อุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตา ในการใช้ชีวิตภายในอาคาร

EQUIPMENT FOR ASSISTING INDIVIDUALS WITH VISUALLY IMPAIRED IN DAILY LIFE INSIDE BUILDING

โดย

นางสาวพรรณธารา เจริญศรีชัยรัตน์

นางสาววรรณวร ลิ่มบุญสืบสาย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2567

อุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตา ในการใช้ชีวิตภายในอาคาร

EQUIPMENT FOR ASSISTING INDIVIDUALS WITH VISUALLY IMPAIRED IN DAILY LIFE INSIDE BUILDING

โดย

นางสาวพรรณธารา เจริญศรีชัยรัตน์ 64010562

นางสาววรรณวร ลิ่มบุญสืบสาย 64010768

อาจารย์ที่ปรึกษา

ศ.ดร.ปราโมทย์ วาดเขียน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2567

ปริญญานิพนธ์ปีการศึกษา 2567

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง อุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตาในการใช้ชีวิตภายในอาคาร
EQUIPMENT FOR ASSISTING INDIVIDUALS WITH VISUALLY IMPAIRED IN DAILY LIFE INSIDE BUILDING

ผู้จัดทำ

1. นางสาวพรรณธารา เจริญศรีชัยรัตน์ 64010562
2. นางสาววรรณวร ลิ่มบุญสืบสาย 64010768

ปราโมทย์ วาดเขียน ..... อาจารย์ที่ปรึกษา

(ศ.ดร.ปราโมทย์ วาดเขียน)

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง “อุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตาในการใช้ชีวิตภายในอาคาร” จะไม่สามารถสำเร็จลุล่วงไปได้ หากไม่ได้รับความอนุเคราะห์อย่างยิ่งจากอาจารย์ที่ปรึกษาปริญญานิพนธ์ คือ ศ.ดร.ปราโมทย์ วาดเขียน ที่กรุณาให้คำปรึกษา และแนวทางการแก้ไขปัญหาตลอดระยะเวลาในการจัดทำปริญญานิพนธ์รวมทั้งสนับสนุนสถานที่ เครื่องมือ และอุปกรณ์ต่างๆ ที่จำเป็นต้องใช้ในระหว่างการจัดทำปริญญานิพนธ์ ขอขอบพระคุณในความห่วงใยและความหวังดีที่มีให้แก่คณะผู้จัดทำเป็นอย่างยิ่ง

ขอขอบคุณอาจารย์ประจำภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนและประสิทธิ์ประสาทวิชาให้ความรู้แก่คณะผู้จัดทำ

ขอขอบคุณผู้มีส่วนเกี่ยวข้องทุกท่าน อาทิ บิดา มารดา และเพื่อนนักศึกษา ที่คอยสนับสนุน แนะนำช่วยเหลือ และให้กำลังใจแก่คณะผู้จัดทำเสมอมา จนกระทั่งปริญญานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี

นางสาวพรรณธารา เจริญศรีชัยรัตน์
นางสาววรรณวร ลิ่มบุญสืบสาย
ผู้จัดทำ

อุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตาในการใช้ชีวิตภายในอาคาร

EQUIPMENT FOR ASSISTING INDIVIDUALS WITH VISUALLY IMPAIRED IN DAILY LIFE INSIDE BUILDING

โดย นางสาวพรรณธารา เจริญศรีชัยรัตน์ 64010562
นางสาววรรณวร ลิ่มบุญสืบสาย 64010768

อาจารย์ที่ปรึกษา ศ.ดร.ปราโมทย์ วาดเขียน

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการใช้การเรียนรู้เชิงลึกสำหรับการจำแนกวัตถุ โดยอัลกอริทึมการเรียนรู้เชิงลึกมีโครงสร้างที่เลือกศึกษา ได้แก่ Convolution Neural Network (CNN) และ ResNet18 รวมถึงการเตรียมข้อมูล การดึงคุณลักษณะ การปรับพารามิเตอร์เพื่อเปรียบเทียบความแม่นยำ และการประเมินประสิทธิภาพของโมเดลที่เลือก เพื่อนำเสนอโมเดลที่มีประสิทธิภาพสำหรับการใช้งานกับอุปกรณ์ช่วยเหลือผู้มีความบกพร่องทางสายตาในการจำแนกวัตถุภายในอาคารและส่งเสียงเตือน

ABSTRACT

This thesis presents the application of deep learning for object classification. The selected deep learning architectures studied include Convolutional Neural Networks (CNN) and ResNet18. It covers data preparation, feature extraction, parameter tuning for accuracy comparison, and performance evaluation of the selected models. The aim is to propose an efficient model for use in devices that assist visually impaired individuals in classifying indoor objects and providing sound alerts.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันปัญหาในการเคลื่อนที่และการรับรู้สภาพแวดล้อมรอบตัวสำหรับผู้บกพร่องทางสายตายังคงเป็นอุปสรรคสำคัญในสังคมส่งผลให้เกิดความเสี่ยงต่อการเดินชนสิ่งกีดขวางและได้รับอันตราย ด้วยการนำเทคโนโลยีปัญญาประดิษฐ์เข้ามาใช้ในการเรียนรู้ การวิเคราะห์ และประมวลผลข้อมูลได้อย่างรวดเร็วเพื่อจำแนกวัตถุ ปริญญานิพนธ์นี้จึงมีจุดประสงค์เพื่อสร้างอุปกรณ์สำหรับช่วยเหลือผู้บกพร่องทางสายตาที่ใช้อัลกอริทึมการเรียนรู้เชิงลึกแบบต่างๆ เพื่อนำไปจำแนกวัตถุที่เป็นอุปสรรคสำหรับผู้บกพร่องทางสายตาในการทำกิจวัตรประจำวันภายในอาคารและส่งสัญญาณเตือนให้ผู้ใช้งานทราบให้เคลื่อนที่อย่างปลอดภัย

1.2 วัตถุประสงค์

1) ศึกษาการประมวลผลภาพ (Image Processing)
2) ศึกษาอัลกอริทึมการเรียนรู้เชิงลึกเพื่อจำแนกวัตถุ
3) เพื่อออกแบบและสร้างอุปกรณ์เพื่อช่วยเหลือผู้บกพร่องทางสายตาสำหรับการจำแนกวัตถุที่เป็นอุปสรรคต่อการเคลื่อนที่ภายในอาคาร

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์นี้ทำการศึกษาอัลกอริทึมการเรียนรู้เชิงลึกโดยใช้ข้อมูลรูปภาพเพื่อจำแนกวัตถุ โดยส่วนของรูปภาพจะใช้ข้อมูลภาพของวัตถุ ได้แก่ ลิฟต์ บันได ประตู และสิ่งกีดขวาง จากออนไลน์และจัดเตรียมด้วยตัวเอง จะใช้หลักการ Image Processing เพื่อการปรับปรุงคุณภาพของข้อมูลช่วยให้การจำแนกวัตถุมีประสิทธิภาพมากขึ้นและทำให้ข้อมูลอยู่ในรูปแบบที่โมเดลเข้าใจได้ ข้อมูลที่ได้จะถูกนำไปทำการฝึกโมเดลและสุดท้ายประเมินประสิทธิภาพโมเดล เมื่อได้โมเดลที่พร้อมใช้งานแล้วจะนำมาประยุกต์ใช้กับอุปกรณ์เพื่อช่วยเหลือผู้บกพร่องทางสายตาที่สร้างขึ้นสำหรับจำแนกวัตถุ และยังสามารถส่งสัญญาณเสียงเตือนสำหรับการใช้งานภายในอาคาร

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

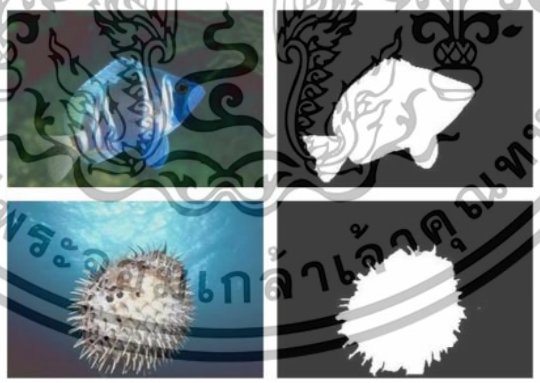
2.1 รูปภาพ (Image)

รูปภาพถูกกำหนดให้เป็นฟังก์ชันสองมิติ F(x, y) โดยที่ x และ y เป็นพิกัดเชิงพื้นที่และแอมพลิจูดของ F ที่พิกัดคู่ใดๆ (x, y) เรียกว่าความเข้มของรูปภาพ ณ จุดนั้นเมื่อค่า x, y และแอมพลิจูดของ F มีจำกัด เรียกว่า ภาพดิจิทัล รูปภาพสามารถกำหนดได้โดยอาร์เรย์สองมิติที่จัดเรียงในรูปแบบแถวและคอลัมน์ รูปภาพดิจิทัลประกอบด้วยองค์ประกอบรูปภาพและพิกเซล ซึ่งพิกเซลแสดงองค์ประกอบของภาพดิจิทัล ยกตัวอย่างเช่น หากขนาดของรูปภาพคือ 500 x 400 (กว้าง x สูง) จำนวนพิกเซลทั้งหมดในรูปภาพคือ 200,000 พิกเซล [1][2]

2.1.1 ประเภทของรูปภาพ

2.1.1.1 รูปภาพไบนารี (Binary Image)

รูปภาพที่มีค่าความเข้มของพิกเซลที่ไม่ซ้ำกันเพียงสองค่า ได้แก่ 0 (แทนสีดำ) และ 1 (แทนสีขาว) เรียกว่าภาพไบนารี โดยทั่วไปจะถูกนำมาใช้เพื่อเน้นส่วนที่แยกแยะของภาพสี แสดงดังรูปที่ 2.1 [3]

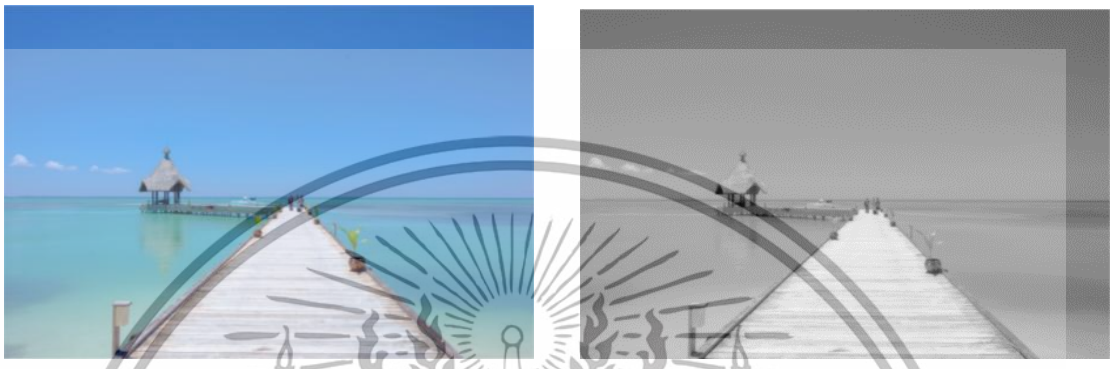


รูปที่ 2.1 รูปภาพไบนารี [3]

2.1.1.2 รูปภาพระดับสีเทา (Grayscale Image)

รูปภาพระดับสีเทาหรือภาพที่มีความลึก 8 บิต ประกอบด้วย 256 สี โดยค่าสีในแต่ละพิกเซลมีค่าอยู่ระหว่าง 0 ถึง 255 แสดงถึงความสว่างตั้งแต่เลข 0 แทนสีดำไปจนเลข 255 แทนสีขาว [3] โดยตัวอย่างภาพ RGB แสดงทิวทัศน์ของสะพานไม้ในทะเลซึ่งมีสีฟ้าของน้ำและท้องฟ้า แสดงดังรูปที่ 2.2(a) ขณะที่รูปภาพระดับสีเทาแสดงดังรูปที่ 2.2(b) แสดงให้เห็นรายละเอียดเช่น

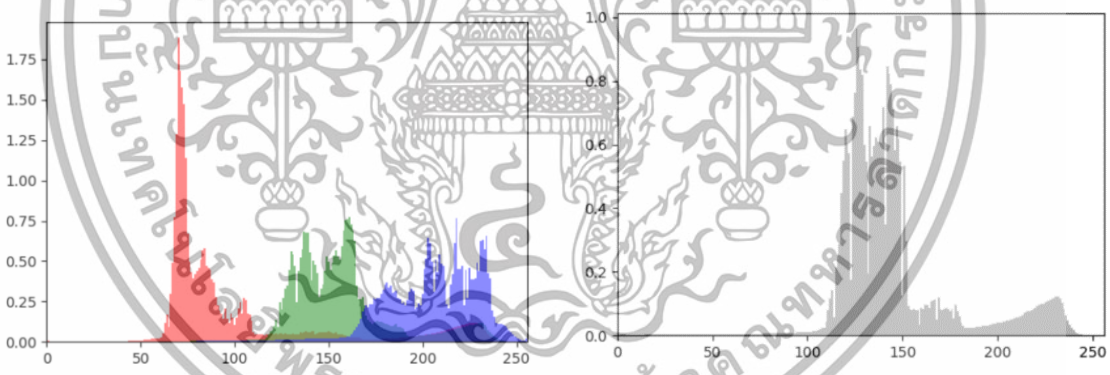
รูปร่างของสะพานและพื้นน้ำได้ชัดเจนกว่า จากฮิสโตแกรมของระดับสี RGB แสดงดังรูปที่ 2.3(a) แสดงการกระจายของสีแดง สีเขียว และสีน้ำเงิน ในขณะที่ฮิสโตแกรมระดับสีเทาแสดงดังรูปที่ 2.3(b) แสดงการกระจายของค่าความเข้มในภาพช่วยให้เห็นการใช้สีและความเข้มที่มีอยู่ในภาพได้อย่างชัดเจน



(a) (b)

รูปที่ 2.2 ความแตกต่างระหว่างรูปภาพ

(a) รูปภาพ RGB (b) รูปภาพระดับสีเทา



(a) (b)

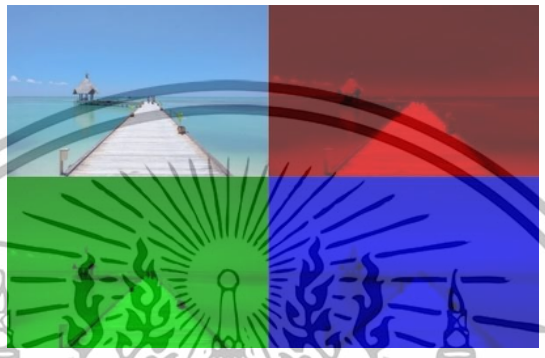
รูปที่ 2.3 ความแตกต่างระหว่างฮิสโตแกรม

(a) ฮิสโตแกรมของรูปภาพ RGB (b) ฮิสโตแกรมของรูปภาพระดับสีเทา

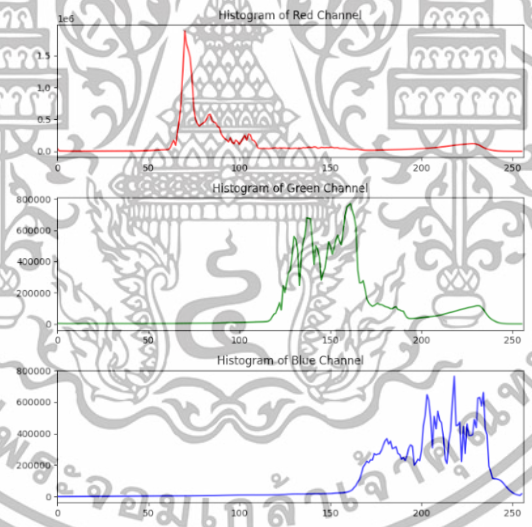
2.1.1.3 รูปภาพสี RGB (RGB Color Image)

ภาพ RGB หรือภาพสีซึ่งเป็นเมทริกซ์ 16 บิตสำหรับคอมพิวเตอร์ คือ 65,536 ค่าสีที่แตกต่างกันในที่นี้ RGB หมายถึงช่องสี 3 สี ได้แก่ ช่องสีแดง ช่องสีเขียว และช่องสีน้ำเงิน ซึ่งภาพ RGB จะเป็นเมทริกซ์ขนาดเท่ากัน 3 เมทริกซ์ เรียกว่า ช่องสัญญาณ ซึ่งแต่ละเมทริกซ์มีค่าตั้งแต่ 0 ถึง 255 จะถูกซ้อนกันทับซ้อนกันเป็นพิกัดที่ไม่ซ้ำกัน เพื่อระบุค่าขององค์ประกอบเมทริกซ์ ดังนั้น

พิกเซลในภาพ RGB จะเป็นสีดำเมื่อค่าพิกเซลเป็น (0, 0, 0) และเป็นสีขาวเมื่อเป็น (255, 255, 255) การรวมกันของตัวเลขใดๆ ก็ตามที่อยู่ระหว่างนั้นทำให้เกิดสีต่างๆ ที่มีอยู่ในธรรมชาติ ตัวอย่างเช่น (255, 0, 0) คือสีแดง, (0, 255, 0) คือสีเขียวและ (0, 0, 255) เป็นสีน้ำเงิน ยกตัวอย่างของภาพ RGB ที่แยกออกเป็นส่วนประกอบของช่อง แสดงดังรูปที่ 2.4 และจะสังเกตว่ารูปร่างของฮิสโตแกรมสำหรับแต่ละช่องจะมีลักษณะแตกต่างกันแสดงดังรูปที่ 2.5 [3]



รูปที่ 2.4 การแยกสีภาพออกเป็นช่องสีแดง เขียว และน้ำเงิน (RGB)



รูปที่ 2.5 ฮิสโตแกรมของภาพในช่องสี RGB

2.2 การประมวลผลภาพ (Image Processing)

การประมวลผลภาพเป็นวิธีการที่เกี่ยวข้องกับการจัดการภาพดิจิทัลโดยใช้ขั้นตอนการประมวลผลคอมพิวเตอร์มาเป็นขั้นตอนสำคัญในการประมวลผลเบื้องต้นในแอปพลิเคชันต่างๆ ยกตัวอย่างเช่น การจดจำใบหน้า (Face recognition) การตรวจจับวัตถุ (Object detection) และการบีบอัดภาพ (Image compression) การประมวลผลภาพทำขึ้นเพื่อปรับปรุงภาพที่มีอยู่หรือเพื่อ

กรองข้อมูลบางอย่างออกจากภาพ ซึ่งเป็นสิ่งสำคัญในแอปพลิเคชันคอมพิวเตอร์วิชันที่ใช้สำหรับการเรียนรู้เชิงลึก (Deep learning) หลายแอปพลิเคชัน [3]

2.2.1 ขั้นตอนการประมวลผลภาพ

ขั้นตอนพื้นฐานในกระบวนการประมวลผลภาพทั่วไปมีดังนี้ [3]

1) การรับภาพ

ภาพจะถูกจับด้วยกล้องและแปลงเป็นดิจิทัล โดยใช้ตัวแปลงแอนะล็อกเป็นดิจิทัลเพื่อประมวลผลเพิ่มเติมในคอมพิวเตอร์

2) การปรับปรุงภาพ

เมื่อได้รับรูปภาพเข้ามาแล้ว รูปภาพจะถูกปรับแต่งให้เป็นไปตามจุดประสงค์ของงาน ซึ่งก็จะมุ่งเน้นรายละเอียดที่ซ่อนอยู่ในภาพหรือสำคัญในภาพ ยกตัวอย่างเช่น การปรับคอนทราสต์และความสว่าง

3) การฟื้นฟูภาพ

เนื่องจากความเสื่อมของรูปภาพสามารถอธิบายได้ด้วยแบบจำลองทางคณิตศาสตร์หรือความน่าจะเป็น เพื่อปรับปรุงคุณลักษณะของภาพให้ดีขึ้น ยกตัวอย่างเช่น การทำ Gaussian Filtering ใช้สำหรับลบสัญญาณรบกวน โดยการใช้ตัวกรอง Gaussian เพื่อทำให้ภาพเนียนเรียบขึ้น แต่ยังคงรักษารายละเอียดบางส่วนไว้ เป็นต้น

4) การประมวลผลภาพสี

การประมวลผลภาพสีเน้นไปที่การจัดการกับภาพสี ยกตัวอย่างเช่น การปรับแก้สีในภาพหรือการสร้างแบบจำลองสี เพื่อให้ภาพมีสีที่ถูกต้องและสมจริงมากขึ้น

5) การประมวลผลเวฟเล็ตและความละเอียดหลายระดับ

การประมวลผลเวฟเล็ตและความละเอียดหลายระดับเป็นวิธีการที่ใช้ในการวิเคราะห์และปรับปรุงภาพโดยการแยกข้อมูลภาพออกเป็นหลายระดับของรายละเอียด ช่วยให้สามารถมองเห็นและจัดการกับรายละเอียดที่ซับซ้อนได้ง่ายขึ้น ยกตัวอย่างเช่น การลดสัญญาณรบกวน การบีบอัดภาพหรือการเพิ่มความคมชัดของภาพในส่วนที่ต้องการ เป็นต้น

2.2.2 เครื่องมือประมวลผลภาพ

2.2.2.1 โอเพ่นซีวี (OpenCV) [4]

ย่อมาจาก Open Source Computer Vision Library ไลบรารีนี้ประกอบด้วยอัลกอริทึมที่ปรับแต่งแล้วมากกว่า 2,000 รายการซึ่งมีประโยชน์สำหรับการมองเห็นด้วยคอมพิวเตอร์และการเรียนรู้ของเครื่อง มีหลายวิธีที่สามารถใช้ OpenCV ในการประมวลผลภาพ ดังต่อไปนี้

1) การแปลงรูปภาพจากพื้นที่สีหนึ่งไปเป็นอีกพื้นที่หนึ่ง ยกตัวอย่างเช่น การแปลงระหว่างรูปภาพ BGR กับรูปภาพสีเทา เป็นต้น

2) การดำเนินการกำหนดขีดจำกัดบนรูปภาพ ยกตัวอย่างเช่น การกำหนดขีดจำกัดแบบง่ายอธิบายได้ว่า หากเลือกค่าความเข้มคงที่ที่ 127 จะหมายความว่า เมื่อพิกเซลที่มีค่าความเข้มมากกว่า 127 จะถูกตั้งเป็นสีขาว (255) และเมื่อพิกเซลที่มีค่าความเข้มน้อยกว่าหรือเท่ากับ 127 จะถูกตั้งเป็นสีดำ (0) และการกำหนดขีดจำกัดแบบปรับเปลี่ยนได้อธิบายได้ว่า เป็นการใช้ค่าเฉลี่ยของพิกเซลที่ผ่านการคำนวณตามสมการทางคณิตศาสตร์มาปรับ

3) การปรับแต่งภาพ ยกตัวอย่างเช่น การใช้ฟิลเตอร์ที่กำหนดเองให้กับภาพและการเบลอภาพ

2.2.3 การประมวลผลภาพด้วยปัญญาประดิษฐ์ (Artificial Intelligence Image Processing)

การใช้เทคโนโลยีปัญญาประดิษฐ์เพื่อเพิ่มประสิทธิภาพในการจัดเก็บและวิเคราะห์ข้อมูลภาพ ประยุกต์เทคนิคการเรียนรู้ของเครื่องและคอมพิวเตอร์วิทัศน์ โดยอัลกอริทึมได้รับการฝึกฝนด้วยชุดข้อมูลขนาดใหญ่ ทำให้โมเดล AI สามารถเรียนรู้และจดจำรูปแบบ วัตถุ และลักษณะเฉพาะภายในภาพได้ ส่งผลให้กระบวนการวิเคราะห์และการตัดสินใจมีความแม่นยำและถูกต้องยิ่งขึ้นผ่านการทำงานแบบอัตโนมัติ [5]

2.2.3.1 หลักการของการประมวลผลภาพด้วยปัญญาประดิษฐ์

การประมวลผลภาพด้วย AI เกิดจากการผสมผสานของอัลกอริทึมขั้นสูง โครงข่ายประสาทเทียม (Artificial Neural Networks: ANN) และการประมวลผลข้อมูล เพื่อนำไปใช้ในการวิเคราะห์ ตีความ และจัดการภาพดิจิทัล ส่งผลให้เกิดกระบวนการทำงานที่อัจฉริยะ สามารถแทนการสังเกตของมนุษย์ในรายละเอียดที่ซับซ้อน ดังนี้

1) รวบรวมและประมวลผลข้อมูลล่วงหน้า

เริ่มต้นด้วยการรวบรวมและจัดการข้อมูลขนาดใหญ่ (Big Data) ในรูปแบบของภาพที่มีป้ายกำกับ เช่น การจดจำวัตถุ และการจัดหมวดหมู่รูปภาพ หลังจากนั้นจึงเข้าสู่กระบวนการประมวลผลล่วงหน้า (Pre-processing) ในการปรับขนาดและการจัดการมาตรฐาน เพื่อให้ข้อมูลสอดคล้องกับการปรับปรุงประสิทธิภาพของโมเดล AI

2) การสกัดคุณลักษณะของลำดับภาพ

การใช้ Convolutional Neural Networks (CNN) ซึ่งเป็นสถาปัตยกรรมการเรียนรู้เชิงลึกประเภทหนึ่ง มักนำมาใช้ในการทำ AI Image Processing โดย CNN จะเรียนรู้และแยก

คุณลักษณะแบบลำดับชั้นจากรูปภาพโดยอัตโนมัติ เริ่มจากเลเยอร์ที่มีตัวกรองการตรวจจับรูปแบบ เช่น ขอบ พื้นผิว และคุณสมบัติที่ซับซ้อนมากขึ้น และส่งผลลัพธ์ที่ได้เข้าสู่กระบวนการฝึกอบรม

3) การฝึกอบรมแบบจำลอง

หลังจากได้ภาพที่ผ่านการประมวลผลล่วงหน้าแล้ว ภาพเหล่านี้จะถูกนำเข้าสู่โมเดล CNN เพื่อฝึกอบรม โดยในขั้นตอนนี้จะมีการปรับน้ำหนักและพารามิเตอร์ภายในโมเดลอย่างต่อเนื่อง เพื่อให้การคาดการณ์มีความแม่นยำมากขึ้น โดยอัลกอริทึมจะปรับค่าพารามิเตอร์ซ้ำ ๆ เพื่อลดข้อผิดพลาดในการคาดการณ์สำหรับการใช้งานจริง

4) การอนุมานและการประยุกต์

เมื่อแบบจำลองผ่านการฝึกอบรมจนพร้อมสำหรับการอนุมานแล้ว โมเดล AI Image Processing จะทำการวิเคราะห์คุณสมบัติของภาพและคาดการณ์ผลอย่างแม่นยำ โดยอ้างอิงจากการฝึกฝนและพัฒนาอย่างต่อเนื่องเพื่อเสริมสร้างการเรียนรู้

2.2.4 การมองเห็นด้วยคอมพิวเตอร์ (Computer Vision)

การมองเห็นด้วยคอมพิวเตอร์เป็นสาขาหนึ่งของปัญญาประดิษฐ์ที่เกี่ยวข้องกับความสามารถของคอมพิวเตอร์ในการรับรู้และตีความภาพหรือวิดีโอในลักษณะคล้ายกับการมองเห็นของมนุษย์ แม้ว่าทั้งคอมพิวเตอร์และมนุษย์จะมีความสามารถในการประมวลผลข้อมูลภาพ แต่ความแตกต่างคือความสามารถของคอมพิวเตอร์มุ่งเน้นไปที่การวิเคราะห์เชิงเทคนิค ขณะที่มนุษย์อาศัยการประเมินและตีความจากประสบการณ์และความเข้าใจในสภาพแวดล้อม ระบบการมองเห็นด้วยคอมพิวเตอร์ในปัจจุบันยังไม่สามารถเข้าใจภาพได้ในระดับที่มนุษย์สามารถทำได้ แม้จะมีการพัฒนาให้สามารถประมวลผลและตีความภาพอย่างถูกต้องมากขึ้น เมื่อรูปภาพมีความไม่ชัดเจนหรือไม่สอดคล้องกับรูปแบบที่ระบบเคยได้รับการฝึกฝนมา ปัญหานี้ยังคงต้องได้รับการปรับปรุง เพื่อให้คอมพิวเตอร์สามารถมองเห็นและตีความภาพได้อย่างใกล้เคียงกับมนุษย์มากที่สุด [6]

2.2.5 การตรวจจับขอบ (Edge Detection)

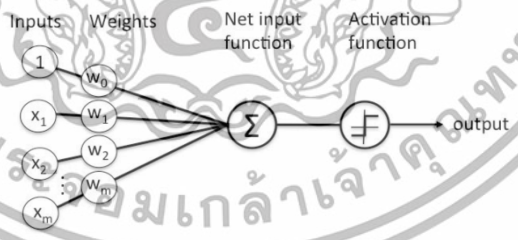
การตรวจจับขอบเป็นกระบวนการที่ใช้ในการประมวลผลภาพ (Image Processing) และการมองเห็นด้วยคอมพิวเตอร์ (Computer Vision) โดยการตรวจจับจะมุ่งเน้นไปที่การหาการเปลี่ยนแปลงที่มีความชัดเจนในระดับความเข้มของรูปภาพ ซึ่งการเปลี่ยนแปลงเหล่านี้มักจะสอดคล้องกับขอบของวัตถุในภาพ กระบวนการตรวจจับขอบช่วยลดความซับซ้อนของภาพโดยการเน้นเฉพาะข้อมูลโครงสร้างที่สำคัญ ยกตัวอย่างเช่น ขอบของวัตถุ ลวดลาย และรูปแบบต่าง ๆ ซึ่งทำให้การวิเคราะห์และตีความภาพเป็นไปได้ง่ายและมีประสิทธิภาพมากขึ้น [7]

2.3 เครือข่ายประสาทเทียม (Neural Network)

เครือข่ายประสาทเทียมหรือเรียกว่าเครือข่ายประสาทเทียมเชิงประดิษฐ์ (Artificial Neural Network) เป็นสถาปัตยกรรมการคำนวณชนิดหนึ่งที่มีพื้นฐานมาจากรูปแบบการทำงานของสมองมนุษย์ ประกอบด้วยหน่วยประมวลผลหลายหน่วยที่เรียกว่า โหนด (Nodes) ซึ่งโหนดเหล่านี้จะส่งผ่านข้อมูลให้กันและกัน คล้ายกับการที่เซลล์ประสาทในสมองส่งสัญญาณไฟฟ้าให้กันและกัน เครือข่ายประสาทเทียมถูกใช้ในระบบการเรียนรู้ของเครื่อง (Machine learning) ซึ่งเป็นหมวดหมู่ของโปรแกรมคอมพิวเตอร์ที่สามารถเรียนรู้ได้โดยไม่ต้องมีคำสั่งที่ชัดเจน เครือข่ายประสาทเทียมถูกใช้ในการเรียนรู้เชิงลึก [8]

2.3.1 การทำงานของเครือข่ายประสาทเทียม

เครือข่ายประสาทเทียมเป็นระบบคำนวณที่ซับซ้อน ออกแบบมาให้ทำงานคล้ายกับสมองมนุษย์ ประกอบด้วยเลเยอร์ 3 ประเภท คือ เลเยอร์รับข้อมูลเข้า (Input layer) ซึ่งทำหน้าที่รับข้อมูลดิบ เลเยอร์ที่ซ่อนอยู่ (Hidden layers) ซึ่งเป็นส่วนที่ประมวลผลข้อมูลภายใน และเลเยอร์ผลลัพธ์ (Output layer) ที่ให้ผลลัพธ์สุดท้าย แสดงดังรูปที่ 2.6 ทั้งหมดนี้ประกอบด้วยเซลล์ประสาทเทียมที่เชื่อมโยงกัน กระบวนการทำงานหลักของเครือข่ายมีสองขั้นตอน คือ การส่งต่อข้อมูลไปข้างหน้า (Forward propagation) ซึ่งข้อมูลจะถูกส่งผ่านเลเยอร์ต่างๆ เพื่อประมวลผล และการย้อนกลับ (Backpropagation) ซึ่งใช้ในการปรับค่าน้ำหนักและพารามิเตอร์ต่างๆ เพื่อให้เครือข่ายทำงานได้แม่นยำยิ่งขึ้น [9]



รูปที่ 2.6 เครือข่ายประสาทเทียม

2.3.1.1 การส่งต่อข้อมูล (Forward Propagation)

1) เลเยอร์รับข้อมูลเข้า (Input Layer)

คุณลักษณะในเลเยอร์รับข้อมูลเข้าจะแสดงด้วยโหนดในเครือข่าย

2) น้ำหนักและการเชื่อมต่อ (Weights and Connections)

ค่าน้ำหนักของการเชื่อมต่อระหว่างเซลล์ประสาทจะแสดงถึงความแข็งแกร่งของการเชื่อมต่อนั้น และค่าของน้ำหนักเหล่านี้จะถูกปรับเปลี่ยนระหว่างการฝึก

3) เลเยอร์ที่ซ่อนอยู่ (Hidden Layers)

เซลล์ประสาทในแต่ละเลเยอร์ที่ซ่อนอยู่จะประมวลผลข้อมูลโดยการคูณข้อมูลเข้ากับค่าน้ำหนัก รวมผลลัพธ์ แล้วผ่านเข้าสู่ฟังก์ชันการกระตุ้น (Activation Function) การทำเช่นนี้จะช่วยให้เครือข่ายสามารถรับรู้รูปแบบที่ซับซ้อนได้

4) ผลลัพธ์ (Output)

ผลลัพธ์สุดท้ายของเครือข่ายประสาทเทียมเกิดขึ้นจากการประมวลผลข้อมูลที่ถูกส่งผ่านแต่ละเลเยอร์ซ้ำๆจนกว่าจะถึงเลเยอร์ผลลัพธ์

2.3.1.2 การย้อนกลับ (Backpropagation)

1) การคำนวณค่าความสูญเสีย (Loss Calculation)

ผลลัพธ์ของเครือข่ายจะถูกประเมินเมื่อเปรียบเทียบกับค่าที่แท้จริง และฟังก์ชันความสูญเสียจะถูกใช้ในการคำนวณความแตกต่าง

2) การลดเกรเดียนต์ (Gradient Descent)

จากนั้นเครือข่ายจะใช้การลดเกรเดียนต์เพื่อลดค่าความสูญเสีย โดยการปรับค่าน้ำหนักตามอนุพันธ์ของค่าความสูญเสียที่สัมพันธ์กับน้ำหนักแต่ละตัวเพื่อลดความไม่ถูกต้อง

3) การปรับน้ำหนัก (Adjusting weights)

น้ำหนักจะถูกปรับที่การเชื่อมต่อแต่ละจุดโดยการใช้กระบวนการวนซ้ำ (iterative process) หรือการย้อนกลับ (backpropagation) ข้ามเครือข่ายไปทางด้านหลัง

4) การฝึกอบรม (Training)

ในระหว่างการฝึกอบรม กระบวนการทั้งหมดของการส่งต่อข้อมูล การคำนวณค่าความสูญเสีย และการย้อนกลับจะถูกทำซ้ำอย่างต่อเนื่อง ซึ่งทำให้เครือข่ายสามารถปรับตัวและเรียนรู้รูปแบบจากข้อมูลได้

5) ฟังก์ชันการกระตุ้น (Activation Functions)

การสร้างความไม่เป็นเชิงเส้นในโมเดลจะเกิดจากฟังก์ชันการกระตุ้น เช่น Rectified Linear Unit (ReLU) หรือ Sigmoid การตัดสินใจว่าจะแสดงผลเซลล์ประสาทหรือไม่ขึ้นอยู่กับข้อมูลนำเข้าที่มีน้ำหนักรวมกันทั้งหมด

2.4 การจำแนกประเภท (Classification)

การจำแนกประเภท (Classification) เป็นกระบวนการเรียนรู้ของเครื่อง (Machine Learning) แบบมีผู้สอน (Supervised Learning) โดยมีวัตถุประสงค์เพื่อให้แบบจำลองสามารถทำนายป้ายกำกับที่ถูกต้องสำหรับข้อมูลที่ป้อนเข้ามา ในขั้นตอนการทำงาน แบบจำลองจะถูกฝึกด้วย

ข้อมูลฝึกอบรมที่มีป้ายกำกับอยู่แล้ว จากนั้นจะนำไปประเมินผลด้วยข้อมูลทดสอบ เพื่อดูว่ามีความแม่นยำเพียงใด ก่อนที่จะนำไปใช้กับข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน [10]

2.4.1 ประเภทของการจำแนก

การจำแนกประเภทสามารถแบ่งออกได้เป็น 2 ประเภท ดังนี้

2.4.1.1 การจำแนกประเภทแบบทวิภาค (Binary Classification)

การจำแนกสิ่งต่างๆออกเป็น 2 ประเภทเท่านั้น ยกตัวอย่างเช่น การจำแนกสุนัขกับแมว และการจำแนกตัวเลขกับตัวอักษร เป็นต้น

2.4.1.2 การจำแนกประเภทแบบหลายคลาส (Multi-class Classification)

การจำแนกประเภทที่มีคลาสที่ต้องการมากกว่า 2 คลาส ยกตัวอย่างเช่น การจำแนกสายพันธุ์สุนัข ซึ่งมีหลายสายพันธุ์ การจำแนกตัวเลขที่เขียนด้วยลายมือ และการจำแนกวัตถุในภาพ เป็นต้น [11]

2.4.2 ขั้นตอนกระบวนการฝึกการจำแนกประเภท

1) การจัดเตรียมข้อมูล

ข้อมูลที่ใช้สำหรับการฝึก เรียกว่า ข้อมูลชุดเรียนรู้ (Training Set) นำมาแยกประเภทผลลัพธ์ด้วยการติดป้ายกำกับ (Labels/Class) เป็นผลเฉลย จากนั้นนำข้อมูลที่ติดป้ายแล้วไปใช้ในการฝึกของเครื่องที่ทำงานผ่านอัลกอริทึมสำหรับสร้างโมเดลที่ใช้ในการทำนายผลลัพธ์ โดยนำข้อมูลใหม่หรือข้อมูลชุดทดสอบ (Test Set) ไปใช้ทดสอบประสิทธิภาพของโมเดล [12]

2) การประมวลผลข้อมูลเบื้องต้น

ข้อมูลที่เก็บรวบรวมจะได้รับการประมวลผลเบื้องต้น ซึ่งอาจเกี่ยวข้องกับงานต่างๆ เช่น การทำให้เป็นมาตรฐาน (Normalization) การปรับขนาด (Resizing) และการเพิ่มข้อมูล (Data Augmentation) เพื่อให้แน่ใจว่าข้อมูลเหมาะสำหรับการฝึกอบรม

3) สถาปัตยกรรมของโมเดล

การกำหนดสถาปัตยกรรมของโมเดลการเรียนรู้เชิงลึก ซึ่งรวมถึงการเลือกประเภทของเลเยอร์ จำนวนเลเยอร์และการเชื่อมต่อระหว่างเลเยอร์

4) ฟังก์ชันการสูญเสีย

การกำหนดฟังก์ชันการสูญเสีย (Loss Function) เพื่อวัดความแตกต่างระหว่างผลลัพธ์ที่คาดการณ์และป้ายกำกับที่แท้จริง เป้าหมายของกระบวนการฝึกอบรมคือ การลดค่าการสูญเสียนั้นให้น้อยที่สุด

5) การเพิ่มประสิทธิภาพ

อัลกอริทึมการเพิ่มประสิทธิภาพ (Optimization Algorithm) เช่น การลดระดับ Gradient Descent จะถูกใช้ในการปรับพารามิเตอร์ของโมเดลแบบวนซ้ำ ซึ่งทำได้โดยการคำนวณเกรเดียนต์ของฟังก์ชันการสูญเสียที่สัมพันธ์กับพารามิเตอร์ของโมเดลและปรับพารามิเตอร์เพื่อลดค่าการสูญเสีย

6) การย้อนกลับ (Backpropagation)

ในระหว่างกระบวนการเพิ่มประสิทธิภาพ จะใช้วิธีการย้อนกลับเพื่อคำนวณค่าเกรเดียนต์ของฟังก์ชันการสูญเสีย (Loss Function) อย่างมีประสิทธิภาพ โดยการแพร่กระจายจากเลเยอร์ผลลัพธ์ไปยังเลเยอร์นำเข้า

7) ลูปการฝึกอบรม

กระบวนการฝึกอบรมจะเกี่ยวข้องกับการป้อนชุดข้อมูลนำเข้าเป็นกลุ่ม (Batch) เข้าไปในโมเดลซ้ำ ๆ คำนวณค่าการสูญเสีย และปรับพารามิเตอร์โดยใช้อัลกอริทึมการเพิ่มประสิทธิภาพ ซึ่งจะทำซ้ำเช่นนี้เป็นจำนวนครั้งที่กำหนดหรือจนกว่าโมเดลจะรวมเข้ากับระดับที่น่าพอใจ [13]

2.4.3 ข้อกำจัดของโมเดลการเรียนรู้เชิงลึกสำหรับการจำแนกประเภท

1) ความซับซ้อนในการคำนวณ (Computational Complexity)

โมเดลการเรียนรู้เชิงลึกอาจใช้ทรัพยากรการคำนวณสูงในการฝึกฝน โดยเฉพาะกับชุดข้อมูลขนาดใหญ่ เนื่องจากโมเดลต้องเรียนรู้ความสัมพันธ์ระหว่างลักษณะต่างๆ ของข้อมูล ซึ่งอาจเป็นงานที่ซับซ้อนมาก

2) ความต้องการข้อมูล (Data Requirements)

โมเดลการเรียนรู้เชิงลึกต้องการข้อมูลที่มีการติดป้ายกำกับจำนวนมากในการฝึกอบรม ข้อมูลเหล่านี้อาจยากและมีค่าใช้จ่ายสูงในการรวบรวม

3) การเรียนรู้เกินจริง (Overfitting)

โมเดลการเรียนรู้เชิงลึกสามารถมีปัญหาเรื่องการเรียนรู้เกินจริง ซึ่งหมายความว่าโมเดลเรียนรู้ข้อมูลฝึกอบรมมากเกินไปและไม่สามารถทำนายกับข้อมูลใหม่ได้ ปัญหานี้อาจเกิดจากหลายปัจจัย เช่น การใช้โมเดลที่ซับซ้อนเกินไปหรือการไม่มีข้อมูลฝึกอบรมเพียงพอ

4) อคติ (Bias)

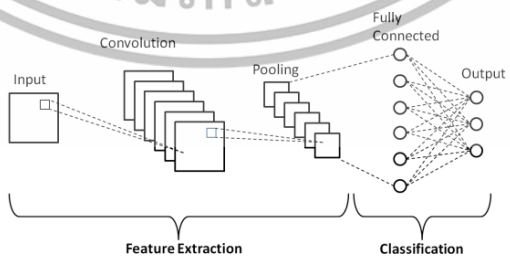
โมเดลการเรียนรู้เชิงลึกอาจมีอคติ ซึ่งหมายความว่าโมเดลอาจทำการทำนายที่แตกต่างกันสำหรับกลุ่มคนที่แตกต่างกัน สาเหตุอาจมาจากวิธีการเก็บข้อมูลหรือวิธีการฝึกอบรมโมเดล [13]

2.5 อัลกอริทึมการเรียนรู้เชิงลึก (Deep Learning Algorithm)

การเรียนรู้เชิงลึกสามารถกำหนดได้ว่าเป็นวิธีการเรียนรู้ของเครื่องและปัญญาประดิษฐ์ โดยอิงจากการทำงานของสมองมนุษย์เพื่อให้ตัดสินใจได้อย่างมีประสิทธิผล อัลกอริทึมการเรียนรู้เชิงลึกถูกสร้างขึ้นแบบไดนามิกเพื่อทำงานผ่านชั้นต่างๆ ของเครือข่ายประสาท ซึ่งเป็นเพียงชุดของเครือข่ายการตัดสินใจที่ได้รับการฝึกไว้ล่วงหน้า มีบทบาทสำคัญในการกำหนดคุณลักษณะ ซึ่งเรียนรู้เกี่ยวกับรูปภาพโดยส่งผ่านเลเยอร์เครือข่ายประสาทแต่ละเลเยอร์ ซึ่งมีความไวสูงในการตรวจจับคุณลักษณะระดับต่ำของรูปภาพ เช่น ขอบและพิกเซล จากนั้นเลเยอร์ที่รวมกันจะนำข้อมูลนี้และสร้างการแสดงแบบองค์รวมโดยการเปรียบเทียบกับข้อมูลก่อนหน้า [14]

2.5.1 สถาปัตยกรรมของ Convolutional Neural Networks (CNN)

ในการเรียนรู้เชิงลึก CNN เป็นเครือข่ายที่ใช้กันมากที่สุดในการจำแนกภาพ ซึ่งได้รับแรงบันดาลใจจากระบบการมองเห็นของมนุษย์ และแตกต่างจากอัลกอริทึมการจดจำรูปแบบอื่นๆ โดยผสมผสานทั้งการสกัดคุณลักษณะ (Feature Extraction) และการจำแนกประเภท (Classification) ประกอบด้วย 5 เลเยอร์ ได้แก่ เลเยอร์อินพุต (Input Layer) เลเยอร์คอนโวลูชั่น (Convolution Layer) เลเยอร์พูลลิ่ง (Pooling Layer) เลเยอร์เชื่อมต่ออย่างสมบูรณ์ (Fully Connected Layer) และเลเยอร์เอาต์พุต (Output Layer) แสดงดังรูปที่ 2.7 โดยการสกัดคุณลักษณะรวมถึงเลเยอร์ Input Convolution และ Pooling ขณะที่การจำแนกประเภทมีเลเยอร์ Fully Connected และ Output เลเยอร์ Input ระบุขนาดคงที่สำหรับภาพ ซึ่งจะมีการคอนโวลูชั่นด้วยเคอร์เนลที่เรียนรู้หลายตัวก่อนที่เลเยอร์ Pooling จะลดขนาดภาพลงเพื่อรักษาข้อมูลไว้ แผนที่คุณลักษณะที่ได้จะถูกนำไปรวมเลเยอร์ Fully Connected โดยมีนิวรอน Output 1 ตัวสำหรับแต่ละหมวดหมู่ในเลเยอร์ Output ซึ่งผลลัพธ์คือการจำแนกประเภท ส่วนการสกัดคุณลักษณะและการจำแนกประเภทจะมีเลเยอร์หลายชั้นเพื่อเพิ่มประสิทธิภาพ [15]



รูปที่ 2.7 สถาปัตยกรรมของ CNN [13]

สถาปัตยกรรมของ Convolutional Neural Network (CNN) แสดงดังรูปที่ 2.8 มีจำนวนชั้นเลเยอร์ทั้งหมด 10 ชั้น (3 Convolution Layers, 3 Max Pooling Layers, 1 Flatten

Layer, 1 Fully Connected Layer และ 1 Dropout Layer) ขนาดอินพุตที่โมเดล CNN ใช้คือ (640, 480, 3) โดยที่ 640 คือความกว้าง, 480 คือความสูง และ 3 คือช่องสัญญาณ RGB โดย Convolution Layer ใช้ฟิลเตอร์ขนาด 3x3 พิกเซล และถูกออกแบบมาให้เอาต์พุตของ Feature Map มีขนาดเท่ากัน ด้วยการใช้ Padding แบบ Same โดยมี Max Pooling Layers ตามด้วย Flatten Layer ที่แปลงข้อมูลจาก 3 มิติเป็น 1 มิติ และเชื่อมต่อไปยัง Fully Connected Layer ที่ใช้ Dropout Layer มีอัตราการดรอป 0.5 เพื่อป้องกันการเกิด Overfitting ซึ่งโมเดลนี้ใช้เฟรมเวิร์ค PyTorch ในการพัฒนา

รูปที่ 2.8 โครงสร้างโมเดล CNN

2.5.1.1 การดึงคุณลักษณะ (Feature Extraction)

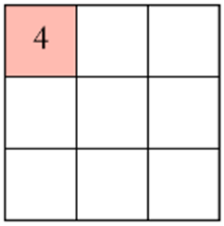
ความซับซ้อนของ CNN อยู่ที่ระบบการคำนวณที่อิงกับหลักการพื้นฐานของโครงข่ายประสาทเทียม ซึ่งต้องใช้คณิตศาสตร์รองรับ CNN ใช้หลักการคอนโวลูชันเชิงพื้นที่ (Spatial Convolution) ในการประมวลผลภาพ โดยเริ่มจากการกำหนดค่าตัวกรอง (Filter) หรือเคอร์เนล (Kernel) เพื่อดึงคุณลักษณะสำคัญจากภาพ [16][17]

1) ลักษณะของฟิลเตอร์

ฟิลเตอร์มักเป็นตาราง 2 มิติ ขนาด 3x3 แสดงดังรูปที่ 2.9 ที่ใช้ในการดึงคุณลักษณะสำคัญของภาพ โดยจะเลื่อนไปทีละจุดทั่วภาพ ผลลัพธ์ที่ได้คือ แผนผังคุณลักษณะ (Feature Map) แสดงดังรูปที่ 2.10

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

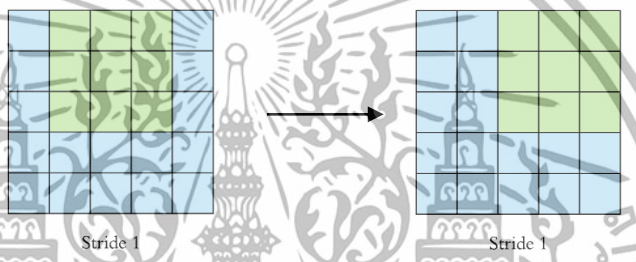
รูปที่ 2.9 ลักษณะของฟิลเตอร์ขนาด 3x3



รูปที่ 2.10 แผนผังคุณลักษณะ

2) Stride

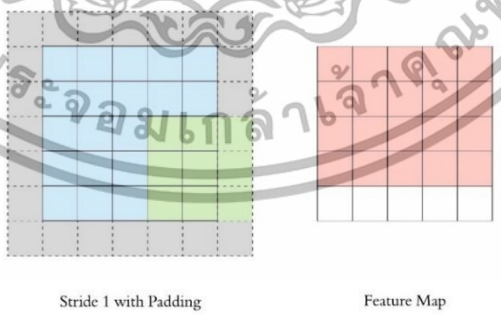
ตัวกำหนดที่ว่าจะทำการเลื่อนฟิลเตอร์ไปด้วยเสต็ปเท่าไหร่ แสดงดังรูปที่ 2.11 เป็นการเคลื่อนด้วยที่กำหนด Stride = 1



รูปที่ 2.11 การเคลื่อนที่ของฟิลเตอร์ที่กำหนด Stride = 1

3) Padding

การเพิ่มพื้นที่รอบๆของภาพ โดยเติม 0 เพื่อให้การทำคอนโวลูชันแล้วมีผลลัพธ์ของแผนผังคุณลักษณะยังคงมีขนาดเท่ากับอินพุตแสดงดังรูปที่ 2.12 แบ่งออกเป็น 2 ประเภท คือ Valid Padding ที่จะเป็นการไม่ใช่ Padding และ Same Padding ที่จะเป็นการใช้งาน Padding



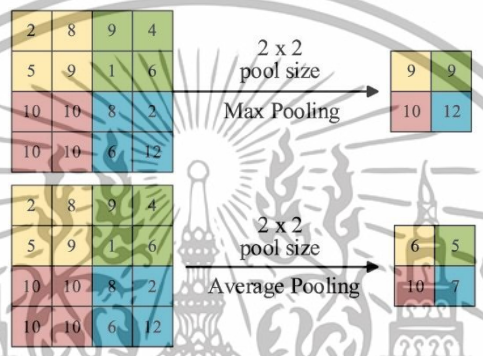
รูปที่ 2.12 ตัวอย่างของการเพิ่ม Padding

4) Pooling

หลังจากกระบวนการคอนโวลูชัน มักจะมีการทำ Pooling เพื่อลดขนาดของข้อมูล มีวัตถุประสงค์เพื่อลดจำนวนพารามิเตอร์ในเครือข่าย ช่วยลดระยะเวลาในการฝึกฝนโมเดล และลดโอกาสเกิดปัญหา overfitting โดยการทำงานของ Pooling จะลดขนาดความสูงและความ

กว้างของผังคุณลักษณะ (Feature Map) แต่ยังคงรักษาความลึก (Depth) ไว้เหมือนเดิม โดยทั่วไป การ Pooling มี 2 แบบคือ Pooling ด้วยค่าสูงสุด (Max Pooling) และ Pooling ด้วยค่าเฉลี่ย (Average Pooling)

ในกรณีที่ต้องการ Pooling ด้วยค่าสูงสุด (Max Pooling) ตัวอย่างการกำหนดให้ตัวกรองมีขนาด 2x2 และ Stride 2 โดยจะทำการเลือกค่าที่สูงที่สุดในแต่ละ Pool ที่ถูกทาบด้วยตัวกรอง และการทำ Pooling ด้วยค่าเฉลี่ย (Average Pooling) แสดงดังรูปที่ 2.13 จะทำการหาค่าเฉลี่ยในแต่ละ Pool ที่ถูกทาบด้วยตัวกรอง



รูปที่ 2.13 ตัวอย่างการทำ Max Pooling และ Average Pooling

2.5.1.2 การเรียนรู้คุณลักษณะและเลเยอร์ใน CNN

1) Convolution Layer

ทำหน้าที่รับอินพุตเข้ามา แปลงรูปภาพให้เป็นพิกเซล ที่กำหนดให้เป็น 0-255 เพราะรูปภาพที่นำเข้ามาเป็นภาพแบบ RGB ที่มีช่องสัญญาณ 3 สี หลังจากนั้นใช้กระบวนการทางคณิตศาสตร์ โดยการนำรูปภาพอินพุตคูณกับฟิลเตอร์หรือเคอร์เนลทำหน้าที่ดึงคุณลักษณะสำคัญในรูปออกมาเป็นแผนผังคุณลักษณะ (Feature Map)

การคำนวณหาขนาดเอาต์พุตของ Convolution Layer สามารถหาได้ดังสมการที่ (2.1), (2.2) และ (2.3) [18]

$$W_2 = \frac{W_1\text{-}F+2P}{S} + 1 \tag{2.1}$$

$$H_2 = \frac{H_1\text{-}F+2P}{S} + 1 \tag{2.2}$$

$$D_2 = K \tag{2.3}$$

โดยที่ $W_1 \times H_1 \times D_1$ คือ ขนาดของอินพุต

K คือ จำนวนของฟิลเตอร์

F คือ ขนาดของฟิลเตอร์

S คือ สเต็ปในการเคลื่อนของฟิลเตอร์

P คือ ส่วนที่ถูกเพิ่มเข้าไปในค่าอินพุต

ซึ่งสามารถหาค่าของ P ได้จากสมการที่ (2.4)

$$H_2 = \frac{F\text{-}S}{2} \tag{2.4}$$

จะได้ค่าขนาดของเอาต์พุตเท่ากับ $W_2 \times H_2 \times K$

2) Activation Layer

ขั้นตอนสำคัญที่ทำให้โครงข่ายประสาทเทียมเกิดการเรียนรู้จากข้อมูลที่ผ่านมาจาก Convolution Layer โดยการเปลี่ยนข้อมูลให้มีความไม่เชิงเส้น (Non-linear) ผ่านฟังก์ชันกระตุ้น (Activation Function) ซึ่งฟังก์ชันที่นิยมใช้ในโครงข่ายประสาทเชิงลึก ได้แก่

2.1) ฟังก์ชันเรคติไฟด์เชิงเส้น (Rectified Linear Unit : ReLU)

ReLU (Rectified Linear Unit) เป็นฟังก์ชันการเปิดใช้งานที่ใช้ในเครือข่ายประสาทเทียม โดยฟังก์ชันนี้จะคืนค่าผลลัพธ์ที่เป็นศูนย์หากอินพุตมีค่าน้อยกว่าศูนย์ และจะคืนค่าของอินพุตเองหากมีค่ามากกว่าหรือเท่ากับศูนย์ แสดงดังสมการที่ (2.5)

$$\mathrm{f}(x) = max(0, x) = \begin{cases} 0 \; for \; x \leq 0 \\ x \; for \; x > 0 \end{cases} \tag{2.5}$$

และสมการ Derivative ReLU function = 1 แสดงดังสมการที่ (2.6)

$$\mathrm{f}'(x) = max(0, x) = \begin{cases} 0 \; for \; x \leq 0 \\ x \; for \; x > 0 \end{cases} \tag{2.6}$$

ฟังก์ชัน ReLU ช่วยเพิ่มความไม่เป็นเชิงเส้นในโมเดล และมีข้อดีคือการคำนวณที่รวดเร็ว อีกทั้งยังช่วยลดปัญหาการเกิด Vanishing Gradient ในระหว่างการฝึกเครือข่ายประสาทเทียม [19]

2.2) ฟังก์ชัน SoftMax

เป็นฟังก์ชันที่ใช้ในการแปลงเวกเตอร์ของค่าโลจิต (Logit) จำนวนจริงให้เป็นความน่าจะเป็น (Probability) ซึ่งผลรวมของความน่าจะเป็นทั้งหมดจะเท่ากับ 1 ฟังก์ชันนี้ช่วยในการจำแนกประเภทโดยการเน้นค่าที่สูงที่สุดในเวกเตอร์และลดค่าที่ต่ำกว่า แสดงดังสมการที่ (2.7)

$$\sigma(z)i = \frac{e^{z_i}}{\sum j=1^{K} e^{z_j}} \tag{2.7}$$

โดยที่ $\sigma(z)i$ คือ เวกเตอร์อินพุต

z_i คือ เวกเตอร์อินพุตตัวที่ i โดยที่ $i=1 \rightarrow n$

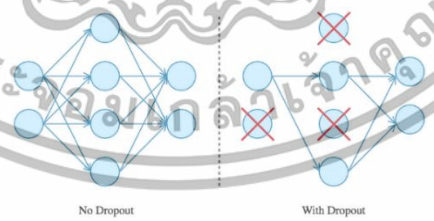
$\sum j$ คือ ผลรวมของค่าเวกเตอร์อินพุตที่ใช้ในการทำ Normalization เพื่อให้ค่าเอาต์พุตของฟังก์ชันที่ได้ออกมามีค่าเท่ากับ 1 โดย $z_j = z_i$ เมื่อ $i = j$ [20]

3) Batch Normalization Layer

เป็นเทคนิคการปรับขนาดของการทำงาน (Activation) ภายใน Hidden Layer ให้อยู่ในระดับที่เหมาะสม ก่อนนำไปเป็นอินพุตของเลเยอร์ถัดไป ทำให้ใน Neural Network สามารถเรียนรู้ได้ด้วยตัวเองอย่างเป็นอิสระ ลดการผูกติดกับเลเยอร์อื่นเพื่อให้การฝึกโมเดลมีประสิทธิภาพและเสถียรยิ่งขึ้น [21]

4) Dropout Layer

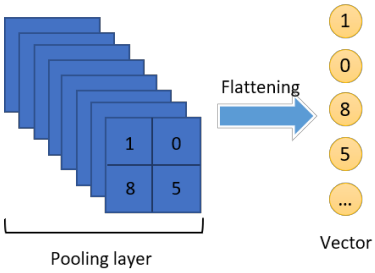
เป็นเทคนิคที่ใช้ใน Neural Networks เพื่อป้องกันการเกิด Overfitting โดยการสุ่มปิด (Drop) หน่วยประมวลผล (Neurons) บางส่วนในระหว่างการฝึกฝน โมเดลจะต้องเรียนรู้ที่จะทำงานโดยไม่พึ่งพาหน่วยเหล่านี้ ซึ่งช่วยให้โมเดลมีความสามารถในการ Generalization ได้ดีขึ้น ในแต่ละรอบการฝึก (Training Epoch) จะมีการสุ่มปิดหน่วยประมวลผลในอัตราส่วนที่กำหนด ทำให้โมเดลไม่สามารถใช้ข้อมูลจากหน่วยที่ถูกปิดในการตัดสินใจระหว่างการเรียนรู้ ทำให้ลดความซับซ้อนของโมเดลและช่วยเพิ่มประสิทธิภาพในข้อมูลที่ไม่เคยเห็นมาก่อน วิธีการทำ Dropout คือการสร้าง Boolean matrix มิติเท่ากับ Matrix ของ Activation ที่ต้องการใช้ Dropout โดยสุ่มค่า Yes/No ตามสัดส่วนที่กำหนด แสดงดังรูปที่ 2.14 ยกตัวอย่างเช่น กำหนดให้ปิด 25% คือให้ทั้ง Matrix มีค่า True 75% ค่า False 25% จากนั้นนำ Dropout matrix นี้ไปคูณแบบ Element-wise เข้ากับ Activation matrix ก็จะได้ Activation matrix ใหม่ที่บาง Element มีค่าเท่ากับศูนย์ในตำแหน่งเดียวกับที่ใน Dropout matrix มีค่าเป็น False จากนั้นนำสัดส่วนการปิดไปหารแบบ Element-wise ออกจาก Activation matrix ใหม่ เพื่อทำการ Normalize [22][23][24]



รูปที่ 2.14 การทำ Dropout Layer

5) Flatten Layer

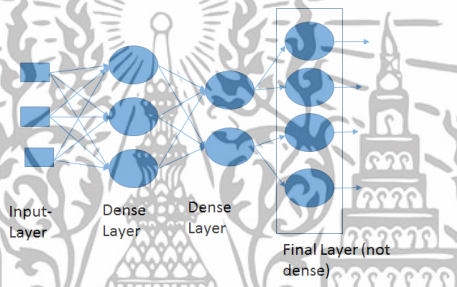
ทำหน้าที่แปลงข้อมูลจากรูปแบบหลายมิติให้เป็นเวกเตอร์หนึ่งมิติ เพื่อเตรียมนำข้อมูลเข้าสู่เลเยอร์ Fully Connected ถัดไป โดยจะไม่เปลี่ยนแปลงค่าของข้อมูล แต่เพียงแค่ปรับรูปแบบให้เหมาะสมกับการประมวลผล แสดงดังรูปที่ 2.15 [25]



รูปที่ 2.15 การทำงานของ Flatten [26]

6) Dense Layer

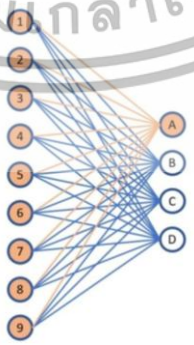
เป็นเลเยอร์ที่เชื่อมต่อโหนดทุกโหนดในชั้นเลเยอร์หนึ่งกับทุกโหนดในเลเยอร์ถัดไป หรือที่เรียกว่า Fully Connected Layer แสดงดังรูปที่ 2.16 [25]



รูปที่ 2.16 ตัวอย่างการใช้งาน Dense Layer

7) Fully Connected Layer

เป็นขั้นตอนสุดท้ายที่ทำให้ทุกชั้นเชื่อมโยงกันอย่างสมบูรณ์ โดยที่ผลลัพธ์จากการคอนโวลูชันและการ Pooling จะทำให้สามารถดึงคุณลักษณะสำคัญจากรูปภาพที่นำเข้าได้ และจะทำการคัดกรองคุณลักษณะเหล่านั้นผ่านฟิลเตอร์ เพื่อให้ได้ผลลัพธ์ในรูปแบบของแต่ละคลาส แสดงดังรูปที่ 2.17 [27] โดยที่ A B C และ D หมายถึงคลาสที่จำแนกออกมา



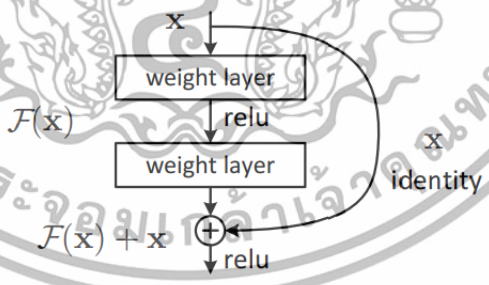
รูปที่ 2.17 การแยกของแต่ละคลาสใน Fully Connected Layer

2.5.2 สถาปัตยกรรมของ Pre-Trained Model

โมเดลที่ผ่านการฝึกอบรมล่วงหน้า (Pre-Trained Model) หมายถึง โมเดลเครือข่ายประสาทที่ได้รับการฝึกอบรมจากชุดข้อมูลขนาดใหญ่เพื่อดำเนินการอย่างการจำแนกวัตถุ โมเดลเหล่านี้ได้รับการฝึกฝนโดยใช้ทรัพยากรการคำนวณและข้อมูลจำนวนมาก ส่งผลให้เข้าใจรูปแบบและคุณลักษณะพื้นฐานภายในข้อมูลและมีศักยภาพในการปรับแต่งเพิ่มเติม [28] การใช้เทคนิค Transfer learning เพื่อลดเวลาในการฝึกโมเดลและลดปริมาณการใช้ข้อมูลด้วยการใช้โมเดลที่ผ่านการฝึกอบรมล่วงหน้าช่วยให้สามารถกำหนดค่าโมเดลใหม่ [29] โดยปริญญานิพนธ์นี้เลือกใช้สถาปัตยกรรม Resnet-18

2.5.2.1 สถาปัตยกรรมของ ResNet

ResNet หรือ Residual Network เป็นสถาปัตยกรรมเครือข่ายประสาทชนิดหนึ่ง ได้รับการออกแบบมาเพื่อแก้ปัญหา Vanishing gradients ในเครือข่ายประสาทเทียมเชิงลึกโดยไม่ได้ทำให้พารามิเตอร์เพิ่มขึ้น สถาปัตยกรรม ResNet มีหลายรูปแบบ เช่น ResNet-18, ResNet-34, ResNet-50, ResNet-101 และ ResNet-152 จำนวนในแต่ละรูปแบบจะสอดคล้องกับจำนวนเลเยอร์ในเครือข่าย ตัวอย่างเช่น ResNet-50 มี 50 เลเยอร์ ในขณะที่ ResNet-152 มี 152 เลเยอร์ [30] โดยโครงสร้างของ Residual learning แสดงดังรูปที่ 2.18 ใน Residual Network อินพุตของบล็อกจะถูกเพิ่มไปยังเอาต์พุตของบล็อกเพื่อสร้าง Residual connection เอาต์พุต H(x) สามารถแสดงได้ดังนี้ H(x) = F(x) + x ซึ่ง F(x) แสดงถึง Residual mapping ที่เครือข่ายเรียนรู้ [31]

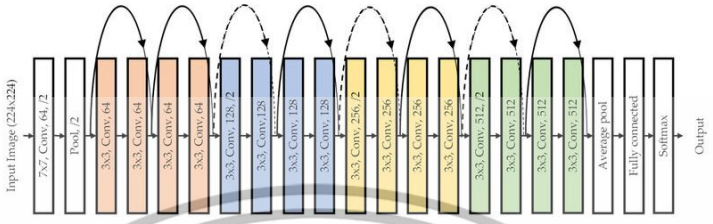


รูปที่ 2.18 Residual Learning

1) Resnet-18

สถาปัตยกรรมของ ResNet-18 แสดงดังรูปที่ 2.19 มีจำนวนชั้นเลเยอร์ทั้งหมด 18 เลเยอร์ (17 Convolution Layers, 1 Fully-Connected Layer และ SoftMax Layer ที่ใช้ในการจำแนก) ขนาดอินพุตที่ ResNet-18 นำมาใช้คือ (224, 224, 3) โดยที่ 224 คือความกว้างและความสูง 3 คือช่องสัญญาณ RBG โดย Convolution Layer ใช้ฟิลเตอร์ขนาด 3x3 พิกเซลและโครงข่ายถูกออกแบบมาให้เอาต์พุตของ Feature Map มีขนาดเท่ากัน มี Residual Blocks จำนวน 4

ชุด โดยแต่ละชุดประกอบด้วย Convolution Layer จำนวน 2 ชั้นพร้อม Skip Connection เพื่อให้ง่ายต่อการส่งค่าข้อมูลไปยังชั้นที่ลึกลงไป มี Average Pooling ตามด้วย Fully-Connected Layer ที่ใช้ SoftMax Layer เพื่อรับอัลกอริทึมของความน่าจะเป็น ซึ่งโมเดลนี้ใช้เฟรมเวิร์ค PyTorch [32]



รูปที่ 2.19 สถาปัตยกรรมของ ResNet-18

2.5.3 Loss Function

Loss Function หรือฟังก์ชันการสูญเสียที่ใช้ในการปรับค่าถ่วงน้ำหนักของโมเดลระหว่างการฝึก โดยที่มีเป้าหมายเพื่อให้โมเดลมีค่า Loss เหลือน้อยที่สุด ซึ่ง Loss Function มีหลากหลายประเภทสามารถเลือกใช้ได้ตามความเหมาะสมกับปัญหา ยกตัวอย่างเช่น Cross-entropy และ MSE

ครอส-เอนโทรปี (Cross-entropy) เป็นฟังก์ชันการสูญเสียที่ใช้บอกประสิทธิภาพของแบบจำลองที่ใช้ในการจำแนก โดยสมการ Cross-entropy Multi-class แสดงดังสมการที่ (2.8) [33]

$$L = -\sum_{i}^{C} t_i \log(s_i) \tag{2.8}$$

2.5.4 Hyperparameter tunning

การปรับแต่งไฮเปอร์พารามิเตอร์ (Hyperparameter Tuning) คือกระบวนการทดลองค่าไฮเปอร์พารามิเตอร์ต่าง ๆ เพื่อหาชุดค่าที่ทำให้โมเดลมีประสิทธิภาพดีที่สุด โดยใช้การวิเคราะห์ทางสถิติ เช่น ฟังก์ชันการสูญเสีย เพื่อประเมินผลลัพธ์ [34]

1) Grid Search

เป็นการทดสอบทุกค่าที่เป็นไปได้ตามชุดไฮเปอร์พารามิเตอร์ที่กำหนดไว้ล่วงหน้า เพื่อค้นหาค่าที่ทำให้โมเดลทำงานได้ดีที่สุด ซึ่งช่วยให้การปรับแต่งไฮเปอร์พารามิเตอร์เป็นไปอย่างเป็นระบบแต่ใช้เวลามาก [35]

2) Bayesian Optimizer

ใช้ข้อมูลจากการทดสอบก่อนหน้าเพื่อคาดการณ์ค่าที่ดีที่สุดในการทดลองครั้งต่อไป ช่วยลดจำนวนการทดลองโดยไม่ต้องทดสอบทุกค่า ยกตัวอย่างเช่น Grid Search ซึ่งเพิ่มความเร็วในการ

ค้นหา และ Optuna เป็นไลบรารีที่ใช้ Bayesian Optimization ในการเลือกค่าที่คาดว่าจะให้ผลลัพธ์ดีที่สุด ช่วยลดการทดลองที่ไม่จำเป็นและเพิ่มประสิทธิภาพในการปรับแต่ง [36]

2.6 เฟรมเวิร์คสำหรับการเรียนรู้เชิงลึก (Deep Learning Framework)

2.6.1 PyTorch

PyTorch คือ Machine Learning Framework แสดงในรูปที่ 2.20 สามารถใช้ในการสร้างโมเดล Machine Learning และ Neural Network ได้ มีเครื่องมือให้สามารถควบคุมการทำงานของโมเดลที่หลากหลายและใช้ง่ายอีกทั้งยังสามารถนำโมเดลไปใช้งานจริง ในปัจจุบันเป็น Open Source ภายใต้ Linux Foundation มีความสามารถของการใช้การประมวลผลด้วย GPU [37]

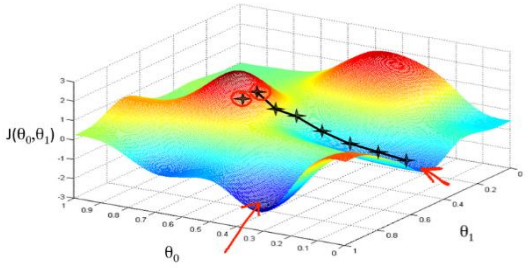
รูปที่ 2.20 สัญลักษณ์ PyTorch

2.7 การเพิ่มประสิทธิภาพของโมเดล

ตัวเพิ่มประสิทธิภาพ (Optimizers) เป็นฟังก์ชันทางคณิตศาสตร์เพื่อปรับเปลี่ยนน้ำหนักของเครือข่าย ถูกสร้างขึ้นจากแนวคิดของการลดระดับ Gradient ตัวเพิ่มประสิทธิภาพที่ดีกว่าจะทำงานเร็วขึ้นและมีประสิทธิภาพ

2.7.1 Gradient Descent

Gradient Descent คือ Optimization algorithm ที่ทำการปรับค่าพารามิเตอร์ในโมเดลเพื่อลดค่าของ Loss function ให้มีค่าน้อยที่สุด ซึ่ง Gradient Descent จะใช้สูตรในการคำนวณค่า Weight ใหม่ที่ดีขึ้น โดยเริ่มจากการสุ่มเดาค่า Weight ค่าแรกขึ้นมาแล้วทำการวัด Performance ของโมเดลด้วยค่า Error จาก Loss Function และจะนำค่า Error ที่ได้มาช่วยในการคำนวณค่า Weight ใหม่ที่ดีกว่าเดิม ทำวนซ้ำไปเรื่อยๆ จนกว่าจะได้ค่า Weight ที่ทำให้ค่า Error ของ Model ต่ำที่สุด [38]



รูปที่ 2.21 กระบวนการ Gradient Descent

2.7.2 Stochastic Gradient Descent (SGD)

SGD คือตัวแปรหนึ่งของอัลกอริทึม Gradient Descent ที่ใช้สำหรับการเพิ่มประสิทธิภาพโมเดลการเรียนรู้ของเครื่อง ใน SGD แทนที่จะใช้ชุดข้อมูลทั้งหมดสำหรับการวนซ้ำแต่ละครั้ง จะมีการเลือกตัวอย่างการฝึกแบบสุ่มเพียงตัวอย่างเดียว เพื่อคำนวณการไล่ระดับและอัปเดตพารามิเตอร์ของโมเดล เป็นการนำความสุ่มมาใช้ในกระบวนการเพิ่มประสิทธิภาพ [39]

2.7.3 Adaptive Moment Estimation (Adam)

Adam optimizer เป็น optimizer ที่สามารถปรับ learning rates สำหรับพารามิเตอร์ในแต่ละครั้งได้และยังสามารถแก้ปัญหา decaying ของ gradients เป็นเครื่องมือเพิ่มประสิทธิภาพที่ได้รับความนิยม ผสมผสานระหว่าง Stochastic Gradient Descent with momentum และ RMSprop ซึ่งทำให้โมเดลไม่หยุดเรียนรู้ อีกทั้งยังไวกว่า Gradient Descent และลดปัญหาการแกว่งของพารามิเตอร์ได้อีกด้วย [40] โดย Adam จะมีสมการแสดงดังสมการที่ (2.9) และมีเวกเตอร์ m และ v เพื่อแสดงถึงพารามิเตอร์ที่เก็บ First Order Gradient และกำลังสองของ First-Order Gradient ตามลำดับ แสดงดังสมการที่ (2.10) และ (2.11)

$$\theta_{t+1}=\theta_t-\frac{\eta}{\sqrt{v_t}+\epsilon}m_t \tag{2.9}$$

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)\frac{\partial L}{\partial w_t} \tag{2.10}$$

$$v_t=\beta_2 v_{t-1}+(1-\beta_2)\left(\frac{\partial L}{\partial w_t}\right)^2 \tag{2.11}$$

ในสมการที่ (2.9) และ (2.10) เวกเตอร์ m_t และ v_t จะมี Bias เล็กน้อย โดยที่ค่าของ β_1 และ β_2 จะมีค่าอยู่ในช่วง 0-1 เพื่อที่จะแก้ปัญหานี้ Adam จะมีสมการใหม่แสดงดังสมการที่ (2.12) และ (2.13) ตามลำดับ [41]

$$\widehat{m_t}=\frac{m_t}{1-\beta_1^t} \tag{2.12}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.13}$$

2.8 การวัดประสิทธิภาพของอัลกอริทึม

2.8.1 Confusion matrix

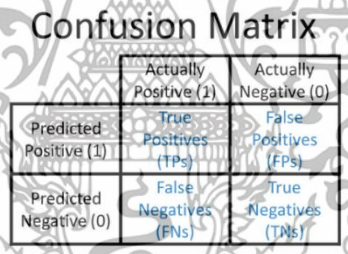
Confusion matrix คือตารางที่ใช้ประเมินประสิทธิภาพของโมเดลแบบ classification ด้วยการประเมินค่าจริงเทียบกับค่าที่ได้จากผลลัพธ์ของโมเดลที่ทำนายได้ แล้วสรุปค่าออกมาในรูปแบบของตารางเมทริกซ์แสดงดังรูปที่ 2.22 และตารางเมทริกซ์ Multiclass แสดงดังรูปที่ 2.23 มีองค์ประกอบดังนี้ [42]

1) True Positive (TP) สิ่งที่ทำนายว่าจริง และสิ่งที่เกิดขึ้นคือจริง

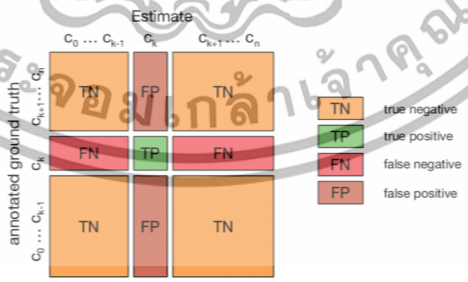
2) True Negative (TN) สิ่งที่ทำนายว่าไม่จริง และสิ่งที่เกิดขึ้นคือไม่จริง

3) False Positive (FP) สิ่งที่ทำนายว่าจริง แต่สิ่งที่เกิดขึ้นคือไม่จริง

4) False Negative (FN) สิ่งที่ทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้นคือจริง [43]



รูปที่ 2.22 Confusion Matrix



รูปที่ 2.23 Multiclass Confusion Matrix [44]

2.8.2 Precision

Precision คือค่าความแม่นยำในการทำนายในกลุ่มเป้าหมายหรือพิจารณาเฉพาะที่เป็น True Positives (TP) โดยสามารถคำนวณได้ดังสมการที่ (2.14)

$$Precision = \frac{TP}{(TP+FP)} \quad (2.14)$$

2.8.3 Recall

Recall คือความสามารถของโมเดลในการตรวจจับความถูกต้อง ซึ่งคือสัดส่วนของ True Positives (TP) กับข้อมูลที่จริงๆ เป็น Positive ทั้งหมด โดยสามารถคำนวณได้ดังสมการที่ (2.15)

$$Recall = \frac{TP}{(TP+FN)} \quad (2.15)$$

2.8.4 F1-score

F1-Score คือค่าเฉลี่ยแบบ harmonic mean ระหว่าง Precision และ Recall ซึ่งเป็น single metric ที่วัดความสามารถของโมเดล โดยสามารถคำนวณได้ดังสมการที่ (2.16)

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.16)$$

2.8.5 Accuracy

ค่า Accuracy คือค่าความแม่นยำเป็นสัดส่วนของข้อมูลทั้งหมดที่ถูกจำแนกถูกต้องโดยระบบ ซึ่งคำนวณได้ดังสมการที่ (2.17)

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \quad (2.17)$$

2.9 การแปลงข้อความเป็นเสียง (Text-to-Speech : TTS)

การเปลี่ยนข้อความดิจิทัลให้เป็นเสียงพูดโดยใช้ปัญญาประดิษฐ์และการเรียนรู้ของเครื่องในการแปลงข้อความให้เป็นเสียงสังเคราะห์ ผ่านกระบวนการประมวลผลด้วยโครงข่ายประสาทเทียมแบบลึก (Deep Neural Networks) ซึ่งเป็นโครงสร้างที่ประกอบด้วยโหนดการคำนวณที่เชื่อมโยงและทำงานร่วมกัน โครงข่ายประสาทเทียมเหล่านี้ได้รับการฝึกฝนด้วยข้อมูลเสียงในหลากหลายภาษา รวมถึงสำเนียง ระดับเสียง (Pitch) และระดับความดัง (Volume) เพื่อให้สามารถแปลงข้อความเป็นเสียงพูดที่มีความเป็นธรรมชาติและใกล้เคียงกับเสียงมนุษย์ [45][46]

2.9.1 หลักการทำงานของการแปลงข้อความเป็นเสียง

การทำงานของการแปลงข้อความเป็นเสียง แบ่งออกเป็น 4 ส่วน ดังนี้

1) การวิเคราะห์ข้อความ

เริ่มต้นด้วยการวิเคราะห์โครงสร้างและไวยากรณ์ของข้อความดิจิทัล เพื่อทำความเข้าใจความหมายและตรวจสอบการจัดเรียงประโยค คำต่างๆ และเครื่องหมายวรรคตอนจะถูกระบุเพื่อช่วยจัดระเบียบประโยคให้ถูกต้องและมีความเข้าใจง่าย

2) การประมวลผลทางภาษา (Linguistic Processing)

คำต่างๆ ในข้อความจะถูกแปลงเป็นโฟเนม (Phonemes) ซึ่งเป็นหน่วยเสียงเล็กที่สุดในภาษาที่ใช้ จากนั้นจะนำโฟเนมเหล่านี้ไปแมปกับคลื่นเสียงที่เหมาะสมโดยใช้ฐานข้อมูลเสียงและเทคโนโลยีการจำลองเสียง (Synthesis Technology) เพื่อให้เสียงที่สร้างขึ้นมีความถูกต้องและสามารถออกเสียงได้อย่างเป็นธรรมชาติ

3) การสร้างคลื่นเสียง (Vocoder)

เมื่อโฟเนมถูกแปลงและแมปไปยังคลื่นเสียง ระบบ Vocoder จะใช้ข้อมูลนี้เพื่อสร้างคลื่นเสียงพูด ระบบจะใช้เทคนิคการสังเคราะห์เสียงเพื่อสร้างคลื่นเสียงที่ตอบสนองต่อข้อความดิจิทัลด้วยเสียงพูดที่สามารถเข้าใจได้

4) การสังเคราะห์เสียงพูด (Synthetic Speech)

เป็นกระบวนการสร้างเสียงจากโฟเนม โดยจะใช้เทคนิคการเรียนรู้เชิงลึก (Deep Learning) มาเพิ่มคุณภาพและความเป็นธรรมชาติของเสียง ทำให้คอมพิวเตอร์สามารถส่งเสียงหรือพูดข้อความดิจิทัลได้อย่างมีประสิทธิภาพและเข้าใจง่าย [47]

2.9.2 ไลบรารีที่ใช้สำหรับการแปลงข้อความเป็นเสียง

ไลบรารี eSpeak คือ ซอฟต์แวร์สังเคราะห์เสียงพูดโอเพ่นซอร์ส แสดงดังรูปที่ 2.24 ที่มีขนาดกะทัดรัดสำหรับการนำเสนอภาษาอังกฤษและภาษาอื่นๆ สำหรับระบบปฏิบัติการทั้ง Linux และ Windows ใช้หลักการการสังเคราะห์ฟอร์แมนต์ (Formant Synthesis) ทำให้เสียงพูดที่ได้มีความชัดเจนและสามารถประมวลผลได้ด้วยความเร็วสูง [48]

รูปที่ 2.24 สัญลักษณ์ eSpeak [49]

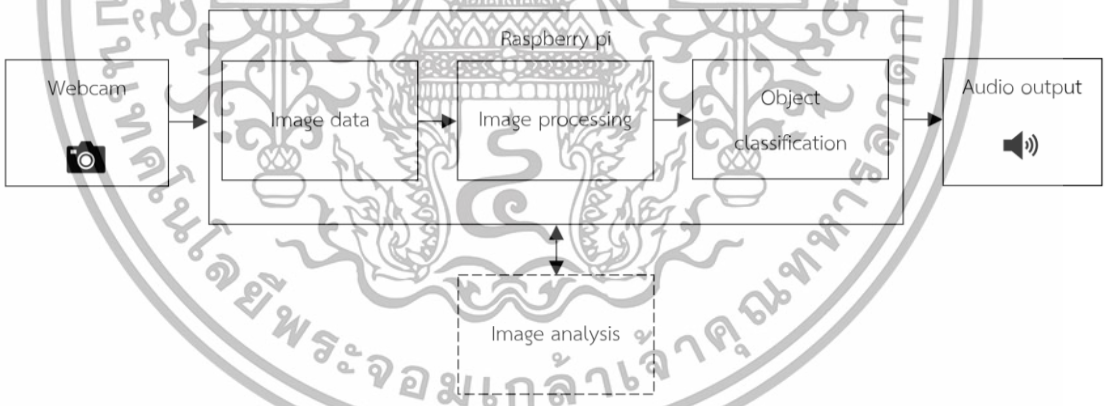
บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

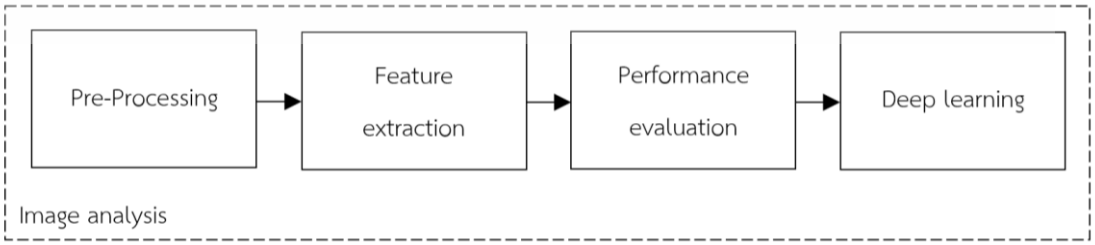
3.1 การออกแบบ

3.1.1 การออกแบบระบบ

ปริญญานิพนธ์นี้เป็นการศึกษาอัลกอริทึมการเรียนรู้เชิงลึกในรูปแบบต่างๆ ที่ใช้ในการประมวลผลรูปภาพเพื่อใช้งานในอุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตาในการใช้ชีวิตภายในอาคาร โดยมีบล็อกไดอะแกรมแสดงดังรูปที่ 3.1 โดยระบบจะเริ่มจากการบันทึกข้อมูลภาพจากกล้อง Webcam มาประมวลผลโดยใช้ Raspberry pi ในการควบคุมการทำงานของระบบ มีกระบวนการย่อยแสดงในรูปที่ 3.2 โดยมีการเตรียมข้อมูลภาพได้แก่ บันได ประตู ลิฟต์ และสิ่งกีดขวาง ทำการดึงคุณลักษณะของข้อมูลและนำไปวิเคราะห์โดยใช้อัลกอริทึมการเรียนรู้เชิงลึกเพื่อจำแนกข้อมูล ทำการฝึกสอนข้อมูล และมีการปรับค่าพารามิเตอร์ของอัลกอริทึมการเรียนรู้เชิงลึกเพื่อเปรียบเทียบความแม่นยำของอัลกอริทึม เมื่อได้โมเดลที่มีประสิทธิภาพพร้อมใช้งานแล้วจะนำไปใช้ร่วมกับอุปกรณ์สำหรับช่วยเหลือผู้มีความบกพร่องทางสายตาเพื่อจำแนกวัตถุและส่งเสียงเตือน



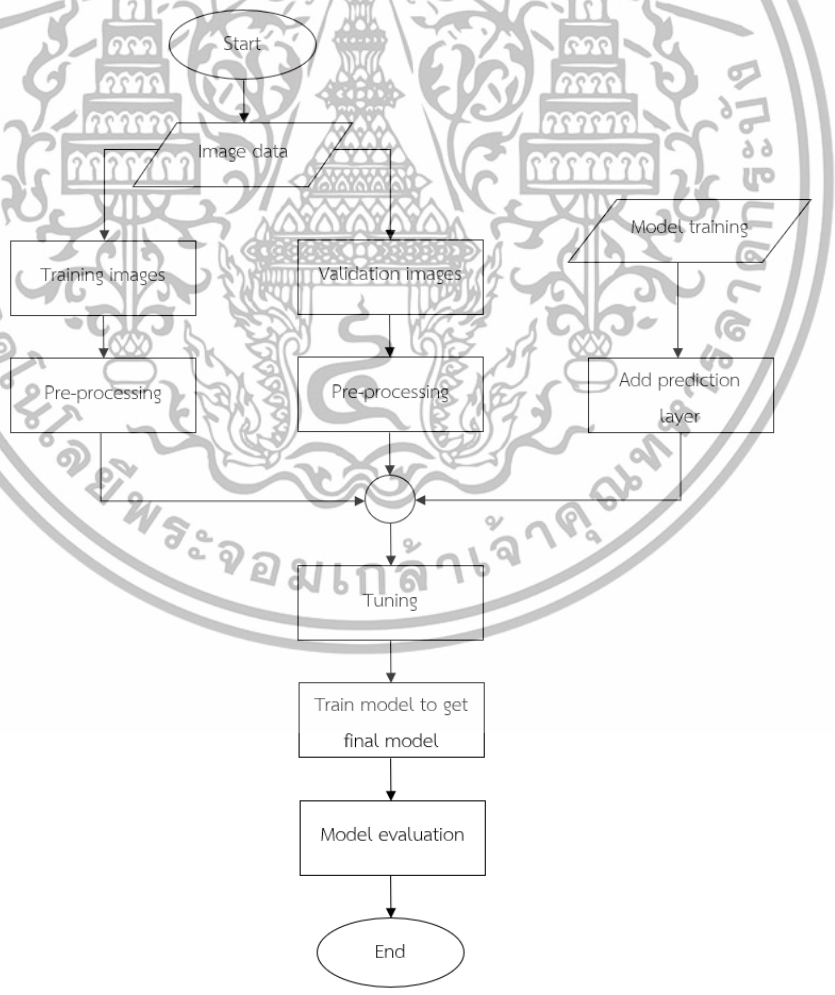
รูปที่ 3.1 บล็อกไดอะแกรมแสดงภาพรวมของปริญญานิพนธ์



รูปที่ 3.2 บล็อกไดอะแกรมส่วนวิเคราะห์รูปภาพ

3.1.2 การออกแบบโมเดลฝึกสอน

การออกแบบโมเดลฝึกสอน มีข้อมูลอินพุตเป็นรูปภาพ ทำการแบ่งออกเป็น 3 ชุดโดยการสุ่มข้อมูล ได้แก่ ชุดข้อมูลที่ใช้สำหรับฝึกสอน (Training set), ชุดข้อมูลที่ใช้ทดสอบความถูกต้องขณะฝึกสอน (Validation set) และชุดข้อมูลที่ใช้สำหรับทดสอบ (Testing set) โดยมีสัดส่วนเป็น 70%, 20% และ 10% ตามลำดับ ข้อมูลอินพุตจะถูกปรับขนาดให้เหมาะสมกับโมเดล ในปริญญานิพนธ์นี้เลือกใช้สถาปัตยกรรม CNN และ ResNet18 ซึ่งมีการเพิ่ม Prediction layer ทำให้โมเดลสามารถจำแนกข้อมูลแบบ Multiclass ได้ โดยมีการกำหนด Neuron ให้เท่ากับจำนวนคลาสที่ใช้ ได้แก่ 6 คลาส จากนั้นทำการหาพารามิเตอร์และจำนวนเลเยอร์ที่เหมาะสมกับโมเดลที่เลือกใช้ให้มีความแม่นยำสูงสุด ซึ่งพารามิเตอร์และจำนวนเลเยอร์ที่ดีที่สุดในแต่ละโมเดลจะทำการเทรนโมเดลรอบสุดท้ายเพื่อให้ได้โมเดลที่พร้อมใช้งาน และนำมาประเมินประสิทธิภาพหรือความแม่นยำของโมเดล โดยแผนผังแสดงการทำงานของโมเดลฝึกสอนแสดงดังรูปที่ 3.3

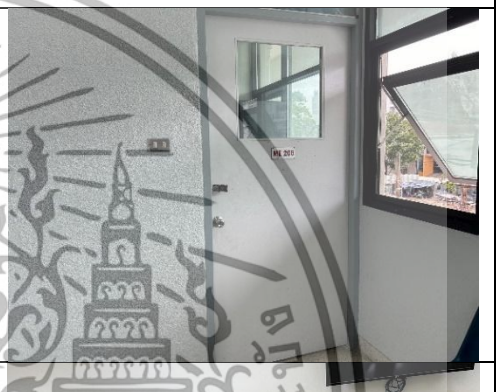
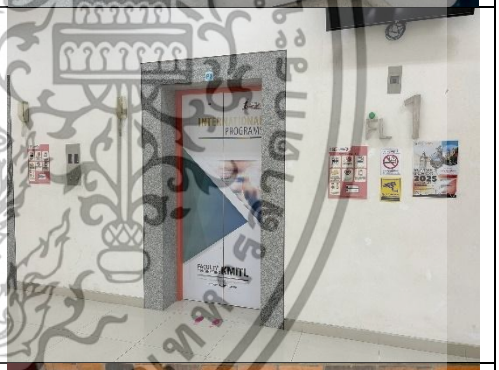


รูปที่ 3.3 แผนผังการออกแบบโมเดลฝึกสอน

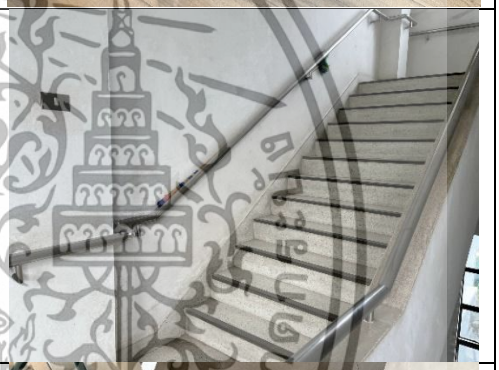
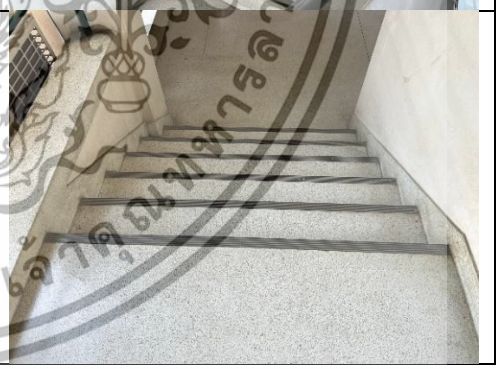
3.1.2.1 การเตรียมข้อมูล (Pre-processing)

ผู้จัดทำได้มีการจัดเตรียมรูปภาพเป็นจำนวน 6 คลาส ได้แก่ ประตู ลิฟต์ โต๊ะ เก้าอี้ บันไดขึ้น และบันไดลง โดยตัวอย่างภาพในชุดข้อมูลของแต่ละคลาสแสดงดังตารางที่ 3.1

ตารางที่ 3.1 ตัวอย่างภาพในแต่ละคลาส

Datasets	Training set 70% (Images)	Validation set 20% (Images)	Test set 10% (Images)	Sample picture
ประตู	254	72	38	
ลิฟต์	249	71	36	
โต๊ะ	245	70	36	

ตารางที่ 3.1 ตัวอย่างภาพในแต่ละคลาส (ต่อ)

Datasets	Training set 70% (Images)	Validation set 20% (Images)	Test set 10% (Images)	Sample picture
เก้าอี้	259	74	38	
บันไดขึ้น	251	72	37	
บันไดลง	247	70	37	

1) การขยายข้อมูล (Augmentation)

เป็นการขยายส่วนของข้อมูลอินพุต ตัวอย่างข้อมูลอินพุตแสดงดังรูปที่ 3.4 เพื่อให้ข้อมูลมีความหลากหลาย เพื่อเพิ่มความสามารถของโมเดลในการเรียนรู้จากข้อมูลที่มีความหลากหลายมากขึ้น ซึ่งช่วยลดปัญหา Overfitting ในการฝึกโมเดลด้วยข้อมูลขนาดเล็กได้ ด้วยการใช้หลักการ Image processing โดยมีการสุ่มทำทั้งหมด 4 รูปแบบ ได้แก่ การพลิกภาพในแนวนอนแสดงดังรูปที่ 3.5 การหมุนภาพภายในช่วง -10 ถึง +10 องศา แสดงดังรูปที่ 3.6 การครอปภาพที่มีขนาดแตกต่างกันระหว่าง 80% ถึง 100% แสดงดังรูปที่ 3.7 การปรับค่าของสี แสดงดังรูปที่ 3.8 และปรับ

ค่าพิกเซลของภาพ (Normalize) เพื่อให้ค่าพิกเซลอยู่ในช่วงมาตรฐานก่อนที่จะนำภาพเข้าโมเดล โดยใช้ค่ามาตรฐานของชุดข้อมูล ImageNet ซึ่งเป็นชุดข้อมูลของ Pre-trained model

รูปที่ 3.4 ตัวอย่างข้อมูลอินพุต

รูปที่ 3.5 ตัวอย่างการพลิกภาพในแนวนอน

รูปที่ 3.6 ตัวอย่างการหมุนภาพ

รูปที่ 3.7 ตัวอย่างการครอปภาพ

รูปที่ 3.8 ตัวอย่างการปรับค่าของสี

2) การสกัดคุณลักษณะ (Feature extraction)

ในการสกัดคุณลักษณะจากข้อมูลอินพุตที่เป็นรูปภาพจะถูกแปลงเป็นรูปแบบของคุณลักษณะที่สามารถนำไปใช้ในการเรียนรู้อัลกอริทึมเชิงลึกได้ โดยการใช้โครงข่ายประสาทเทียมเพื่อให้ได้คุณลักษณะซึ่งนำไปสู่การทำนายและการจำแนกประเภทที่แม่นยำมากขึ้น จากสถาปัตยกรรมที่ผู้จัดทำเลือกใช้ ในสถาปัตยกรรม CNN มี Convolution Layer ทำหน้าที่สกัดคุณลักษณะจากรูปภาพ โดยมีการลดขนาดผ่าน MaxPooling และเชื่อมต่อกับ Fully Connected Layer เพื่อทำการทำนาย ตัวอย่างพารามิเตอร์ของโครงสร้าง CNN แสดงดังรูปที่ 3.9 และในสถาปัตยกรรม ResNet18 ที่มี Convolution Layer ทำหน้าที่สกัดคุณลักษณะจากรูปภาพเช่นเดียวกัน และโครงสร้างมีการเชื่อมต่อแบบ Residual Connection ที่ช่วยให้โมเดลสามารถส่งข้อมูลข้ามชั้นหรือข้ามผ่านบาง layer ได้ ตัวอย่างพารามิเตอร์ 1 block ใน ResNet18 แสดงดังรูปที่ 3.10

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 224, 224]	448
MaxPool2d-2	[-1, 16, 112, 112]	0
Conv2d-3	[-1, 32, 112, 112]	4,640
MaxPool2d-4	[-1, 32, 56, 56]	0
Linear-5	[-1, 128]	12,845,184
Dropout-6	[-1, 128]	0
Linear-7	[-1, 5]	645

รูปที่ 3.9 ตัวอย่างพารามิเตอร์ 1 block ใน CNN

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 112, 112]	9,408
BatchNorm2d-2	[-1, 64, 112, 112]	128
ReLU-3	[-1, 64, 112, 112]	0
MaxPool2d-4	[-1, 64, 56, 56]	0
Conv2d-5	[-1, 64, 56, 56]	36,864
BatchNorm2d-6	[-1, 64, 56, 56]	128
ReLU-7	[-1, 64, 56, 56]	0
Conv2d-8	[-1, 64, 56, 56]	36,864
BatchNorm2d-9	[-1, 64, 56, 56]	128
ReLU-10	[-1, 64, 56, 56]	0
BasicBlock-11	[-1, 64, 56, 56]	0

รูปที่ 3.10 ตัวอย่างพารามิเตอร์ 1 block ใน ResNet18

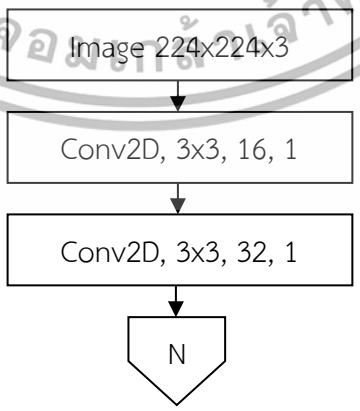
3.1.2.2 การเขียนโปรแกรมการเรียนรู้เชิงลึก

การเขียนโปรแกรมสร้างโครงสร้างทั้ง 2 สถาปัตยกรรม ในโครงสร้าง CNN จะประกอบด้วย Convolution 2 มิติหลายชั้นเพื่อเพิ่มประสิทธิภาพการเรียนรู้ของโมเดล จากนั้นทำการเชื่อมต่อข้อมูลไปยัง Flatten Layer เพื่อแปลงข้อมูลให้เป็นเวกเตอร์ 1 มิติ ก่อนเชื่อมต่อไปยัง Fully Connected Layer (Dense Layer) และในส่วนโครงสร้าง ResNet18 จะ ทำการดาวน์โหลดโมเดล ResNet18 พร้อมพารามิเตอร์ที่ถูกฝึกมาก่อนแล้ว (pretrained) บนชุดข้อมูล ImageNet ผ่านไลบรารี PyTorch บนภาษา Python และทำการปรับโครงสร้างใน Fully connected layer เดิมของ ResNet18 ด้วยโครงสร้างใหม่ที่ออกแบบมาให้เหมาะสมกับการจำแนกข้อมูลใน 6 คลาส ด้วย Flatten layer, Dense layer และ Prediction layer ซึ่งใน Prediction layer ในทั้ง 2 โครงสร้างมี 6 หน่วยประมวลผลสำหรับจำแนกข้อมูล 6 คลาส สุดท้ายการทำนาย output ใช้ฟังก์ชัน CrossEntropyLoss จะทำการแปลงค่าโลจิต เป็นค่าความน่าจะเป็นด้วย SoftMax ภายในทำให้สามารถระบุ Output ได้

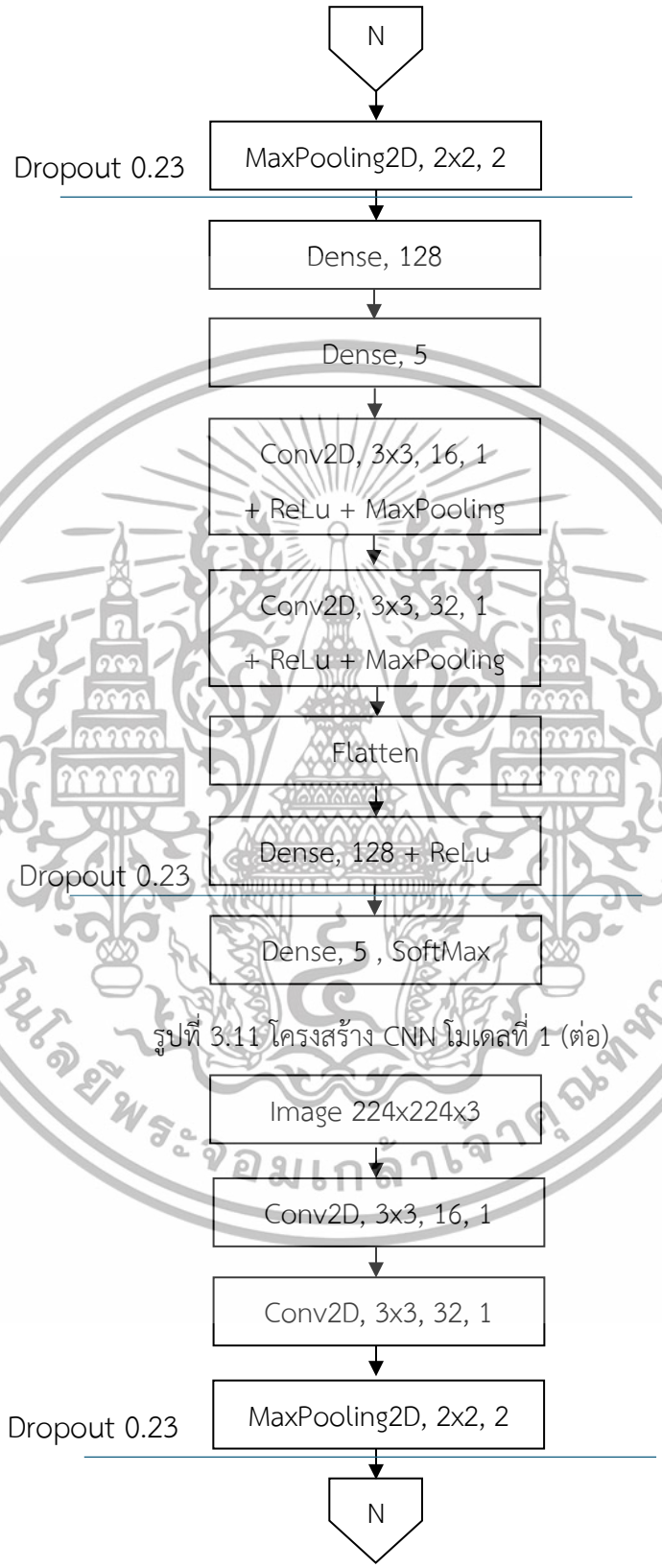
3.1.2.3 การเขียนโปรแกรมการปรับแต่งโมเดล

1) โครงสร้าง CNN

เลือกใช้ Bayesian Optimizer ผ่านไลบารี Optuna ในการหาค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมสำหรับการทดสอบการฝึกสอนโมเดล CNN ด้วยการสร้างโครงสร้างเลเยอร์ที่แตกต่างกัน โดยค่าไฮเปอร์พารามิเตอร์สำหรับการฝึกได้แก่ Learning rate = 0.0001, Batch size = 4, Dropout rate = 0.23, Epochs = 100, Patience = 5 และ Optimizer คือ Adam เพื่อให้สามารถวัดผลความแตกต่างที่เกิดจากการปรับโครงสร้างเลเยอร์เพียงอย่างเดียว โดยที่การกำหนดแต่ละเลเยอร์แสดงดังรูปที่ 3.11 - 3.14 ดังนี้

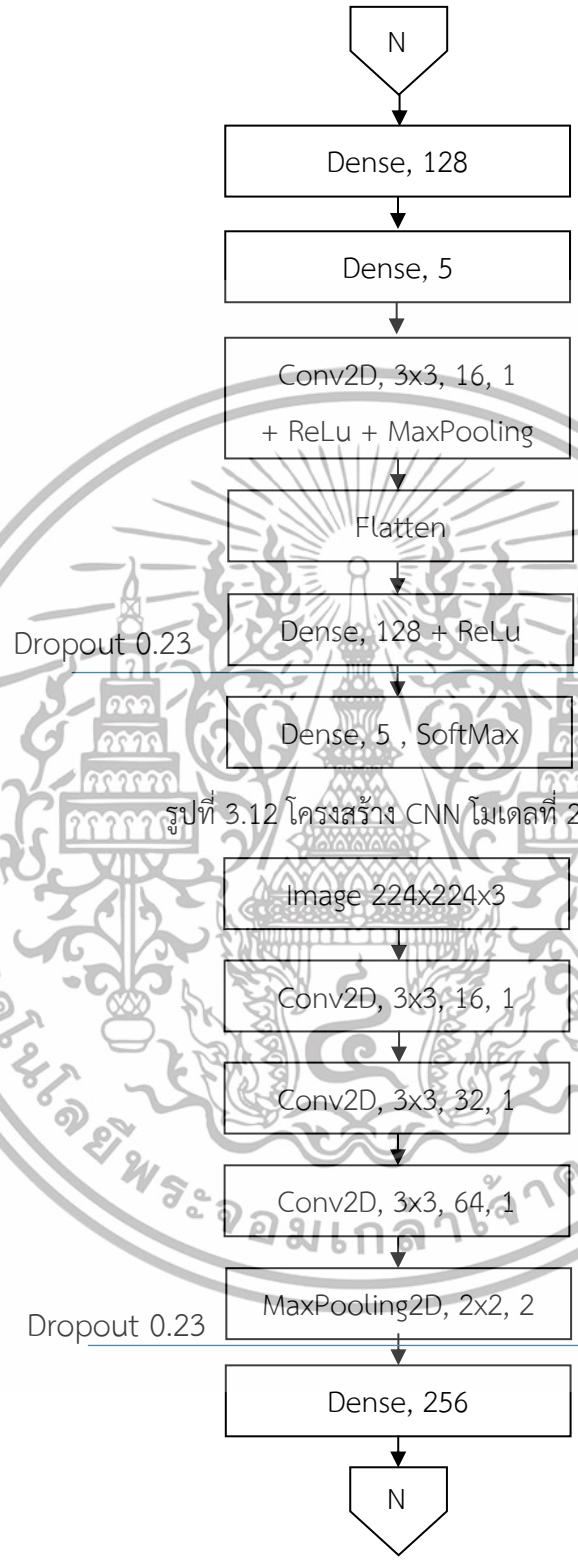


รูปที่ 3.11 โครงสร้าง CNN โมเดลที่ 1



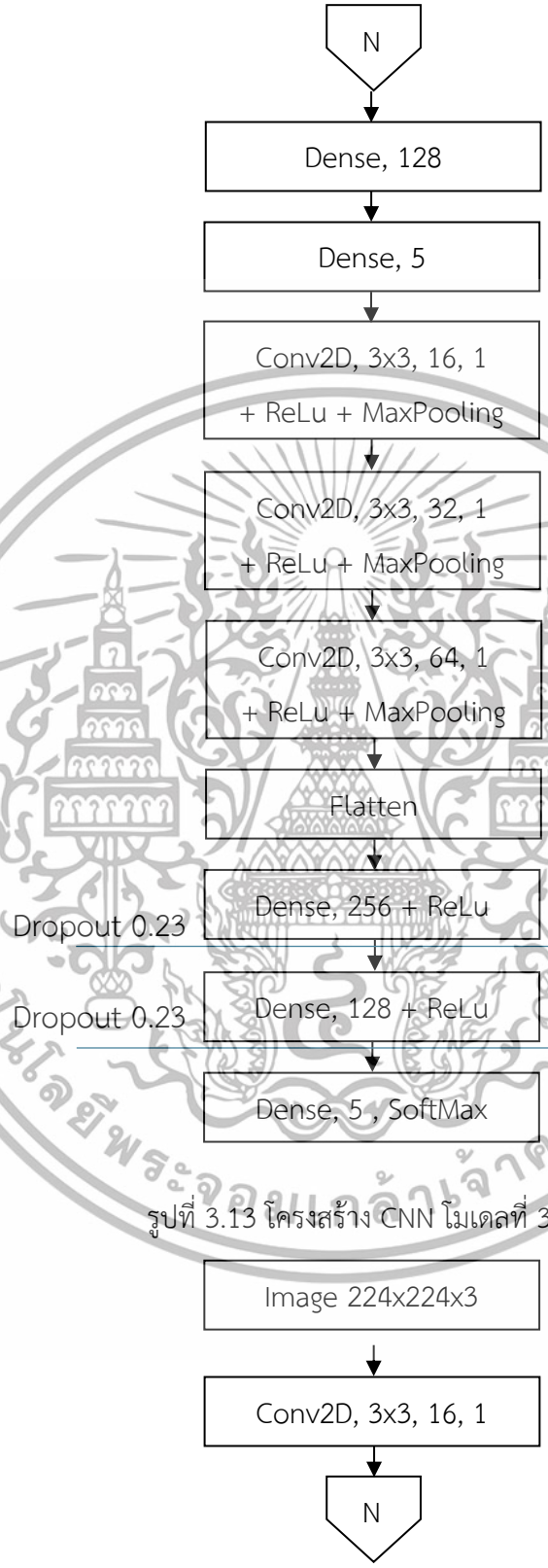
รูปที่ 3.11 โครงสร้าง CNN โมเดลที่ 1 (ต่อ)

รูปที่ 3.11 โครงสร้าง CNN โมเดลที่ 2



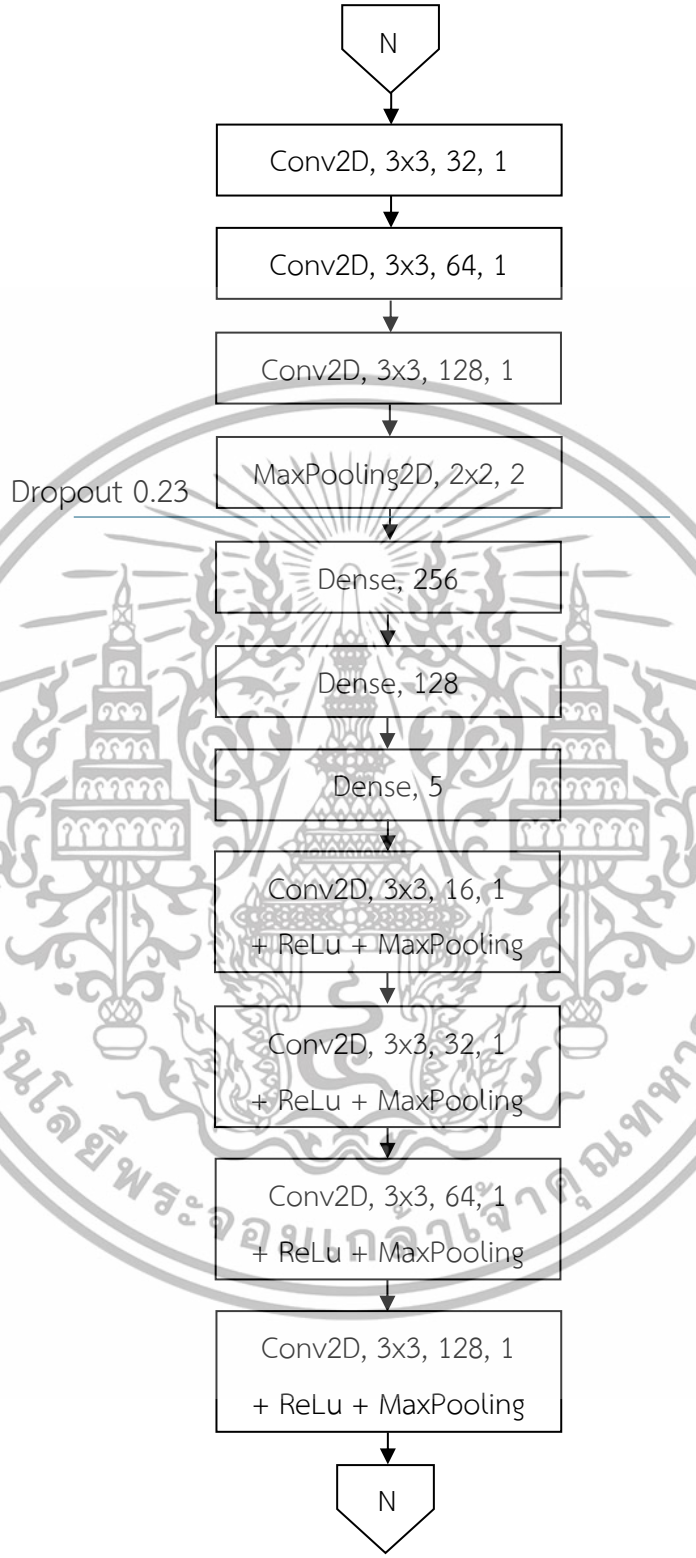
รูปที่ 3.12 โครงสร้าง CNN โมเดลที่ 2 (ต่อ)

รูปที่ 3.13 โครงสร้าง CNN โมเดลที่ 3

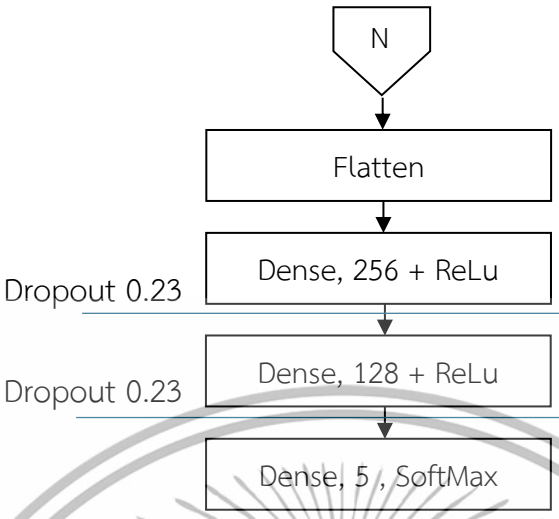


รูปที่ 3.13 โครงสร้าง CNN โมเดลที่ 3 (ต่อ)

รูปที่ 3.14 โครงสร้าง CNN โมเดลที่ 4



รูปที่ 3.14 โครงสร้าง CNN โมเดลที่ 4 (ต่อ)



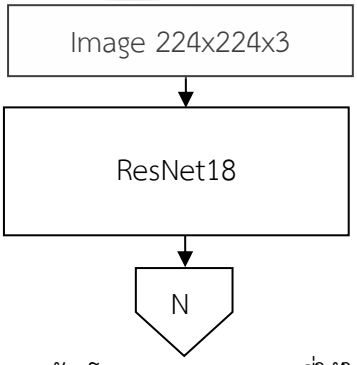
รูปที่ 3.14 โครงสร้าง CNN โมเดลที่ 4 (ต่อ)

2) โครงสร้าง ResNet18

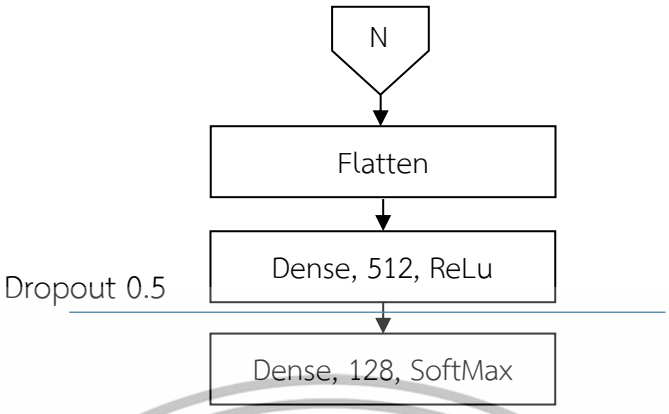
เลือกใช้วิธี Grid search ในการหาค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมสำหรับโมเดล ResNet18 โดยผู้จัดทำได้เลือกใช้ Optimizer ได้แก่ Adam โดยกำหนดค่าอัตราการเรียนรู้ (Learning rate) เป็นค่า Default ได้แก่ 0.001 และกำหนดค่า Epoch ที่แตกต่างกัน 3 ค่า ได้แก่ 15, 20 และ 30 และสำหรับค่า Batch size ที่ใช้ในการฝึกสอนของแต่ละการวนซ้ำเลือกใช้ 2 ค่า ได้แก่ 16 และ 32 โดยมีการออกแบบโครงสร้าง ResNet18 แสดงดังรูปที่ 3.15

การเปลี่ยนพารามิเตอร์ในการฝึกสอนแบ่งออกเป็น 6 กรณี ได้แก่

- กรณีที่ 1 : Optimizer = Adam, Epoch = 15, Batch size = 16
- กรณีที่ 2 : Optimizer = Adam, Epoch = 20, Batch size = 16
- กรณีที่ 3 : Optimizer = Adam, Epoch = 30, Batch size = 16
- กรณีที่ 4 : Optimizer = Adam, Epoch = 15, Batch size = 32
- กรณีที่ 5 : Optimizer = Adam, Epoch = 20, Batch size = 32
- กรณีที่ 6 : Optimizer = Adam, Epoch = 30, Batch size = 32



รูปที่ 3.15 โครงสร้างโมเดล ResNet18 ที่ใช้ในการทดลอง



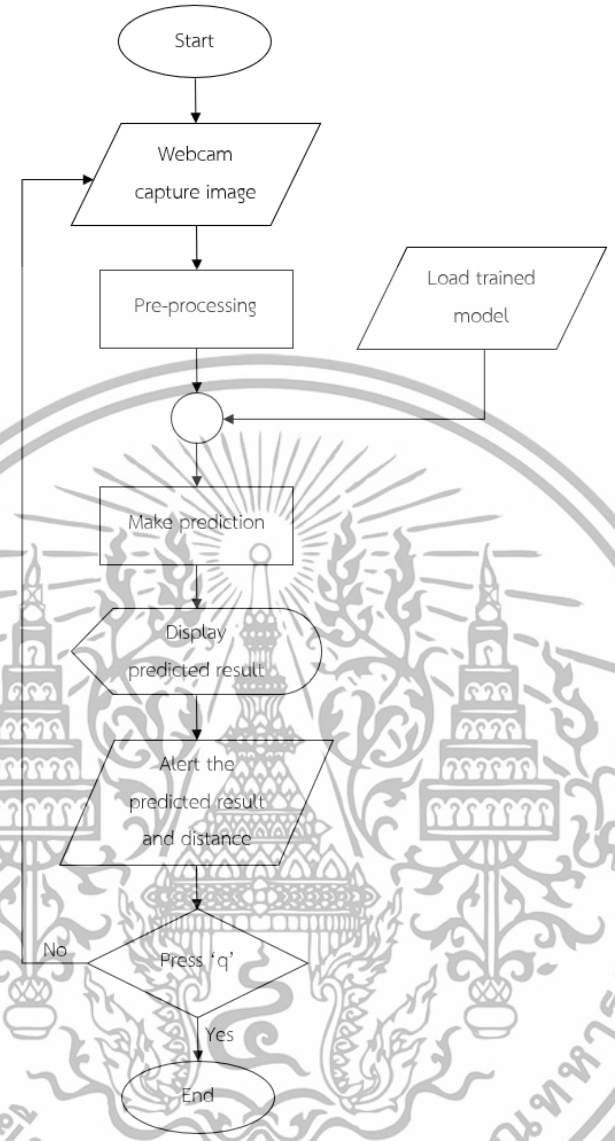
รูปที่ 3.15 โครงสร้างโมเดล ResNet18 ที่ใช้ในการทดลอง (ต่อ)

3.1.2.4 การเขียนโปรแกรมประเมินผล

เมื่อได้ค่าพารามิเตอร์ที่เหมาะสมที่สุดสำหรับแต่ละโมเดล ผู้จัดทำได้ทำการนำค่าเหล่านี้ไปฝึกกับโมเดลเพื่อปรับปรุงประสิทธิภาพของการจำแนกรูปภาพ จากนั้นโมเดลจะถูกทดสอบด้วยชุดข้อมูลทดสอบเพื่อประเมินความแม่นยำในการจำแนกรูปภาพที่ไม่เคยเห็นมาก่อน และในการประเมินความแม่นยำของระบบจะประเมินผ่าน Classification Report ซึ่งจะแสดงผลการจำแนกที่รวมถึง Precision, Recall, F1-score และ Support สำหรับแต่ละคลาสและทั้งระบบ พร้อมทั้งแสดง Confusion Matrix

3.1.3 การเขียนโปรแกรมสำหรับจำแนกวัตถุ

จากการทำการประเมินผลของโมเดลสถาปัตยกรรม CNN และ ResNet18 ผู้จัดทำจะเลือกใช้โมเดลที่มีประสิทธิภาพสูงสุดเพื่อนำมาใช้กับอุปกรณ์สำหรับช่วยเหลือผู้บกพร่องทางสายตา โดยมีแผนผังการทำงานของอุปกรณ์แสดงดังรูปที่ 3.16 เริ่มต้นจากกล้อง Webcam จับภาพอินพุต ภาพจะถูกนำมาเข้ากระบวนการ Pre-processing โดยทำการปรับขนาดเป็น 224x224 พิกเซล ทำการ Normalize และทำการปรับแสง จากนั้นใช้โมเดลที่ผ่านฝึกสอนทำการประมวลผลภาพเพื่อทำนายผล แสดงผลการทำนายบนหน้าจอคอมพิวเตอร์ และทำการแจ้งเตือนในรูปแบบเสียงถึงวัตถุพร้อมกับระยะห่างของวัตถุ

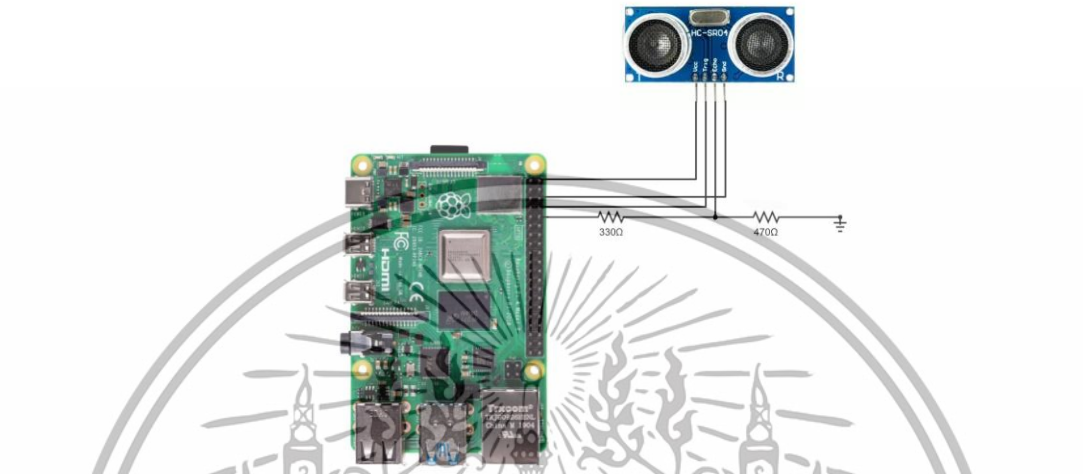


รูปที่ 3.16 แผนผังแสดงการทำงานของอุปกรณ์

3.1.4 การออกแบบวงจรสำหรับการคำนวณระยะ

ผลลัพธ์ของการทำงานของอุปกรณ์สำหรับช่วยเหลือผู้บกพร่องทางสายตานอกจากจะส่งเสียงแจ้งเตือนถึงวัตถุที่เป็นอุปสรรคต่อการเคลื่อนไหวตรงหน้าที่โมเดลสามารถทำนายได้ ยังมีการแจ้งเตือนถึงระยะห่างระหว่างอุปกรณ์กับวัตถุ โดยผู้จัดทำได้เลือกใช้ Ultrasonic sensor ในการคำนวณระยะห่างที่มีหลักการทำงานคือจะส่งพัลส์อัลตราซาวนด์ซึ่งมนุษย์ไม่ได้ยิน และตรวจจับเสียงสะท้อนที่ส่งกลับมาเมื่อเสียงสะท้อนออกจากวัตถุที่อยู่ใกล้เคียง จากนั้นจะใช้ความเร็วของเสียงเพื่อคำนวณระยะห่างจากวัตถุ

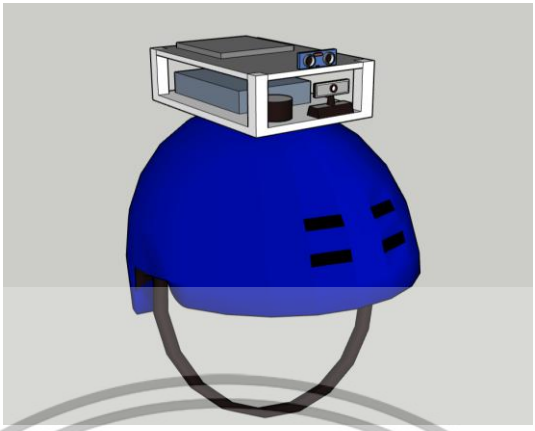
การออกแบบใช้การเชื่อมต่อกับพิน GPIO ของ Raspberry pi สำหรับพิน Echo ใช้ตัวต้านทาน 330 โอห์ม และ 470 โอห์ม เพื่อทำการแบ่งศักย์ไฟฟ้า การเชื่อมต่อ Ultrasonic sensor กับ Raspberry pi แสดงดังรูปที่ 3.17



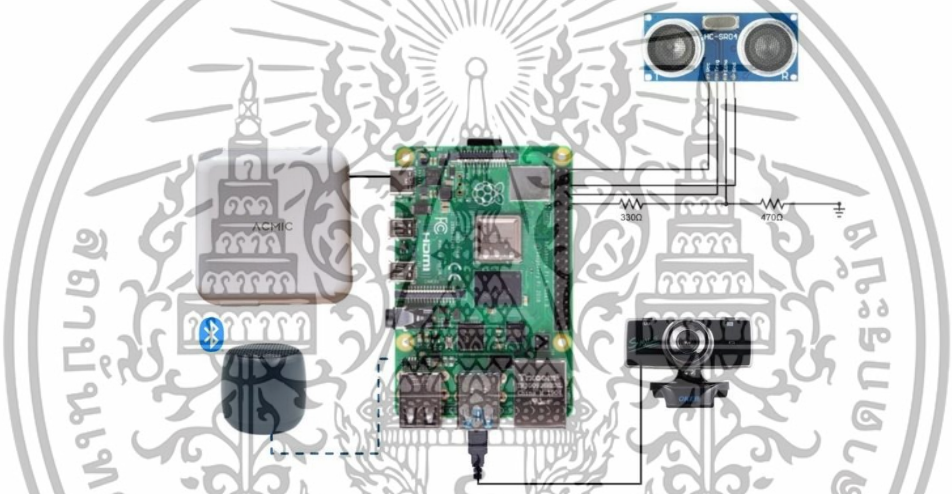
รูปที่ 3.17 การเชื่อมต่อ Ultrasonic sensor กับ Raspberry pi

3.1.5 การออกแบบอุปกรณ์

อุปกรณ์สำหรับผู้ที่มีความบกพร่องทางสายตานี้ได้รับการออกแบบให้ติดตั้งบนหมวกกันน็อก โดยใช้โครงสร้างอะคริลิกสองชั้นเป็นฐานสำหรับติดตั้งอุปกรณ์ต่าง ๆ แสดงดังรูปที่ 3.18 ซึ่งชั้นล่างประกอบด้วยกล้อง Webcam ติดตั้งบริเวณด้านหน้าซ้ายเพื่อช่วยตรวจจับวัตถุ ลำโพงติดตั้งบริเวณด้านหน้าด้านขวาเพื่อให้สามารถได้ยินเสียงแจ้งเตือนได้อย่างชัดเจน และบริเวณด้านหลังของชั้นล่างติดตั้งพาวเวอร์แบงก์ทำหน้าที่เป็นแหล่งพลังงานของอุปกรณ์ทั้งหมด ส่วนชั้นบนประกอบด้วย Ultrasonic Sensor ติดตั้งบริเวณด้านหน้าเพื่อใช้ตรวจจับระยะของสิ่งกีดขวาง และด้านหลังติดตั้ง Raspberry Pi ซึ่งออกแบบให้สามารถเชื่อมต่อกับพาวเวอร์แบงก์ได้สะดวก การเชื่อมต่อของเครื่องมือทั้งหมดแสดงดังรูปที่ 3.19 และอุปกรณ์ที่ประกอบเสร็จสมบูรณ์แสดงในรูปที่ 3.20



รูปที่ 3.18 การออกแบบอุปกรณ์สำหรับผู้มีความบกพร่องทางสายตา



รูปที่ 3.19 การเชื่อมต่อของเครื่องมือทั้งหมด

รูปที่ 3.20 อุปกรณ์จริง

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 Raspberry pi 4 model B

Raspberry Pi 4 Model B เป็นไมโครคอนโทรลเลอร์ที่ใช้ในการประมวลผลและควบคุมการทำงานของอุปกรณ์ต่างๆ มีหน่วยประมวลผลที่มีประสิทธิภาพและสามารถเชื่อมต่อกับอุปกรณ์เสริมได้หลากหลาย ได้แก่ ลำโพง กล้อง Webcam และพาวเวอร์แบงค์สำหรับการจ่ายพลังงาน

3.2.2 กล้อง Webcam

กล้อง Webcam ถูกใช้ในการเก็บรวบรวมข้อมูลรูปภาพและวิดีโอที่เกี่ยวข้องกับการทดลอง โดยมีความละเอียดเพียงพอสำหรับการประมวลผลและการวิเคราะห์ข้อมูลต่อไป

3.2.3 Visual Studio Code

ใช้สำหรับการเขียนโปรแกรมโดยใช้ภาษา Python สำหรับการนำเข้ารูปภาพที่ต้องการนำมาฝึก แยกคุณลักษณะที่สำคัญออกจากรูปภาพ และจำแนกวัตถุทั้ง 5 คลาส โดยใช้อัลกอริทึมการเรียนรู้เชิงลึก

3.2.4 Thonny

เป็นซอฟต์แวร์ที่ใช้สำหรับพัฒนาโปรแกรมภาษา Python ซึ่ง Thonny ถูกใช้สำหรับการพัฒนาและทดสอบโค้ดบน Raspberry Pi เนื่องจากมีอินเทอร์เฟซที่ใช้งานง่าย ช่วยให้การพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์เป็นไปอย่างสะดวกและมีประสิทธิภาพ โดยเฉพาะในขั้นตอนการตรวจสอบและทดสอบโปรแกรมที่เกี่ยวข้องกับการประมวลผลข้อมูลจากกล้อง

3.2.5 Ultrasonic HC-SR04

โมดูลตรวจวัดระยะทางด้วยคลื่นอัลตราโซนิกถูกใช้เพื่อวัดระยะห่างระหว่างอุปกรณ์ช่วยเหลือผู้มีความบกพร่องทางสายกับวัตถุที่เป็นอุปสรรคต่อการเคลื่อนไหว

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบการเตรียมข้อมูลแต่ละโมเดล

3.3.1.1 ทำการทดสอบการฝึกโมเดล CNN ด้วยโมเดล Default โดยการเปรียบค่าความแม่นยำระหว่างการเขียนโปรแกรมให้มีการขยายข้อมูล (With Augmentation) และไม่มีการขยายข้อมูล (Without Augmentation)

3.3.1.2 ทำการทดสอบการฝึกโมเดล ResNet18 ด้วยค่าไฮเปอร์พารามิเตอร์ที่เป็นค่า Default โดยการเปรียบค่าความแม่นยำระหว่างการเขียนโปรแกรมให้มีการขยายข้อมูล (With Augmentation) และไม่มีการขยายข้อมูล (Without Augmentation)

3.3.2 การทดสอบการดึงคุณลักษณะของการเรียนรู้เชิงลึกในแต่ละโมเดล

3.3.2.1 ทดสอบการดึงคุณลักษณะด้วยโมเดล CNN

3.3.2.2 ทดสอบการดึงคุณลักษณะด้วยโมเดล ResNet18

3.3.3 การทดสอบการปรับแต่งโมเดล

3.3.3.1 การทดสอบของโมเดล CNN จะนำไปทดสอบในรูปแบบโครงสร้างเลเยอร์ที่แตกต่างกัน 4 โครงสร้าง เพื่อเปรียบเทียบประสิทธิภาพของแต่ละโครงสร้างในการเรียนรู้และจำแนกรูปภาพ

3.3.3.2 การทดสอบของโมเดล ResNet18 จะนำไปทดสอบในรูปแบบของการปรับเปลี่ยนพารามิเตอร์ในการการฝึกสอนซึ่งแบ่งออกเป็น 6 กรณี เพื่อหาค่าไฮเปอร์พารามิเตอร์ที่มีผลต่อประสิทธิภาพของโมเดล ResNet18

3.3.4 การทดสอบการประเมินประสิทธิภาพอัลกอริทึมการเรียนรู้เชิงลึก

ทดสอบการประเมินประสิทธิภาพด้วยชุดข้อมูลทดสอบของโมเดล CNN ที่มีโครงสร้างที่เหมาะสมรวมถึงโมเดล ResNet18 ที่มีการตั้งค่าไฮเปอร์พารามิเตอร์อย่างเหมาะสม

3.3.5 การทดสอบการทำงานของอุปกรณ์

3.3.5.1 เลือกโมเดลที่มีประสิทธิภาพสูงสุดจากการประเมินประสิทธิภาพระหว่างโมเดล CNN และ ResNet18 เพื่อนำมาทดสอบการทำงานกับอุปกรณ์ช่วยเหลือผู้มีความบกพร่องทางสายตาเพื่อประเมินความสามารถในการใช้งานจริง

3.3.5.2 ทดสอบการแจ้งเตือนของอุปกรณ์

บทที่ 4

ผลการทดลอง

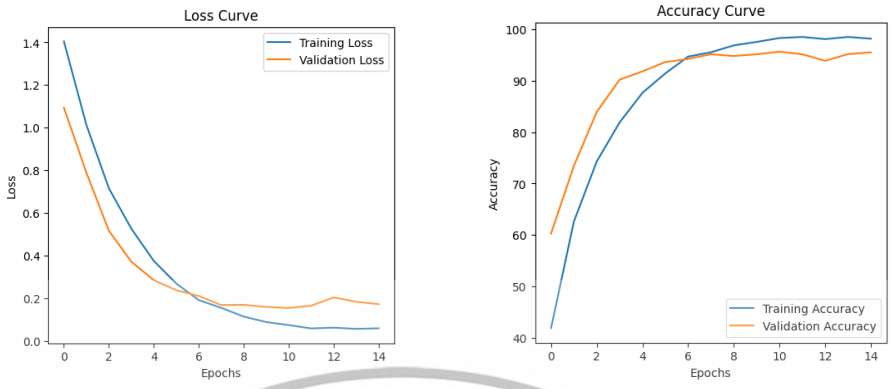
ผู้จัดทำได้ทำการเก็บผลการทำงานของระบบ โดยแบ่งการทดลองและจัดเก็บผลการทดลองเป็นส่วนๆ ดังต่อไปนี้

4.1 ผลการทดสอบการเตรียมข้อมูลแต่ละโมเดล

หลังจากที่ผู้จัดทำจัดเตรียมชุดข้อมูลทั้ง 6 คลาส และแบ่งออกเป็น 3 ชุดโดยการสุ่มข้อมูล ได้แก่ ชุดข้อมูลที่ใช้สำหรับฝึกสอน (Training set), ชุดข้อมูลที่ใช้ทดสอบความถูกต้องขณะฝึกสอน (Validation set) และชุดข้อมูลที่ใช้สำหรับทดสอบ (Testing set) โดยมีสัดส่วนเป็น 70%, 20% และ 10% ตามลำดับ ผู้จัดทำใช้เทคนิค Early stop ในทุกๆ การเทรนโมเดลเพื่อลดการเกิด Overfitting และปรับปรุงประสิทธิภาพการฝึกโมเดลโดยการหยุด Training เมื่อโมเดลเริ่มประสิทธิภาพไม่ดีขึ้นในชุดข้อมูล Validation ในการเทรนโมเดลมีการใช้หลักการ Image processing ในการเตรียมข้อมูล (Pre-processing) ช่วยให้ข้อมูลอยู่ในรูปแบบที่เหมาะสมสำหรับโมเดลและในส่วนของการขยายข้อมูล (Augmentation) เพื่อสร้างความหลากหลายให้กับข้อมูล หลักการ Image processing มีความสำคัญในการเตรียมข้อมูล จึงทำการทดสอบระหว่างการเทรนโมเดลด้วยชุดข้อมูลที่มีการขยายข้อมูลและไม่มีการขยายข้อมูลเพื่อเปรียบเทียบประสิทธิภาพในการเตรียมข้อมูล

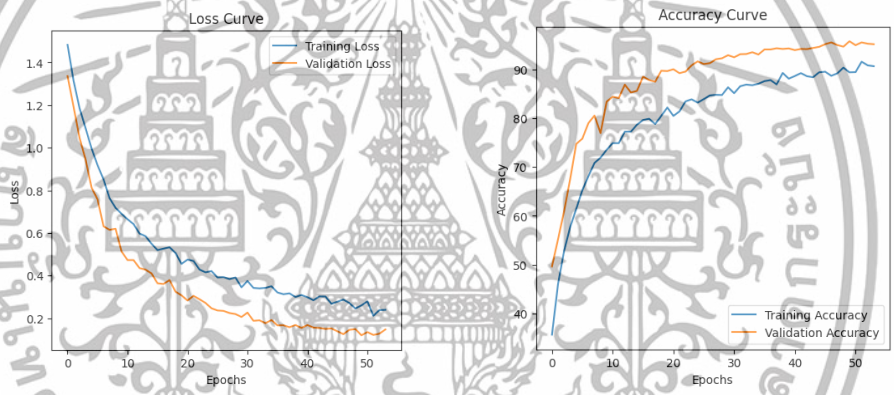
4.1.1 ผลการทดสอบการเตรียมข้อมูลสำหรับฝึกโมเดล CNN

ผลการทดสอบการฝึกโมเดล CNN ด้วยโมเดล Default และมีการกำหนดค่าไฮเปอร์พารามิเตอร์ให้เหมือนกัน ได้แก่ Learning rate = 0.0001, Batch size = 4, Epochs = 100, Dropout rate = 0.23, Patience = 5 และใช้ Optimizer คือ Adam เพื่อเปรียบเทียบประสิทธิภาพในการเตรียมข้อมูล ผลลัพธ์ที่ได้จากการเทรนโมเดลแบบไม่มีการขยายข้อมูล ค่า Loss ของชุดข้อมูล Validation มีค่าสูงกว่าชุดข้อมูล Training สะท้อนให้เห็นถึงการเกิด Overfitting แสดงดังรูปที่ 4.1(a) นอกจากนี้กราฟ Accuracy Curve แสดงดังรูปที่ 4.1(b) ก็แสดงให้เห็นถึงการเกิด Overfitting เช่นกัน โดยค่า Accuracy ในชุดข้อมูล Training มีค่าสูงกว่าชุดข้อมูล Validation ซึ่งค่าความแม่นยำของโมเดลอยู่ที่ 95.23% และผลลัพธ์ที่ได้จากการเทรนโมเดลแบบมีการขยายข้อมูล โมเดลมีการเรียนรู้ที่ดีไม่มีการเกิด Overfitting แสดงดังรูปที่ 4.2(a) และกราฟ Accuracy ของชุด Validation และ Training แสดงดังรูปที่ 4.2(b) แสดงให้เห็นว่าโมเดลสามารถปรับปรุงทั้งในชุดข้อมูลฝึกและชุดตรวจสอบได้อย่างดี ซึ่งค่าความแม่นยำของโมเดลอยู่ที่ 95.53% ดังนั้นการขยายข้อมูลในขั้นตอน Pre-processing จึงมีความสำคัญที่ทำให้โมเดลสามารถเรียนรู้ได้ดี



(a) (b)

รูปที่ 4.1 ผลลัพธ์การเทรนโมเดลที่ไม่มีการขยายข้อมูลของโมเดล CNN
(a) กราฟ Loss curve (b) กราฟ Accuracy curve



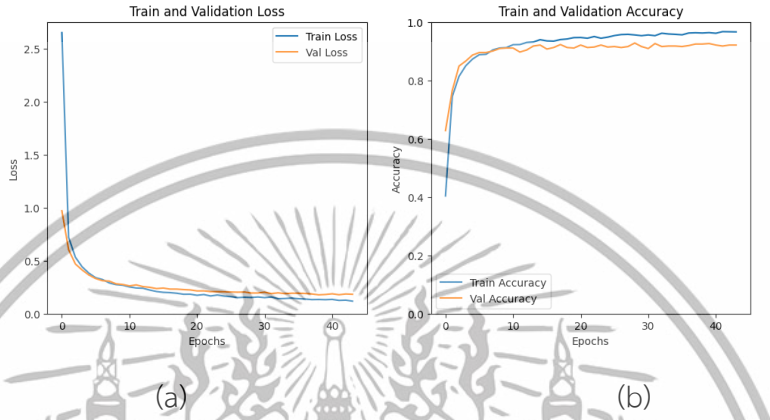
(a) (b)

รูปที่ 4.2 ผลลัพธ์การเทรนโมเดลที่มีการขยายข้อมูลของโมเดล CNN
(a) กราฟ Loss curve (b) กราฟ Accuracy curve

4.1.2 ผลการทดสอบการเตรียมข้อมูลสำหรับฝึกโมเดล ResNet18

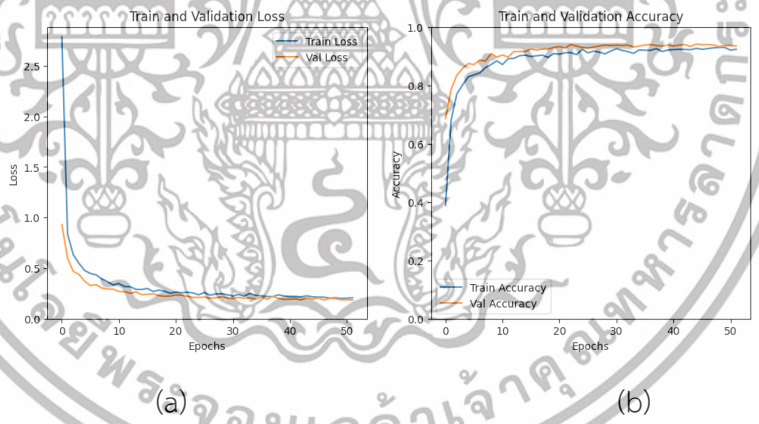
ผลทดสอบการฝึกโมเดล ResNet18 ด้วยการกำหนดค่าไฮเปอร์พารามิเตอร์ที่เป็นค่า Default ได้แก่ Batch size เท่ากับ 32 Epochs เท่ากับ 100 Patience เท่ากับ 5 และใช้ Optimizer คือ SGD เพื่อทำการเปรียบเทียบประสิทธิภาพในการเตรียมข้อมูล โดยผลลัพธ์ที่ได้จากการเทรนโมเดลจากข้อมูลที่ไม่มีการขยายข้อมูล (Without Augmentation) ค่า Loss ของ Validation สูงกว่าค่า Loss ของ Training แสดงถึงโมเดลเกิดการ Overfitting กราฟ Loss curve แสดงดังรูปที่ 4.3 (a) และมีความสัมพันธ์กับกราฟ Accuracy curve แสดงดังรูปที่ 4.3 (b) ที่แสดงถึงการเกิด Overfitting เนื่องจากค่า Accuracy ของ Validation ต่ำกว่า Training ซึ่งค่าความแม่นยำของโมเดลมีค่าเท่ากับ 92.11% และผลลัพธ์ที่ได้จากการเทรนโมเดลจากข้อมูลที่มีการขยายข้อมูล (Augmentation) โมเดล

สามารถเรียนรู้ในลักษณะทั่วไปได้ดี ไม่มีการ Overfitting มีกราฟ Loss curve แสดงดังรูปที่ 4.4 (a) และมีความสัมพันธ์กับกราฟ Accuracy curve แสดงดังรูปที่ 4.4 (b) แสดงถึงโมเดลกำลังเรียนรู้และทำนายผลลัพธ์ได้ถูกต้องมากขึ้น ซึ่งค่าความแม่นยำของโมเดลมีค่าเท่ากับ 93.48% ดังนั้นการขยายข้อมูลในขั้นตอน Pre-processing จึงมีความสำคัญที่ทำให้โมเดลสามารถเรียนรู้ได้ดี



(a) (b)

รูปที่ 4.3 ผลลัพธ์ของการเทรนโมเดลจากข้อมูลที่ไม่มีการขยายข้อมูลของโมเดล ResNet18
(a) กราฟ Loss curve (b) กราฟ Accuracy curve



(a) (b)

รูปที่ 4.4 ผลลัพธ์ของการเทรนโมเดลจากข้อมูลที่มีการขยายข้อมูลของโมเดล ResNet18
(a) กราฟ Loss curve (b) กราฟ Accuracy curve

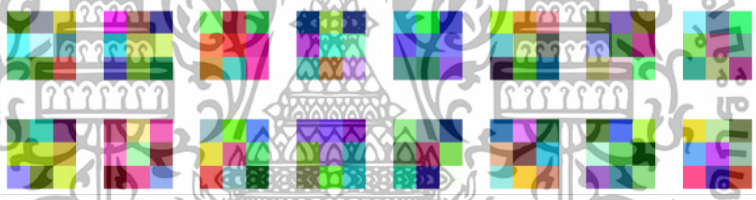
4.2 ผลการทดสอบการดึงคุณลักษณะของการเรียนรู้เชิงลึกในแต่ละโมเดล

4.2.1 ผลการทดสอบการดึงคุณลักษณะด้วยโมเดล CNN

ภาพต้นฉบับของรูปประตูที่ถูกปรับให้มีขนาดเท่ากับ 224x224x3 พิกเซล ซึ่งเป็นภาพที่จะใช้ทดสอบเบื้องต้นสำหรับการประมวลผลแต่ละเลเยอร์ แสดงดังรูปที่ 4.5

รูปที่ 4.6-4.8 เป็นตัวอย่างของ Feature Map ที่ได้จาก Convolution Layer และ MaxPooling โดยจะแสดงฟิลเตอร์ทั้งหมดใน Convolution Layer1 มีจำนวน 16 ฟิลเตอร์ ขนาด 3x3 แสดงดังรูปที่ 4.6 รูป Feature Map ของ Convolution Layer1 แสดงดังรูปที่ 4.7 ภาพเอาต์พุตทั้งหมดหลังผ่านฟิลเตอร์ใน Convolution Layer1 มีขนาด 224x224x16 พิกเซล แสดงดังรูปที่ 4.8

รูปที่ 4.5 ตัวอย่างภาพต้นฉบับของรูปประตูหลังถูกปรับขนาด

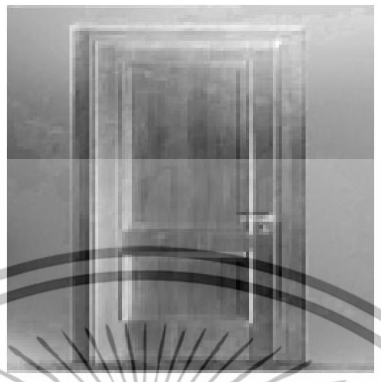


รูปที่ 4.6 ฟิลเตอร์ทั้งหมดใน Convolution Layer1 จำนวน 16 ฟิลเตอร์

รูปที่ 4.7 Feature Map ของ Convolution Layer1

รูปที่ 4.8 ภาพเอาต์พุตของฟิลเตอร์ที่ 1 จาก Convolution Layer1

ตัวอย่างภาพเอาต์พุตขนาด 112x112x16 พิกเซล ของ MaxPooling Layer1 แสดงดังรูปที่ 4.9 โดย MaxPooling มีขนาด 2x2 ค่า Stride เท่ากับ 2



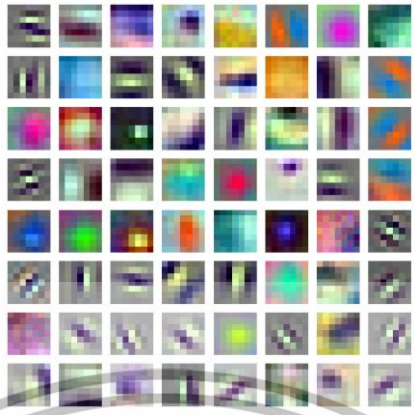
รูปที่ 4.9 ภาพเอาต์พุตของฟิลเตอร์ที่ 1 จาก MaxPooling Layer1

4.2.2 ผลการทดสอบการดึงคุณลักษณะด้วยโมเดล ResNet18

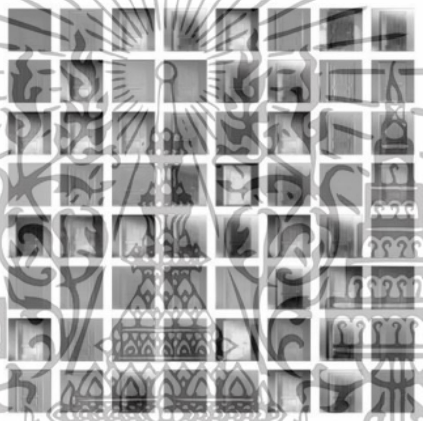
ภาพต้นฉบับของรูปประตูที่ถูกปรับให้มีขนาดเท่ากับ 224x224x3 พิกเซล ซึ่งเป็นภาพที่จะใช้ทดสอบเบื้องต้นสำหรับการประมวลผลแต่ละเลเยอร์ แสดงดังรูปที่ 4.10

รูปที่ 4.11-4.13 เป็นตัวอย่างของ Feature Map ที่ได้จาก Convolution Layer โดยจะแสดงฟิลเตอร์ทั้งหมดในเลเยอร์ Convolution (Conv2D) โดยมีจำนวน 64 ฟิลเตอร์ ขนาด 7x7 แสดงดังรูปที่ 4.11 รูปภาพเอาต์พุตทั้งหมดหลังผ่านแต่ละฟิลเตอร์ในเลเยอร์ Convolution (Conv2D) แสดงดังรูปที่ 4.12 และรูปภาพเอาต์พุตขนาด 112x112x64 พิกเซล หลังผ่านเลเยอร์ Convolution (Conv2D) แสดงดังรูปที่ 4.13

รูปที่ 4.10 ตัวอย่างภาพต้นฉบับของรูปประตูหลังถูกปรับขนาด



รูปที่ 4.11 ฟิลเตอร์ทั้งหมดในเลยอร์ Convolution (Conv2D) จำนวน 64 ฟิลเตอร์

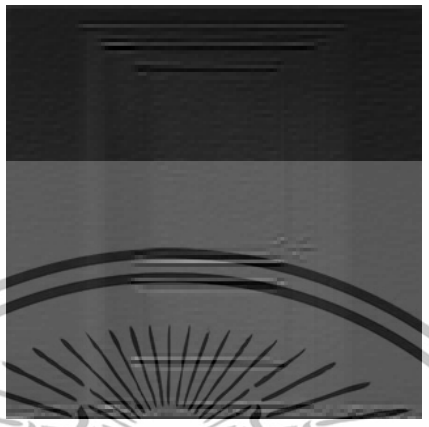


รูปที่ 4.12 ภาพเอาต์พุตทั้งหมดหลังผ่านแต่ละฟิลเตอร์ในเลเยอร์ Convolution (Conv2D)

รูปที่ 4.13 ภาพเอาท์พุตของฟิลเตอร์ที่ 1 จากเลเยอร์ Convolution (Conv2D)

ตัวอย่างเอาต์พุตขนาด 112x112x64 พิกเซล ของเลเยอร์ BatchNorm2D ซึ่งเป็น Batch Normalization layer ของโมเดล ResNet18 แสดงดังรูปที่ 4.14 ตัวอย่างเอาต์พุตขนาด 112x112x64 พิกเซล ของเลเยอร์ ReLU (Activation layer) ซึ่งเป็น ReLU layer แรกของโมเดล ResNet18 แสดงดังรูปที่ 4.15 และตัวอย่างเอาต์พุตขนาด 56x56x64 พิกเซล ของเลเยอร์

MaxPooling2D ซึ่งเป็น Max Pooling layer แรกของโมเดล ResNet18 ซึ่งมีขนาด 3x3 และ Stride เท่ากับ 2 แสดงดังรูปที่ 4.16



รูปที่ 4.14 ภาพเอาต์พุตของเลเยอร์ BatchNorm2D

รูปที่ 4.15 ภาพเอาต์พุตของเลเยอร์ ReLU

รูปที่ 4.16 ภาพเอาต์พุตของเลเยอร์ MaxPooling2D

4.3 ผลการทดสอบการปรับแต่งโมเดล

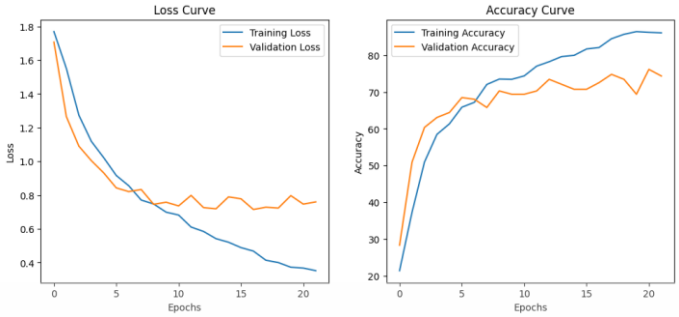
ผู้จัดทำได้ทำการทดสอบอัลกอริทึมการเรียนรู้เชิงลึกทั้ง 2 สถาปัตยกรรม โดยทำการหาโครงสร้างที่เหมาะสมสำหรับโมเดล CNN และพารามิเตอร์ที่เหมาะสมสำหรับโมเดล ResNet18 ที่ส่งผลให้มีความแม่นยำสูงหรือให้มีค่า Loss ต่ำที่สุด โดยการทดสอบของโมเดล CNN ผู้จัดทำทดสอบด้วยรูปแบบโครงสร้างเลเยอร์ที่แตกต่างกัน 4 โครงสร้าง เพื่อเปรียบเทียบประสิทธิภาพของแต่ละโครงสร้าง โดยโครงสร้างที่มีความแม่นยำสูงถือเป็นโมเดลที่ดีที่สุด และในการทดสอบโมเดล ResNet18 ผู้จัดทำเลือกใช้วิธี Grid Search คือวิธีการหาพารามิเตอร์ที่ดีที่สุดด้วยการลองใช้พารามิเตอร์ที่กำหนดไว้ล่วงหน้าทุกชุดทั้งหมด 6 กรณี พารามิเตอร์ชุดที่ดีที่สุดจะถูกนำมาเทรนโมเดลเพื่อให้ได้โมเดลที่ดีที่สุดและนำมาทดสอบด้วยชุดข้อมูลทดสอบเพื่อประเมินประสิทธิภาพของโมเดลต่อไป

4.3.1 ผลการทดสอบของการปรับแต่งโมเดล CNN

ตารางที่ 4.1 สรุปผลการเปรียบเทียบโมเดล CNN

โมเดลที่	เวลาที่ใช้ในการฝึก (hh:mm:ss)	ค่าความแม่นยำ (%)	F1 score (%)
1	1:44:13	73.89	74
2	4:21:02	61.77	62
3	1:33:09	75.29	75
4	**1:17:01**	**75.52**	**76**

จากตารางที่ 4.1 พบว่าการเปรียบเทียบประสิทธิภาพของโมเดลที่ใช้โครงสร้าง CNN โมเดลที่ 4 ซึ่งมีโครงสร้าง CNN ประกอบด้วย Convolution Layer และ MaxPooling Layer จำนวน 4 ชุด มีค่าความแม่นยำและ F1 score สูงที่สุดเท่ากับ 75.52% และ 76% ตามลำดับ ใช้เวลาในการฝึก 1 ชั่วโมง 17 นาที 1 วินาที จะเห็นว่าโครงสร้าง CNN ที่มีหลายชั้น Convolution Layer และ MaxPooling Layer สามารถช่วยเพิ่มประสิทธิภาพในการเรียนรู้ของโมเดลได้มากขึ้น เมื่อเปรียบเทียบกับโครงสร้างที่มีความซับซ้อนน้อยกว่า และกราฟของผลการฝึกสอนโมเดลของโมเดล CNN ที่ 4 แสดงดังรูปที่ 4.17



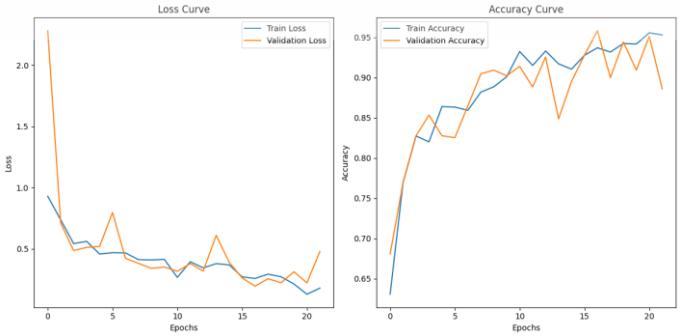
รูปที่ 4.17 กราฟของผลการฝึกสอนโมเดลของโมเดล CNN ที่ 4

4.3.2 ผลการทดสอบของการปรับแต่งโมเดล ResNet18

ตารางที่ 4.2 สรุปผลการเปรียบเทียบโมเดล ResNet18

กรณีที่	Optimizer	Epoch	Batch size	Fit time (hh:mm:ss)	Test accuracy (%)
1	Adam	15	16	0:55:26	74.83
2	Adam	15	32	0:44:34	89.70
3	Adam	20	16	0:42:56	86.48
4	Adam	20	32	0:58:50	89.51
5	Adam	30	16	0:45:51	91.38
6	**Adam**	**30**	**32**	**1:26:26**	**95.34**

จากตารางที่ 4.2 พบว่ากรณีที่ 6 มีค่าความแม่นยำสูงที่สุดเท่ากับ 95.34% ดังนั้นจึงได้ค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมสำหรับโมเดล ResNet18 ได้แก่ Optimizer คือ Adam, Epoch เท่ากับ 30, Batch size เท่ากับ 32 และค่า Fit time ที่ใช้เท่ากับ 1 ชั่วโมง 26 นาที 26 วินาที และกราฟของผลการฝึกสอนโมเดลของโมเดล ResNet18 กรณีที่ 6 แสดงดังรูปที่ 4.18



รูปที่ 4.18 กราฟของผลการฝึกสอนโมเดลของโมเดล ResNet18 กรณีที่ 6

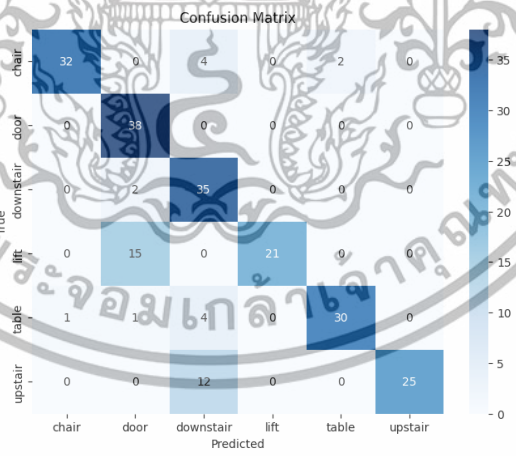
4.4 ผลการทดสอบการประเมินประสิทธิภาพอัลกอริทึมการเรียนรู้เชิงลึก

หลังจากได้โครงสร้างโมเดลที่เหมาะสมสำหรับโมเดล CNN และชุดพารามิเตอร์ที่เหมาะสมสำหรับโมเดล ResNet18 นำทั้ง 2 โมเดลมาทดสอบประเมินประสิทธิภาพของโมเดลด้วยชุดข้อมูลทดสอบ และผลการประเมินประสิทธิภาพของโมเดลผ่าน Metric ได้แก่ Precision, Recall, F1 score และ Accuracy พร้อมทั้งแสดง Confusion matrix

ตารางที่ 4.3 สรุปผลการทดสอบการประเมินประสิทธิภาพอัลกอริทึมการเรียนรู้เชิงลึก

Architecture	Precision	Recall	F1 score	Accuracy (%)
CNN	0.75	0.74	0.74	74
ResNet18	**0.87**	**0.82**	**0.82**	**82**

จากตารางที่ 4.3 ผลการทดสอบการประเมินประสิทธิภาพอัลกอริทึมการเรียนรู้เชิงลึกของ CNN และ ResNet18 พบว่าค่าความแม่นยำที่สูงที่สุดคือโมเดล ResNet18 มีค่าเท่ากับ 82% และโมเดล CNN 74% ซึ่งค่า Precision, Recall และ F1 score ที่ได้จากการประเมินประสิทธิภาพของโมเดล ResNet18 มีค่า 0.87, 0.82 และ 0.82 ตามลำดับ สามารถแสดง Confusion Matrix ของโมเดล ResNet18 แสดงดังรูปที่ 4.19 สรุปได้ว่าโครงสร้างที่ให้ความแม่นยำและมีประสิทธิภาพในการจำแนกวัตถุมากที่สุดคือ โครงสร้าง ResNet18



รูปที่ 4.19 Confusion Matrix ของโมเดล ResNet18

4.5 ผลการทดสอบการทำงานของอุปกรณ์

4.5.1 ผลการทดสอบการจำแนกวัตถุของอุปกรณ์

ผู้จัดทำได้เลือกใช้โมเดล ResNet18 ที่ผ่านการปรับแต่งโมเดลด้วยชุดพารามิเตอร์ที่ดีที่สุดและจากการประเมินประสิทธิภาพของโมเดลได้ค่าความแม่นยำสูงสุด โดยจะนำโมเดล ResNet18 มาใช้สำหรับการจำแนกวัตถุทั้ง 6 คลาส ได้แก่ ประตู ลิฟต์ โต๊ะ เก้าอี้ บันไดขึ้น และบันไดลง ร่วมกับอุปกรณ์สำหรับช่วยเหลือผู้บกพร่องทางสายตา เมื่อนำอุปกรณ์มาใช้งานในสถานการณ์จริง อุปกรณ์สามารถจำแนกวัตถุทั้ง 6 คลาสได้จริง โดยตัวอย่างผลลัพธ์การจำแนกรูปประตูแสดงดังรูปที่ 4.20 ตัวอย่างผลลัพธ์การจำแนกรูปลิฟต์แสดงดังรูปที่ 4.21 ตัวอย่างผลลัพธ์การจำแนกรูปโต๊ะแสดงดังรูปที่ 4.22 ตัวอย่างผลลัพธ์การจำแนกรูปเก้าอี้แสดงดังรูปที่ 4.23 ตัวอย่างผลลัพธ์การจำแนกรูปบันไดขึ้นแสดงดังรูปที่ 4.24 และตัวอย่างผลลัพธ์การจำแนกรูปบันไดลงแสดงดังรูปที่ 4.25

รูปที่ 4.20 ตัวอย่างผลลัพธ์การจำแนกรูปประตู

รูปที่ 4.21 ตัวอย่างผลลัพธ์การจำแนกรูปลิฟต์

รูปที่ 4.22 ตัวอย่างผลลัพธ์การจำแนกรูปโต๊ะ

รูปที่ 4.23 ตัวอย่างผลลัพธ์การจำแนกรูปเก้าอี้

รูปที่ 4.24 ตัวอย่างผลลัพธ์การจำแนกรูปบันไดขึ้น

รูปที่ 4.25 ตัวอย่างผลลัพธ์การจำแนกรูปบันไดลง

4.5.2 ผลการทดสอบการแจ้งเตือนของอุปกรณ์

เมื่ออุปกรณ์สามารถใช้งานได้จริงจำแนกวัตถุได้ อุปกรณ์จะแจ้งเตือนออกมาเป็นเสียงด้วยหลักการ TTS โดยจะแจ้งเตือนถึงวัตถุที่จำแนกได้พร้อมกับระยะห่างของวัตถุที่คำนวณระยะห่างโดยใช้ Ultrasonic sensor ซึ่งระยะห่างได้ทำการกำหนดให้อุปกรณ์แจ้งเตือนทุกๆ ระยะ ดังนี้ 3, 2 และ 1 เมตร จากที่ผู้จัดทำได้ทำการทดสอบสามารถแบ่งออกเป็น 3 กรณีตามพื้นผิวของวัตถุทั้ง 6 คลาส ได้แก่

- กรณีที่ 1 : พื้นผิวระนาบแนวตั้ง ได้แก่ ประตู และลิฟต์
- กรณีที่ 2 : พื้นผิวไม่สม่ำเสมอ ได้แก่ โต๊ะ และเก้าอี้
- กรณีที่ 3 : พื้นผิวลาดเอียง ได้แก่ บันไดขึ้น และบันไดลง

ตารางที่ 4.4 สรุปผลลัพธ์การแจ้งเตือนของอุปกรณ์

กรณีที่	ค่าความผิดพลาดเฉลี่ย (%)	ตัวอย่างผลลัพธ์การแจ้งเตือนของอุปกรณ์
1	0	Speaking door in 3 meters Predicted: door, Confidence: 0.86, Time: 0.3673 Predicted: door, Confidence: 0.84, Time: 0.3761
2	6.67	Speaking chair in 3 meters Predicted: chair, Confidence: 0.93, Time: 0.3898 Predicted: chair, Confidence: 0.83, Time: 0.3517
3	20	Speaking upstair in 3 meters Predicted: upstair, Confidence: 0.99, Time: 0.3298 Predicted: upstair, Confidence: 0.99, Time: 0.2776

จากตารางที่ 4.4 ในกรณีที่ 1 วัตถุในคลาสประตูและลิฟต์มีลักษณะพื้นผิวระนาบแนวตั้งและอยู่ในระดับที่พอดีกับระดับสายตาทำให้ Ultrasonic sensor สามารถวัดระยะห่างระหว่างวัตถุได้

แม่นยำมากที่สุด ไม่มีความผิดพลาดในทุกๆระยะ ในกรณีที่ 2 วัตถุในคลาสโต๊ะและเก้าอี้มีลักษณะพื้นผิวไม่สม่ำเสมอเนื่องจากมีหลากหลายรูปร่างและยังอยู่ต่ำกว่าระดับสายตา สามารถวัดค่าความผิดพลาดเฉลี่ยของระยะห่างในกรณีนี้ได้เท่ากับ 6.67% และในกรณีที่ 3 วัตถุในคลาสบันไดขึ้นและบันไดลงมีลักษณะเป็นพื้นผิวลาดเอียง จะเกิดความผิดพลาดมากขึ้นเมื่อเริ่มเข้าใกล้วัตถุเนื่องจากมีความชันของวัตถุ สามารถวัดค่าความผิดพลาดเฉลี่ยของระยะห่างในกรณีนี้ได้เท่ากับ 20% และยังสรุปได้ว่าการทำงานของอุปกรณ์ใช้เวลาในการประมวลผลประมาณ 0.36 วินาที

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้นำเสนอการศึกษาและการใช้งานอัลกอริทึมเชิงลึกเพื่อประมวลผลรูปภาพและเปรียบเทียบประสิทธิภาพอัลกอริทึมการเรียนรู้เชิงลึกในการจำแนกสิ่งกีดขวางสำหรับผู้มีความบกพร่องทางสายตาและส่งเสียงเตือนให้ผู้ใช้งานทราบ ผู้จัดทำได้เลือกใช้โครงสร้างทั้งหมด 2 โครงสร้าง ได้แก่ CNN และ ResNet18 ในการเรียนรู้โมเดลและการจำแนกข้อมูล ซึ่งจากการทดสอบเพื่อหาโมเดลที่ดีที่สุด พบว่าโครงสร้าง ResNet18 ให้ความแม่นยำสูงที่สุด โดยใช้วิธี Grid Search เพื่อหาค่าพารามิเตอร์ที่เหมาะสมกับโมเดล ได้แก่ Optimizer คือ Adam, Epoch = 30, Batch size = 32 มีค่าความแม่นยำในการฝึกสอน (Accuracy) = 95.34% มีค่าผลการประเมินประสิทธิภาพของโมเดลดังนี้ Precision = 0.87, Recall = 0.82, F1-score = 0.82 และค่าความแม่นยำ = 82% รองลงมาคือโครงสร้าง CNN และจากการทดสอบการทำงานอุปกรณ์สำหรับช่วยเหลือผู้บกพร่องทางสายตา เมื่อนำมาใช้งานในสถานการณ์จริงพบว่าอุปกรณ์สามารถจำแนกวัตถุได้ทั้ง 6 คลาส ยังสามารถส่งเสียงเตือนถึงวัตถุที่จำแนกพร้อมกับระยะห่างของวัตถุได้ และใช้เวลาในการประมวลผลประมาณ 0.36 วินาที

5.2 ข้อเสนอแนะ

ในปริญญานิพนธ์นี้ ผู้จัดทำได้ใช้ชุดข้อมูลรูปภาพ จำนวน 2976 รูป ซึ่งทำให้ความหลากหลายของข้อมูลมีไม่มาก ประสิทธิภาพของโมเดลสามารถปรับปรุงให้ดียิ่งขึ้นได้โดยการใช้เทคนิคการขยายข้อมูล (Data Augmentation) เพื่อเพิ่มความหลากหลายให้กับข้อมูล ทำให้การเรียนรู้ของโมเดลมีประสิทธิภาพมากขึ้น อีกทั้งยังช่วยป้องกันการเกิดปัญหา Overfitting ได้ และในการคำนวณระยะห่างมีข้อจำกัดในด้านฮาร์ดแวร์ของ Raspberry pi ในการประมวลผลจึงจำเป็นต้องใช้ Ultrasonic sensor เพื่อให้สามารถประมวลผลได้อย่างรวดเร็ว หากพัฒนาไมโครคอนโทรลเลอร์ให้มีการประมวลผลที่ดีขึ้นอาจช่วยเพิ่มประสิทธิภาพในการคำนวณระยะห่างด้วย Library ของ Python เพื่อลดความผิดพลาดลงได้

บรรณานุกรม

[1] GeeksforGeeks. “รูปภาพคืออะไร?.” Digital Image Processing Basics - GeeksforGeeks

[2] Simplilearn. “What is an image.” What is Image Processing? Everything you need to Know! (simplilearn.com)

[3] Rohit Kundu. “Image Processing.” Image Processing: Techniques, Types, & Applications [2024] (v7labs.com)

[4] Neetika Khandelwal. “Image Processing in python.” Image Processing in Python: Algorithms, Tools, and Methods You Should Know (neptune.ai)

[5] AIGEN Team. “AI Image Processing คือ.” Image Processing คือตัวช่วยยกระดับการทำงานในทุกอุตสาหกรรม!

[6]Farooq Alvi. “Computer Vision.” Computer Vision and Image Processing: A Beginner's Guide

[7] Cloudflare, Inc. “What is a neural network?.” What is a neural network? | Types of neural networks | Cloudflare

[8] GeeksforGeeks. “What is a neural network?.” What is a neural network? - GeeksforGeeks

[9] Zoumana Kelta. “Classification in Machine Learning: An Introduction.” Classification in Machine Learning: A Guide for Beginners | DataCamp

[10] อาจารย์ ดร.ปภังกร อิ่นแก้ว. “วิทยาการข้อมูลเบื้องต้น.” fetch.php (cmu.ac.th)

[11] Papon SMC. “What’s AI, Machine learning, Deep learning (เบื้องต้น).” What’s AI, Machine learning, Deep learning (เบื้องต้น) | by Papon Smc | Medium

[12] Metana. “Deep Learning Models for Classification.” Deep Learning Models for Classification : A Comprehensive Guide - Metana

[13] Javatpoint. “Deep Learning Algorithms.” Deep Learning Algorithms - Javatpoint

[14] Van Hiep Phung และ Eun Joo Rhee. “A High-Accuracy Model Average Ensemble of Convolutional of Cloud Image Patches on Small Datasets.” A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets (researchgate.net)

[15] Natthawat Phongchit. “Convolutional Neural Network (CNN) คืออะไร.” Convolutional Neural Network (CNN) คืออะไร | by Natthawat Phongchit | Medium

[16] Sumit Saha. “A Comprehensive Guide to Convolutional Neural Networks” A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science

[17] cs231n. “Convolutional Neural Networks.” CS231n Convolutional Neural Networks for Visual Recognition

[18] Surapong Kanoktipsathaporn. “ReLU Function คืออะไร.” ReLU Function คืออะไร ทำไมถึงนิยมใช้ใน Deep Neural Network ต่างกับ Sigmoid อย่างไร – Activation Function ep.3 - BUA Labs

[19] Surapong Kanoktipsathaporn. “Softmax Function คืออะไร.” Softmax Function คืออะไร เราจะใช้งาน Softmax Function อย่างไร ประโยชน์ของ Softmax - BUA Labs

[20] Surapong Kanoktipsathaporn. “BatchNorm คืออะไร.” BatchNorm คืออะไร สอน Batch Normalization เทรน Machine Learning โมเดล Deep Convolutional Neural Network - ConvNet ep.5 - BUA Labs

[21] Harsh Yadav. “Dropout in Neural Networks.” Dropout in Neural Networks. Dropout layers have been the go-to... | by Harsh Yadav | Towards Data Science

[22] ชิตพงษ์ กิตตินราดร. “Neural Network Regularisation.” Machine Learning *บทที่* 18: Neural Network Regularisation (guopai.github.io)

[23] Arden Dertat. “Applied Deep Learning.” Applied Deep Learning - Part 4: Convolutional Neural Networks | by Arden Dertat | Towards Data Science

[24] Prudhviraju Srivatsavaya. “Flatten Layer — Implementation, Advantage and Disadvantages.” Flatten Layer — Implementation, Advantage and Disadvantages | by Prudhviraju Srivatsavaya | Medium

[25] Satsawat Natakarnkitkul. “Beginners Guide to Convolutional Neural Network from Scratch — Kuzushiji-MNIST.” Beginners Guide to Convolutional Neural Network from Scratch — Kuzushiji-MNIST | Towards AI

[26] Diego Unzueta. “Fully Connected Layer vs. Convolutional Layer: Explained.” Fully Connected Layer vs Convolutional Layer: Explained | Built In

[27] Lark Editorial Team. “Pre Trained Models.” Pre Trained Models (larksuite.com)

[28] Amazon Web Service, Inc. “Transfer Learning (การถ่ายโอนความรู้) คืออะไร?.” Transfer Learning คืออะไร - อธิบายการถ่ายโอนการเรียนรู้ในแมชชีนเลิร์นนิง - AWS (amazon.com)

[29] Javatpoint. “ResNet: Residual Network.” ResNet: Residual Network - Javatpoint

[30] Kaimimg He, Xiangyu Zhang, Shaoqing Ren และ Jian Sun. “Deep Residual Learning for Image Recognition.” 1512.03385 (arxiv.org)

[31] Allena Vekata Sai Abhishek, Dr. Vekateswara Rao Gurrala และ Dr. Laxman Sahoo. “Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset.” IJRTI (ijcrt.org)

[32] Raúl Gombru. “Cross-entropy loss.” Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names

[33] Amazon Web Service, Inc. “Hyperparameter tunning คืออะไร.” Hyperparameter Tuning คืออะไร - อธิบาย Hyperparameter Tuning - AWS

[34] Inna Logunova. “Bayesian optimizer.” Bayesian optimization

[35] Taeef Najib. “Hyperparameter Tuning Using Optuna.” Hyperparameter Tuning Using Optuna | by Taeef Najib | Medium

[36] ThemeIsle. “Pytorch คืออะไร.” Pytorch วิธีใช้งานเบื้องต้นทดลองทำโมเดลจำแนกรูปภาพด้วย ConvNet (sub-brain.com)

[37] Chaiyaphop Jamjumrat. “Gradient Descent Optimization algorithm ที่จะช่วยให้มี Error น้อยที่สุด.” Gradient Descent Optimization algorithm ที่จะช่วยให้มี Error น้อยที่สุด – BorntoDev เริ่มต้นเรียน เขียนโปรแกรม ขั้นเทพ !

[38] GeeksforGeeks. “ML | Stochastic Gradient Descent (SGD).” ML | Stochastic Gradient Descent (SGD) - GeeksforGeeks

[39] Gunand Mayanglambam. “Deep Learning Optimizers.” Deep Learning Optimizers. SGD with momentum, Adagrad, Adadelta... | by Gunand Mayanglambam | Towards Data Science

[40] Jiawei Zhang. “Gradient Descent based Optimization Algorithms.” 1903.03614

[41] everydaymarketing.co. “Confusion Matrix คืออะไร?.” Confusion Matrix และหลักการประเมินประสิทธิภาพ ML Model (everydaymarketing.co)

[42] Pagon Gatchalee. “Confusion Matrix เครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนาย ใน ML” Confusion Matrix *เครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนาย ใน* Machine learning | by Pagon Gatchalee | Medium

[43] Frank Krüger. “multiclass confusion matrix.” Confusion-matrix-for-multi-class-classification-The-confusion-matrix-of-a-classification.png (850×512)

[44] GeeksforGeeks. “What is Edge Detection.” Comprehensive Guide to Edge Detection Algorithms - GeeksforGeeks

[45] Jamine Martin. “”What is text to speech.” What is text-to-speech technology (TTS)?

[46] Amazon Web Services, Inc. “What is Text-to-Speech (TTS)?” What is Text to Speech? - TTS Explained - AWS

[47] Max Krupenko. “How does tts work.” What is Text to Speech and How It Works: An In-Depth Guide

[48] SOURCEFORGE. “eSpeak text to speech.” eSpeak: Speech Synthesizer

[49] Shivdureja. “espeak.” Everything about espeak-ng linux command | by Shivdu | Medium

ภาคผนวก ก
โปรแกรมสำหรับการหาพารามิเตอร์ที่ดีที่สุด

```
# CNN
import os
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader, random_split
import matplotlib.pyplot as plt
import torch.nn.functional as F
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
import optuna
from collections import Counter
from PIL import Image
from torchsummary import summary

img_height = 480
img_width = 640
num_classes = 6
num_epochs = 100
batch_sizes = [4, 6]
patience = 10
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

def load_data(data_dir, val_split=0.1):
    transform = transforms.Compose([
        transforms.Resize((img_height, img_width)),
        transforms.ToTensor(),
    ])
    dataset = ImageFolder(root=data_dir, transform=transform)
```

```
    val_size = int(len(dataset) * val_split)
    train_size = len(dataset) - val_size
    train_dataset, val_dataset = random_split(dataset, [train_size, val_size])
    return train_dataset, val_dataset


class CNN(nn.Module):
    def __init__(self, dropout_rate=0.5):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16,
kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=16, out_channels=32,
kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.dropout = nn.Dropout(dropout_rate)
        self.fc1 = nn.Linear(32 * (img_height // 4) * (img_width // 4), 128)
        self.fc2 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1)
        x = self.dropout(F.relu(self.fc1(x)))
        x = self.fc2(x)
        return x


model = CNN().to(device)
summary(model, input_size=(3, img_height, img_width))


train_dataset, val_dataset =
load_data(r'C:\Users\DELL\Desktop\Project4T1\datasample\train')
```

```
class_counts = Counter(train_dataset.dataset.targets)
class_distribution = {train_dataset.dataset.classes[i]: class_counts[i] for i in
range(len(train_dataset.dataset.classes))}
num_files = len(train_dataset)
num_classes = len(train_dataset.dataset.classes)

print(f"Found {num_files} files belonging to {num_classes} classes.")
for class_name, count in class_distribution.items():
    print(f"{class_name}: {count} files")

class EarlyStopping:
    def __init__(self, patience=patience, verbose=False):
        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_loss = None

    def __call__(self, val_loss, model):
        if self.best_loss is None:
            self.best_loss = val_loss
        elif val_loss > self.best_loss:
            self.counter += 1
            if self.verbose:
                print(f'Validation loss did not improve. Counter:
{self.counter}/{self.patience}')
            if self.counter >= self.patience:
                print("Early stopping!")
                return True
        else:
            self.best_loss = val_loss
            self.counter = 0
```

```
            return False

        # ฟังก์ชันการฝึกและประเมินผล ใช้ร่วมกับ Optuna
        def train_and_evaluate(trial):
            # Optuna กำหนดค่า hyperparameters
            learning_rate = trial.suggest_categorical('learning_rate', [0.001, 0.0001])
            batch_size = trial.suggest_categorical('batch_size', [4, 6])
            dropout_rate = trial.suggest_uniform('dropout_rate', 0.1, 0.5)

            train_dataset, val_dataset =
load_data(r'C:\Users\DELL\Desktop\Project4T1\datasample\train')
            train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
            val_loader = DataLoader(val_dataset, batch_size=batch_size,
shuffle=False)

            model = CNN(dropout_rate=dropout_rate).to(device)
            optimizer = optim.Adam(model.parameters(), lr=learning_rate)
            criterion = nn.CrossEntropyLoss()
            early_stopping = EarlyStopping(patience=patience, verbose=True)

            for epoch in range(num_epochs):
                model.train()
                running_loss = 0.0
                correct = 0
                total = 0

                for inputs, labels in train_loader:
                    inputs, labels = inputs.to(device), labels.to(device)
                    optimizer.zero_grad()
                    outputs = model(inputs)
```

```
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

        epoch_loss = running_loss / len(train_loader)
        epoch_accuracy = 100 * correct / total

        model.eval()
        running_val_loss = 0.0
        val_correct = 0
        val_total = 0

        with torch.no_grad():
            for inputs, labels in val_loader:
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                val_loss = criterion(outputs, labels)
                running_val_loss += val_loss.item()
                _, val_predicted = torch.max(outputs.data, 1)
                val_total += labels.size(0)
                val_correct += (val_predicted == labels).sum().item()

        val_epoch_loss = running_val_loss / len(val_loader)
        val_epoch_accuracy = 100 * val_correct / val_total

        print(f'Epoch [{epoch+1}/{num_epochs}], '
```

```
                f'Train Loss: {epoch_loss:.4f}, Train Accuracy:
{epoch_accuracy:.4f}%, '
                f'Val Loss: {val_epoch_loss:.4f}, Val Accuracy:
{val_epoch_accuracy:.4f}%')

            if early_stopping(val_epoch_loss, model):
                break

    return val_epoch_loss

# ฟังก์ชันหลักสำหรับเรียกใช้งาน Optuna
def main():
    study = optuna.create_study(direction='minimize')
    study.optimize(train_and_evaluate, n_trials=5)
    print("Best hyperparameters:", study.best_params)

if __name__ == "__main__":
    main()

# ResNet18
import time
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.models as models
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import numpy as np
from sklearn.metrics import accuracy_score

train_transform = transforms.Compose(
```

```
            [
                transforms.Resize((224, 224)),
                transforms.RandomHorizontalFlip(),
                transforms.RandomRotation(10),
                transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
                transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,
hue=0.1),
                transforms.ToTensor(),
                transforms.Normalize(
                    [0.485, 0.456, 0.406], [0.229, 0.224, 0.225]
                ),
            ]
        )
        test_transform = transforms.Compose(
            [
                transforms.Resize((224, 224)),
                transforms.ToTensor(),
                transforms.Normalize(
                    [0.485, 0.456, 0.406], [0.229, 0.224, 0.225]
                ),
            ]
        )
        train_dataset = ImageFolder(root="D:/test resnet18/dataset/train",
transform=train_transform)
        test_dataset = ImageFolder(root="D:/test resnet18/dataset/val",
transform=test_transform)

        epochs_list = [15, 20, 30]
        batch_sizes = [16, 32]
        optimizers = ["Adam"]
        patience = 5
```

```
best_loss = None
trigger_times = 0
results = []

for batch_size in batch_sizes:
    for epoch in epochs_list:
        for opt in optimizers:
            train_loader = DataLoader(
                train_dataset, batch_size=batch_size, shuffle=True
            )
            test_loader = DataLoader(test_dataset, batch_size=batch_size,
shuffle=False)

            model = models.resnet18(pretrained=True)
            num_features = model.fc.in_features
            model.fc = nn.Sequential(
                nn.Flatten(),
                nn.Linear(num_features, 512),
                nn.ReLU(),
                nn.Dropout(p=0.5),
                nn.Linear(512, 128),
                nn.ReLU(),
                nn.Linear(128, 6)
            )

            device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
            model = model.to(device)

            criterion = nn.CrossEntropyLoss()
```

```
if opt == "Adam":
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

start_time = time.time()
train_losses = []
val_losses = []
model.train()
best_loss = None
trigger_times = 0

for e in range(epoch):
    train_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)
        # Backward pass
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
    avg_train_loss = train_loss / len(train_loader)
    train_losses.append(avg_train_loss)

    model.eval()
    val_loss = 0.0
    y_true_train, y_pred_train = [], []
    with torch.no_grad():
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)
```

```
                outputs = model(images)
                loss = criterion(outputs, labels)
                val_loss += loss.item()
                _, preds = torch.max(outputs, 1)
                y_true_train.extend(labels.cpu().numpy())
                y_pred_train.extend(preds.cpu().numpy())

            train_accuracy = accuracy_score(y_true_train, y_pred_train)
            avg_val_loss = val_loss / len(train_loader)
            val_losses.append(avg_val_loss)

            print(
                f"Epoch [{e+1}/{epoch}], "
                f"Train Loss: {avg_train_loss:.4f}, "
                f"Validation Loss: {avg_val_loss:.4f}, "
                f"Train Accuracy: {train_accuracy:.4f}"
            )

            # Early Stopping
            if best_loss is None or avg_val_loss < best_loss:
                best_loss = avg_val_loss
                trigger_times = 0
            else:
                trigger_times += 1
                print(f"EarlyStopping counter: {trigger_times} out of
{patience}")
                if trigger_times >= patience:
                    print("Early stopping triggered!")
                    break
        model.train()
```

```
# ประเมินผลชุด train
model.eval()
y_true_train, y_pred_train = [], []
with torch.no_grad():
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)
        y_true_train.extend(labels.cpu().numpy())
        y_pred_train.extend(preds.cpu().numpy())
train_accuracy = accuracy_score(y_true_train, y_pred_train)

# ประเมินผลชุด validation
y_true_test, y_pred_test = [], []
test_loss = 0.0
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)
        y_true_test.extend(labels.cpu().numpy())
        y_pred_test.extend(preds.cpu().numpy())
        loss = criterion(outputs, labels)
        test_loss += loss.item()
test_accuracy = accuracy_score(y_true_test, y_pred_test)

elapsed_time = time.time() - start_time
results.append(
    {
        "batch_size": batch_size,
        "epoch": epoch,
```

```
                "optimizer": opt,
                "train_accuracy": train_accuracy,
                "test_accuracy": test_accuracy,
                "train_loss": avg_train_loss,
                "test_loss": test_loss / len(test_loader),
                "time_elapsed": elapsed_time,
            }
        )

for result in results:
    print(
        f"Batch Size: {result['batch_size']}, Epoch: {result['epoch']}, Optimizer:
{result['optimizer']}, "
        f"Train Accuracy: {result['train_accuracy']:.4f}, Test Accuracy:
{result['test_accuracy']:.4f}, "
        f"Time Elapsed: {result['time_elapsed']:.2f} seconds"
    )
```

ภาคผนวก ข

โปรแกรมสำหรับทำการฝึกสอนโมเดล

```
# CNN
import os
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader, random_split
import matplotlib.pyplot as plt
import numpy as np
from collections import Counter
from torchsummary import summary
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay
from PIL import Image

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
learning_rate = 0.0001
img_height, img_width = 224, 224
batch_size = 4
val_split = 0.2
num_epochs = 100
dropout_rate = 0.2311959186265376
num_classes = 6

# สร้างโมเดล CNN
class CNN_FourConvLayers(nn.Module):
    def __init__(self, dropout_rate=0.2311959186265376, num_classes=6):
        super(CNN_FourConvLayers, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=16,
kernel_size=3, padding=1)
```

```
                self.conv2 = nn.Conv2d(in_channels=16, out_channels=32,
kernel_size=3, padding=1)
                self.conv3 = nn.Conv2d(in_channels=32, out_channels=64,
kernel_size=3, padding=1)
                self.conv4 = nn.Conv2d(in_channels=64, out_channels=128,
kernel_size=3, padding=1)
                self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
                self.dropout = nn.Dropout(p=dropout_rate)

                self.fc1 = nn.Linear(128 * (img_height // 16) * (img_width // 16), 256)
                self.fc2 = nn.Linear(256, 128)
                self.fc3 = nn.Linear(128, num_classes)

            def forward(self, x):
                x = self.pool(nn.ReLU()(self.conv1(x)))
                x = self.pool(nn.ReLU()(self.conv2(x)))
                x = self.pool(nn.ReLU()(self.conv3(x)))
                x = self.pool(nn.ReLU()(self.conv4(x)))
                x = torch.flatten(x, 1)
                x = self.dropout(nn.ReLU()(self.fc1(x)))
                x = self.dropout(nn.ReLU()(self.fc2(x)))
                x = self.fc3(x)
                return x

        model = CNN_FourConvLayers(dropout_rate).to(device)
        summary(model, input_size=(3, img_height, img_width))

        transform_train = transforms.Compose([
            transforms.Resize((img_height, img_width)),
            transforms.RandomHorizontalFlip(),
            transforms.RandomRotation(10),
```

```
            transforms.RandomCrop((img_height, img_width), padding=4),
            transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,
hue=0.2),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
        ])

        transform_val_test = transforms.Compose([
            transforms.Resize((img_height, img_width)),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
        ])

        full_train_dataset =
ImageFolder(root="C:/Users/DELL/Desktop/Project4T1/datasample/train",
transform=transform_train)
        test_dataset =
ImageFolder(root="C:/Users/DELL/Desktop/Project4T1/datasample/test",
transform=transform_val_test)

        train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size,
shuffle=True)
        val_loader = DataLoader(dataset=val_dataset, batch_size=batch_size,
shuffle=True)
        test_loader = DataLoader(dataset=test_dataset, batch_size=32,
shuffle=False)

        criterion = nn.CrossEntropyLoss()
        optimizer = optim.Adam(model.parameters(), lr=learning_rate)
```

```
# Early Stopping Class
class EarlyStopping:
    def __init__(self, patience=5, verbose=False):
        self.patience = patience
        self.verbose = verbose
        self.counter = 0
        self.best_loss = None

    def __call__(self, val_loss, model):
        if self.best_loss is None:
            self.best_loss = val_loss
        elif val_loss > self.best_loss:
            self.counter += 1
            if self.verbose:
                print(f'Validation loss did not improve. Counter:
{self.counter}/{self.patience}')
            if self.counter >= self.patience:
                print("Early stopping!")
                return True
        else:
            self.best_loss = val_loss
            self.counter = 0
        return False

# เริ่มการเทรนโมเดล
train_losses, train_accuracies, val_losses, val_accuracies = [], [], [], []
early_stopping = EarlyStopping(patience=4, verbose=True)

for epoch in range(num_epochs):
    model.train()
```

```
running_loss, correct, total = 0.0, 0, 0

for inputs, labels in train_loader:
    inputs, labels = inputs.to(device), labels.to(device)
    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

    running_loss += loss.item()
    _, predicted = torch.max(outputs.data, 1)
    total += labels.size(0)
    correct += (predicted == labels).sum().item()

epoch_loss = running_loss / len(train_loader)
epoch_accuracy = 100 * correct / total
train_losses.append(epoch_loss)
train_accuracies.append(epoch_accuracy)

# Validation
model.eval()
running_val_loss, val_correct, val_total = 0.0, 0, 0

with torch.no_grad():
    for inputs, labels in val_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        val_loss = criterion(outputs, labels)
        running_val_loss += val_loss.item()
        _, val_predicted = torch.max(outputs.data, 1)
```

```
            val_total += labels.size(0)
            val_correct += (val_predicted == labels).sum().item()

        val_epoch_loss = running_val_loss / len(val_loader)
        val_epoch_accuracy = 100 * val_correct / val_total
        val_losses.append(val_epoch_loss)
        val_accuracies.append(val_epoch_accuracy)

        print(f'Epoch [{epoch+1}/{num_epochs}], '
              f'Train Loss: {epoch_loss:.4f}, Train Accuracy:
{epoch_accuracy:.4f}%, '
              f'Val Loss: {val_epoch_loss:.4f}, Val Accuracy:
{val_epoch_accuracy:.4f}%')

        if early_stopping(val_epoch_loss, model):
            break

    print("Training completed!")

    model.eval()
    test_path =
r'C:\Users\DELL\Desktop\Project4T1\test\chair\LINE_ALBUM_chairrr_241013_60.jpg'
    class_names = ['Chair', 'Door', 'Lift', 'Stair', 'Table']

    image = Image.open(test_path)
    image = transform_val_test(image).unsqueeze(0).to(device)

    with torch.no_grad():
        outputs = model(image)
        probabilities = torch.nn.functional.softmax(outputs[0], dim=0)
```

```
            predicted_class = torch.argmax(probabilities).item()
            predicted_score = probabilities[predicted_class].item()
            print(f"Predicted class: {class_names[predicted_class]} with confidence:
{predicted_score * 100:.8f}%")
```

```
            img = Image.open(test_path)
            plt.imshow(img)
            plt.title(f"Predicted: {class_names[predicted_class]} ({predicted_score *
100:.4f}%)")
            plt.axis('off')
            plt.show()

            # ResNet18
            import torch
            import torch.nn as nn
            import torch.optim as optim
            from torchvision import datasets, transforms, models
            import os
            import matplotlib.pyplot as plt
            import seaborn as sns
            from sklearn.metrics import confusion_matrix, classification_report

            data_transforms = {
                'train': transforms.Compose([
                    transforms.Resize((244, 244)),
                    transforms.RandomHorizontalFlip(),
                    transforms.RandomRotation(10),
                    transforms.RandomResizedCrop(224, scale=(0.8, 1.0)),
                    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2,
hue=0.1),
                    transforms.ToTensor(),
```

```
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ]),
        'val': transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ]),
    }
    data_dir = 'D:/test resnet18/dataset'
    image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
data_transforms[x]) for x in ['train', 'val']}
    dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x],
batch_size=32, shuffle=True, num_workers=4) for x in ['train', 'val']}

    # ResNet-18 model
    model = models.resnet18(pretrained=True)
    num_features = model.fc.in_features
    model.fc = nn.Sequential(
        nn.Flatten(),
        nn.Linear(num_features, 512),
        nn.ReLU(),
        nn.Dropout(p=0.5),
        nn.Linear(512, 128),
        nn.ReLU(),
        nn.Linear(128, 6)
    )

    # Freeze all layers except the final classification layer
    for name, param in model.named_parameters():
        if "fc" in name:  # Unfreeze the final classification layer
            param.requires_grad = True
```

```
    else:
        param.requires_grad = False

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = model.to(device)

train_losses = []
val_losses = []
train_accs = []
val_accs = []

# Early Stopping parameters
patience = 5  # Number of epochs to wait before stopping
best_val_loss = float('inf')
patience_counter = 0

# Training loop
num_epochs = 30
for epoch in range(num_epochs):
    for phase in ['train', 'val']:
        if phase == 'train':
            model.train()
        else:
            model.eval()

        running_loss = 0.0
        running_corrects = 0
        all_preds = []
```

```
all_labels = []

for inputs, labels in dataloaders[phase]:
    inputs = inputs.to(device)
    labels = labels.to(device)
    optimizer.zero_grad()
    with torch.set_grad_enabled(phase == 'train'):
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        loss = criterion(outputs, labels)

        if phase == 'train':
            loss.backward()
            optimizer.step()

    running_loss += loss.item() * inputs.size(0)
    running_corrects += torch.sum(preds == labels.data)

    if phase == 'val':
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

epoch_loss = running_loss / dataset_sizes[phase]
epoch_acc = running_corrects.double() / dataset_sizes[phase]

if phase == 'train':
    train_losses.append(epoch_loss)
    train_accs.append(epoch_acc.item())
else:
    val_losses.append(epoch_loss)
    val_accs.append(epoch_acc.item())
```

```
                if epoch_loss < best_val_loss:
                    best_val_loss = epoch_loss
                    patience_counter = 0
                else:
                    patience_counter += 1

            print(f'{epoch}/{num_epochs} - {phase} Loss: {epoch_loss:.4f} Acc:
{epoch_acc:.4f}')

            if patience_counter >= patience:
                print(f'Early stopping triggered at epoch {epoch+1}')
                break
        else:
            continue
        break
    print("Training complete")

    # Classification Report for precision, recall, F1-score
    print("Classification Report:")
    print(classification_report(all_labels, all_preds,
target_names=class_names))

    # Plotting train loss and val loss
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(range(len(train_losses)), train_losses, label='Train Loss')
    plt.plot(range(len(val_losses)), val_losses, label='Val Loss')
    plt.title('Train and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
```

```
plt.ylim(0, max(max(train_losses), max(val_losses)) + 0.1)
plt.legend()

# Train and Validation Accuracy Plot
plt.subplot(1, 2, 2)
plt.plot(range(len(train_accs)), train_accs, label='Train Accuracy')
plt.plot(range(len(val_accs)), val_accs, label='Val Accuracy')
plt.title('Train and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.legend()
plt.show()

# Plotting Confusion Matrix
class_names = ['chair', 'door', 'downstair', 'lift', 'table', 'upstair']
cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix - Final Validation Set')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Save the model
torch.save(model.state_dict(), 'project_model.pth')
```

ภาคผนวก ค

โปรแกรมสำหรับประเมินประสิทธิภาพโมเดล

```
# CNN
test_data_path = r'C:\Users\DELL\Desktop\Project4T1\test'

model = CNN_FourConvLayers(dropout_rate=0.2311959186265376).to(device)
model.load_state_dict(torch.load('model_four_weight.pth'))
model.eval()

# การแปลงภาพสำหรับชุดทดสอบ
test_transform = transforms.Compose([
    transforms.Resize((224, 224)),  # ขนาดภาพที่โมเดลต้องการ
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])  # Normalize
])

# โหลดชุดทดสอบ
test_dataset = datasets.ImageFolder(root=test_data_path, transform=test_transform)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
class_names_test = test_dataset.classes

# ประเมินผลโมเดล
all_labels = []
all_preds = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
```

```
            all_labels.extend(labels.cpu().numpy())
            all_preds.extend(preds.cpu().numpy())

        #Classification Report
        print("Accuracy of the model on the test set:")
        print(classification_report(all_labels, all_preds,
target_names=class_names_test))
        accuracy = np.mean(np.array(all_preds) == np.array(all_labels)) * 100
        print(f'Overall Accuracy: {accuracy:.2f}%')

        # Confusion Matrix
        cm = confusion_matrix(all_labels, all_preds)
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names_test, yticklabels=class_names_test)
        plt.title('Confusion Matrix')
        plt.xlabel('Predicted')
        plt.ylabel('True')
        plt.show()

        # ResNet18
        def evaluate_model(model, test_loader, device):
            model.eval()
            all_preds = []
            all_labels = []
            with torch.no_grad():
                for images, labels in test_loader:
                    images, labels = images.to(device), labels.to(device)
                    outputs = model(images)
                    _, preds = torch.max(outputs, 1)
                    all_preds.extend(preds.cpu().numpy())
```

```
                all_labels.extend(labels.cpu().numpy())

            # Classification report
            class_names = test_dataset.classes
            report = classification_report(all_labels, all_preds,
target_names=class_names)
            print("Classification Report:\n", report)

            cm = confusion_matrix(all_labels, all_preds)
            plt.figure(figsize=(8, 6))
            sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
            plt.xlabel('Predicted')
            plt.ylabel('True')
            plt.title('Confusion Matrix')
            plt.show()

        # โหลดโมเดล ResNet18
        model = models.resnet18(pretrained=True)
        num_features = model.fc.in_features
        model.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(num_features, 512),
            nn.ReLU(),
            nn.Dropout(p=0.5),
            nn.Linear(512, 128),
            nn.ReLU(),
            nn.Linear(128, 6)
        )
        state_dict = torch.load('project_model.pth')
        model.load_state_dict(state_dict)
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

test_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

test_dataset = datasets.ImageFolder(root='D:/test resnet18/dataset/test',
transform=test_transforms)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
evaluate_model(model, test_loader, device)
```

ภาคผนวก ง

โปรแกรมการทำงานของอุปกรณ์

```
import torch
import torch.nn as nn
from torchvision import models, transforms
import cv2
import numpy as np
from PIL import Image
import os
import time
import RPi.GPIO as io

io.setmode(io.BCM)
io.setup(14, io.OUT)
io.setup(15, io.IN)

# Define the class names and corresponding sound files
class_names = ["chair", "door", "downstair", "lift", "table", "upstair"]

# Load the trained model
model = models.resnet18(pretrained=False)
num_features = model.fc.in_features
model.fc = nn.Sequential(
    nn.Flatten(),
    nn.Linear(num_features, 512),
    nn.ReLU(),
    nn.Dropout(p=0.5),
    nn.Linear(512, 128),
    nn.ReLU(),
    nn.Linear(128, len(class_names)),
)
```

```
        # Load the pre-trained model's state_dict
        model.load_state_dict(torch.load("project_model.pth",
map_location=torch.device("cpu")))

        # Move model to the appropriate device (CPU or GPU)
        device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
        model = model.to(device)
        model.eval()

        # Define image transformations for preprocessing
        data_transforms = transforms.Compose(
            [
                transforms.Resize((224, 224)),
                transforms.ToTensor(),
                transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
            ]
        )

        # Image processing functions
        def adjust_brightness(image, factor):
            hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
            h, s, v = cv2.split(hsv)
            v = cv2.multiply(v, factor)
            v = np.clip(v, 0, 255).astype(hsv.dtype)
            hsv = cv2.merge([h, s, v])
            return cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)


        def sharpen_image(image):
            kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
            return cv2.filter2D(image, -1, kernel)
```

```
# Initialize the webcam
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break

    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    gray_image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    mean_brightness = np.mean(gray_image)

    # Adjust brightness based on mean brightness level
    if mean_brightness > 200:
        frame = adjust_brightness(frame, 0.8)
    elif mean_brightness < 130:
        frame = adjust_brightness(frame, 1.5)

    # Sharpen the image
    frame = sharpen_image(frame)

    # Convert image to PIL format for transformation
    pil_image = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    input_tensor = data_transforms(pil_image).unsqueeze(0).to(device)
```

```
                start = time.time()

                # Run the image through the model
                with torch.no_grad():
                    outputs = model(input_tensor)

                end = time.time()
                ptime = end - start

                # Apply softmax to get class probabilities
                probabilities = torch.nn.functional.softmax(outputs, dim=1)
                confidence, preds = torch.max(probabilities, 1)
                predicted_class = class_names[preds[0].item()]

                # ultrasonic

                io.output(14, 1)
                time.sleep(0.00001)
                io.output(14, 0)
                trig = 0
                while (io.input(15) == 0):
                    start = time.time()
                while (io.input(15) == 1):
                    stop = time.time()
                ult = stop - start
                distance = ((ult * 343) / 2)
                #time.sleep(0.5)

                # Display the predicted class and confidence on the frame if
confidence > 0.8
                if confidence.item() > 0.8:
```

```
            text = "" #TTS
            d = ""
            cv2.putText(
                frame,
                f"Predicted: {predicted_class}",
                (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX,
                1,
                (0, 255, 0),
                2,
                cv2.LINE_AA,
            )
            s = f"Predicted: {predicted_class}, Confidence: {confidence.item():.2f},
Time: {ptime:.4f}"
            print(s)
            if 1.9 < distance < 2.1:
                d = f"2 meters"
            if 0.9 < distance < 1.1:
                d = f"1 meter"
            if 2.9 < distance < 3.1:
                d = f"3 meters"
            if d != "":
                text = f"{predicted_class} in {d}"
                os.system(f'espeak "{text}" -v en')
                print(f"Speaking {text}")

        # Display the resulting frame
        cv2.imshow("Webcam Object Classification", frame)

        # Break the loop with 'q'
        if cv2.waitKey(1) & 0xFF == ord("q"):
```

```
        break

    # Release the capture and close windows
    cap.release()
    cv2.destroyAllWindows()
```