเสนอ สามารถอธิบายรายละเอียดได้ดังขั้นตอนต่อไปนี้

1. การเตรียมข้อมูลทดลอง

ชุดข้อมูลทดลองจะประกอบไปด้วยชุดข้อมูลดอกไม้ตั้งต้น ซึ่งเป็นชุดรูปภาพดอกไม้จาก Kaggle ชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang และชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีที่เสนอ ซึ่งชุดข้อมูลเหล่านี้ประกอบไปด้วยรูปภาพจำนวน 4,931 รูป ชุดข้อมูลประกอบไปด้วยรูปภาพดอกไม้ 5 ประเภท ได้แก่ ดอกเดซี่ ดอกทานตะวัน ดอกทิวลิป ดอกกุหลาบ และดอกแดนดิไลออน ก่อนนำชุดข้อมูลรูปภาพไปจำแนกประเภทด้วยแบบจำลอง ได้มีการเพิ่มจำนวนชุดข้อมูล (Data augmentation) ด้วยการสุ่มรูปภาพดอกไม้มาประเภทละ 500 รูปภาพ จากนั้นทำการแปลงรูปภาพด้วยการกลับด้านรูปภาพ (Flip) หมุนรูปภาพ (Rotate) ด้วยมุม 90° ปรับค่าความสว่าง (Brightness) ที่ช่วง 0.4 และครอบตัด (Crop) จากศูนย์กลางด้วยอัตราส่วน 0.8 ดังแสดงตัวอย่างไว้ในภาพที่ 41 ด้วยวิธีการเหล่านี้ทำให้ชุดข้อมูลทั้ง 3 แบบมีข้อมูลเพิ่มขึ้นจากเดิมเป็น 7,431 รูป จากนั้นทำการนอมัลไลซ์สีของรูปภาพให้มีค่าสีในแต่ละพิกเซลอยู่ในช่วง [0, 1] เพื่อลดระยะเวลาในการประมวลผลของแบบจำลอง

ภาพที่ 41 แสดงตัวอย่าดอกไม้ที่ผ่านขั้นตอนการเพิ่มข้อมูลด้วยวิธีการกลับด้านรูปภาพ หมุนรูปภาพด้วยมุม 90° ปรับค่าความสว่างที่ช่วง 0.4 และครอบตัดจากศูนย์กลางด้วยอัตราส่วน 0.8

2. ชุดข้อมูลสำหรับใช้ในการทดลอง

งานวิจัยชิ้นนี้จะทำการแบ่งข้อมูลทั้ง 3 ชุด ได้แก่ ชุดข้อมูลดอกไม้ตั้งต้น และชุดข้อมูลที่ผ่านขั้นตอนการแบ่งส่วนรูปภาพดอกไม้ด้วยวิธีของ Yangyang และวิธีที่นำเสนอด้วยอัตราส่วนเดียวกัน นั่นคือ 60% เป็นชุดข้อมูลสอน (Training set) 20% เป็นชุดข้อมูลทดสอบ (Testing set) และ 20% เป็นชุดตรวจสอบความถูกต้อง (Validation set) จะได้ชุดข้อมูลสอนจำนวน 4,458 รูป ชุดข้อมูลทดสอบจำนวน 1,487 รูป และชุดตรวจสอบความถูกต้องจำนวน 1,486 รูป ดังแสดงในตารางที่ 3 ถึง 5

ตารางที่ 3 แสดงอัตราส่วนการแบ่งชุดข้อมูลตั้งต้น

ชุดข้อมูลตั้งต้นจาก Kaggle	อัตราส่วนการแบ่งข้อมูล	จำนวนรูปภาพ
ชุดข้อมูลสอน (Training set)	60%	4,458 รูป
ชุดข้อมูลทดสอบ (Testing set)	20%	1,487 รูป
ชุดตรวจสอบความถูกต้อง (Validation set)	20%	1,486 รูป

ตารางที่ 4 แสดงอัตราส่วนการแบ่งชุดข้อมูลที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang

ชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang	อัตราส่วนการแบ่งข้อมูล	จำนวนรูปภาพ
ชุดข้อมูลสอน (Training set)	60%	4,458 รูป
ชุดข้อมูลทดสอบ (Testing set)	20%	1,487 รูป
ชุดตรวจสอบความถูกต้อง (Validation set)	20%	1,486 รูป

ตารางที่ 5 แสดงอัตราส่วนการแบ่งชุดข้อมูลที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ

ชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ	อัตราส่วนการแบ่งข้อมูล	จำนวนรูปภาพ
ชุดข้อมูลสอน (Training set)	60%	4,458 รูป
ชุดข้อมูลทดสอบ (Testing set)	20%	1,487 รูป
ชุดตรวจสอบความถูกต้อง (Validation set)	20%	1,486 รูป

3. ขั้นตอนการทดลอง

ในการทดลองใช้ Python เป็นซอร์ฟแวร์โปรแกรมและไลบรารีการเขียนโปรแกรมส่วนใหญ่เป็น Keras ที่มีแบ็คเอนด์ TensorFlow และ Scikit-learn โดยการทำงานจะประมวลผลบนหน่วยประมวลผลกราฟฟิก (GPU) แทนการประมวลผลส่วนกลาง (CPU) เนื่องจากการใช้ GPU จะเป็นประโยชน์สำหรับการเรียนรู้เชิงลึก เนื่องจากจะช่วยลดเวลาในการประมวลผล โดย GPU ที่ใช้ในการทดลองสำหรับงานวิจัยชิ้นนี้คือ NVidia K80

3.1 แบบจำลองที่ใช้ในการจำแนกประเภทดอกไม้

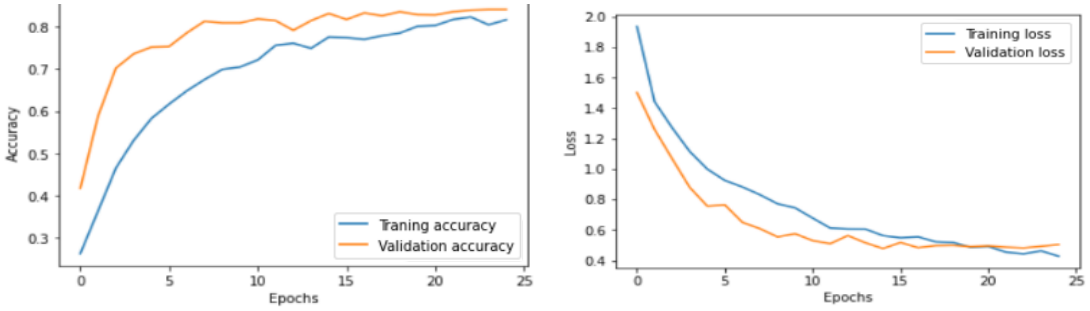
เนื่องจากชุดข้อมูลรูปภาพที่ใช้ในงานวิจัยของ Yangyang กับชุดข้อมูลรูปภาพที่ใช้ในงานวิจัยชิ้นนี้เป็นข้อมูลคนละชุด อีกทั้งแบบจำลองที่ใช้ในงานวิจัยของ Yangyang คือแบบจำลอง PCANet เพื่อที่จะเปรียบความสามารถในการจำแนกประเภทดอกไม้ ในงานวิจัยชิ้นนี้จึงเลือกใช้แบบจำลอง VGG16 ด้วยการถ่ายทอดความรู้จากแบบจำลองที่ถูกสอนมาแล้ว (Transfer learning) ซึ่งได้แสดงรายละเอียดไว้ในหัวข้อที่ 2.3 ข้อ 2.3.2 ด้วยการทำ Fine-tuning ซึ่งเป็นกระบวนการที่ใช้เมื่อมีการนำแบบจำลอง Pre-trained มาใช้สอนชุดข้อมูลของตัวเอง โดยจะใช้ชั้นคอนโวลูชันเป็นบล็อกแรกในขณะที่บล็อกอื่นจะถูกบล็อค ในชั้นเชื่อมโยงสมบูรณ์ (Fully connected) จะประกอบด้วยชั้นประมวลผลที่ซ่อนอยู่ (Hidden layer) จำนวน 2 ชั้น โดยในชั้นแรกจะปรับให้จำนวนโหนด (Node) เท่ากับ 256 โหนด และในชั้นที่สองจะปรับจำนวนโหนดให้เท่ากับ 64 โหนด โดยชั้นแรกและชั้นที่สองจะ Drop out จำนวนโหนดด้วยสัดส่วน 0.65 % และ 0.3 % ตามลำดับเพื่อลดปัญหา Overfit ซึ่งเกิดจากแบบจำลองมีค่าน้ำหนัก (Weight) มากจนเกินไป และใช้ Activation เป็น ReLU ทั้งสองชั้น ในชั้น Output ประกอบด้วยโหนดจำนวน 5 โหนดซึ่งเป็นจำนวนประเภทของดอกไม้และใช้ฟังก์ชันซอร์ฟแมกซ์เป็นฟังก์ชันกระตุ้น แบบจำลองจะถูกสอนด้วย Optimizer Adam ด้วย Learning rate 0.001 จำนวน 40 Epoch และ Batch size สำหรับเป็นชุดข้อมูลสอนจำนวน 128 ตัวอย่าง (Sample) ดังแสดงในตารางที่ 6

ตารางที่ 6 แสดงโครงสร้างแบบจำลอง CNN ที่ใช้ในงานวิจัย

Layer (type)	Function	Output shape
Input		224x224x3
VGG16	Functional	7x7x512
Flatten_1	Flatten	25088x1x1
Dense_1	Fully connected	256x1x1
Activation_3	Activation	256x1x1
Dropout_1	Drop out	256x1x1
Dense_2	Fully connected	64x1x1
Activation_4	Activation	64x1x1
Droupout_2	Drop out	64x1x1
Output	Softmax regression	5x1x1

3.2 วิธีการทดลอง

นำชุดข้อมูลสอนของชุดข้อมูลตั้งต้นจำนวน 4,458 รูป มาทำการสอนแบบจำลองโดยกำหนดรอบในการสอน (Epoch) เท่ากับ 40 รอบ และตั้งค่าหยุดรอบในการสอนหากค่า Loss ไม่มีการปรับค่าให้น้อยลงมากกว่า 20 รอบ โดยในแต่ละรอบที่สอนจะใช้ข้อมูลสอนครั้งละ 128 ข้อมูล ในการสอนแบบจำลองจะตรวจสอบความแม่นยำของแบบจำลองระหว่างการสอนด้วยการใช้ชุดข้อมูลตรวจสอบความถูกต้องของชุดข้อมูลตั้งต้นจำนวน 1,486 รูป เพื่อตรวจสอบปัญหา Underfit หรือ Overfit ของแบบจำลอง ดังแสดงในภาพที่ 42

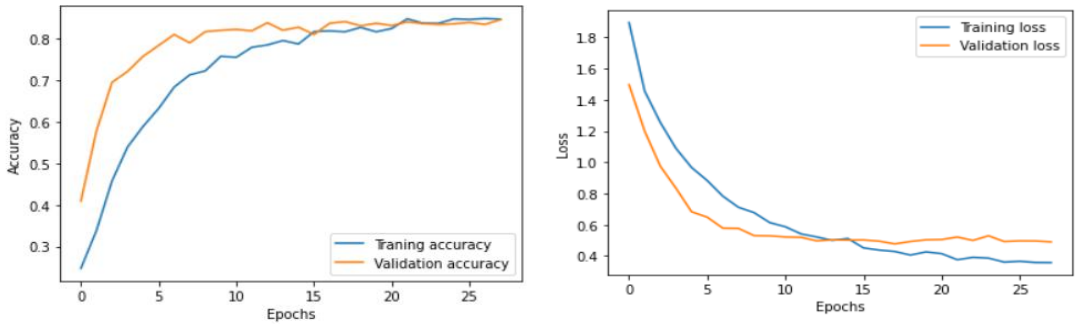


(ก) (ข)

ภาพที่ 42 (ก) แสดงกราฟเปรียบเทียบค่าความถูกต้องของชุดข้อมูลสอนกับค่าความถูกต้องของชุดข้อมูลตรวจสอบความถูกต้องในแต่ละรอบการสอน และ (ข) กราฟเปรียบเทียบค่า Loss ของชุดข้อมูลสอนกับค่า Loss ของชุดข้อมูลตรวจสอบความถูกต้องในแต่ละรอบการสอน จากนั้นจึงนำชุดข้อมูลทดสอบของชุดข้อมูลตั้งต้นจำนวน

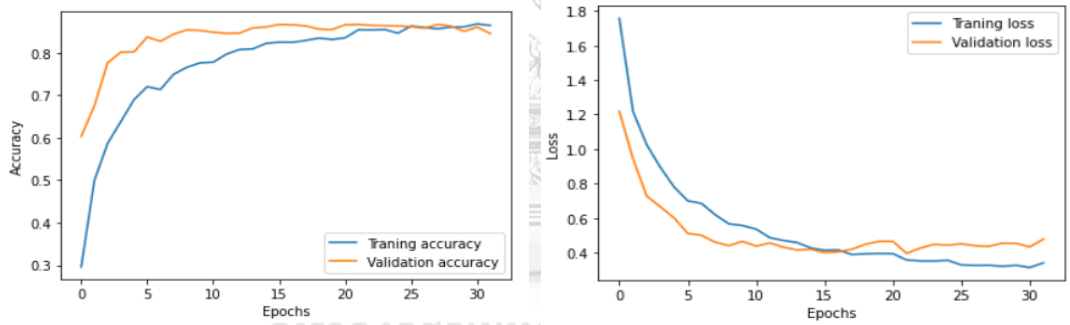
จากภาพที่ 42 (ก) และ (ข) พบว่ามีการหยุดรอบการสอนเอาไว้ที่รอบการสอนที่ 25 จากกราฟพบว่าแบบจำลองมีปัญหา Overfit เล็กน้อยเนื่องจากกราฟของค่าความถูกต้องของชุดข้อมูลสอนนั้นมีค่ามากกว่ากราฟของค่าความถูกต้องของชุดข้อมูลตรวจสอบความถูกต้อง และกราฟของค่า Loss ของชุดข้อมูลสอนนั้นมีค่าน้อยกว่ากราฟของค่า Loss ของชุดข้อมูลตรวจสอบความถูกต้อง จากนั้นจึงนำชุดข้อมูลทดสอบของชุดข้อมูลตั้งต้นจำนวน 1,487 รูป มาทำการประเมินประสิทธิภาพของการจำแนกประเภทรูปภาพดอกไม้ของชุดข้อมูลตั้งต้น ด้วยการวัดประสิทธิภาพด้วยค่าความถูกต้อง ความแม่นยำ ความครบถ้วน และค่า F1

จากนั้นจึงนำชุดข้อมูลอีก 2 ชุด ได้แก่ ชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang และชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ มาทำการจำแนกประเภทรูปภาพดอกไม้ ด้วยการใช้แบบจำลอง VGG16 ที่มีโครงสร้างและพารามิเตอร์ (Parameter) เดียวกับชุดข้อมูลตั้งต้น โดยมีกระบวนการสอน การตรวจสอบความแม่นยำของแบบจำลองระหว่างการสอนและการประเมินประสิทธิภาพของการจำแนกประเภทรูปภาพดอกไม้เช่นเดียวกับชุดข้อมูลรูปภาพตั้งต้น ดังแสดงในภาพที่ 43 และ 44



(ก) (ข)

ภาพที่ 43 (ก) แสดงกราฟเปรียบเทียบค่าความถูกต้องของชุดข้อมูลสอนกับค่าความถูกต้องของชุดข้อมูลตรวจสอบความถูกต้องของชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนด้วยวิธีของ Yangyang ในแต่ละรอบการสอน และ (ข) กราฟเปรียบเทียบค่า Loss ของชุดข้อมูลสอนกับค่า Loss ของชุดข้อมูลตรวจสอบความถูกต้องของชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang ในแต่ละรอบการสอน



(ก) (ข)

ภาพที่ 44 (ก) แสดงกราฟเปรียบเทียบค่าความถูกต้องของชุดข้อมูลสอนกับค่าความถูกต้องของชุดข้อมูลตรวจสอบความถูกต้องของชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนด้วยวิธีที่นำเสนอ ในแต่ละรอบการสอน และ (ข) กราฟเปรียบเทียบค่า Loss ของชุดข้อมูลสอนกับค่า Loss ของชุดข้อมูลตรวจสอบความถูกต้องของชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีนำเสนอ ในแต่ละรอบการสอน

จากภาพที่ 43 และ 44 (ก) และ (ข) พบว่ามีการหยุดรอบการสอนของแบบจำลองที่ใช้ชุดข้อมูลตั้งต้นที่ผ่านการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang และวิธีที่นำเสนอที่รอบการสอนที่ 28 จากกราฟพบว่าแบบจำลองที่ใช้สอนข้มูลทั้ง 2 ชุดมีปัญหา Overfit เล็กน้อยเนื่องจากกราฟของค่าความถูกต้องของชุดข้อมูลสอนนั้นมีค่ามากกว่ากราฟของค่าความมถูกต้องของชุดข้อมูลตรวจสอบความถูกต้อง

และกราฟของค่า Loss ของชุดข้อมูลสอนนั้นมีค่าน้อยกว่ากราฟของค่า Loss ของชุดข้อมูลตรวจสอบความถูกต้อง จากนั้นจึงนำชุดข้อมูลทดสอบของชุดข้อมูลตั้งต้นจำนวน 1,487 รูป มาทำการประเมินประสิทธิภาพของการจำแนกประเภทรูปภาพดอกไม้ของชุดข้อมูลตั้งต้น โดยการวัดประสิทธิภาพจากค่าความถูกต้อง ความแม่นยำ ความครบถ้วน และค่า F1

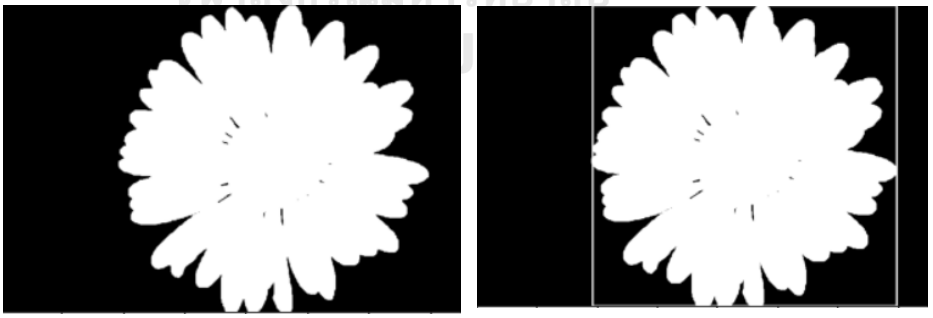
เมื่อทำการทดลองด้วยการแบ่งส่วนรูปภาพดอกไม้ และนำชุดข้อมูลรูปภาพทั้ง 3 ชุดไปทำการจำแนกประเภทดอกไม้อันเป็นการจบกระบวนการทดลองแล้ว ในขั้นตอนถัดไปจะเป็นการวัดประสิทธิภาพการแบ่งส่วนรูปภาพและการจำแนกประเภทดอกไม้ เพื่อเปรียบเทียบผลลัพธ์ที่ได้การวัดประสิทธิภาพในการสกัดคุณลักษณะของพื้นที่ดอกไม้ด้วยวิธีการที่นำเสนอโดย Yangyang และวิธีที่นำเสนอในงานชิ้นนี้กับชุดข้อมูลตั้งต้นจาก Kaggle ซึ่งจะอธิบายรายละเอียดผลการวิจัยพร้อมทั้งอภิปรายผลการศึกษาของกระบวนการทั้งหมดในบทถัดไป

บทที่ 4
ผลการวิจัยและอภิปรายผล

ในงานวิจัยชิ้นนี้การประเมินผลการทดลองจะถูกแบ่งออกเป็น 2 ขั้นตอน ขั้นตอนแรกจะป็นการเปรียบเทียบประสิทธิภาพของการแบ่งส่วนรูปภาพ และขั้นตอนที่สองจะเป็นการเปรียบเทียบประสิทธิภาพในการจำแนกประเภทของดอกไม้ ซึ่งทั้งสองขั้นตอนจะทำการเปรียบเทียบผลลัพธ์ระหว่างวิธีที่นำเสนอโดย Yangyang และวิธีที่นำเสนอในงานวิจัยชิ้นนี้

4.1 ผลการทดลองและการเปรียบเทียบประสิทธิภาพของการแบ่งรูปภาพ

ในขั้นตอนนี้จะทำการหาผลการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสอโดย Yangyang และวิธีการที่นำเสนอในงานชิ้นนี้ โดยจะทำการเปรียบเทียบผลลัพธ์ที่ได้กับชุดข้อมูลภาพที่ถูกต้อง (Ground truth) ด้วยการคำนวณค่า Bounding box ด้วยวิธี Intersection over Union (IoU) ซึ่งได้นำเสนอไว้ในหัวข้อที่ 2.4 ข้อที่ 2.4.1 เนื่องจากชุดข้อมูลรูปภาพที่นำมาใช้ในการทดลองไม่มีชุดข้อมูลรูปภาพจริง ในงานวิจัยชิ้นนี้จึงทำการสร้างชุดข้อมูลจริงด้วยการสุ่มรูปภาพดอกไม้มาจำนวน 240 รูปภาพ ชนิดละ 48 รูป สำหรับนำไปทำการปรับแต่งภาพด้วยโปรแกรม Photoshop เพื่อให้มีชุดข้อมูลสำหรับนำไปใช้ในการวัดประสิทธิภาพการแบ่งส่วนรูปภาพ และทำการเก็บค่า Bounding box ของชุดข้อมูลจริงเพื่อนำไปใช้ในการวัดประสิทธิภาพของการแบ่งส่วนรูปภาพ ดังแสดงในภาพที่ 45

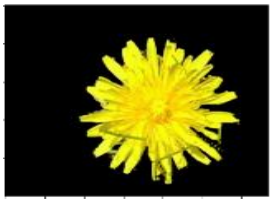
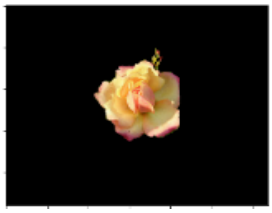


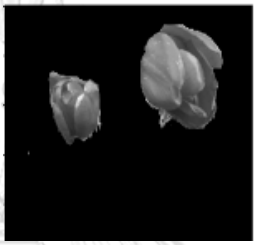
ภาพที่ 45 แสดงตัวอย่างขอชุดข้อมูลจริงที่ใช้ในการวัดประสิทธิภาพการแบ่งส่วนรูปภาพ

ผลการทดลอง

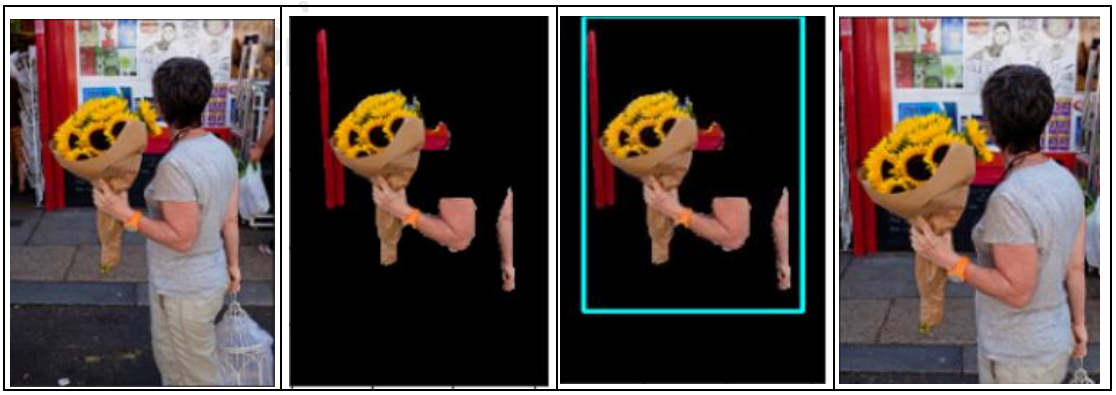
เมื่อพิจารณาค่าเฉลี่ย IoU ของทั้ง 2 วิธี พบว่าวิธีการที่นำเสนอโดย Yangyang ให้ค่าเฉลี่ย IoU เท่ากับ 41% ในขณะที่วิธีการที่นำเสนอให้ค่าเฉลี่ย IoU เท่ากับ 54% เนื่องจากในงานวิจัยของ Yangyang การแบ่งส่วนรูปภาพดอกไม้นั้นทดลองบนชุดข้อมูลดอกไม้ที่ชื่อว่า Oxford 17 ซึ่งมีองค์ประกอบของพื้นหลังรูปภาพที่ซับซ้อนน้อยกว่าชุดข้อมูลจาก Kaggle จึงอาจเป็นหนึ่งสาเหตุที่ทำให้ค่าเฉลี่ย IoU ของวิธีการที่นำเสนอโดย Yangyang มีค่าน้อยกว่าเมื่อนำมาทดลองด้วยชุดข้อมูลที่มีความซับซ้อนของพื้นหลังรูปภาพมากขึ้น โดยวิธีที่นำเสนอให้ผลลัพธ์ที่ดีเมื่อใช้ชุดรูปภาพตั้งต้นจาก Kaggle แสดงให้เห็นว่าวิธีนี้สามารถลดรายละเอียดของพื้นหลังรูปภาพที่มีความซับซ้อนมากได้มากได้ค่อนข้างดี ด้วยเหตุนี้จึงทำให้ Bounding box สามารถล้อมรอบบริเวณที่สนใจโดยมีรายละเอียดที่ไม่เกี่ยวข้องในรูปภาพน้อยกว่าวิธีการของ Yangyang โดยแสดงตัวอย่างการแบ่งส่วนรูปภาพของทั้ง 2 วิธีไว้ในตารางที่ 7

ตารางที่ 7 แสดงตัวอย่างการแบ่งส่วนรูปภาพของทั้ง 2 วิธี

ตัวอย่างประเภทของดอกไม้	รูปภาพตั้งต้น	วิธีการที่นำเสนอโดย Yang Yang	วิธีการที่นำเสนอ
เดซี่			
แดนดิไลออน			
กุหลาบ			

ตัวอย่างประเภทของดอกไม้	รูปภาพตั้งต้น	วิธีการที่นำเสนอโดย Yang Yang	วิธีการที่นำเสนอ
ทานตะวัน			
ทิวลิป			

สำหรับข้อจำกัดของงานวิจัยที่นำเสนอ พบว่าวิธีการสามารถแบ่งส่วนรูปภาพได้ดีหากภายในรูปภาพมีวัตถุที่สนใจเป็นองค์ประกอบส่วนใหญ่ภายในภาพ หากบริเวณที่สนใจเป็นเพียงบริเวณเล็กภายในรูปภาพโอกาสที่จะเลือกบริเวณนั้นจะน้อยลงไป ซึ่งปัญหาเหล่านี้เป็นผลจากการใช้วิธีเลือกพื้นที่โดยการเลือกพื้นที่ที่ติดกันมากที่สุด ดังแสดงในภาพที่ 46



ภาพที่ 46 แสดงตัวอย่างรูปภาพที่เป็นข้อจำกัดของงานวิจัยที่นำเสนอ

4.2 ผลการทดลองและการเปรียบเทียบประสิทธิภาพของการจำแนกประเภทรูปภาพ

ในการวัดประสิทธิภาพในการสกัดคุณลักษณะของพื้นที่ดอกไม้ด้วยวิธีการที่นำเสนอโดย Yangyang และวิธีที่นำเสนอในงานชิ้นนี้กับชุดข้อมูลตั้งต้น ด้วยการนำชุดข้อมูลรูปภาพทั้ง 3 ชุด ไปจำแนกประเภทดอกไม้โดยใช้แบบจำลอง CNN โดยจะวัดประสิทธิภาพด้วยค่าความถูกต้อง ความแม่นยำ ความครบถ้วน และ F1 ซึ่งได้นำเสนอเอาไว้ในหัวข้อที่ 2.4 ข้อที่ 2.4.2

ผลการทดลอง

จากการทดลองพบว่าเมื่อจำแนกข้อมูลตั้งต้นจาก Kaggle ที่ไม่ผ่านขั้นตอนการแบ่งส่วนรูปภาพด้วยแบบจำลอง CNN จะให้ความถูกต้องเท่ากับ 84% และความถูกต้องของชุดข้อมูลที่ได้จากวิธีการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang คือ 85% ในขณะที่ความถูกต้องของวิธีที่นำเสนอในงานชิ้นนี้คือ 87% นอกจากนี้ค่าความแม่นยำ ค่าความครบถ้วน และF1 ของชุดข้อมูลทั้ง 3 ชุดได้แสดงผลลัพธ์ไว้ ดังแสดงในตารางที่ 8

ตารางที่ 8 แสดงค่า Precision, Recall และ F1 ที่ได้จากการจำแนกข้อมูลทั้ง 3 แบบด้วยแบบจำลอง CNN

ชุดข้อมูล	ความแม่นยำ	ความครบถ้วน	F1
ชุดข้อมูลตั้งต้น	84%	84%	84%
วิธีการแบ่งส่วนรูปภาพของ Yangyang	85%	85%	85%
วิธีที่นำเสนอในงานวิจัยชิ้นนี้	87%	87%	87%

จากผลการทดลองได้แสดงให้เห็นว่าการนำชุดข้อมูลรูปภาพดอกไม้ไปผ่านขั้นตอนการแบ่งส่วนรูปภาพก่อนนำชุดรูปภาพไปจำแนกประเภทจะช่วยให้สามารถจำแนกประเภทดอกไม้ได้ดีมากยิ่งขึ้น ด้วยการใช้วิธีที่นำเสนอในงานวิจัยนี้ในขั้นตอนการเตรียมรูปภาพ (Preprocessing) สามารถเพิ่มค่าความถูกต้อง ความแม่นยำ ความครบถ้วน และ F1 ได้ดีมากยิ่งขึ้นเช่นเดียวกัน ด้วยเหตุนี้การแบ่งส่วนรูปภาพจึงเป็นอีก 1 ขั้นตอนที่เป็นประโยชน์สำหรับการเตรียมข้อมูลให้พร้อมก่อนนำรูปภาพไปจำแนกประเภท

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ในงานวิจัยชิ้นนี้ได้นำเสนอแนวคิดการแบ่งส่วนรูปภาพ โดยอิงการใช้ประโยชน์จาก Saliency map ในการเก็บรายละเอียดของบริเวณที่สนใจภายในภาพ และการใช้ประโยชน์จากปริภูมิสี HSV ผนวกกับการประยุกต์ใช้ Color mask ในการช่วยลดรายละเอียดที่ไม่สำคัญภายในพื้นหลังของรูปภาพออกไป โดยทำการทดลองกับชุดรูปภาพดอกไม้ที่มีความหลากหลายของลักษณะทางกายภาพ เพื่อให้ชุดข้อมูลมีความสมจริงและใกล้เคียงกับรูปภาพโดยทั่วไปมากที่สุด ในการศึกษาจะแบ่งออกเป็น 2 ขั้นตอน คือขั้นตอนการแบ่งส่วนรูปภาพดอกไม้ และขั้นตอนการนำภาพที่ได้จากการแบ่งส่วนรูปภาพมาจำแนกประเภทดอกไม้ พร้อมทั้งวัดประสิทธิภาพของการจำแนกรูปภาพดอกไม้ และวัดประสิทธิภาพในการจำแนกประเภทดอกไม้ ซึ่งสามารถสรุปผลการวิจัยได้ ดังนี้

5.1 สรุปผลการวิจัย

จากการทดลองพบว่าวิธีนี้สามารถเก็บบริเวณของวัตถุที่สนใจได้ดีและสามารถลดรายละเอียดที่ไม่เกี่ยวข้องของพื้นหลังที่มีความซับซ้อนมากได้ค่อนข้างมาก ทำให้สามารถเลือกบริเวณที่สำคัญภายในภาพสำหรับนำไปใช้ในขั้นตอนการจำแนกประเภทดอกไม้ได้ค่อนข้างดีเมื่อเปรียบเทียบผลลัพธ์ระหว่างวิธีการแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang ซึ่งใช้วิธีการปรับเฉดสีของ Saliency map ให้เป็นเฉดสีเทาแล้วจึงปรับพื้นหลังของรูปภาพให้เป็นสีดำ ซึ่งให้ผลลัพธ์ที่ดีกับชุดรูปภาพที่มีความซับซ้อนของพื้นหลังในระดับปานกลาง จากผลการทดลองทำการวัดประสิทธิภาพการแบ่งส่วนรูปภาพวิธีการที่นำเสนอด้วยค่าเฉลี่ย IoU พบว่าค่าเฉลี่ย IoU ของวิธีการที่นำเสนอเท่ากับ 54% ในขณะที่ค่าเฉลี่ย IoU ของวิธีการของ Yangyang มีค่าเท่ากับ 41% ซึ่งเป็นผลจาก Bounding box ของวิธีการที่นำเสนอ สามารถล้อมรอบบริเวณของดอกไม้ได้ใกล้เคียงกับ Bounding box ของรูปภาพจริง และเมื่อนำชุดรูปภาพที่ไม่ผ่านขั้นตอนการแบ่งส่วนรูปภาพกับชุดข้อมูลที่ผ่านขั้นตอนการเตรียมข้อมูลด้วยการแบ่งส่วนรูปภาพจากทั้ง 2 วิธีไปใช้จำแนกประเภทของดอกไม้ด้วยแบบจำลอง VGG16 ที่ผ่านการทำ Fine-tuned พบว่าการเตรียมข้อมูลรูปภาพด้วยการแบ่งส่วนรูปภาพช่วยให้แบบจำลองสามารถจำแนกประเภทดอกไม้ได้ดีมากยิ่งขึ้น โดยผลลัพธ์แสดงให้เห็นได้จากค่าความถูกต้อง ความแม่นยำ ความครบถ้วน และ F1 ที่เพิ่มขึ้น โดยชุดข้อมูลที่แบ่งส่วนรูปภาพด้วยวิธีการที่นำเสนอช่วยให้จำแนกประเภทดอกไม้ได้ถูกต้อง 87% ในขณะที่ชุดข้อมูลที่ใช้การแบ่งส่วนรูปภาพด้วยวิธีการของ Yangyang สามารถจำแนกได้ถูกต้อง 85% ซึ่งวิธีการแบ่งส่วนทั้ง 2 วิธีได้ผลลัพธีที่ดีเมื่อเทียบกับชุด

ข้อมูลที่ไม่ผ่านขั้นตอนการแบ่งส่วนรูปภาพที่จำแนกได้ถูกต้อง 84% ด้วยเหตุนี้การแบ่งส่วนรูปภาพจึงเป็นอีก 1 ขั้นตอนที่เป็นประโยชน์สำหรับการเตรียมข้อมูลให้พร้อมก่อนรูปภาพไปจำแนกประเภท

สำหรับข้อจำกัดของงานวิจัยที่นำเสนอ ผู้วิจัยพบว่าวิธีการนี้สามารถแบ่งส่วนรูปภาพได้ดีหากภายในรูปภาพมีวัตถุที่สนใจเป็นองค์ประกอบส่วนใหญ่ภายในภาพ หากบริเวณที่สนใจเป็นเพียงบริเวณเล็กภายในรูปภาพโอกาสที่จะเลือกบริเวณนั้นก็จะน้อยลงไป ซึ่งปัญหาเหล่านี้เป็นผลจากการใช้วิธีเลือกพื้นที่โดยการเลือกพื้นที่ที่ติดกันมากที่สุด ดังนั้นสำหรับงานวิจัยที่จะขยายต่อในอนาคตอาจมีการนำประเด็นนี้ไปพิจารณาและศึกษาต่อเพื่อปรับปรุงงานให้ดีมากยิ่งขึ้น

5.2 ข้อเสนอแนะ

1. เนื่องจากงานวิจัยชิ้นนี้ทดลองกับชุดข้อมูลดอกไม้เพียงชุดเดียว อีกทั้งงานวิจัยชิ้นนี้มุ่งเน้นศึกษาพียงชุดข้อมูลดอกไม้ ดังนั้นผู้ที่สนใจสามารถลองใช้วิธีการที่นำเสนอกับชุดข้อมูลดอกไม้หลากหลายประเภท หรือชุดข้อมูลประเภทอื่นเพื่อเป็นการยืนยันประสิทธิภาพของวิธีการแบ่งส่วนรูปภาพ

2. เนื่องจากข้อจำกัดของซอร์ฟแวร์ทำให้งานวิจัยชิ้นนี้ไม่ได้ทดลองวิธีการจำแนกประเภทดอกไม้ด้วยแบบจำลอง Pre-trained ประเภทอื่น ซึ่งอาจจะให้ผลลัพธ์ของการทดลองที่แตกต่างกันออกไป

3. การทำ Fine-tuned แบบจำลองในงานวิจัยนี้อาจจะยังไม่ใช่วิธีการปรับค่าที่ดีที่สุด ดังนั้นผู้สนใจสามารถต่อยอดและพัฒนาขั้นตอนการปรับปรุงแบบจำลองในส่วนนี้ ให้สามารถจำแนกประเภทได้แม่นยำมากยิ่งขึ้น

ภาคผนวก ก.

คำสั่งการแบ่งส่วนรูปภาพด้วยวิธี RC ด้วยโปรแกรม Python เวอร์ชัน 3.7.6

```
# disjoint-set forests using union-by-rank and path compression (sort of)
class Uni_elt:
    def __init__(self,rank=0,p=0,size=0):
        self.rank= rank
        self.p = p
        self.size = size

class Universe:
    def __init__(self,elements):
        self.num = elements
        self.elts = [Uni_elt(rank=0,p=i,size=1) for i in range(elements)]

    def join(self,x,y):
        if(self.elts[x].rank > self.elts[y].rank):
            self.elts[y].p = x
            self.elts[x].size += self.elts[y].size
        else:
            self.elts[x].p = y
            self.elts[y].size += self.elts[x].size
            if self.elts[x].rank == self.elts[y].rank:
                self.elts[y].rank += 1
        self.num -= 1
```

```
    def size(self,x):
        return self.elts[x].size

    def nu_sets(self):
        return self.num

    def find(self,x):
        y = x
        while(y != self.elts[y].p):
            y = self.elts[y].p
        self.elts[x].p = y
        return y
```

```
import disjointSet
class Edge:
    def __init__(self,w=0.0,a=0,b=0):
        self.w = w
        self.a = a
        self.b = b


def threshold(size,c):
    return c / size

"""
Segment a graph
```

```
Returns a disjoint-set forest representing the segmentation
nu_vertices:number of vertices in the graph
nu_edges: number of edges in the graph
c: constant for threshold function
"""
def  segment_graph(nu_vertices,nu_edges,edges,c):
    tmp = edges[:nu_edges]
    #sorted by weight
    tmp.sort(key=lambda edge: edge.w)
    edges[:nu_edges] = tmp
    #make a disjoint-set forest
    u = disjointSet.Universe(nu_vertices)
    thresholds = [threshold(1,c) for _ in range(nu_vertices)]
    loop_range= range(nu_edges)
    for i in loop_range:
        edge = edges[i]
        a = u.find(edge.a)
        b = u.find(edge.b)
        if a != b:
            if edge.w <= thresholds[a] and edge.w <= thresholds[b]:
                u.join(a,b)
                a = u.find(a)
                thresholds[a] = edge.w + threshold(u.size(a),c)
    return u
```

```
import segmentGraph
import cv2
import numpy as np
from math import sqrt,pow

# dissimilarity measure between pixels
def diff(img3f,x1,y1,x2,y2):
    p1 = img3f[y1,x1]
    p2 = img3f[y2,x2]
    return np.sqrt(np.sum(np.power(p1-p2,2)))

def SegmentImage(src3f,imgInd,sigma=0.5,c=200,min_size=50):
    width = src3f.shape[1]
    height = src3f.shape[0]
    smImg3f = np.zeros(src3f.shape,dtype=src3f.dtype)

cv2.GaussianBlur(src3f,(0,0),sigma,dst=smImg3f,borderType=cv2.BORDER_REP
LICATE)
    # build graph
    edges = [segmentGraph.Edge() for _ in range(width * height * 4)]
    num = 0
    width_range = range(width)
    height_range = range(height)
    for y in height_range:
        for x in width_range:
```

```
        if x < width - 1:
            edges[num].a = y * width + x
            edges[num].b = y * width + (x+1)
            edges[num].w = diff(smImg3f,x,y,x+1,y)
            num += 1
        if y < height - 1:
            edges[num].a = y * width + x
            edges[num].b = (y+1) * width + x
            edges[num].w = diff(smImg3f,x,y,x,y+1)
            num += 1
        if (x < (width - 1)) and (y < (height-1)):
            edges[num].a = y * width + x
            edges[num].b = (y+1)*width + (x+1)
            edges[num].w = diff(smImg3f,x,y,x+1,y+1)
            num += 1
        if (x < (width - 1)) and y > 0:
            edges[num].a = y * width + x
            edges[num].b = (y-1) * width + (x+1)
            edges[num].w = diff(smImg3f,x,y,x+1,y-1)
            num += 1
# segment
u = segmentGraph.segment_graph(width * height,num,edges,c)

# post process small components
num_range = range(num)
for i in num_range:
    a = u.find(edges[i].a)
```

```
        b = u.find(edges[i].b)
        if ((a != b) and ((u.size(a) < min_size) or (u.size(b) < min_size))):
            u.join(a,b)


    # pick random colors for each components
    marker = {}
    imgInd = np.zeros((smImg3f.shape[0],smImg3f.shape[1]),np.int32)
    idxNum = 0
    for y in height_range:
        for x in width_range:
            comp = u.find(y * width + x)
            if comp not in marker.keys():
                marker[comp] = idxNum
                idxNum += 1
            idx = marker[comp]
            imgInd[y,x] = idx
    return idxNum,imgInd
```

```
import segmentImage
import cv2
import numpy as np
from utils import sqr,sqrDist,dist
import copy
```

```
class Region:
    def __init__(self,pixNum=0,ad2c=(0,0)):
        self.pixNum = pixNum
        self.ad2c = [ad2c[0],ad2c[1]]
        self.freIdx = []
        self.centroid = [0,0]


def GetHC(img3f):
    binN,idx1i,binColor3f,colorNums1i = Quantize(img3f)
    print(binN)
    cv2.cvtColor(binColor3f,cv2.COLOR_BGR2Lab,binColor3f)
    weight1f = np.zeros(colorNums1i.shape,np.float32)
    cv2.normalize(colorNums1i.astype(np.float32),weight1f,1,0,cv2.NORM_L1)
    colorSal = np.zeros((1,binN),np.float64)
    similar = [[] for _ in range(binN)]
    for i in range(binN):
        similar[i].append([0.0,i])
        for j in range(binN):
            if i != j:
                dij = dist(binColor3f[0,i],binColor3f[0,j])
                similar[i].append([dij,j])
                colorSal[0,i] += weight1f[0,j] * dij
        similar[i].sort()
    SmoothBySaliency(np.ones(colorSal.shape,np.int32),colorSal,0.25,similar)
    salHC1f = np.zeros((img3f.shape[0],img3f.shape[1]),np.float64)
    width = img3f.shape[1]
```

```
    height = img3f.shape[0]
    h_range = range(height)
    w_range = range(width)
    for y in h_range:
        for x in w_range:
            salHC1f[y,x] = colorSal[0,idx1i[y,x]]
    cv2.GaussianBlur(salHC1f,(3,3),0,salHC1f)
    cv2.normalize(salHC1f,salHC1f,0,1,cv2.NORM_MINMAX)
    return salHC1f


def GetFT(img3f):
    sal = np.zeros((img3f.shape[0],img3f.shape[1]),np.float32)
    tImg = cv2.GaussianBlur(img3f,(3,3),0)
    colorM = cv2.mean(tImg)
    height = img3f.shape[0]
    width = img3f.shape[1]
    for y in range(height):
        for x in range(width):
            sal[y,x] = float(sqr(colorM[0] - tImg[y,x,0]) + sqr(colorM[1] -
tImg[y,x,1])+sqr(colorM[2] - tImg[y,x,2]))
    cv2.normalize(sal,sal,0,1,cv2.NORM_MINMAX)
    return sal
def GetRC(img3f,sigmaDist=0.4,segK=200,segMinSize=50,segSigma=0.5):
    imgLab3f = img3f.copy()
    cv2.cvtColor(img3f,cv2.COLOR_BGR2Lab,imgLab3f)
    regNum,regIdx1i =
segmentImage.SegmentImage(imgLab3f,None,segSigma,segK,segMinSize)
```

```
Quatizenum,colorIdx1i,color3fv,tmp = Quantize(img3f)
if Quatizenum == 2:
    sal = colorIdx1i.copy()
    cv2.compare(colorIdx1i,1,cv2.CMP_EQ,sal)
    sal = sal.astype(np.float32)
    mn = np.min(sal)
    mx = np.max(sal)
    sal = (sal-mn)*255/(mx-mn)
if Quatizenum <= 2:
    return np.zeros(img3f.shape,img3f.dtype)
cv2.cvtColor(color3fv,cv2.COLOR_BGR2Lab,color3fv)
regs = BuildRegions(regIdx1i,colorIdx1i,color3fv.shape[1],regNum)
regSal1v = RegionContrast(regs,color3fv,sigmaDist)
sal1f = np.zeros((img3f.shape[0],img3f.shape[1]),img3f.dtype)
cv2.normalize(regSal1v,regSal1v,0,1,cv2.NORM_MINMAX)
#regSal1v = regSal1v / regSal1v.max(0)
width = img3f.shape[1]
height = img3f.shape[0]
height_range = range(height)
width_range = range(width)
for y in height_range:
    for x in width_range:
        sal1f[y,x] = regSal1v[0,regIdx1i[y,x]]
bdgReg1u = GetBorderReg(regIdx1i,regNum,0.02,0.4)
idxs = np.where(bdgReg1u == 255)
sal1f[idxs] = 0
# np.set_printoptions(threshold=np.nan)
```

```
    np.set_printoptions(threshold=np.inf, linewidth=np.nan)
    #cv2.imwrite('rc.png',sal1f)
    SmoothByHist(img3f,sal1f,0.1)
    SmoothByRegion(sal1f,regIdx1i,regNum)
    sal1f[idxs] = 0
    cv2.GaussianBlur(sal1f,(3,3),0,sal1f)
    return sal1f


def Quantize(img3f,ratio=0.95,colorNums=(12,12,12)):
    clrTmp = [i - 0.0001 for i in colorNums]
    w = [colorNums[1] * colorNums[2],colorNums[2],1]
    idx1i = np.zeros((img3f.shape[0],img3f.shape[1]),np.int32)
    width = img3f.shape[1]
    height = img3f.shape[0]
    height_range =range(height)
    width_range = range(width)
    #build color pallet
    pallet = {}
    for y in height_range:
        for x in width_range:
            idx1i[y,x] = int(img3f[y,x,0]*clrTmp[0]) * w[0] + int(img3f[y,x,1] * clrTmp[1]) *
w[1] + int(img3f[y,x,2] * clrTmp[2] )
            if idx1i[y,x] not in pallet.keys():
                pallet[idx1i[y,x]] = 1
            else:
                pallet[idx1i[y,x]] += 1
    #Find significant colors
```

```
    maxNum = 0
    num = [(pallet[key],key) for key in pallet] #(num,color) pairs in num
    num.sort(reverse=True)
    maxNum = len(num)
    maxDropNum = int(np.round(height*width*(1-ratio)))
    crnt = num[maxNum-1][0]
    while crnt < maxDropNum and maxNum > 1:
        crnt += num[maxNum-2][0]
        maxNum -= 1
    maxNum = 256 if maxNum > 256 else maxNum # To avoid very rarely case
    if maxNum <= 10:
        maxNum = 10 if len(num) > 10 else len(num)
    pallet.clear()
    for i in range(maxNum):
        pallet[num[i][1]] = i
    color3i = [[int(num[i][1] / w[0]),int(num[i][1] % w[0] / w[1]),int(num[i][1] % w[1])] for
i in range(len(num))]
    for i in range(maxNum,len(num)):
        simIdx = 0
        simVal = (1 << 31) - 1 # int32 max
        for j in range(maxNum):
            d_ij = sqrDist(color3i[i],color3i[j])
            if d_ij < simVal:
                simVal = d_ij
                simIdx = j
        pallet[num[i][1]] = pallet[num[simIdx][1]]
    color3f = np.zeros((1,maxNum,3),np.float32)
```

```
    colorNum = np.zeros((1,maxNum),np.int32)
    for y in height_range:
        for x in width_range:
            idx1i[y,x] = pallet[idx1i[y,x]]
            color3f[0,idx1i[y,x]] += img3f[y,x]
            colorNum[0,idx1i[y,x]] += 1
    for i in range(color3f.shape[1]):
        color3f[0,i] /= colorNum[0,i]
    return color3f.shape[1],idx1i,color3f,colorNum


def BuildRegions(regIdx1i,colorIdx1i,colorNum,regNum):
    width = regIdx1i.shape[1]
    height = regIdx1i.shape[0]
    cx = width / 2.0
    cy = height / 2.0
    width_range = range(width)
    height_range = range(height)
    regColorFre1i = np.zeros((regNum,colorNum),np.int32)
    regs = [Region() for _ in range(regNum)]
    for y in height_range:
        for x in width_range:
            regidx = regIdx1i[y,x]
            coloridx = colorIdx1i[y,x]
            reg = regs[regidx]
            reg.pixNum += 1
            reg.centroid[0] += x # region center x coordinate
            reg.centroid[1] += y # region center y coordinate
```

```
            regColorFre1i[regidx,coloridx] += 1
            reg.ad2c[0] = abs(x - cx)
            reg.ad2c[1] = abs(y - cy)
    for i in range(regNum):
        reg = regs[i]
        reg.centroid[0] /= reg.pixNum * width
        reg.centroid[1] /= reg.pixNum * height
        reg.ad2c[0] /= reg.pixNum * width
        reg.ad2c[1] /= reg.pixNum * height
        for j in range(colorNum):
            fre = float(regColorFre1i[i,j]) / reg.pixNum
            EPS = 1e-200
            if regColorFre1i[i,j] > EPS:
                reg.freIdx.append((fre,j))
    return regs


def RegionContrast(regs,color3fv,sigmaDist):
    cDistCache1f = np.zeros((color3fv.shape[1],color3fv.shape[1]),np.float64)
    for i in range(cDistCache1f.shape[0]):
        for j in range(i+1,cDistCache1f.shape[1]):
            cDistCache1f[i,j] = dist(color3fv[0, i], color3fv[0, j])
            cDistCache1f[j,i] = cDistCache1f[i,j]
    regNum = len(regs)
    rDistCache1d = np.zeros((regNum,regNum),np.float64)
    regSal1d = np.zeros((1,regNum),np.float64)
    for i in range(regNum):
        for j in range(regNum):
```

```
            if i < j:
                dd = 0.0
                range_m = range(len(regs[i].freIdx))
                range_n = range(len(regs[j].freIdx))
                c1 = regs[i].freIdx
                c2 = regs[j].freIdx
                for m in range_m:
                    for n in range_n:
                        dd += cDistCache1f[c1[m][1],c2[n][1]] * c1[m][0] * c2[n][0]
                        #dd += c1[m][0] * c2[n][0]
                #print(dd)
                tmp = dd * np.exp(-1.0 * sqrDist(regs[i].centroid,regs[j].centroid)/sigmaDist)
                #print(tmp)
                rDistCache1d[i][j] = tmp
                rDistCache1d[j][i] = tmp
            regSal1d[0,i] += regs[j].pixNum * rDistCache1d[i,j]
        regSal1d[0,i] *= np.exp(-9.0 * (sqr(regs[i].ad2c[0])+sqr(regs[i].ad2c[1])))
    return regSal1d


def GetBorderReg(idx1i,regNum,ratio=0.02,thr=0.3):
    EPS = 1e-200
    #variance of x and y
    vX = [0.0 for i in range(regNum)]
    vY = copy.deepcopy(vX)
    # mean value of x and y, pixel number of region
    mX = copy.deepcopy(vX)
    mY = copy.deepcopy(vX)
```

```
n = copy.deepcopy(vX)
w = idx1i.shape[1]
h = idx1i.shape[0]
h_range = range(h)
w_range = range(w)

for y in h_range:
    for x in w_range:
        mX[idx1i[y,x]] += x
        mY[idx1i[y,x]] += y
        n[idx1i[y,x]] += 1
for i in range(regNum):
    mX[i] /= n[i]
    mY[i] /= n[i]
for y in h_range:
    for x in w_range:
        idx = idx1i[y,x]
        vX[idx] += abs(x - mX[idx])
        vY[idx] += abs(y - mY[idx])
for i in range(regNum):
    vX[i] = vX[i] / n[i] + EPS
    vY[i] = vY[i] / n[i] + EPS
# Number of border pixels in x and y border region
xbNum = [0 for i in range(regNum)]
ybNum = copy.deepcopy(xbNum)
wGap = np.round(w * ratio)
hGap = np.round(h * ratio)
```

```
bPnts = []
pnt = [0,0]
sx = pnt[0]
sy = pnt[1]
#top region
while pnt[1] != hGap:
    pnt[0] = sx
    while pnt[0] != w:
        ybNum[idx1i[pnt[1],pnt[0]]] += 1
        bPnts.append(copy.deepcopy(pnt))
        pnt[0] += 1
    pnt[1] +=1
pnt = [0,int(h-hGap)]
sx = pnt[0]
sy = pnt[1]
#Bottom region
while pnt[1] != h:
    pnt[0] = sx
    while pnt[0] != w:
        ybNum[idx1i[pnt[1],pnt[0]]] += 1
        bPnts.append(copy.deepcopy(pnt))
        pnt[0] += 1
    pnt[1] += 1
pnt = [0,0]
sx = pnt[0]
sy = pnt[1]
#Left Region
```

```
while pnt[1] != h:
    pnt[0] = sx
    while pnt[0] != wGap:
        ybNum[idx1i[pnt[1],pnt[0]]] += 1
        bPnts.append(copy.deepcopy(pnt))
        pnt[0] += 1
    pnt[1] += 1
pnt = [int(w - wGap),0]
sx = pnt[0]
sy = pnt[1]
#Right Region
while pnt[1] != h:
    pnt[0] = sx
    while pnt[0] != w:
        ybNum[idx1i[pnt[1],pnt[0]]] += 1
        bPnts.append(copy.deepcopy(pnt))
        pnt[0] += 1
    pnt[1] += 1
bReg1u = np.zeros((h,w),np.uint8)
#likelihood map of border region
xR = 1.0 / (4 * hGap)
yR = 1.0 / (4 * wGap)
regL = [0 for _ in range(regNum)]
for i in range(regNum):
    lk = xbNum[i] * xR / vY[i] + ybNum[i] * yR / vX[i]
    regL[i] = 255 if lk / thr > 1 else 0
for y in h_range:
```

```
        for x in w_range:
            bReg1u[y,x] = regL[idx1i[y,x]]
    length = len(bPnts)
    len_range = range(length)
    for i in len_range:
        bReg1u[bPnts[i][1],bPnts[i][0]] = 255
    return bReg1u


def SmoothByHist(img3f,sal1f,delta):
    #Quantize colors
    binN, idx1i, binColor3f, colorNums1i = Quantize(img3f)
    _colorSal = np.zeros((1,binN),np.float64)
    height = img3f.shape[0]
    width = img3f.shape[1]
    h_range = range(height)
    w_range = range(width)
    for y in h_range:
        for x in w_range:
            _colorSal[0,idx1i[y,x]] += sal1f[y,x]
    for i in range(binN):
        _colorSal[0,i] /= colorNums1i[0,i]
    cv2.normalize(_colorSal,_colorSal,0,1,cv2.NORM_MINMAX)
    #Find similar colors & Smooth saliency value for color bins
    similar = [[] for i in range(binN)]
    cv2.cvtColor(binColor3f,cv2.COLOR_BGR2Lab,binColor3f)
    for i in range(binN):
        similar[i].append([0.0,i])
```

```
        for j in range(binN):
            if i != j:
                similar[i].append([dist(binColor3f[0,i],binColor3f[0,j]),j])
        similar[i].sort()

    cv2.cvtColor(binColor3f,cv2.COLOR_Lab2BGR,binColor3f)
    SmoothBySaliency(colorNums1i,_colorSal,delta,similar)
    #reassign pixel saliency values
    for y in h_range:
        for x in w_range:
            sal1f[y,x] = _colorSal[0,idx1i[y,x]]

def SmoothByRegion(sal1f,idx1i,regNum,bNormalize = True):
    saliecy = [0.0 for _ in range(regNum)]
    counter = [0 for _ in range(regNum)]
    h = sal1f.shape[0]
    w = sal1f.shape[1]
    h_range = range(h)
    w_range = range(w)
    for y in h_range:
        for x in w_range:
            saliecy[idx1i[y,x]] += sal1f[y,x]
            counter[idx1i[y,x]] += 1
    for i in range(len(counter)):
        saliecy[i] /= counter[i]

    rSal = np.array([saliecy],dtype=np.float64)
```

```
    if(bNormalize):
        cv2.normalize(rSal,rSal,0,1,cv2.NORM_MINMAX)
    for y in h_range:
        for x in w_range:
            sal1f[y,x] = saliecy[idx1i[y,x]]


def SmoothBySaliency(colorNums1i,sal1f,delta,similar):
    if sal1f.shape[1] < 2:
        return
    binN = sal1f.shape[1]
    newSal1d = np.zeros((1,binN),np.float64)
    tmpNum = int(np.round(binN*delta))
    n =  tmpNum if tmpNum > 2 else 2
    dist = [0.0 for _ in range(n)]
    val = copy.deepcopy(dist)
    w = copy.deepcopy(dist)
    binN_range = range(binN)
    n_range = range(n)
    for i in binN_range:
        totalDist = 0.0
        totalWeight = 0.0
        for j in n_range:
            ithIdx = similar[i][j][1]
            dist[j] = similar[i][j][0]
            val[j] = sal1f[0,ithIdx]
            w[j] = colorNums1i[0,ithIdx]
            totalDist += dist[j]
```

```
            totalWeight += w[j]
        valCrnt = 0.0
        for j in n_range:
            valCrnt += val[j] * (totalDist - dist[j]) * w[j]
        #print(valCrnt,totalDist,totalWeight)
        newSal1d[0,i] = valCrnt / (totalDist * totalWeight)
    cv2.normalize(newSal1d,sal1f,0,1,cv2.NORM_MINMAX)
```

คำสั่งการนำเข้ารูปภาพ ด้วยโปรแกรม Python เวอร์ชัน 3.7.6

```
import os

def import_flower_images(path):
        flower_images = []
        labels = []

        for root, directories, files in os.walk(path, topdown=False):
                for name in files:
                        flower_images.append(os.path.join(root, name))
        for item in flower_images:
                labels.append(item.split('\\')[6])

        return flower_images, labels
```

คำสั่งการแบ่งส่วนรูปภาพด้วยวิธีของ Yangyang ด้วยโปรแกรม Python เวอร์ชัน 3.7.6

```
from import_images import import_flower_images
import segmentGraph, saliencyRC, segmentGraph, segmentImage
import cv2
import matplotlib.pyplot as plt
import numpy as np
import random
from utils import sqr,sqrDist,dist
import copy
import os
from matplotlib.patches import Rectangle
from scipy import ndimage


# Regional contrast
def saliency_map(flower_image):
    img3i = flower_image
    img3f = img3i.astype(np.float32)
    img3f *= 1. / 255
    sal = saliencyRC.GetRC(img3f,segK=20,segMinSize=200)
    start = cv2.getTickCount()
    plt.imsave('sal.jpg', sal)
    return sal


# Gray-scale map
```

```
def gray_scale_map(flower_image):
    gray = cv2.cvtColor(flower_image, cv2.COLOR_BGR2GRAY)
    plt.imsave('gray.jpg', gray)
    return gray


# Map saliency location with c=gray-scale map
def threshold_img():
    sal_img = cv2.imread('sal.jpg')
     sal_img = cv2.cvtColor(sal_img, cv2.COLOR_BGR2GRAY)
    ret, mask = cv2.threshold(sal_img, 120, 255,
                        cv2.THRESH_TOZERO)
    plt.imsave('thresh.jpg', mask)
    return mask


def combine_sal_and_gray(mask):
    result = cv2.bitwise_and(flower_image, flower_image, mask=mask)
    plt.imsave('result.jpg', result)
    return result


def largest_connected_area():
    image =  cv2.imread('result.jpg')
    img_gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
```

```
    contours, hierarchy = cv2.findContours(img_gray,
                                        cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnt = max(contours, key=cv2.contourArea)
    x,y,w,h = cv2.boundingRect(cnt)
    bbox = (x, y, w, h)
    cv2.rectangle(image, (x,y), (x+w, y+h), (255,255,0), 2);
    return image, bbox
# Crop image in bounding box
def crop_image(flower_image, bbox):
    (x, y, w, h) = bbox
    cropped_image = flower_image[y:y+h, x:x+w]
    return cropped_image
# resize image
def resize_image(cropped_image):
    croped = cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB)
    resized_img = cv2.resize(cropped_image, (224, 224),
                interpolation = cv2.INTER_CUBIC)
    return resized_img


def main(flower_image):
    sal = saliency_map(flower_image)
    gray = gray_scale_map(flower_image)
```

```
    mask = threshold_img(dst)
    result = combine_sal_and_gray()
    image, bbox = largest_connected_area()
    croped = crop_image(flower_image, bbox)
    resized_img = resize_image(croped)
    return resized_img
```

คำสั่งการแบ่งส่วนรูปภาพด้วยวิธีที่นำเสนอ ด้วยโปรแกรม Python เวอร์ชัน 3.7.6

```
from import_images import import_flower_images
import segmentGraph, saliencyRC, segmentGraph, segmentImage
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random
from utils import sqr,sqrDist,dist
import copy
import os
from matplotlib.patches import Rectangle

export_path = 'C:\\Users\\LENOVO\\Documents\\Thesis_papers\\\my_solution_img\\'
flower_images, labels =
import_flower_images('C:\\Users\\LENOVO\\Documents\\Thesis_papers\\Flowers_Kaggle')
df = pd.DataFrame({'data': flower_images, 'labels': labels})
```

```
features, labels = df.data, df.labels


def saliency_map(flower_image):
    img3i = flower_image
    img3f = img3i.astype(np.float32)
    img3f *= 1. / 255
    sal = saliencyRC.GetRC(img3f,segK=20,segMinSize=200)
    start = cv2.getTickCount()
    plt.imsave('sal.jpg', sal)
    return sal


def morphological(flower_image):
    img = cv2.imread('sal.jpg')
    ## convert to hsv
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, (15,0,0), (90, 255, 255))
    result = cv2.bitwise_and(flower_image, flower_image, mask = mask)
    plt.imsave('result.jpg', result)
    return result



def largest_connected_area():
    image =  cv2.imread('result.jpg')
    img_gray = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    contours, hierarchy = cv2.findContours(img_gray,
                                           cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

```
    cnt = max(contours, key=cv2.contourArea)
    x,y,w,h = cv2.boundingRect(cnt)
    bbox = (x, y, w, h)
    cv2.rectangle(image, (x,y), (x+w, y+h), (255,255,0), 2);
    return image, bbox


def crop_image(flower_image, bbox):
    (x, y, w, h) = bbox
    croped_image = flower_image[y:y+h, x:x+w]
    return croped_image


def resize_image(croped_image):
    croped = cv2.cvtColor(croped_image, cv2.COLOR_BGR2RGB)
    resized_img = cv2.resize(croped_image, (224, 224),
                    interpolation = cv2.INTER_CUBIC)
    return resized_img


def main(flower_image):
    sal = saliency_map(flower_image)
    result = morphological(flower_image)
    image, bbox = largest_connected_area()
    croped_image = crop_image(flower_image, bbox)
    resized_image = resize_image(croped_image)
    return resized_image
```

คำสั่งการเพิ่มจำนวนข้อมูลรูปภาพ (Data Augmentation) ด้วยโปรแกรม Python เวอร์ชัน 3.7.6

```
import cv2
import skimage as sk
from skimage import io
import glob

from import_images import import_flower_images
import random   # ใช้ในการ shuffle รูปภาพเรียงติดกันเหมือนเดิมทุกครั้ง
import numpy as np

import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt

export_path =
'C:\\Users\\LENOVO\\Documents\\Thesis_papers\\flower_kaggle_augmented\\'

images = [file for file in
glob.glob('C:\\Users\\LENOVO\\Documents\\Thesis_papers\\flower_kaggle\*.jpg')]

data = []
labels = []

for img in images:
    data.append(img)
    labels.append(img.split('\\')[6])

df_org = pd.DataFrame({'data': data, 'label': labels})
```

```
new = df_org["label"].str.split("_", n = 1 , expand = True)
df_org["labels"]= new[0]
df_org = df_org.drop(columns= ['label'])
```

Sampling Images

```
flower_classes = df_org.labels.drop_duplicates().to_list()
num_files_desired = 500
randomed_images = []
randomed_labels = []

for i in flower_classes:
    random_img = random.sample(df_org[df_org.labels == i]['data'].tolist(), k =
num_files_desired)
    for img in random_img:
        randomed_images.append(img)
        randomed_labels.append(i)
df_randomed = pd.DataFrame({'data': randomed_images, 'labels': randomed_labels})
```



```
### Image Augmentations

def flipped(image):
    return tf.image.flip_left_right(image)
def rotated(image):
    return tf.image.rot90(image)
def bright(image):
    return tf.image.adjust_brightness(image, 0.4)
def crop(image):
    return tf.image.central_crop(image, central_fraction=0.8)

def augmentation(img):
```

```
    augmentation_method = {'flipped': flipped, 'rotated': rotated,
                           'bright': bright, 'crop' : crop}
    image = cv2.imread(img)
    key = random.choice(list(augmentation_method))
    augmented_img = augmentation_method[key](image)
    return augmented_img



num = 0
for idx in range(df_randomed.shape[0]):
    img = augmentation(df_randomed.data[idx])
    io.imsave(export_path + randomed_labels[idx]+ '_' + str(num)
                          + '.jpg', img)
    num += 1
```

คำสั่งการจำแนกประเภทรูปภาพและประเมินผลรูปภาพ ด้วยโปรแกรม Python


```
import cv2
from fastai.vision.all import *
import random
import pandas as pd
import matplotlib.pyplot as plt
from keras.utils import np_utils
from PIL import Image
from numpy import mean
from numpy import std
import numpy as np
import glob
import joblib
```

```
import os

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split,
from sklearn.metrics import classification_report, confusion_matrix,\
                            f1_score ,accuracy_score, plot_confusion_matrix

from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.utils import to_categorical
import tensorflow.keras as keras

from keras.models import Sequential, load_model, Model
from keras.layers import Dense, Flatten, MaxPool2D, Conv2D,\
                        Activation, Dropout
from keras import backend, regularizers
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.optimizers import Adam
from keras.callbacks import ReduceLROnPlateau
from keras.applications import VGG16
from livelossplot.inputs.keras import PlotLossesCallback



def data_frame(path):
    df = pd.DataFrame({'data': get_image_files(path), 'label': os.listdir(path)})
    new = df["label"].str.split("_", n = 1, expand = True)
    df['labels'] = new[0]
    df = df.drop(columns= ['label'])
    return df
```

```
# load train and test dataset
def load_dataset():
    # resize image
    data_arr =[] #input array
    resized_images = []

    for img in df.data:
        img_array= Image.open(img).convert('RGB')
        dim = (224, 224)
        image_resized = img_array.resize(dim, Image.LANCZOS)
        np_img = np.array(image_resized)
        resized_images.append(image_resized)
        data_arr.append(np_img)

    data = np.array(data_arr)
    target = np.array(df.labels)

    X_train, X_test, y_train, y_test = train_test_split(data, target,test_size = 0.2,
random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25,
random_state=42)


    le = LabelEncoder()
    le.fit(y_train)
    y_train = le.transform(y_train)
    y_test = le.transform(y_test)
    y_val = le.transform(y_val)

    y_train = to_categorical(y_train)
    y_test = to_categorical(y_test)
```

```
        y_val = to_categorical(y_val)

        return X_train, y_train, X_test, y_test, X_val, y_val

# scale pixels
def prep_pixels(X_train, X_test, X_val):
    # convert from integers to floats
    train_norm = X_train.astype('float32')
    test_norm = X_test.astype('float32')
    val_norm = X_val.astype('float32')
    # normalize to range 0-1
    train_norm = train_norm / 255.0
    test_norm = test_norm / 255.0
    val_norm = val_norm / 255.0
    return train_norm, test_norm, val_norm


def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix',
cmap=plt.cm.Blues):
    #Add Normalization Option
    if normalize:
     cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
     print('Normalized confusion matrix')
    else:
     print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation= 'nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
```

```
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, format(cm[i, j], fmt), horizontalalignment='center', color= 'white' if
cm[i, j] > thresh else 'black')
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Data Preparation

```
path_org = '../input/flower-kaggle/flower_kaggle/flower_kaggle'
path_aug =
'../input/flowerkaggle/flower_kaggle_augmented/flower_kaggle_augmented'

df = pd.concat([df_org, df_aug], axis = 0)
X_train, y_train, X_test, y_test, X_val, y_val = load_dataset()
train_norm, test_norm, val_norm = prep_pixels(X_train, X_test, X_val)
```

Modeling

```
vgg16_model = VGG16(weights = "imagenet", include_top = False, input_shape = (224,
224, 3))

# Create the model
full_model = Sequential()
```

```
# Add the vgg convolutional base model
full_model.add(vgg16_model)

# Add new layers
full_model.add(Flatten())
full_model.add(Dense(256, activation='relu'))
full_model.add(Dropout(0.65))
full_model.add(Dense(64, activation='relu'))
full_model.add(Dropout(0.3))
full_model.add(Dense(5, activation='softmax'))

# Show a summary of the model. Check the number of trainable parameters
full_model.summary()

for layer in vgg16_model.layers[-15:]:
  layer.trainable = False
for i, layer in enumerate(vgg16_model.layers):
    print(i, layer.name, layer.trainable)

full_model.compile(optimizer = Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
                metrics=['accuracy'])



plot_loss_1 = PlotLossesCallback()
# ModelCheckpoint callback - save best weights
tl_checkpoint_1 = ModelCheckpoint(filepath='cnn_vgg16_mine.weights.best.h5',
                                  save_best_only = True,
                                  verbose=1)
EarlyStopping
early_stop = EarlyStopping(monitor='val_loss',
```

```
                    patience=10,
                    restore_best_weights=True,
                    mode='min')


fitted_model = full_model.fit(train_norm, y_train,
                     batch_size= 128,
                     epochs= 40,
                     validation_data= (val_norm, y_val),
                     callbacks = [tl_checkpoint_1, early_stop],
                     verbose=1)

acc = fitted_model.history['accuracy']
val_acc = fitted_model.history['val_accuracy']
plt.plot(acc)
plt.plot(val_acc)
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Accuracy','loss'])

loss = fitted_model.history['loss']
val_loss = fitted_model.history['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['loss', 'validation'])
```

Evaluating

```
model = load_model('cnn_vgg16_mine.weights.best.h5')
```

```
model.evaluate(test_norm, y_test)

predictions = model.predict(test_norm)
y_pred=np.argmax(predictions, axis=1)
y_true=np.argmax(y_test, axis=1 )
class_names = df.labels.drop_duplicates().to_list()
classification_metrices = classification_report(y_true, y_pred, target_names = class_names)
print(classification_metrices)

categorical_test_labels = pd.DataFrame(y_test).idxmax(axis=1)
categorical_preds = pd.DataFrame(predictions).idxmax(axis=1)
confusion_matrix= confusion_matrix(categorical_test_labels, categorical_preds)

plot_confusion_matrix(confusion_matrix, class_names, normalize= True)
```

คำสั่งการประเมินผลการแบ่งส่วนรูปภาพ ด้วยโปรแกรม Python


```
import pandas as pd
import numpy as np
import cv2


def IOU(df):
    # determining the minimum and maximum -coordinates of the intersection rectangle
    xmin_inter = max(df.xmin, df.xmin_pred)
    ymin_inter = max(df.ymin, df.ymin_pred)
    xmax_inter = min(df.xmax, df.xmax_pred)
```

```
    ymax_inter = min(df.ymax, df.ymax_pred)

    # calculate area of intersection rectangle
    inter_area = max(0, xmax_inter - xmin_inter + 1) * max(0, ymax_inter - ymin_inter +
1)

    # calculate area of actual and predicted boxes
    actual_area = (df.xmax - df.xmin + 1) * (df.ymax - df.ymin + 1)
    pred_area = (df.xmax_pred - df.xmin_pred + 1) * (df.ymax_pred - df.ymin_pred+ 1)

    # computing intersection over union
    iou = inter_area / float(actual_area + pred_area - inter_area)

    # return the intersection over union value
    return iou

path_method2=
'C:\\Thesis_practice\\oxford17_segmented_method2\\bbox_method2.xlsx'
path_method_mine=
'C:\\Thesis_practice\\oxford17_segmented_mine\\bbox_method_mine.xlsx'

data = pd.read_excel(path_method_mine).iloc[:, 1:]

# creating a evaluation table
eval_table = pd.DataFrame()
eval_table['image_name'] = data.labels

#applying IOU function over each image in the dataframe
eval_table['IOU'] = data.apply(IOU, axis = 1)

eval_table.IOU.mean()
```

บรรณานุกรม

Bhargavi K. & Jyothi S. (2014). A survey on threshold based segmentation technique in image processing. I*nternational Journal of Innovative Research and Development*, 3(12), 234-239.

Chen T.-W., Chen Y.-L. & Chien S.-Y. (2008). Fast image segmentation based on K-Means clustering with histograms in HSV color space. *2008 IEEE 10th Workshop on Multimedia Signal Processing.*

Chen Y., Xu W., Kuang F. & Gao S. (2013). The Study of Randomized Visual Saliency Detection Algorithm. *Computational and mathematical methods in medicine 2013.*

Cheng M.-M., Mitra N. J., Huang X., Torr P. H. & Hu S.-M. (2014). Global contrast based salient region detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3), 569-582.

Chris S. & Toby B. (2011). *Fundamentals of Digital Image Processing*. In (pp. 6-7). John Wiley & Sons.

Cooley C., Coleman S., Gardiner B. & Scotney B. (2017). Saliency detection and object classification. *Proc. 19th Irish Machine Vision and Image Processing Conf.(IMVIP 2017).*

Deswal M. & Sharma N. (2014). A fast HSV image color and texture detection and image conversion algorithm. *International Journal of Science and Research (IJSR)*, 3(6).

Garg I. & Kaur B. (2016). Color based segmentation using K-mean clustering and watershed segmentation. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom).*

Ghosh A., Sufian A., Sultana F., Chakrabarti A. & De D. (2020). Fundamental concepts of convolutional neural network. *In Recent Trends and Advances in Artificial Intelligence and Internet of Thing.*519-567. Springer.

Hassan M. R., Ema R. R. & Islam T. (2017). Color image segmentation using automated K-means clustering with RGB and HSV color spaces. *Global Journal of Computer Science and Technology*.

Kaur D. & Kaur Y. (2014). Various image segmentation techniques: a review. *International Journal of Computer Science and Mobile Computing*, 3(5), 809-814.

Linda S., & George S. (2001). *Computer Vision*. In (pp. 46-47). Prentice-Hall.

Liu Y., Tang F., Zhou D., Meng Y. & Dong W. (2016). Flower classification via convolutional neural network. *2016 IEEE International Conference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications (FSPMA).*

Milan S., Vaclav H. & Roger B. (2015). *Image Processing, Analysis, and Machine Vision*. In (pp. 178). Cengage Learning.

Najjar A. & Zagrouba E. (2012). Flower image segmentation based on color analysis and a supervised evaluation. *2012 International Conference on Communications and Information Technology (ICCIT).*

Ornek A. H. & Ceylan M. (2019). Comparison of traditional transformations for data augmentation in deep learning of medical thermography. *2019 42nd International Conference on Telecommunications and Signal Processing (TSP).*

Pan J.-S., Feng Q., Yan L. & Yang, J.-F. (2014). Neighborhood feature line segment for image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(3), 387-398.

Panwar P., Gopal G. & Kumar R. (2016). Image Segmentation using K-means clustering and Thresholding. I*mage*, 3(05), 1787-1793.

Ren S., He K., Girshick R. & Sun J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6), 1137-1149.

Sabri N., Ibrahim Z. & Rosman N. N. (2016). K-means vs. fuzzy C-means for segmentation of orchid flowers. *2016 7th IEEE Control and System Graduate Research Colloquium (ICSGRC).*

Sachs J. (1996-1999). *Digital Image Basics.* In (pp. 10-11).

Sathya P. & Kayalvizhi R. (2011). Modified bacterial foraging algorithm based multilevel thresholding for image segmentation. *Engineering Applications of Artificial Intelligence,* 24(4), 595-615.

Sharma A. & Sehgal S. (2016). Image segmentation using firefly algorithm. *2016 International Conference on Information Technology (InCITe)-The Next Generation IT Summit on the Theme-Internet of Things: Connect your Worlds.*

Tammina S. (2019). Transfer learning using vgg-16 with deep convolutional neural network for classifying images. *International Journal of Scientific and Research Publications (IJSRP)*, 9(10), 143-150.

Thanh L. T. & Thanh D. N. (2020). An adaptive local thresholding roads segmentation method for satellite aerial images with normalized HSV and lab color models. *In Intelligent Computing in Engineering* (pp. 865-872). Springer.

Vu Q. D., Graham S., Kurc T., To M. N. N., Shaban M., Qaiser T., Koohbanani N. A., Khurram S. A., Kalpathy-Cramer J. & Zhao T. (2019). Methods for segmentation and classification of digital microscopy tissue images. *Frontiers in bioengineering and biotechnology*, 7, 53.

Wang W., Duan L., & Wang Y. (2017). Fast image segmentation using two-dimensional Otsu based on estimation of distribution algorithm. *Journal of Electrical and Computer Engineering*, 2017.

Wang X., Peng Y., Lu L., Lu Z., Bagheri M. & Summers R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. P*roceedings of the IEEE conference on computer vision and pattern recognition*.

Wilhelm B. & Mark J. B. (2016). *Digital Image An Algorithmic Introduction Using Java*. Springer Nature.

Yadav J. & Sharma M. (2013). A Review of K-mean Algorithm. *Int. J. Eng. Trends Technol*, 4(7), 2972-2976.

Yangyang, Y., & Xiang, F. (2019). A Flower Image Classification Algorithm Based on Saliency Map and PCANet. Journal of Communication and Computer, 15, 14-24.

Zheng X., Lei Q., Yao R., Gong Y. & Yin Q. (2018). Image segmentation based on adaptive K-means algorithm. *EURASIP Journal on Image and Video Processing*, 2018(1), 1-10.

อรวรรณ เลียบศิริ. (2559). *การจำแนกอารม์ตัวการ์ตูนด้วยฮิสโตแกรมของทิศทางตามค่าเกรเดียนท์*.ธรรมศาสตร์.

ประวัติผู้เขียน

ชื่อ-สกุล	ธนณัฏฐ์ หงษ์ทอง
วัน เดือน ปี เกิด	5 พฤษภาคม 2540
สถานที่เกิด	สุราษฎร์ธานี
วุฒิการศึกษา	วิทยาศาสตรบัญฑิต(คณิตศาสตร์ประยุกต์) มหาวิทยาลัยธรรมศาสตร์