การตรวจจับความเสียหายของรถยนต์อัตโนมัติโดยวิธีเรียนรู้ของเครื่อง

AUTOMATED CAR DAMAGE DETECTION USING MACHINE LEARNING

ธุวานนท์ พัฒนนวพงศ์

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2567

KMITL-2024-SC-M-017-067

AUTOMATED CAR DAMAGE DETECTION USING MACHINE LEARNING

THUWANON PATTANANAWAPONG

AN INDEPENDENT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN DATA SCIENCE AND ANALYTICS
KMITL DIGITAL ANALYTICS AND INTELLIGENCE CENTER SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2024

KMITL-2024-SC-M-017-067

หัวข้อการค้นคว้าอิสระ การตรวจจับความเสียหายของรถยนต์อัตโนมัติโดยวิธีเรียนรู้ของเครื่อง
ชื่อนักศึกษา นายธุวานนท์ พัฒนนวพงศ์
รหัสประจำตัว 65056048
ปริญญา วิทยาศาสตรมหาบัณฑิต (วิทยาการข้อมูลและการวิเคราะห์)
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง
พ.ศ. 2567
อาจารย์ที่ปรึกษาการค้นคว้าอิสระ ผู้ช่วยศาสตราจารย์.ดร. จิรภัทร์ หยกรัตนศักดิ์

บทคัดย่อ

อุตสาหกรรมประกันภัยรถยนต์ในประเทศไทยมีการแข่งขันสูง เนื่องจากเบี้ยประกันภัยรถยนต์คิดเป็น 56.48% ของเบี้ยประกันภัยวินาศภัยในปี 2565 ซึ่งกระบวนการเคลมมักใช้เวลานาน การใช้ Machine Learning โดยเฉพาะ Convolutional Neural Networks (CNN) และเทคนิค Transfer Learning จากโมเดล ImageNet เช่น VGG16, ResNet50 และ InceptionV3 สามารถช่วยจำแนกภาพรถยนต์ที่เสียหายและไม่เสียหายได้อย่างมีประสิทธิภาพ การทดลองนี้เปรียบเทียบประสิทธิภาพของโมเดลด้วยการปรับแต่ง Custom Head และ Fine Tune โดยใช้ข้อมูลที่ผ่านและไม่ผ่านการทำ Data Augmentation

ผลการทดลองพบว่าโมเดล VGG16 ที่ปรับแต่งด้วย Fine Tune และใช้ชุดข้อมูลที่ไม่แปลงข้อมูลต้นฉบับ พร้อม Custom Head แบบ Average Pooling ให้ค่า Accuracy เท่ากับ 0.8789 Precision เท่ากับ 0.8955 Recall เท่ากับ 0.8780 และ F1-Score เท่ากับ 0.8867 และยังแสดงผลการเรียนรู้ที่ดีจากการวิเคราะห์ประสิทธิภาพจากกราฟ Learning Curve ในการวัดค่า Loss และ Accuracy โดยมีค่า Training Loss และ Validation Loss ลดลงอย่างต่อเนื่อง และในทางตรงกันข้ามค่า Training Accuracy และ Validation Accuracy เพิ่มอย่างต่อเนื่องเช่นกัน ซึ่งแสดงถึงการเรียนรู้ที่ดีและความเหมาะสม ในการใช้จำแนกความเสียหายจากรูปภาพของรถยนต์ในกระบวนการเคลมประกันภัย

คำสำคัญ : การเรียนรู้ของเครื่อง การประมวลผลภาพ ประกันภัยรถยนต์

Independent Study Title	Automated Car Damage Detection using Machine Learning
Student Name	Mr. Thuwanon Pattananawapong
Student ID	65056048
Degree	Master of Science (Data Science and Analytics)
	KMITL Digital Analytics and Intelligence Center
Year	2024
Independent Study Advisor	Asst. Prof. Dr. Jiraphat Yokrattanasak

Abstract

The automobile insurance industry in Thailand is highly competitive, with motor insurance premiums accounting for 56.48% of all non-life insurance premiums in 2022. The claims process for automobile insurance is often time-consuming. Using machine learning, especially convolutional neural networks (CNNs) with transfer learning techniques from ImageNet models such as VGG16, ResNet50, and InceptionV3, can effectively classify car images as damaged or undamaged. This study compares the performance of these models by employing custom head adjustments and Fine Tune, using with Data Augmentation and without Data Augmentation.

The results show that the VGG16 model with Fine Tune and the original dataset without data augmentation along with an average pooling custom head, achieved an accuracy of 0.8789, a precision of 0.8955, a recall of 0.8780, and an F1-score of 0.8867. Additionally, the model demonstrated good learning performance, as analyzed from the Learning Curve graphs measuring Loss and Accuracy. The Training Loss and Validation Loss consistently decreased, while the Training Accuracy and Validation Accuracy steadily increased. This indicates effective learning and suitability for classifying vehicle damage from images in the insurance claim process.

Keywords : Machine Learning, Image Processing, Motor Insurance

กิตติกรรมประกาศ

การศึกษาค้นคว้าอิสระในครั้งนี้สำเร็จลุล่วงไปด้วยดี ด้วยการสนับสนุนจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทั้งนี้ผู้จัดทำขอขอบพระคุณ ผศ.ดร. จิรภัทร์ หยกรัตนศักดิ์ อาจารย์ที่ปรึกษาของการศึกษาค้นคว้าอิสระครั้งนี้ที่ให้คำปรึกษา ชี้แนะแนะนำแนวทางในการทดลอง พร้อมทั้งให้เรียนรู้ และแก้ไขปัญหาวิธีการหรือข้อบกพร่องต่าง ๆ ในขณะศึกษาและทดลองครั้งนี้ รวมถึงให้คำปรึกษาในด้านอื่นๆ และเป็นผลทำให้การศึกษาค้นคว้าอิสระครั้งนี้สำเร็จลุล่วงด้วยดี

ผู้จัดทำขอขอบพระคุณคณาจารย์ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ให้คำแนะนำปรึกษาอันเป็นผลประโยชน์ในการพัฒนาและแก้ไขเนื้อหาฉบับนี้ให้สมบูรณ์ขึ้น ทางผู้จัดทำขอขอบพระคุณเป็นอย่างสูง ณ โอกาสนี้

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา และครอบครัวที่ให้กำลังใจ ให้การสนับสนุนส่งเสริมในการศึกษามาโดยตลอด รวมถึงเพื่อน ๆ ในสาขาวิทยาการข้อมูลและการวิเคราะห์ทุกท่านที่ให้กำลังใจ และสนับสนุน ในด้านการศึกษาและทดลอง ตลอดจนเป็นผลให้ประสบความสำเร็จลุล่วงไปด้วยดี ผู้จัดทำขอขอบพระคุณทุกท่านมา ณ โอกาสนี้

นายธุวานนท์ พัฒนนวพงศ์

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

เป้าหมายในการศึกษาค้นคว้าอิสระครั้งนี้เพื่อศึกษาแนวทางและพัฒนาการตรวจจับความเสียหายของรถยนต์ด้วยวิธีการเรียนรู้ของเครื่อง (Machine Learning) โดยจะกล่าวถึงความสำคัญและที่มาของการศึกษาค้นคว้าอิสระ วัตถุประสงค์ของการศึกษาค้นคว้าอิสระ ขอบเขตของการศึกษา ประโยชน์ที่คาดว่าจะได้รับ และนิยามคำศัพท์เฉพาะ

1.1 ความสำคัญและที่มาของการศึกษาค้นคว้าอิสระ

ปัจจุบันในประเทศไทยอุตสาหกรรมประกันภัยเป็นอุตสาหกรรมที่มีการแข่งขันกันสูง การประกันภัยมีการแบ่งประเภทตามประมวลกฎหมายแพ่งและพาณิชย์ แบ่งเป็น 2 ประเภท คือ การประกันชีวิต (Life Insurance) และ การประกันวินาศภัย (Non-Life Insurance) โดยธุรกิจการประกันภัยทั้ง 2 ประเภท มีหลักเกณฑ์และมีการกำกับดูแลภายใต้กฎหมายที่แตกต่างกัน โดยส่วนของประกันภัยรถยนต์ (Motor insurance) อยู่ในประเภทประกันวินาศภัย และปี พ.ศ.2565 มีเบี้ยประกันวินาศภัยสะสม ทั้งสิ้นอยู่ที่ประมาณ 274,218,000,000 บาท สัดส่วนของประกันภัยรถยนต์ (Motor Insurance) อยู่ที่ 56.48% หรือมีเบี้ยประมาณ 154,886,000,000 บาท (สำนักงานคณะกรรมการกำกับและส่งเสริมการประกอบธุรกิจประกันภัยและสำนักงานสภาพัฒนาการเศรษฐกิจและสังคมแห่งชาติ, 2565) ซึ่งจะเห็นได้ว่าประกันภัยรถยนต์มีความสำคัญอย่างยิ่งในธุรกิจประกันภัย

เนื่องจากประกันภัยรถยนต์มีเบี้ยที่มากบอกถึงจำนวนที่บริษัทประกันภัยต้องรับผิดชอบในกรณีมี เคลมประกันภัยเกิดขึ้นก็มีโอกาสมากขึ้นไปด้วย บริษัทประกันภัยจะสูญเสียเงินในส่วนของเคลมที่มีการจ่ายเคลมเกินกว่ายอดการเคลมที่แท้จริงในปริมาณมากมายมหาศาล และในการแจ้งเคลมในปัจจุบันส่วนใหญ่ลูกค้าต้องแจ้งเคลมประกันภัยผ่านทางโทรศัพท์ ซึ่งตามมาด้วยกระบวนการที่เจ้าหน้าที่ต้องเดินทางไปที่เกิดเหตุเพื่อเก็บหลักฐานและถ่ายภาพความเสียหายของรถยนต์ กระบวนการนี้ใช้เวลานานและต้องดำเนินการกับเอกสารหลายอย่าง และใช้เวลาในการพิจารณาเคลม ซึ่งทำให้เสียเวลาเป็นอย่างมาก ดังนั้นการมีเครื่องมือที่ช่วยจำแนกความเสียหายอาจลดเวลาและภาระงานของเจ้าหน้าที่ งานวิจัยนี้เน้นการจำแนกความเสียหายจากรูปภาพเพื่อลดขั้นตอนการลงพื้นที่และลดกระบวนการอื่นๆที่เกี่ยวข้อง เพื่อให้มีความแม่นยำในการวิเคราะห์จำแนกความเสียหาย บริษัทประกันภัยจึงมองหาการแก้ปัญหาด้วยนำเทคโนโลยีจากการเรียนรู้ของเครื่อง (Machine Learning) มาช่วยแก้ปัญหาดังกล่าว

ดังนั้นในการศึกษาค้นคว้าอิสระนี้จึงมีแนวคิดที่จะศึกษา และพัฒนาการตรวจจับความเสียหายของรถยนต์ด้วยวิธีการเรียนรู้ของเครื่อง (Machine Learning) สำหรับจำแนกความเสียหายของรถยนต์ เพื่อนำไปเพิ่มประสิทธิภาพในกระบวนการทำงานของพนักงาน

1.2 วัตถุประสงค์ของการศึกษาค้นคว้าอิสระ

วัตถุประสงค์ของการศึกษาค้นคว้าอิสระครั้งนี้ มีวัตถุประสงค์ ดังนี้

1) เพื่อให้สามารถจำแนกรูปภาพรถยนต์ที่มีความเสียหาย และที่ไม่มีความเสียหายโดยใช้วิธีการเรียนรู้ของเครื่อง (Machine Learning)

2) เพื่อสามารถตรวจสอบความเสียหายของรถยนต์อัตโนมัติได้

3) เพื่อนำไปประยุกต์ใช้และเพิ่มประสิทธิภาพในการทำงานในองค์กร

1.2 ขอบเขตของการศึกษา

วัตถุประสงค์ของการศึกษาค้นคว้าอิสระครั้งนี้ มีขอบเขตของการศึกษา ดังนี้

1) ด้านข้อมูลในการศึกษา : ภาพถ่ายของรถยนต์ที่เกิดความเสียหาย โดยภาพถ่ายได้รับการจำแนกตามประเภทดังนี้

1.1) ภาพถ่ายของรถที่มีการเกิดความเสียหาย จำนวน 979 รูป

1.2) ภาพถ่ายของรถที่ไม่มีการเกิดความเสียหาย จำนวน 919 รูป

ข้อมูลได้รับจากเว็บไซต์ที่มา : https://github.com/rehmanzafar/DetectDamage/blob/master/README.md

2) ด้านเนื้อหาในการศึกษา : ศึกษาและพัฒนาเกี่ยวกับแบบจำลองโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) สำหรับจำแนกความเสียหายของรถยนต์ เพื่อนำไปลดภาระและเพิ่มประสิทธิภาพในกระบวนการทำงานของพนักงานประกันภัย

1.3 ประโยชน์ที่คาดว่าจะได้รับ

วัตถุประสงค์ของการศึกษาค้นคว้าอิสระครั้งนี้ มีประโยชน์ที่คาดว่าจะได้รับ ดังนี้

1) แบบจำลองสามารถนำไปใช้ตรวจสอบและจำแนกความเสียหายจากภาพถ่ายของรถยนต์ได้ว่าเกิดความเสียหายหรือไม่

2) แบบจำลองสามารถช่วยลดภาระและเพิ่มประสิทธิภาพในกระบวนการทำงานของพนักงานประกันภัยได้

3) ลูกค้าประกันภัยได้รับการบริการเคลมได้รวดเร็วมากยิ่งขึ้น

1.4 นิยามศัพท์เฉพาะ

วัตถุประสงค์ของการศึกษาค้นคว้าอิสระครั้งนี้ มีนิยามศัพท์เฉพาะ ดังนี้

1) การเรียนรู้ของเครื่อง (Machine Learning) หมายถึง ระบบที่สามารถเรียนรู้จากตัวอย่างได้เองโดยปราศจากการป้อนคำสั่งของมนุษย์ ด้วยข้อมูลและเครื่องมือทางสถิติเพื่อทำนายผลลัพธ์ออกมา ในงานค้นคว้าอิสระนี้จัดอยู่ในประเภท การเรียนรู้แบบมีผู้สอน (Supervised Learning)

2) การเรียนรู้เชิงลึก (Deep Learning) หมายถึง การเรียนรู้ของคอมพิวเตอร์ที่มีหลักการมาจากการทำงาน ของสมองมนุษย์ โดยจะเป็นการรับข้อมูลเข้ามา แล้วประมวลผลซ้ำๆ ในรูปแบบต่างๆ จำนวนมาก เพื่อทำการหาจุดที่จะบ่งบอกคุณลักษณะของข้อมูลที่นำเข้าได้ และแสดงผลลัพธ์ออกมาเป็นกลุ่มต่างๆ ว่าข้อมูลที่ใส่เข้าไปนั้นเป็นข้อมูลที่อยู่ในกลุ่มไหน เช่น การใส่รูป สุนัข เข้าไป เมื่อทำการประมวลผล แล้วผลลัพธ์จะออกมาเป็น สุนัข หรือ สัตว์อื่น เป็นต้น

3) ปัญญาประดิษฐ์ (Artificial Intelligence) หมายถึง เป็นศาสตร์แขนงหนึ่งของวิทยาศาสตร์คอมพิวเตอร์ที่เกี่ยวข้องกับวิธีการทำให้คอมพิวเตอร์มีความสามารถ คล้ายมนุษย์หรือเลียนแบบพฤติกรรมมนุษย์คือโปรแกรมซอฟแวร์ (Software) ต่างๆที่ใช้กับคอมพิวเตอร์ โดยเฉพาะความสามารถในการคิดเองได้ หรือมีปัญญา โดยปัญญาเกอดจากมนุษย์เป็นผู้สร้างให้คอมพิวเตอร์ จึง เรียกว่า ปัญญาประดิษฐ์หรือเรียกว่า AI

4) โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) หมายถึง โมเดลประสาทเทียมที่ถูกออกแบบมาเพื่อการจำแนกและสกัดลักษณะจากรูปภาพ โครงข่ายนี้มักถูกนำมาใช้ในงานทางด้านการมองเห็นคอมพิวเตอร์ เนื่องจากความสามารถในการจัดการกับข้อมูลภาพอย่างมีประสิทธิภาพ โดยมีนิยามสำคัญต่างๆดังนี้

4.1) Convolutional Layers (ชั้นคอนโวลูชัน) : ใช้การสัมผัสเพื่อสกัดลักษณะจากรูปภาพโดยให้ตัวกรอง (filter) เคลื่อนที่ทั่วภาพ

4.2) Pooling Layers (ชั้นพูลลิ่ง) : ใช้เพื่อลดขนาดของลักษณะที่ถูกสกัด, ลดการคำนวณ, และเพิ่มความทนทานต่อการเปลี่ยนแปลงขนาด

4.3) Fully Connected Layers (ชั้นเชื่อมต่อเต็มรูปแบบ) : ชั้นที่เป็นประสาทเทียมเต็มรูปแบบ, ทำหน้าที่ในการจัดการกับลักษณะที่ถูกสกัด

4.4) Activation Functions (ฟังก์ชันกระตุ้น) : เพิ่มความซับซ้อนในโมเดลและสามารถเรียนรู้ลักษณะที่ซับซ้อนได้

4.5) Weight Sharing (การแชร์น้ำหนัก) : ใช้ตัวกรองเดียวกันที่ทุกตำแหน่งของภาพเพื่อลดจำนวนพารามิเตอร์และทำให้โมเดลทนทานต่อการเรียนรู้

โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) นั้นได้รับความนิยมในการแก้ปัญหาทางด้านการมองเห็น เช่น การจำแนกวัตถุ, การตรวจจับวัตถุ, และการทำนายหมวดหมู่ของภาพ การใช้โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) ช่วยลดความซับซ้อนในการดึงเอาลักษณะอย่างมีประสิทธิภาพจากรูปภาพ ทำให้เหมาะสำหรับการประยุกต์ใช้ในงานวิจัยทางศาสตร์และการพัฒนาเทคโนโลยีที่เกี่ยวข้องกับการมองเห็นคอมพิวเตอร์

5) การดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) หมายถึง เป็นกระบวนการที่ใช้เพื่อสร้างข้อมูลเพิ่มเติมจากข้อมูลที่มีอยู่แล้วโดยการทำการปรับแต่งหรือเปลี่ยนแปลงลักษณะของข้อมูล เพื่อเพิ่มความหลากหลายและประสิทธิภาพในการฝึกโมเดลประสาทเทียมหรือวิธีการ

ประมวลผลข้อมูลอื่นๆ ที่มีความเกี่ยวข้อง เช่น การหมุนภาพ (Rotation) เพื่อสร้างภาพที่ถูกหมุนไปที่มุมต่างๆ เพื่อทำให้โมเดลรู้จักลักษณะจากรูปภาพที่หมุน การเทรนสลับภาพ (Flipping) ในทิศทางต่างๆ แนวนอนหรือแนวตั้ง เพื่อเพิ่มความหลากหลายของข้อมูล การย่อหรือขยาย (Scaling) เป็นการปรับขนาดข้อมูลต้นฉบับในลักษณะขยายหรือย่อเพื่อสร้างข้อมูลที่มีขนาดแตกต่าง การตัดภาพ (Cropping) เพื่อให้ได้ข้อมูลที่มีส่วนที่สนใจ การเปลี่ยนแปลงลักษณะสีของภาพ (Color Transformation) การดัดแปลงข้อมูลต้นฉบับมีประโยชน์มากในการปรับปรุงความสามารถในการฝึกโมเดลโดยทำให้โมเดลมีความทนทานต่อข้อมูลที่ไม่เหมือนกัน และลดโอกาสการเกิดการต่อ Overfitting ในข้อมูลที่มีจำนวนน้อย

6) การเรียนรู้แบบถ่ายโอน (Transfer Learning) หมายถึง เป็นกระบวนการในการนำความรู้ที่ได้จากการฝึกโมเดลในงานหนึ่งมาใช้ในงานอื่นๆ โดยโมเดลที่ถูกฝึกนั้นมักเรียกว่า โมเดลที่ถูกฝึกไว้ล่วงหน้า (Pre-Trained Model) กระบวนการนี้มีประโยชน์มากในกรณีที่ข้อมูลฝึกมีขนาดใหญ่และมีความซับซ้อน และใช้ในงานที่ข้อมูลมีจำนวนน้อยหรือทำงานในสาขาที่คล้ายคลึงกันนิยามของ Transfer Learning รวมถึง

6.1) โมเดลที่ถูกฝึกไว้ล่วงหน้า (Pre-trained Model) : โมเดลที่ถูกฝึกเตรียมไว้ในงานหนึ่ง โดยมักจะใช้ข้อมูลขนาดใหญ่และมีโครงสร้างที่ซับซ้อน

6.2) การถ่ายโอนรูปแบบ (Knowledge Transfer) : การนำความรู้ที่ได้จากการฝึกโมเดลในงานหนึ่งมาใช้ในงานอื่นๆ โดยใช้โมเดลที่ถูกฝึกไว้ล่วงหน้า

6.3) การปรับแต่ง (Fine-tuning) : กระบวนการปรับแต่งพารามิเตอร์ของโมเดลที่ถูกฝึกไว้ล่วงหน้าเพื่อให้เข้ากับงานหรือข้อมูลใหม่

6.4) Transfer Learning แบบแชร์แบบหลายงาน (Multi-task Transfer Learning) : การนำความรู้ที่ได้มาใช้ในหลายงานต่างๆ ที่มีความเกี่ยวข้องกัน

การเรียนรู้แบบถ่ายโอน (Transfer Learning) เป็นเทคนิคที่มีความสามารถในการลดเวลาและทรัพยากรในการฝึกโมเดลใหม่, ช่วยให้โมเดลที่ถูกฝึกไว้ล่วงหน้ามีความสามารถในการทำนายสูง โดยไม่ต้องใช้ข้อมูลฝึกจำนวนมากในงานที่มีข้อมูลมีจำนวนน้อยหรือทำงานในสาขาที่คล้ายคลึงกัน

บทที่ 2
แนวคิด ทฤษฎีที่เกี่ยวข้อง

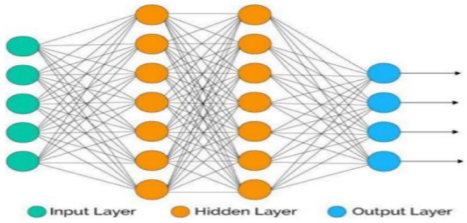
การศึกษาค้นคว้าอิสระครั้งนี้ผู้วิจัยได้ศึกษาค้นคว้าข้อมูลจากบทความ เอกสารทางวิชาการ และวิจัยที่เกี่ยวข้องกับการตรวจจับความเสียหายของรถยนต์ด้วยวิธีการเรียนรู้ของเครื่อง (Machine Learning) โดยผู้วิจัยได้ศึกษาแนวคิดดังต่อไปนี้

2.1 การเรียนรู้เชิงลึก (Deep Learning)
2.2 โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network)
2.3 การเรียนรู้แบบถ่ายโอน (Transfer Learning)
2.4 การดัดแปลงข้อมูลต้นฉบับ (Data Augmentation)
2.5 การปรับแต่ง Fine Tune
2.6 ประเมินประสิทธิภาพแบบจำลอง
2.7 งานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้เชิงลึก (Deep Learning)

การเรียนรู้เชิงลึกหรือ Deep Learning คือกระบวนการที่เลียนแบบการทำงานของเครือข่ายประสาทในสมองมนุษย์ โดยใช้เครือข่ายประสาทเทียมหลายชั้นที่เรียกว่า Neural Networks โครงสร้างพื้นฐานประกอบด้วยชั้นการรับข้อมูล (input layer) ชั้นซ่อน (hidden layers) หนึ่งชั้นหรือมากกว่า และชั้นผลลัพธ์ (output layer) การมีชั้นซ่อนมากกว่าสองชั้นในเครือข่ายถือเป็นเกณฑ์หนึ่งของ Deep Learning ทั้งนี้ การเพิ่มชั้นซ่อนทำให้เครือข่ายสามารถสกัดคุณลักษณะ (Feature Extraction) ที่ซับซ้อนและมีความละเอียดสูงขึ้นจากข้อมูลที่รับเข้า

Deep learning มีความสามารถพิเศษในการให้คอมพิวเตอร์เรียนรู้และเข้าใจข้อมูลที่ได้รับผ่านสถาปัตยกรรมของเครือข่ายประสาทเทียมที่หลากหลาย โดยเฉพาะอย่างยิ่ง Convolutional Neural Networks ถูกออกแบบมาเพื่อการสกัดคุณลักษณะเด่นจากภาพถ่าย โดยกระบวนการ feed-forward ใน Convolutional Neural Networks ประกอบด้วย Convolutional layers ที่ใช้ฟังก์ชัน kernel เพื่อทำการฟิลเตอร์และแปลงคุณลักษณะที่สำคัญ เช่น ขอบภาพ, สี, และรูปทรง ข้อมูลที่ผ่านการแปลงนี้จะถูกส่งผ่านฟังก์ชันการกระตุ้น (activation function) เพื่อปรับให้เหมาะสมกับประมวลผลต่อไป และตามมาด้วยชั้นพูลลิ่งที่ทำหน้าที่ลดขนาดข้อมูลโดยยังคงรักษารายละเอียดไว้ ในขั้นตอนสุดท้าย, Fully Connected Layers จะทำหน้าที่เชื่อมต่อข้อมูลจากทุกชั้นเข้าด้วยกัน เพื่อให้ได้ผลลัพธ์สุดท้าย



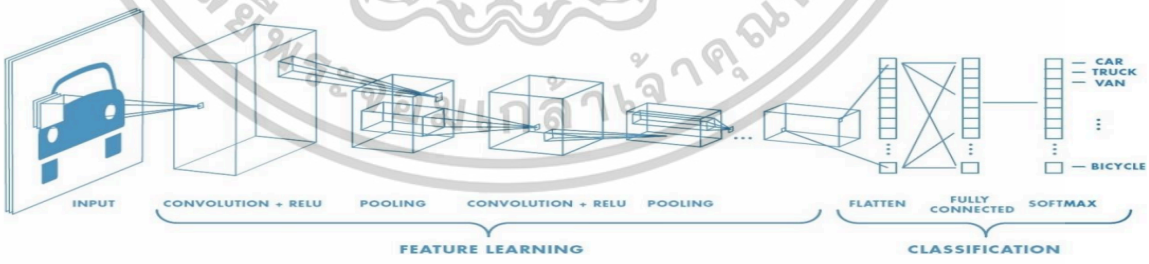
รูปที่ 2.1 แสดงส่วนประกอบของโครงสร้างประสาทเทียม

ที่มา : https://csit.nu.ac.th/kraisak/ds/ds/chapter07/Chapter07.pdf

2.2 โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network)

โครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network) เป็นโครงข่ายประสาทเทียมเชิงลึก (Deep Neural Network) ที่ประกอบด้วยเซ็ตของฟิลเตอร์ (Filter) ซึ่งฟิลเตอร์เหล่านี้ ในจะใช้เพื่อทำคอนโวลูชันกับอินพุต และในระหว่างขั้นตอนการฝึกสอน (Train Neural Network) ฟิลเตอร์เหล่านี้ก็จะถูกปรับค่าเพื่อให้ได้ฟิลเตอร์ที่เหมาะสมที่สุด โดยมีจุดประสงค์คือการสกัดคุณลักษณะจากภาพ (Feature Extraction)

โครงข่ายประสาทเทียมแบบคอนโวลูชันได้รับความนิยมอย่างมากหลังจาก Krizhevsky ได้นำเสนอ AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) ซึ่งสามารถเอาชนะในการแข่งขัน ImageNet Large Scale Visual Recognition Challenge (ILSVRC) โดยสามารถทำคะแนนได้สูงกว่าเทคนิคแบบดั้งเดิม และ CNN Based Architecture ยังถูกพัฒนาอย่างต่อเนื่องเห็นได้จาก Top 5 error rate ของชุดข้อมูล ImageNet ลดจาก 25% จนเหลือ 2.25% ภายใน 5 ปี หลังจาก AlexNet ถูกนำเสนอ ซึ่ง error rate ที่ 2.5 % นั้นถือว่าน้อยกว่า human error อีกด้วย ปัจจุบัน CNN ถูกประยุกต์ไปใช้ในคอมพิวเตอร์วิทัศน์ (Computer vision) ไม่ว่าจะเป็นการทำ Image Classification, Object Detection, Image and Video Recognition ฯลฯ



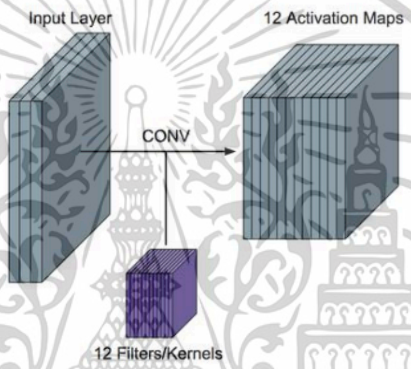
รูปที่ 2.2 แสดงโครงสร้าง Convolution Neural Network

ที่มา : https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

จากรูปที่ 2.2 ได้แสดงโครงสร้างของโครงข่ายประสาทเทียมแบบคอนโวลูชัน โดยเริ่มต้นที่ชั้นข้อมูลเข้า (Input Layer) จะรับข้อมูลเข้าเป็นรูปภาพเพื่อเข้าสู่การดำเนินการที่ชั้นต่างๆ โดยสามารถจัดประเภทของชั้นต่างๆ ได้เป็นชั้นคอนโวลูชัน (Convolution Layer) , ชั้นพูลลิ่ง (Pooling Layer) และชั้นเชื่อมโยงโดยสมบูรณ์ (Fully-Connected Layer)

2.2.1 ชั้นคอนโวลูชัน (Convolution Layer)

ชั้นคอนโวลูชัน (Convolution Layer) มีไว้เพื่อทำการคอนโวลูชันระหว่างอินพุต (Input) กับฟิลเตอร์/เคอร์เนล (Filter/Kernel) ผลลัพธ์ที่ได้คือฟีเจอร์แมพ/แอคทิเวชันแมพ (Feature Map/Activation Map) ดังที่แสดงในรูปที่ 2.3 โดยที่ฟิลเตอร์จะมีการสุ่มและปรับเปลี่ยนในขั้นตอนการเรียนรู้



รูปที่ 2.3 แสดงชั้นคอนโวลูชัน (Convolution Layer)
ที่มา : Khan, S., Rahmani, H., Shah, S.A.A., Bennamoun, M., Medioni, G., & Dickinson, S. (2018). A Guide to Convolutional Neural Networks for Computer Vision. from https://ieeexplore.ieee.org/document/8295029

ขนาดอินพุต ($W_{input} \times H_{input} \times D_{input}$) จะถูกกำหนดโดยความกว้าง , ความสูงและความลึก โดยปกติแล้วขนาดความกว้างและความสูงจะมีขนาดเท่ากันที่ชั้นอินพุต (Input Layer) ความลึกของอินพุตจะมาจากจำนวน Color depth เช่น หากเป็นภาพสี RGB ความลึก (D_{input}) จะเท่ากับ 3 และในชั้นที่ลึกขึ้น (Hidden Layer) ความลึก (D_{input}) จะเท่ากับจำนวนฟิลเตอร์ที่ใช้ในชั้นก่อนหน้าที่ชั้นคอนโวลูชัน (Convolution Layer) จะมีพารามิเตอร์สำหรับฟิลเตอร์ดังต่อไปนี้

จำนวนฟิลเตอร์ (K) จำนวนฟิลเตอร์ที่ใช้จะส่งผลต่อความลึกของเอาต์พุต (Feature Map/Activation Map) ดังที่แสดงในภาพประกอบที่ 2 ความลึกของเอาต์พุต (12 Activation map) จะเท่ากับจำนวนฟิลเตอร์ที่ใช้ (12 Filter)

ขนาดฟิลเตอร์ ($F \times F$) เป็นการกำหนดขนาดความกว้างและความสูงของฟิลเตอร์ เช่น ฟิลเตอร์ขนาด 3x3, 5x5, 7x7, 11x11

Stride (*S*) ฟิลเตอร์จะมีการขยับเพื่อดำเนินการคอนโวลูชันกับอินพุต ดังนั้นจะต้องมีการกำหนดว่าขนาดของ Stride เพื่อกำหนดการขยับของฟิลเตอร์

Zero-Padding (*P*) คือ การกำหนดว่าให้มีการเติม 0 เข้าไปที่บริเวณขอบของอินพุตหรือไม่ ปกติแล้วถ้าไม่มีการทำ padding ผลที่ได้จากการคอนโวลูชันจะทำให้ได้ฟีเจอร์แมพที่มีขนาดเล็กลงกว่าขนาดอินพุต จึงมีทางเลือกในการทำ padding ในกรณีที่ต้องการรักษาขนาดของฟีเจอร์แมพไม่ให้หดลงและจำนวน padding จะหาได้จากสมการที่ 1 เมื่อต้องการรักษาขนาดของเอาต์พุตให้เท่ากับขนาดของอินพุต

$$\mathrm{p} = \frac{f-1}{2} \tag{1}$$

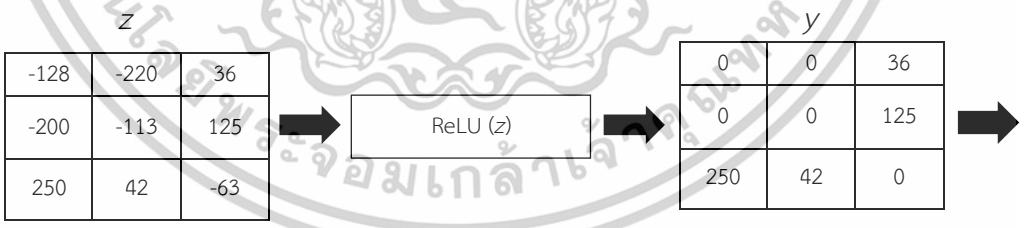
เอาต์พุต (Output) ที่ได้จากการทำคอนโวลูชันจะเรียกว่า ฟีเจอร์แมพหรือแอคทิเวชันแมพ (Feature Map : Activation Map) ซึ่งจะมีขนาดเป็น ($W_{input} \times H_{input} \times D_{input}$) โดยที่

$$w_{output} = \left(\frac{w_{input} - \mathrm{F} + 2\mathrm{P}}{S}\right) + 1 \tag{2}$$

$$H_{output} = \left(\frac{H_{input} - \mathrm{F} + 2\mathrm{P}}{S}\right) + 1 \tag{3}$$

$$D_{output} = K \tag{4}$$

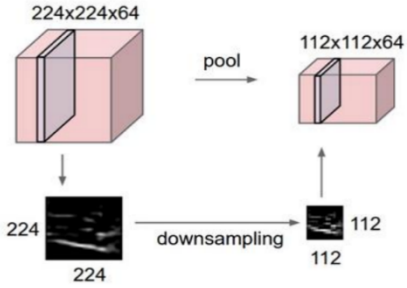
ผลที่ได้จากการทำคอนโวลูชันจะต้องผ่านฟังก์ชันกระตุ้น (Activation Function) เพื่อทำการปรับค่าที่ได้จากการทำคอนโวลูชันและส่งต่อไปเป็นอินพุตในชั้นถัดไป โดยฟังก์ชันกระตุ้นที่นิยมใช้ใน CNN มีหลายฟังก์ชัน เช่น ฟังก์ชันเรคติไฟด์เชิงเส้น (ReLU) ตามรูปที่ 2.4 แสดงค่าที่ได้หลังผ่านฟังก์ชันเรคติไฟด์เชิงเส้น



รูปที่ 2.4 ผลลัพธ์ที่ได้จากฟังก์ชันเรคติไฟด์เชิงเส้น (ReLU)

2.2.2 ชั้นพูลลิ่ง (Pooling Layer)

ชั้นพูลลิ่ง (Pooling Layer) มักวางต่อจากชั้นคอนโวลูชัน จะถูกใช้เพื่อลดขนาดของฟีเจอร์แมพ (Feature Map, Activation Map) กล่าวคือฟีเจอร์แมพที่ได้จากชั้นคอนโวลูชันจะถูกส่งไปยังชั้นพูลลิ่งเพื่อทำ การลดขนาดความกว้างและความสูง ทำให้ข้อมูลที่จะส่งไปยังชั้นถัดไปทำให้เป็นการมีขนาดเล็กลง ซึ่งจะเป็นการลดพารามิเตอร์ไปด้วย



รูปที่ 2.5 Pooling Operation

ที่มา : CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved, from http://cs231n.github.io/convolutional-networks/#architectures

ในชั้นพูลลิ่งจะรับข้อมูลเข้าซึ่งจะมีขนาดเป็น ($W_{input} \times H_{input} \times D_{input}$) และจะมีพารามิเตอร์ที่ต้องกำหนดดังต่อไปนี้

Pool size ขนาดของพูล ($F \times F$)

Stride (S) เพื่อกำหนดการขยับของ Pool

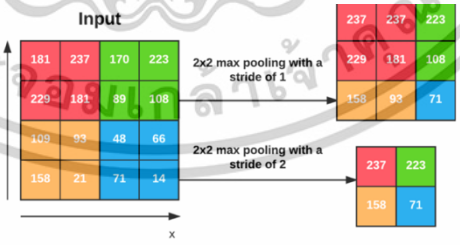
เอาต์พุตทจากชั้นพูลลิ่งจะมีขนาดเป็น ($W_{output} \times H_{output} \times D_{output}$) โดยที่

$$w_{output} = \left(\frac{w_{input} - \mathrm{F}}{S}\right) + 1 \quad (5)$$

$$H_{output} = \left(\frac{H_{input} - \mathrm{F}}{S}\right) + 1 \quad (6)$$

$$D_{output} = D_{input} \quad (7)$$

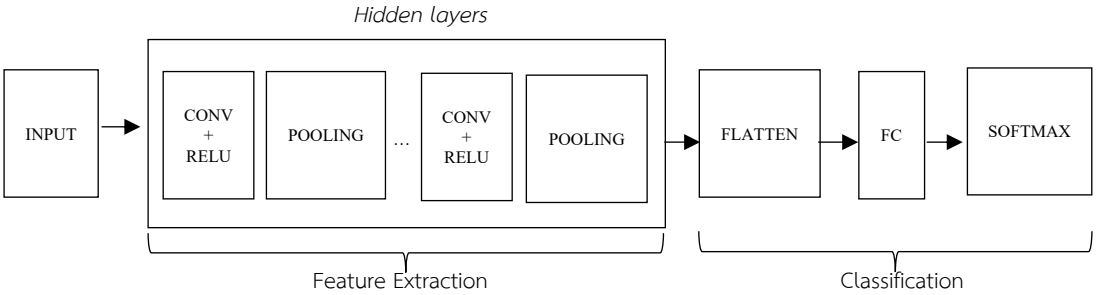
การดำเนินการบนชั้นพูลลิ่งสามารถทำได้ 2 วิธีคือพูลค่ามากที่สุด (Max Pooling) และพูลค่าเฉลี่ย (Average Pooling)



รูปที่ 2.6 การทำพูลลิ่ง

ที่มา : CS231n Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved, from http://cs231n.github.io/convolutional-networks/#architectures

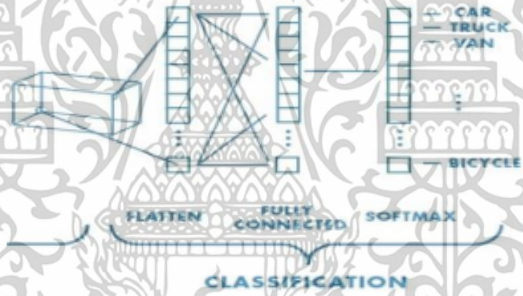
จุดประสงค์ของชั้นคอนโวลูชันและชั้นพูลลิ่งนั้นก็เพื่อทำการสกัดฟีเจอร์จากภาพโดยในชั้นซ่อนแรกๆ จะได้พีเจอร์ระดับล่างเช่น เส้นในชั้นที่ลึกขึ้นลงไปก็จะเป็นการสกัดฟีเจอร์ที่ซับซ้อนขึ้น ซึ่งจำนวนชั้นคอนโวลูชันและชั้นพูลลิ่งที่เพิ่มขึ้นก็จะส่งผลให้ความลึกของโครงข่ายที่เพิ่มขึ้น



รูปที่ 2.7 โครงสร้างอย่างง่ายของโครงข่ายประสาทเทียมแบบคอนโวลูชัน

2.2.3 ชั้นเชื่อมโยงสมบูรณ์ (Fully Connected Layer)

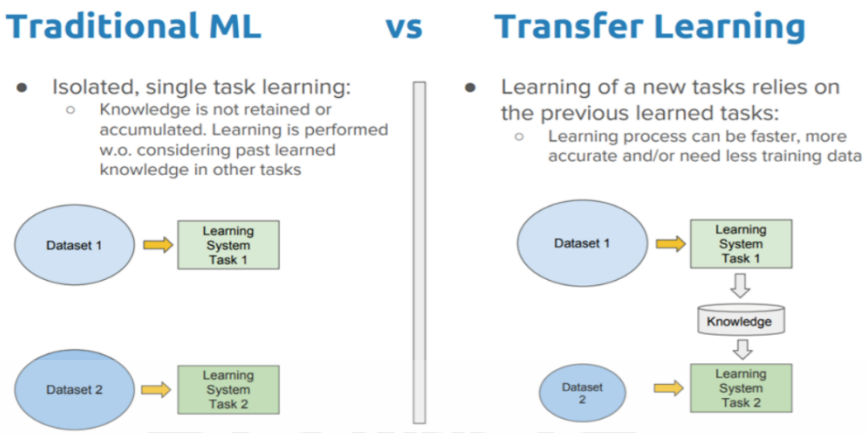
ชั้นสุดท้ายในเครือข่ายประสาทเทียมที่นำมาใช้ในการวิจัยนี้เป็นชั้นที่เชื่อมต่ออย่างสมบูรณ์ (Fully Connected Layer) ตามที่แสดงในรูปที่ 2.8 ชั้นนี้รับคุณลักษณะที่สกัดมาจากชั้นคอนโวลูชันและชั้นพูลลิ่ง แล้วใช้คุณลักษณะเหล่านี้ในการจำแนกประเภทหรือทำการทำนายผลลัพธ์ การประมวลผลที่เกิดขึ้นในชั้นนี้เป็นการรวมและแปลงข้อมูลคุณลักษณะให้อยู่ในรูปแบบที่เหมาะสมสำหรับการวิเคราะห์ขั้นสุดท้าย ซึ่งมีผลสำคัญต่อความแม่นยำในการจำแนกประเภทของเครือข่าย



รูปที่ 2.8 Fully Connected Layer
ที่มา : https://towardsml.com/2018/10/16/deep-learning-series-p2-understandingconvolutional-neural-networks/

2.3 การเรียนรู้แบบถ่ายโอน (Transfer Learning)

การเรียนรู้แบบถ่ายโอนเป็นเทคนิคการเรียนรู้ของเครื่อง ซึ่งใช้การเรียนรู้ด้วยการนำโครงสร้างของโมเดลที่ฝึกเรียบร้อยแล้วนำข้อมูลของใหม่มาฝึก จะช่วยเรื่องในการประหยัดเวลาอย่างมาก เนื่องจากระยะเวลาในการฝึกตั้งแต่ต้นจนจบกระบวนการใช้เวลานานและมีความซับซ้อน อีกทั้งยังต้องใช้ชุดข้อมูลที่มีขนาดใหญ่และใช้เวลาในการประมวลผลหลายวันจนถึงหลายสัปดาห์ ดังรูปที่ 2.9 เป็นการเปรียบเทียบระหว่างการเรียนรู้ของเครื่องที่ไม่ใช้เทคนิคการเรียนรู้ของเครื่อง (traditional ML) และการเรียนรู้ของเครื่องใช้เทคนิคการถ่ายโอน (transfer Learning) โดยโมเดลที่นำมาใช้ในการการศึกษาค้นคว้าอิสระครั้งนี้คือ VGG16 , ResNet50 และ InceptionV3

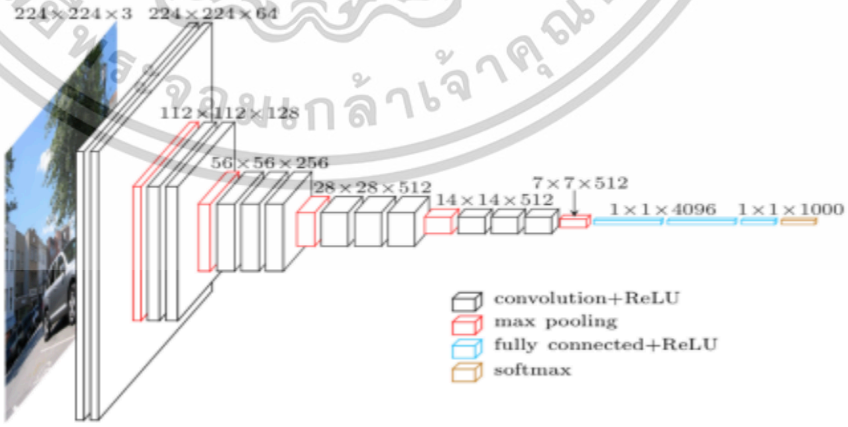


รูปที่ 2.9 การเปรียบเทียบระหว่าง Tradition ML และ Transfer Learning

ที่มา : https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transferlearning-with-real-world-applications-in-deep-learning-212bf3b2f27a

2.3.1 โครงสร้างการถ่ายโอนโมเดล VGG16

Visual Geometry Group หรือ VGG16 (Simonyan และ Zisserman, 2014) โดยกลุ่มนักวิจัย Oxford ได้ทำการพัฒนาโครงสร้างสถาปัตยกรรมนี้ขึ้นมา และได้รับความสนใจมากจากการแข่งขัน ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ปี ค.ศ.2014 มีความแม่นยำในชุดข้อมูลทดสอบที่ 92.7% และเป็นที่นิยมจนถึงปัจจุบัน โดยสิ่งที่เป็นจุดเด่นของ VGG16 มีเอกลักษณ์เฉพาะแทนที่ hyperparameter จำนวนมาก เน้นไปที่การออกแบบชั้น conv2D 3x3 pixels การก้าว 1 stride และการใช้ same padding และ max pooling ขนาด 2x2 pixels การก้าว 2 stride แบบเดียวกันตลอดทั้งโครงสร้าง และมี 2 ชั้นสุดท้ายเป็นชั้นเชื่อมต่ออย่างสมบูรณ์ (Fully connected layer) VGG16 หมายถึงมี 16 ชั้นตามชื่อ ที่มีน้ำหนักเครือข่ายที่ใหญ่และมีพารามิเตอร์ประมาณ 138 ล้าน โครงสร้างสถาปัตยกรรม VGG16 ได้แสดงดังรูปที่ 2.10

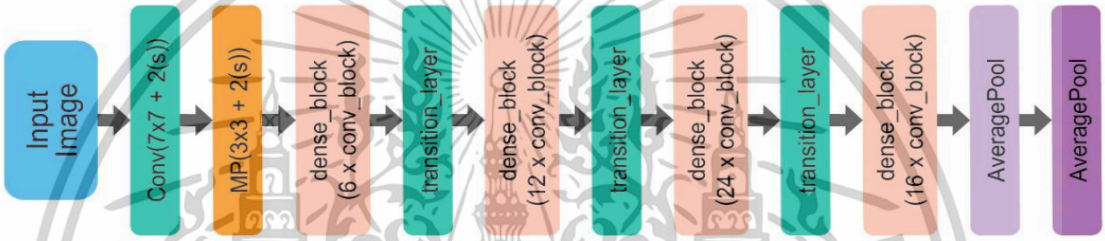


รูปที่ 2.10 สถาปัตยกรรม VGG16

ที่มา : https://towardsdatascience.com/understand-the-architecture-of-cnn-90a25e244c7

2.3.2 โครงสร้างการถ่ายโอนโมเดล ResNet50

Deep Residual Networks หรือ ResNet (He, Zhang, Ren, และ Sun, 2016) ได้นำเสนอในงานวิจัย Deep residual learning for image recognition เป็นโครงสร้างที่มีขนาดใหญ่มีจำนวนชั้นในเครือข่ายถึง 152 ชั้น ถูกสร้างมาเพื่อแก้ปัญหา Vanishing Gradient คือปัญหาในระหว่างการเทรน Gradient มีขนาดเล็กลงเรื่อย ๆ จนเท่ากับ 0 ทำให้น้ำหนัก (weight) ไม่ถูกอัพเดทอีกต่อไป ทำให้โมเดลเทรนต่อไม่ได้ซึ่งโครงสร้างสถาปัตยกรรมนี้ประกอบด้วย 4 บล็อกใหญ่ จำนวนชั้นที่มีพารามิเตอร์ใช้สำหรับการฝึกทั้งหมดจะเป็นจำนวนชั้นที่ใช้ในการเรียกชื่อของ Resnet เช่น Resnet 34, Resnet50, ResNet101, Resnet152 เป็นต้น โดยงานวิจัยนี้เลือกใช้ ResNet50 ซึ่งจะมี 50 ชั้น เป็น [3, 4, 6, 3] ซึ่งก็คือ (3+4+6+3)x3 = 48 ชั้น บวกชั้นคอนโวลูชั่นที่ติดกับชั้นอินพุต และชั้น Dense ที่ติดกับชั้นเอาต์พุต

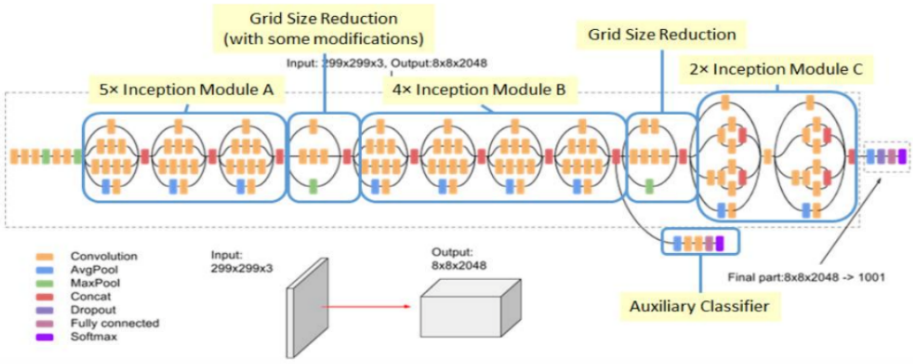


รูปที่ 2.11 สถาปัตยกรรม ResNet50

ที่มา : https://khoon.msu.ac.th/_dir/fulltext/2023/01/Pichada_Saichua65.pdf

2.3.3 โครงสร้างการถ่ายโอนโมเดล InceptionV3

โมเดลนี้พัฒนาโดย Google ถูกกล่าวในงานวิจัย InceptionV3 (1st ILSVRC 2015) เป็นโมเดลที่แสดงให้เห็นว่ามีความแม่นยำมากกว่า 78.1% ในชุดข้อมูล ImageNet โมเดลนี้มีแนวคิดมากมายที่นักวิจัยหลายคนพัฒนาขึ้นในช่วงหลายปีที่ผ่านมา มีพื้นฐานมาจากบทความต้นฉบับ (Szegedy, Vanhoucke, Ioffe, Shlens, และ Wojna, 2016) โดยการลดโครงสร้างภายในออกเป็น 5 ขั้นตอน คือ 1) Inception Module A จำนวน 5 Module, 2) Grid Size of Reductiion Step 1 จำนวน 1 Module, 3) Inception Module B จำนวน 4 Module, 4) Grid Size of Reduction Step 2 จำนวน 1 Moduel, 5) Inception Module C จำนวน 2 Module และ Head (8x8x2048) จำแนกประเภทโดยใช้ softmax function คำนวณเป็นความน่าจะเป็นในการจำแนกประเภทสามารถแยก output ได้ 1,000 Class รูปที่ 2.12



รูปที่ 2.12 สถาปัตยกรรม InceptionV3

ที่มา : https://stackoverflow.com/questions/54793602/tensorflow-hub-inception-v3-structure-compared-to-keras-inception-v3-structure

2.4 การดัดแปลงข้อมูลต้นฉบับ (Data Augmentation)

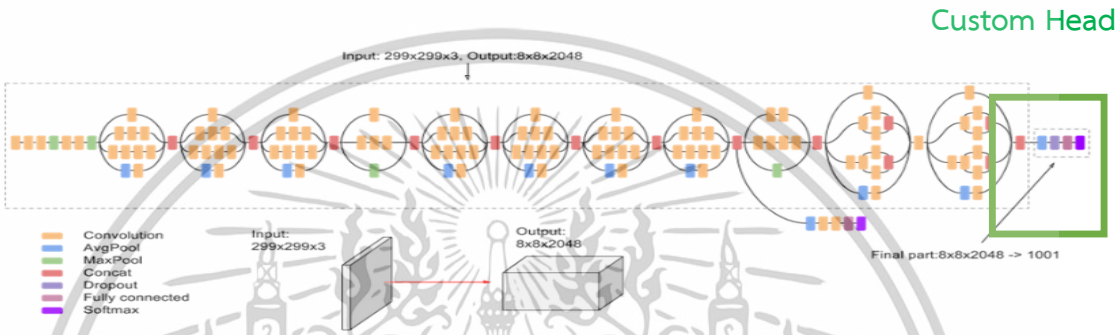
Data Augmentation คือ เป็นเทคนิคที่ใช้เพื่อเพิ่มปริมาณข้อมูลโดยการเพิ่มสำเนาที่แก้ไขเล็กน้อยของข้อมูลที่มีอยู่แล้วหรือข้อมูลสังเคราะห์ที่สร้างขึ้นใหม่จากข้อมูลที่มีอยู่ โดยการนำรูปมาย่อ ขยาย พลิกซ้าย ขวา ล่าง บน, หมุนซ้าย หมุนขวา, Crop มุม, ปรับสีเข้ม สีอ่อน, ปรับสว่าง ปรับมืด, ปรับ Contrast, เพิ่มลด noise, เบลอภาพ ดังรูปที่ 2.13 เพื่อเพิ่มประสิทธิภาพของโมเดลโครงสร้างประสาทเทียมแบบคอนโวลูชัน และยังมีประโยชน์ในการลด Overfitting อีกด้วย

รูปที่ 2.13 ตัวอย่าง Data Augmentation

ที่มา : https://www.researchgate.net/figure/Types-of-data-augmentation-used-in-this-work_fig4_343687059

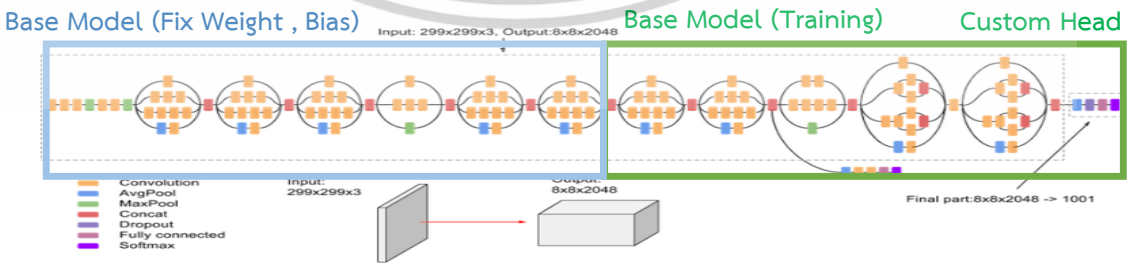
2.5 การปรับแต่ง Fine Tune

การปรับแต่ง Fine Tune ในหัวข้อนี้ ขอยกตัวอย่างการใช้ Transfer Learning ด้วยตัวอย่างของโมเดล InceptionV3 ซึ่งประกอบด้วย 2 ส่วนคือ Base Model และ Head Model โดย Base Model จะถูกนำมาเรียกใช้ในโครงงานนี้ ส่วน Head Model จะถูกนำมาปรับการใช้งานของโมเดลให้เหมาะสมกับโครงงานนี้ ตามที่เราต้องการ ในกรณีที่ต้องการเปลี่ยนแปลง Head Model เราสามารถทำได้โดยการตั้งค่า include top ให้เป็น false ตามที่แสดงรูปที่ 2.14 ซึ่งจะช่วยให้เราสามารถปรับแต่งโมเดลได้อย่างยืดหยุ่นตามความต้องการของโครงงานได้อย่างเหมาะสม และมีประสิทธิภาพ



รูปที่ 2.14 โครงสร้าง Networks ของโมเดล InceptionV3 ที่มีการ custom head
ที่มา : https://cloud.google.com/tpu/docs/inception-v3-advanced

จากนั้นได้ทำการเพิ่ม Layers ตามที่ต้องการและทำการฝึกสอนแบบ Fix Weight และ Bias ในส่วนของ Base Model แล้ว เราจะตรวจสอบค่าความแม่นยำของโมเดลที่ได้เพิ่ม Head layers เข้าไป เพื่อทำการประเมินประสิทธิภาพของโมเดลในการแยกแยะหรือทำนายผลลัพธ์ของข้อมูลภาพที่ใช้ในโครงงาน หากพบว่าค่าความแม่นยำมีระดับต่ำเราสามารถพิจารณาทำ Fine Tune โดยการฝึกสอนโมเดลอีกครั้งโดยใช้ชุดข้อมูล (Data Set) ใหม่ เพื่อปรับปรุงความแม่นยำของโมเดลให้ดียิ่งขึ้น ในกรณีที่มีความต้องการให้โมเดลมีประสิทธิภาพที่ดีขึ้น เราสามารถ Fine Tune โมเดล โดยการ Train บางส่วนของ Base Model หรือทั้งหมดของ Network เพื่อปรับแต่งและปรับปรุงโมเดลให้สอดคล้องกับโครงงานของเราให้เหมาะสมและมีประสิทธิภาพสูงสุดเท่าที่เป็นไปได้ ดังรูปที่ 2.15



รูปที่ 2.15 โครงสร้าง Networks ของโมเดล InceptionV3 ที่มีการ Fine Tune
ที่มา : https://cloud.google.com/tpu/docs/inception-v3-advanced

2.6 ประเมินประสิทธิภาพแบบจำลอง

2.6.1 คอนฟิวชันเมทริกซ์ (Confusion Matrix)

คอนฟิวชันเมทริกซ์ เป็นการประเมินผลลัพธ์การทำนายของโมเดล โดยนำผลลัพธ์ที่ได้จากโมเดลไปทำการเปรียบเทียบกับผลลัพธ์ที่เป็นความจริง และในการศึกษาค้นคว้าอิสระครั้งนี้เป็นการจำแนกประเภทรูปภาพออกเป็น 2 คลาส จึงสามารถเกิดเหตุการณ์ได้ดังตารางที่ 2.1

ตารางที่ 2.1 ตารางคอนฟิวชันเมทริกซ์การจำแนกแบบ 2 คลาส

		คลาสที่ทำนาย	
		เกิดความเสียหาย	ไม่เกิดความเสียหาย
คลาสจริง	เกิดความเสียหาย	TP	FN
	ไม่เกิดความเสียหาย	FP	TN

โดย

True Positive (TP) คือค่าที่ทำนายว่าจริง และข้อมูลนั้นเป็นจริง

True Negative (TN) คือค่าที่ทำนายว่าจริง แต่ข้อมูลนั้นเป็นเท็จ

False Positive (FP) คือค่าค่าที่ทำนายว่าเท็จ และข้อมูลนั้นเป็นเท็จ

False Negative (FN) คือค่าที่ทำนายว่าเท็จ แต่ข้อมูลนั้นเป็นจริง

2.6.2 ค่าความถูกต้อง (Accuracy)

การวัดความถูกต้องของโมเดล โดยพิจารณาทุกคลาส ได้จากสมการที่ 9

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

2.6.3 ค่าความแม่นยำ (Precision)

การวัดค่าความแม่นยำของโมเดล โดยพิจารณาแยกทีละคลาส ได้จากสมการที่ 10

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

2.6.4 ค่าความระลึก (Recall)

การวัดค่าความถูกต้องของโมเดล โดยพิจารณาแยกทีละคลาส ได้จากสมการที่ 11

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

2.6.5 ค่าเฉลี่ยระหว่าง precision และ recall (F1-Score)

คือ ค่าเฉลี่ยระหว่าง precision และ recall เพื่อวัดความสามารถของโมเดล ได้จากสมการที่ 12

$$\text{F1} - \text{Score} = 2 \text{ x } \frac{Precision\ x\ Recall}{Precision + Recall} \tag{12}$$

2.7 งานวิจัยที่เกี่ยวข้อง

จากการศึกษางานวิจัยที่เกี่ยวกับการตรวจจับความเสียหายของรถยนต์จากรูปภาพและน่าสนใจนำมาศึกษา ดังนี้

2.7.1 งานวิจัยของ Kyu และ Woraratpanya (2020) ได้ศึกษาแนวโน้มการเติบโตของอุตสาหกรรมรถยนต์ ซึ่งส่งผลต่อการเพิ่มขึ้นของจำนวนอุบัติเหตุ ทำให้บริษัทประกันต้องเผชิญหน้ากับการเรียกร้องค่าสินไหมทดแทน ในปริมาณที่เพิ่มสูงขึ้น วิธีการในการแก้ปัญหานี้ ก็คือการใช้ปัญญาประดิษฐ์ (Artificial Intelligence) การเรียนรู้ของเครื่อง (Machine Learning) และการเรียนรู้เชิงลึก (Deep learning) ในการตรวจจับและประเมินความเสียหายของรถยนต์ ในงานวิจัยนี้ได้นำอัลกอริธึม VGG16 และ VGG19 มาประยุกต์ใช้ได้แสดงให้เห็นถึงความแม่นยำที่สูงในการตรวจจับความเสียหาย โดยโมเดล VGG19 ได้ความแม่นยำอยู่ที่ 95.22% ส่วน VGG16 มีความแม่นยำที่ 94.56% ในการจำแนกตำแหน่งของความเสียหาย VGG19 แสดงความแม่นยำที่ 76.48% ในขณะที่ VGG16 มีความแม่นยำที่ 74.39% และเมื่อพิจารณาความรุนแรงของความเสียหาย VGG19 ได้ความแม่นยำ 58.48% และ VGG16 อยู่ที่ 54.8% โดยรวมแล้วประสิทธิภาพของ VGG19 พบว่าดีกว่า VGG16

2.7.2 งานวิจัยของ Patil, Kulkarni, Sriraman และ Karande (2017) ได้นำเสนอเกี่ยวกับการประยุกต์ใช้เทคนิคการเรียนรู้เชิงลึก (Deep Learning) เพื่อวิเคราะห์และจำแนกประเภทความเสียหายของรถยนต์จากรูปภาพโดยใช้โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) โดยงานวิจัยนี้มีปริมาณข้อมูลรูปภาพที่จำกัด ทำให้ต้องทำการดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) ที่เป็นวิธีเพิ่มปริมาณข้อมูลรูปภาพเพื่อปรับปรุงประสิทธิภาพของโมเดล และการทดลองได้เน้นการจำแนกประเภทความเสียหายที่เกิดขึ้นกับรถยนต์ในหลาย ๆ ประเภท เช่น กันชนบุบ, ประตูบุบ, กระจกแตก, และรอยชน เป็นต้น โดยมีการวิเคราะห์ผลลัพธ์จากโมเดลต่างๆ เช่น Car, Inception, Alexnet, VGG19, VGG16, และ Resnet และผลลัพธ์ของการทดลองวิธีที่ดีที่สุดคือ Resnet สามารถทำงานได้ดีที่สุด โดยมีความแม่นยำถึง 88.24% เมื่อไม่ใช้ data augmentation และได้ประสิทธิภาพสูงขึ้นเป็น 89.53% เมื่อใช้วิธีการถ่ายโอนความรู้ (transfer learning) ร่วมกับการเรียนรูปแบบรวม (ensemble learning) ทำให้งานวิจัยนี้ยืนยันว่าการประยุกต์ใช้เทคนิคการเรียนรู้เชิงลึกสามารถนำไปใช้วิเคราะห์และจำแนกความเสียหายของรถยนต์จากรูปภาพได้อย่างมีประสิทธิภาพ

2.7.3 งานวิจัยของ Sarath P, Soorya M, Shaik Abdul Rahman A, S Suresh Kumar, K Devaki (2020) ได้ศึกษาวิธีการจัดการป้องกันรถยนต์โดยใช้รูปภาพเพื่อเพิ่มศักยภาพในการประมวลผลอัตโนมัติ โดยเน้นที่การกำหนดลักษณะความเสียหายของรถยนต์ด้วยการจำแนกประเภทท การศึกษานี้ได้ทำการทดสอบการเรียนรู้เชิงลึก (Deep learning) และเริ่มต้นด้วยการเตรียมการฝึกโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network) อย่างเข้มงวด แต่พบว่าประสิทธิภาพไม่เพียงพอเนื่องจากข้อมูลที่มีการประมวลผลมีจำนวนน้อย จากนั้น

จึงตรวจสอบผลกระทบของการเตรียมการเรียนรู้ล่วงหน้าและปรับแต่งโมเดลเพิ่มเติมเพื่อเพิ่มความแม่นยำในการจำแนกประเภท และสุดท้ายได้ทดลองทางเลือกต่างๆ ในการปรับปรุงโมเดลผ่านการเรียนรู้แบบถ่ายโอน (Transfer Learning) และการปรับแต่งเฉพาะ ผลการทดลองชี้ให้เห็นว่าการเรียนรู้แบบถ่ายโอน (Transfer Learning) ให้ผลลัพธ์ที่ดีกว่าการปรับแต่งแบบเฉพาะ และได้ความแม่นยำสูงสุดถึง 89.5% โดยใช้วิธีการผสมผสานการเรียนรู้แบบถ่ายโอน (Transfer Learning) และการเรียนรู้การรวบรวมข้อมูล

2.7.4 งานวิจัยของ คุณธนัช เบญจอนุอาชา, คุณวราภรณ์ วิยานนท์ (2022) ได้ศึกษาการจำแนกรูปภาพรถยนต์ที่มีความเสียหายและไม่มีความเสียหาย โดยใช้เทคนิคการเรียนรู้เชิงลึก เพื่อพัฒนาแบบจำลองการจำแนกความเสียหายรถยนต์โดยใช้โครงข่ายประสาทเทียมแบบสังวัฒนาการหรือ Convolutional Neural Network (CNN) ได้แก่ VGG16, ResNet50 และ InceptionV3 ร่วมกับการใช้เทคนิคการเรียนรู้แบบถ่ายโอนโดยจำแนกออกเป็น 2 ประเภท ได้แก่ มีความเสียหายและ ไม่มีความเสียหาย โดยได้ใช้ชุดข้อมูลจากเว็บไซต์ Kaggle เปรียบเทียบประสิทธิภาพโมเดลซึ่งผู้วิจัยได้ทำการปรับพารามิเตอร์ต่างๆเพื่อให้เหมาะสมก็ชุดข้อมูลมีประสิทธิภาพดีที่สุด ซึ่งผลลัพธ์ที่ได้คือโมเดล VGG16 มีประสิทธิภาพมากที่สุด วัดค่าความแม่นยำ (accuracy) เท่ากับ 0.83 ตามมาด้วยโมเดล InceptionV3 เท่ากับ 0.81 และ ResNet50 เท่ากับ 0.68

2.7.5 งานวิจัยของ Muhammad Mujahid, Furqan Rustam, Roberto Álvarez, Juan Luis Vidal Mazón, Isabel de la Torre Díez และ Imran Ashraf (2022) ได้ศึกษาเกี่ยวกับการตรวจวินิจฉัยโรคปอดบวม ซึ่งเป็นสาเหตุการเสียชีวิตอันดับต้น ๆ ของทารกและผู้สูงอายุ โดยมีผู้เสียชีวิตราว 4 ล้านรายต่อปี โรคนี้อาจเกิดจากไวรัส แบคทีเรีย หรือเชื้อรา ทำลายถุงลมเล็กๆ ในปอด กลุ่มเสี่ยงคือผู้ป่วยโรคเรื้อรัง ผู้มีภูมิคุ้มกันอ่อนแอ ทารก และผู้สูงอายุที่ใช้เครื่องช่วยหายใจ หากไม่สามารถตรวจพบได้ตั้งแต่ระยะแรก อาจมีความเสี่ยงต่อชีวิต แม้ปัจจุบันจะมีแนวทางการวินิจฉัยอยู่ แต่ยังขาดความแม่นยำและประสิทธิภาพ การศึกษานี้ใช้ภาพเอกซเรย์และโมเดล Convolutional Neural Network (CNN) ที่ฝึกฝนล่วงหน้า ได้แก่ VGG16, Inception-V3 และ ResNet50 พร้อมประเมินผลด้วย Cohen's kappa และ AUC ผลการทดลองแสดงว่า Inception-V3 มีความแม่นยำและคะแนนการเรียกคืนสูงสุดอยู่ที่ 99.29% และ 99.73% ตามลำดับ

จากการศึกษางานวิจัยที่เกี่ยวข้อง พบว่ามีการใช้โครงข่ายประสาทแบบคอนโวลูชัน (CNN) และเทคนิคถ่ายโอนความรู้ (Transfer Learning) โดยใช้อัลกอริธึม VGG16, ResNet50 และ InceptionV3 ซึ่งให้ผลลัพธ์ที่น่าพอใจ ผู้วิจัยจึงสนใจนำอัลกอริธึมเหล่านี้มาประยุกต์ใช้ในการศึกษาค้นคว้าอิสระครั้งนี้

บทที่ 3
วิธีการดำเนินการ

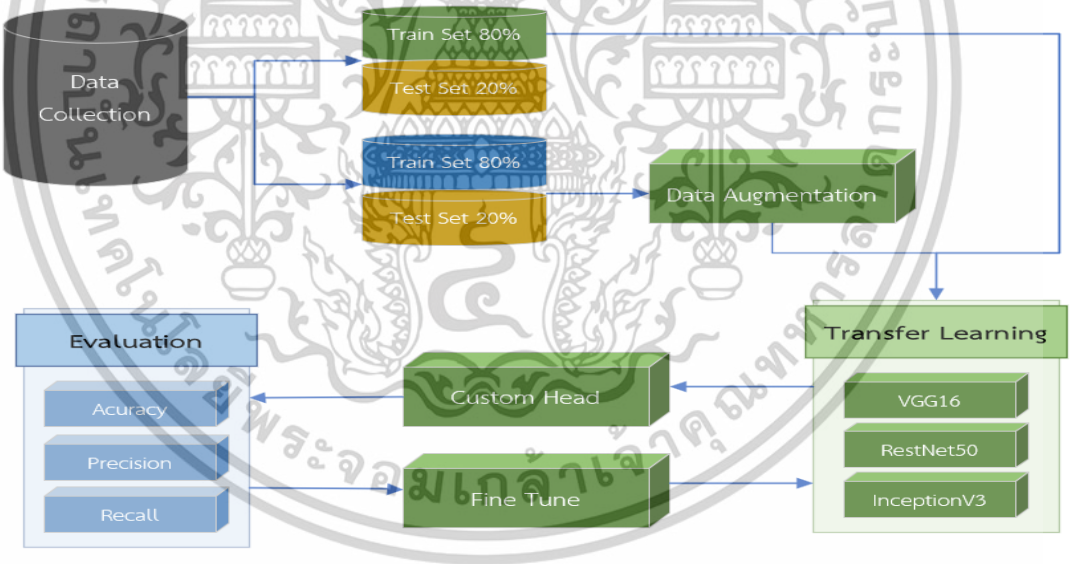
การศึกษาค้นคว้าอิสระครั้งนี้เป็นการศึกษาค้นคว้าเชิงการทดลอง (Experimental Research) มีวัตถุประสงค์เพื่อศึกษาเกี่ยวกับการตรวจจับความเสียหายของรถยนต์ด้วยวิธีการเรียนรู้ของเครื่อง (Machine Learning) โดยใช้โครงสร้างประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network) ใช้เทคนิคการถ่ายโอน 3 โครงสร้าง (Transfer Learning) ประกอบด้วย โครงสร้างการถ่ายโอนโมเดล VGG16 , โครงสร้างการถ่ายโอนโมเดล ResNet50 และโครงสร้าการถ่ายโอนโมเดล InceptionV3 โดยกำหนดแนวทางในการศึกษา ดังนี้

3.1 ขั้นตอนการดำเนินการเบื้องต้น

3.2 การทดลองปรับพารามิเตอร์และเปรียบเทียบประสิทธิภาพ

3.3 ขั้นตอนการดำเนินการ Fine Tune

3.1 ขั้นตอนการดำเนินการเบื้องต้น



รูปที่ 3.1 ขั้นตอนการดำเนินการ

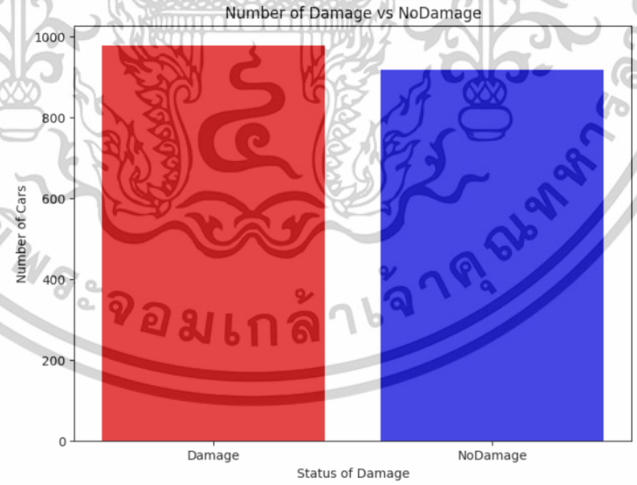
3.1.1 การเก็บรวบรวมข้อมูล (Data Collection)

การศึกษาค้นคว้าอิสระครั้งนี้ดำเนินการเก็บรวบรวมข้อมูลจากเว็บไซต์ GitHub ซึ่งเป็นแหล่งเก็บข้อมูลโค้ดโปรแกรมและเอกสารที่เกี่ยวข้องกับ DetectDamage ของรถยนต์ จากเว็บไซต์ : https://github.com/rehmanzafar/DetectDamage/blob/master/README.md ได้แสดงตัวอย่างภาพถ่ายเหล่านี้ในรูปที่ 3.2

รูปที่ 3.2 ตัวอย่างภาพถ่ายของรถ

3.1.2 ตรวจสอบและสำรวจข้อมูลเบื้องต้น (Exploratory Data Analysis)

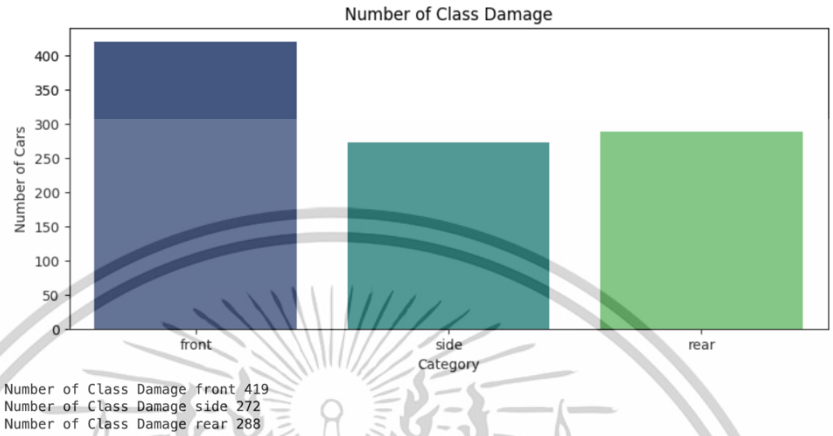
จากการสำรวจข้อมูลเบื้องต้นจากภาพถ่ายสามารถแบ่งออกเป็นรถที่เกิดความเสียหาย จำนวน 979 รูป และ และภาพถ่ายของรถที่ไม่เกิดความเสียหาย จำนวน 919 รูป โดยแสดงจากรูปที่ 3.3




```
Number of Damage  979
Number of NoDamage  919
```

รูปที่ 3.3 กราฟวัดผลการเกิดความเสียหายและไม่เกิดความเสียหาย

จากการสำรวจข้อมูลเบื้องต้นจากภาพถ่ายสามารถแบ่งออกเป็นรถที่เกิดความเสียหายส่วนด้านหน้า (Front) จำนวน 419 รูป ภาพถ่ายของรถที่เกิดความเสียหายส่วนด้านข้าง (Side) จำนวน 272 รูป และภาพถ่ายของรถที่เกิดความเสียหายส่วนด้านหลัง (Rear) จำนวน 288 รูป และโดยแสดงจากรูปที่ 3.4



รูปที่ 3.4 กราฟแสดงผลของภาพถ่ายของรถที่เกิดความเสียหายในส่วนต่างๆ

3.1.3 การเตรียมข้อมูล (Data Preprocessing)

ขั้นตอนการเตรียมข้อมูลจากข้อมูลภาพถ่ายทั้งหมด จำนวน 1,898 รูป ได้นำมาสร้างชุดข้อมูล (Data Set) โดยทำการแบ่งข้อมูลเป็นชุดข้อมูลในการฝึกสอน (Train Set) 80% ประกอบด้วยภาพถ่ายที่เกิดความเสียหาย จำนวน 778 รูป ภาพถ่ายที่ไม่เกิดความเสียหาย จำนวน 740 รูป และแบ่งเป็นชุดข้อมูลในการทดสอบ (Test Set) อีก 20% ประกอบด้วยภาพถ่ายที่เกิด ความเสียหาย จำนวน 201 รูป ภาพถ่ายที่ไม่เกิดความเสียหาย จำนวน 179 รูป ตามตารางที่ 3.1

ตารางที่ 3.1 ตารางการแบ่งชุดข้อมูลข้อมูลการฝึกสอน (Train Set) และ
ชุดข้อมูลการทดสอบ (Test Set)

Class	Train Set	Test Set
Damage	778	201
Nodamage	740	179
Total	1,518	380

หลังจากแบ่งชุดข้อมูล (Data Set) ได้มีการจัดการภาพถ่ายออกเป็น 2 ประเภทเพื่อใช้เปรียบเทียบในการทดสอบ

1) ไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)

สำหรับข้อมูลที่ไม่มีการแปลงข้อมูลต้นฉบับเพียงทำปรับขนาดภาพถ่าย (Target_Size) ให้เป็น 224x224x3 และปรับค่าพิกเซลของภาพถ่ายแทนค่าความเข้มสี ด้วยตัวเลขระหว่าง 0 ถึง 255 (rescale) โดยใช้คำสั่ง ImageDataGenerator ในโปรแกรมไพธอน (python)

2) แปลงข้อมูลต้นฉบับ (With Data Augmentation)

สำหรับการเตรียมข้อมูลแบบการดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) ใช้คำสั่ง ImageDataGenerator ในปรแกรมไพธอน (python) ในการปรับขนาดภาพถ่าย (Target_Size) ให้เป็น 224x224x3 และปรับค่าพิกเซลของภาพถ่ายแทนค่าความเข้มสี ด้วยตัวเลขระหว่าง 0 ถึง 255 (rescale) หลังจากนั้นนำข้อมูลไปดัดแปลงข้อมูลต้นฉบับ (data augmentation) ปรับข้อมูลด้วยคำสั่งดังนี้ ปรับหมุนภาพ 40 องศา (Rotation_Range = 40) ปรับเลื่อนรูปภาพแนวกว้าง 20% (Width_Shift_Range = 0.2) ปรับเลื่อนภาพถ่ายแนวสูง 20% (Height_Shift_Range = 0.2) เฉือนภาพถ่าย 20% (Shear_Range = 0.2) ปรับขยายรูปภาพ 20% (Zoom_Range = 0.2) ปรับพลิกภาพถ่ายกลับจากซ้ายไปขวาหรือขวา 19 ไปซ้าย (Horizontal_Flip=True) และคำสั่งเติมพิกเซลที่ขาดหายไปหลังจากการย้ายหรือเฉือนออกด้วยพิกเซลใกล้เคียง (Fill_Mode=nearest)

หลังจากทำการดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) ได้ทำการกำหนดค่าพารามิเตอร์ (Parameters) ของข้อมูลภาพถ่าย เพื่อปรับแต่งลักษณะของการโหลดข้อมูลถ่ายภาพในข้อมูลที่จัดเก็บในรูปแบบตาราง (DataFrame) โดยการกำหนดพารามิเตอร์ DataFrame เป็นชุดข้อมูลในการฝึกสอน (Train) และชุดข้อมูลในการทดสอบ (Test) และระบุคอลัมน์ที่เก็บที่อยู่ของภาพถ่าย (X_Col='image_path') และระบุคอลัมน์ที่เก็บป้ายกำกับ (Y_Col='damage') รวมถึงระบุขนาดภาพที่ต้องการให้ทุกภาพถ่ายมีขนาด 224x224 (Target_Size=(224, 224)) และระบุประเภทของป้ายกำกับเป็นแบบประเภท Binary เนื่องจากมีเพียงสองคลาสคือ Damage หรือ NoDamage (Class_Mode='binary') และกำหนด Batch Size หรือว่ากลุ่มข้อมูลที่ถูกนำเข้าไปในโมเดลในเวลาเดียวกันในแต่ละรอบของการฝึกสอน (Training) โดยกำหนดขนาด Batch Size ที่ 32 รูป (Batch_Size=32) หลังจากขั้นตอนปรับพารามิเตอร์(Parameters) ของภาพถ่ายแล้วจะนำข้อมูลไปเข้ากระบวนการสกัดคุณลักษณะ (Feature Extraction)

การทำการดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) มีความสำคัญอย่างยิ่งในการศึกษาค้นคว้าอิสระในครั้งนี้ เนื่องจากข้อมูลภาพถ่ายของรถยนต์มีเพียง 1,898 รูป ซึ่งถือว่าเป็นจำนวนที่น้อยเกินไปในการฝึกอบรมโมเดลการเรียนรู้ของเครื่องให้มีประสิทธิภาพและความแม่นยำสูง ในกรณีที่ข้อมูลมีจำนวนจำกัด การทำ Data Augmentation จะช่วยขยายชุดข้อมูลให้มีความหลากหลายมากขึ้น การเพิ่มข้อมูลจะทำให้ได้ตัวอย่างใหม่ๆ จากข้อมูลเดิม โดยการปรับเปลี่ยนลักษณะของภาพ เช่น การหมุนภาพ การปรับแสง หรือการเปลี่ยนขนาด ซึ่งจะช่วยให้โมเดลสามารถเรียนรู้ลักษณะที่หลากหลายและลดความเสี่ยงการเกิดปัญหา Overfitting ที่อาจเกิดขึ้นได้ หากโมเดลถูกฝึกด้วยข้อมูลที่มีความหลากหลายไม่เพียงพอ การทำ Data Augmentation จะช่วยให้โมเดลมีความสามารถในการสรุปผลในสถานการณ์ที่หลากหลาย โดยเฉพาะอย่างยิ่งเมื่อเผชิญกับข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน นอกจากนี้ ยังสามารถช่วยจัดการกับปัญหาความไม่สมดุลของคลาส ซึ่งอาจเกิดขึ้นได้ในกับการศึกษาค้นคว้าอิสระในครั้งนี้ เนื่องจากข้อมูลที่มีอยู่จำกัด การเพิ่มข้อมูลสำหรับคลาสที่มีการแสดงน้อยจะทำให้สามารถปรับปรุงความสามารถของโมเดลในการจำแนกประเภทได้ดีขึ้น

3.1.4 การสกัดคุณลักษณะ (Feature Extraction)

การศึกษาค้นคว้าอิสระครั้งนี้ได้เลือกใช้การเรียนรู้แบบถ่ายโอน (Transfer Learning) จากโครงสร้างของโครงข่ายประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Network) ที่ถูกฝึกสอนด้วยข้อมูลจาก ImageNet ซึ่งประกอบไปด้วย VGG16, ResNet50, และ InceptionV3 เพื่อสกัดคุณลักษณะ (Feature Extraction) จากภาพถ่าย

การเลือกใช้ VGG16 แทน VGG19 ในการศึกษาค้นคว้าอิสระครั้งนี้ เนื่องจาก VGG16 มีจำนวนเลเยอร์และพารามิเตอร์น้อยกว่า ทำให้การประมวลผลรวดเร็วขึ้นและใช้ทรัพยากรน้อยกว่า ซึ่งเหมาะสมกับงานที่มีข้อจำกัดทางทรัพยากร นอกจากนี้ VGG16 ยังช่วยลดโอกาสการเกิด Overfitting ได้ดี เนื่องจากพารามิเตอร์ที่น้อยกว่า ทำให้โมเดลสามารถเรียนรู้จากข้อมูลที่มีขนาดจำกัดได้ดียิ่งขึ้น ในขณะเดียวกัน ResNet50 และ InceptionV3 ก็เป็นตัวเลือกที่ดีในการสกัดคุณลักษณะ (Feature Extraction) จากภาพ เนื่องจากทั้งสองโมเดลมีความสามารถในการจัดการกับปัญหาความลึกของโมเดลได้อย่างมีประสิทธิภาพ โดย ResNet50 ใช้การเชื่อมต่อข้ามเลเยอร์ (Residual Connections) เพื่อช่วยให้การฝึกสอนในโมเดลที่ลึกขึ้นสามารถทำได้ง่ายขึ้น ขณะที่ InceptionV3 ใช้โครงสร้างที่มีหลายขนาดของฟิลเตอร์ในแต่ละเลเยอร์เพื่อจับคุณลักษณะที่หลากหลายของภาพได้ดี

ทั้งสามโมเดลนี้มีหลักการทำงานที่แตกต่างกันและมีจุดเด่นเฉพาะตัว ทำให้การนำมาทดสอบเปรียบเทียบกันจะช่วยในการค้นหาโมเดลที่มีประสิทธิภาพสูงสุดในการตรวจจับความเสียหายจากภาพถ่ายของรถยนต์

3.1.5 ขั้นตอนการจำแนกประเภท (Classification)

ขั้นตอนสุดท้ายของการจำแนกประเภท (Classification) โมเดลจะรับข้อมูลที่ได้จากการสกัดคุณลักษณะ (Feature Extraction) ซึ่งได้รับการสร้างจากโครงสร้างของคอนโวลูชัน (Convolutional Network) โดยไม่มีการฝึกสอนในชุดข้อมูลเป้าหมาย โมเดลจะใช้น้ำหนัก (Weights) ที่เรียนรู้มาจากฐานข้อมูล ImageNet เพื่อส่งคุณลักษณะที่ได้มายังส่วนการจำแนกประเภท ในขั้นตอนนี้ได้ดำเนินการสร้าง custom head โดยการเพิ่มชั้น pooling และ outputs dense = 1 เพื่อให้โมเดลสามารถทำนายประเภทของภาพถ่ายได้อย่างแม่นยำ โดยใช้ Activation sigmoid เพื่อให้ได้ผลลัพธ์ในรูปแบบของความน่าจะเป็น (Probability) ในขั้นตอนนี้ โมเดลจะถูกฝึกสอน (Train) ด้วยชุดข้อมูลฝึกสอน (Train Set) เพื่อให้โมเดลสามารถเรียนรู้และสามารถทำนายชุดข้อมูลทดสอบ (Test Set) ที่เตรียมสำหรับทดสอบได้อย่างแม่นยำและมีประสิทธิภาพที่สูง

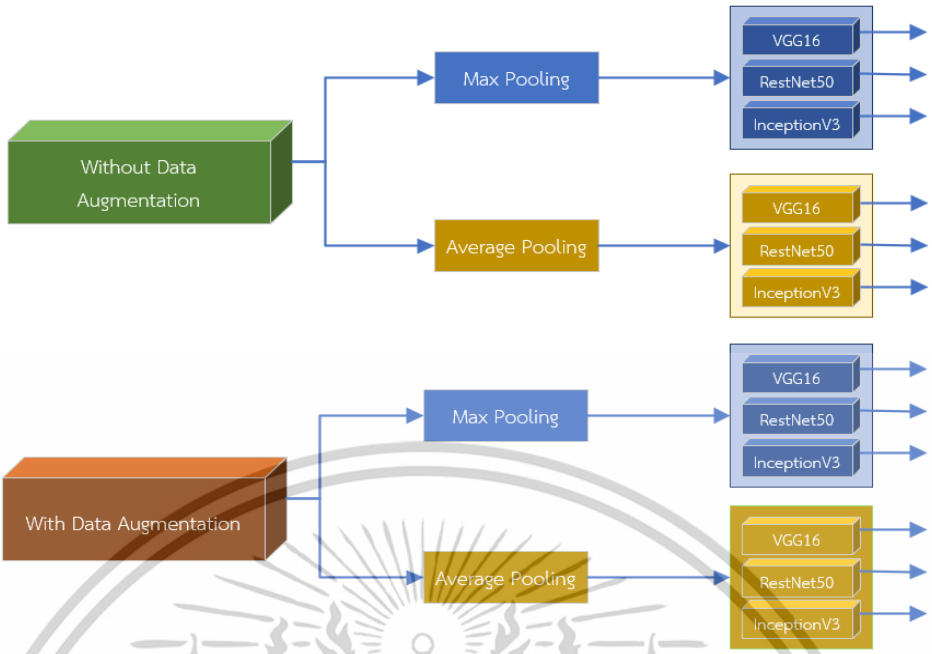
3.2 การทดลองปรับพารามิเตอร์และเปรียบเทียบประสิทธิภาพ

ขั้นตอนนี้ได้ทำการทดลองเพื่อปรับจูนพารามิเตอร์ต่างๆ ได้แก่ Max Pooling, Average Pooling เพื่อปรับให้โมเดลมีประสิทธิภาพที่ดีที่สุด โดยจะเปรียบเทียบผลพารามิเตอร์ต่างๆ ดังนี้

3.2.1 การเปรียบเทียบระหว่าง Max Pooling และ Average Pooling

ได้มีการทดลองเปรียบเทียบโดยการปรับพารามิเตอร์เปลี่ยนชั้นประมวลผล Pooling ของโมเดล ระหว่าง Max Pooling และ Average Pooling เพื่อเพิ่มประสิทธิภาพในงาน Classification และได้ลองทดลองทั้งโครงสร้าง 3 โมเดล ประกอบด้วย VGG16, ResNet50, และ InceptionV3 และกำหนดพารามิเตอร์ค่า Batch size ที่ 32 ค่า Learning rate เป็น RMSprop เท่ากับ 0.0001 และกำหนดรอบการเรียนรู้ที่ 500 รอบ (epochs) หลังจากขั้นตอนการทดลองปรับพารามิเตอร์ (Parameters) โดยการเปลี่ยนชั้นประมวลผล Pooling แล้ว ขั้นตอนถัดไปคือการปรับแต่งละเอียด (Fine Tune) แต่ละโมเดลที่ได้รับการปรับปรุงแล้ว เพื่อเพิ่มประสิทธิภาพและความแม่นยำในงาน Classification

การศึกษาค้นคว้าในครั้งนี้ ได้มีการปรับเปลี่ยนพารามิเตอร์ชั้นประมวลผลแบบ Average Pooling มาใช้ในการทดสอบเพื่อเปรียบเทียบกับชั้นประมวลผลแบบ Max Pooling ในการตรวจจับภาพถ่ายความเสียหายของรถยนต์ เนื่องจาก ชั้นประมวลผลแบบ Average Pooling มีคุณสมบัติในการเก็บข้อมูลแบบครอบคลุม ซึ่งช่วยให้โมเดลสามารถรักษาคุณลักษณะภาพความเสียหายที่กระจายตัวได้หลากหลายโดยไม่มุ่งเน้นเพียงค่าความเข้มสูงสุดเพียงจุดเดียว การเก็บค่าเฉลี่ยนี้ยังช่วยลดผลกระทบจาก Noise ที่อาจเกิดขึ้นในภาพ เช่น จุดสะท้อนหรือจุดที่มีค่าสูงผิดปกติซึ่งไม่เกี่ยวข้องกับความเสียหายจริง ทำให้การตรวจจับแม่นยำมากยิ่งขึ้น ในกรณีที่มีข้อมูลภาพจำกัดเพียง 1,898 รูป การใช้ ชั้นประมวลผลแบบ Average Pooling ช่วยลดการเกิด Overfitting ทำให้โมเดลสามารถ Generalize ได้ดีขึ้น โดยไม่จำเป็นต้องพึ่งพาข้อมูลจำนวนมาก โมเดลจึงมีความยืดหยุ่นและพร้อมสำหรับการทำนายกับข้อมูลใหม่ๆ นอกจากนี้ โมเดลที่ใช้ในการทดสอบ เช่น VGG16, ResNet50, และ InceptionV3 มีโครงสร้างที่ซับซ้อนอยู่แล้ว การใช้ Average Pooling จึงช่วยลดความซับซ้อนของข้อมูลที่ส่งต่อไปยังชั้นลึกๆ ทำให้โมเดลสามารถเรียนรู้ลักษณะของความเสียหายได้ง่ายขึ้นโดยไม่ต้องประมวลผลข้อมูลที่ซับซ้อนเกินไป และการปรับค่า Hyperparameter ร่วมกับการใช้ Average Pooling เช่น Learning Rate และ Optimizer ยังเป็นปัจจัยสำคัญที่ช่วยให้โมเดลสามารถทำงานได้อย่างมีประสิทธิภาพสูงสุด การทดลองปรับเปลี่ยนค่าเหล่านี้เพื่อหาค่าที่เหมาะสมจะช่วยให้โมเดลมีความแม่นยำและมีประสิทธิภาพในการตรวจจับความเสียหายของรถยนต์ได้มากที่สุด



รูปที่ 3.5 โมเดลทั้งหมดก่อนทำ Fine Tune

จากรูปที่ 3.5 สรุปได้ว่าในการศึกษาค้นคว้าอิสระในครั้งนี้ มีการทดสอบเพื่อวัดประสิทธิภาพของโมเดลทั้งหมดจำนวน 12 โมเดล เพื่อเปรียบเทียบประสิทธิภาพและหาโมเดลที่เหมาะสมที่สุดในการนำไปใช้กับการตรวจจับความเสียหายของรถยนต์

3.3 ขั้นตอนการดำเนินการ Fine Tune

หลังจากทำการทดลองปรับพารามิเตอร์ได้ผลลัพธ์ทุกโมเดลแล้วจึงดำเนินการ Fine Tune เพื่อเพิ่มประสิทธิภาพของโมเดลโดยจะทำการนำชุดข้อมูลทั้งแบบไม่มีการแปลงข้อมูลต้นฉบับและข้อมูลแบบการดัดแปลงข้อมูลต้นฉบับ (Data Augmentation) เข้าไปฝึกสอนในชั้น (Layer) ท้ายๆ ของทั้งโครงสร้าง 3 โมเดล ประกอบด้วย VGG16, ResNet50, และ InceptionV3 โดยดำเนินการ ดังนี้

3.3.1 โมเดล VGG16

จากโมเดล VGG16 มีจำนวนชั้น (Layer) ของโมเดลทั้งหมด 19 ชั้น (layer) รวม Max Pooling Layer หรือ Average Pooling Layer และ Outputs Dense รวมทั้งหมด 21 ชั้น ได้ดำเนินการทำการปรับแต่งชั้นเพื่อฝึกสอน (train) โมเดลตั้งแต่ชั้นที่ 16 ถึงชั้นที่ 21 รวมทั้งหมด 6 ชั้น

3.3.2 โมเดล ResNet50

จากโมเดล ResNet50 มีจำนวนชั้น (Layer) ของโมเดลมีทั้งหมด 175 ชั้น (layer) รวม Max Pooling Layer หรือ Average Pooling และ Outputs Dense รวมทั้งหมด 177 ชั้น ได้ดำเนินการทำการปรับแต่งชั้นเพื่อฝึกสอน (train) โมเดลตั้งแต่ชั้นที่ 144 ถึงชั้นที่ 177 รวมทั้งหมด 34 ชั้น

3.3.3 โมเดล InceptionV3

จากโมเดล InceptionV3 มีจำนวนชั้น (Layer) ของโมเดลมีทั้งหมด 311 ชั้น (layer) รวม Max Pooling Layer หรือ Average Pooling และ Outputs Dense รวมทั้งหมด 313 ชั้น ได้ดำเนินการทำการปรับแต่งชั้นเพื่อฝึก (train) โมเดลตั้งแต่ชั้นที่ 281 ถึงชั้นที่ 313 รวมทั้งหมด 33 ชั้น

โดยทุกโมเดลได้ทำการ Compile โมเดล optimizer RMSprop learning late 0.0001 ทำการฝึกสอน 500 epochs ตามเดิม และกำหนดให้สิ้นสุดการฝึกโดยใช้ early stopping กำหนดค่า loss ในชุดข้อมูลทดสอบ (Test) หากไม่ลดลง 3 รอบสุดท้ายของการฝึกสอน (patience = 3) ผลลัพธ์ที่ได้นั้นจะกล่าว ในหัวข้อผลการทดลองในหัวข้อถัดไป

บทที่ 4
ผลการทดลอง

จากการศึกษาค้นคว้าอิสระในบทที่ 3 ได้แสดงวิธีการดำเนินการในการทดลองและในบทนี้จะเป็นการแสดงผลลัพธ์จากการใช้งานการเรียนรู้แบบถ่ายโอน (Transfer Learning) บนโมเดลประสิทธิภาพสูง ได้แก่ VGG16, ResNet50 และ InceptionV3 โดยใช้ weights จาก ImageNet เพื่อให้เข้ากับการจำแนกภาพถ่ายในงานโครงงานนี้ โดยโมเดลทั้งสามได้รับการปรับปรุงด้วยการเพิ่ม Custom Head ที่ประกอบไปด้วยการใช้ Max Pooling Layer และ Average Pooling พร้อมทั้งเพิ่มชั้น Dense Layer เท่ากับ 1 และใช้ฟังก์ชันเอาต์พุตเป็น sigmoid เพื่อการจำแนกผลลัพธ์เป็นสองคลาส การ Compile โมเดลทำโดยใช้ Optimizer อย่าง RMSprop ที่มี Learning Rate 0.0001 และการวัดผลใช้ loss function เป็น binary crossentropy ประสิทธิภาพของโมเดลจะวัดจากความแม่นยำ (Accuracy) ในการทดลองระหว่างการฝึกสอนที่ 500 epochs โดยกำหนดแนวทางผลการทดลอง ดังนี้

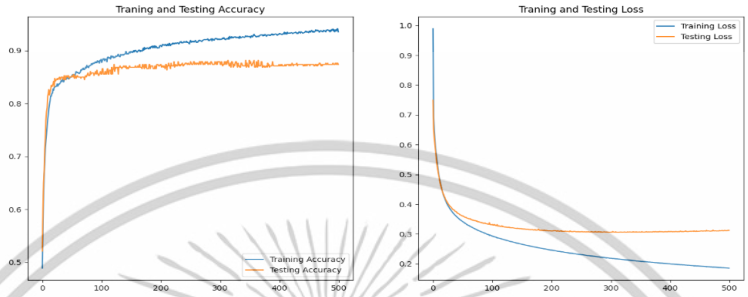
4.1 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)
4.2 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และแปลงข้อมูลต้นฉบับ (With Data Augmentation)
4.3 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)
4.4 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และแปลงข้อมูลต้นฉบับ (With Data Augmentation)
4.5 เปรียบเทียบประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling
4.6 เปรียบเทียบประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling
4.7 เปรียบเทียบประสิทธิภาพของโมเดล หลังจาก Fine Tune

4.1 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)

จากการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล มีการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation) ทำการเปรียบเทียบผลลัพธ์ประสิทธิภาพโมเดล ได้ผลการทดลองตามดังนี้

4.1.1 ผลลัพธ์ประสิทธิภาพของโมเดล VGG16 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.87 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.31 ตามรูปที่ 4.1 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.87 Precision เท่ากับ 0.90 Recall เท่ากับ 0.87 และ F1-Score เท่ากับ 0.88 ตามรูปที่ 4.2



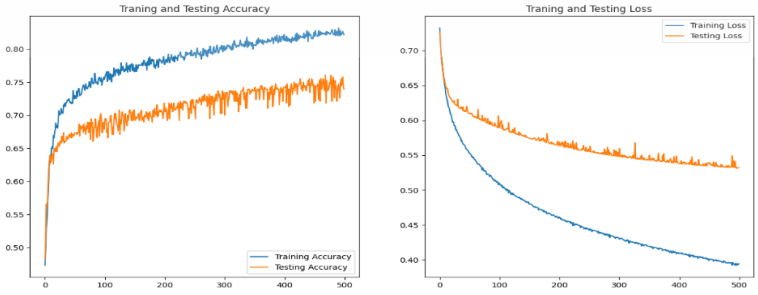
รูปที่ 4.1 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.88	0.84	0.86	179		
Damage	0.87	0.90	0.88	201		
accuracy			0.87	380		
macro avg	0.87	0.87	0.87	380		
weighted avg	0.87	0.87	0.87	380		

รูปที่ 4.2 Classification Report ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

4.1.2 ผลลัพธ์ประสิทธิภาพของโมเดล ResNet50 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.74 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.53 ตามรูปที่ 4.3 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.74 Precision เท่ากับ 0.74 Recall เท่ากับ 0.76 และ F1-Score เท่ากับ 0.75 ตามรูปที่ 4.4



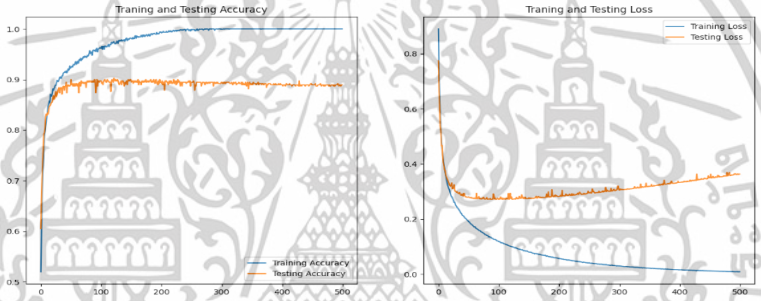
รูปที่ 4.3 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.72	0.74	0.73	179		
Damage	0.76	0.74	0.75	201		
accuracy			0.74	380		
macro avg	0.74	0.74	0.74	380		
weighted avg	0.74	0.74	0.74	380		

รูปที่ 4.4 Classification Report ของโมเดล ResNet50
ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

4.1.3 ผลลัพธ์ประสิทธิภาพของโมเดล InceptionV3 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.90 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.36 ตามรูปที่ 4.5 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.89 Precision เท่ากับ 0.90 Recall เท่ากับ 0.89 และ F1-Score เท่ากับ 0.90 ตามรูปที่ 4.6



รูปที่ 4.5 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.89	0.88	0.88	179		
Damage	0.89	0.90	0.90	201		
accuracy			0.89	380		
macro avg	0.89	0.89	0.89	380		
weighted avg	0.89	0.89	0.89	380		

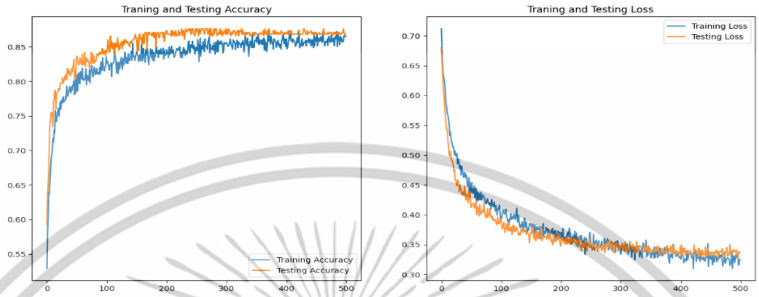
รูปที่ 4.6 Classification Report ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation

4.2 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และแปลงข้อมูลต้นฉบับ (With Data Augmentation)

จากการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล มีการเปลี่ยนชั้นประมวลผลเป็น Max Pooling และในขั้นตอน Preprocessing ได้ เพิ่มขั้นตอนการทำแปลงข้อมูลต้นฉบับ (With Data Augmentation) กับชุดข้อมูลฝึกสอน (training set) ทำการเปรียบเทียบผลลัพธ์ประสิทธิภาพโมเดล ได้ผลการทดลองตามดังนี้

4.2.1 ผลลัพธ์ประสิทธิภาพของโมเดล VGG16 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.87 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.33 ตามรูปที่ 4.7 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.87 Precision เท่ากับ 0.85 Recall เท่ากับ 0.90 และ F1-Score เท่ากับ 0.87 ตามรูปที่ 4.8



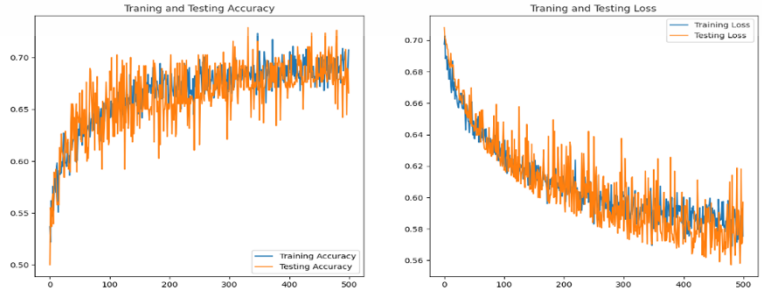
รูปที่ 4.7 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

Classification_report	precision	recall	f1-score	support
Nodamage	0.84	0.89	0.86	179
Damage	0.90	0.85	0.87	201
accuracy			0.87	380
macro avg	0.87	0.87	0.87	380
weighted avg	0.87	0.87	0.87	380

รูปที่ 4.8 Classification Report ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

4.2.2 ผลลัพธ์ประสิทธิภาพของโมเดล ResNet50 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.67 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.60 ตามรูปที่ 4.9 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.67 Precision เท่ากับ 0.50 Recall เท่ากับ 0.79 และ F1-Score เท่ากับ 0.61 ตามรูปที่ 4.10



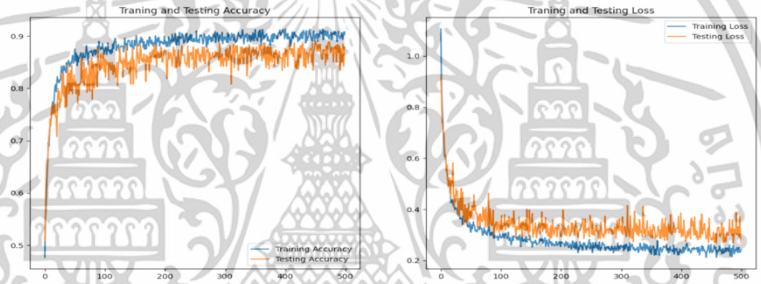
รูปที่ 4.9 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

Classification_report	precision	recall	f1-score	support
Nodamage	0.60	0.85	0.71	179
Damage	0.79	0.50	0.61	201
accuracy			0.67	380
macro avg	0.70	0.68	0.66	380
weighted avg	0.70	0.67	0.66	380

รูปที่ 4.10 Classification Report ของโมเดล ResNet50
ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

4.2.3 ผลลัพธ์ประสิทธิภาพของโมเดล InceptionV3 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.89 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.36 ตามรูปที่ 4.11 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.88 Precision เท่ากับ 0.88 Recall เท่ากับ 0.89 และ F1-Score เท่ากับ 0.88 ตามรูปที่ 4.12



รูปที่ 4.11 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

Classification_report	precision	recall	f1-score	support
Nodamage	0.86	0.88	0.87	179
Damage	0.89	0.88	0.88	201
accuracy			0.88	380
macro avg	0.88	0.88	0.88	380
weighted avg	0.88	0.88	0.88	380

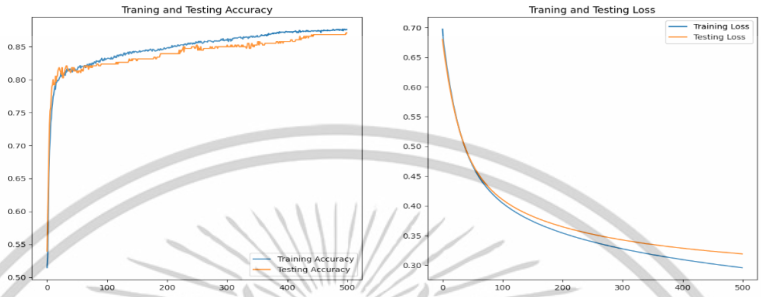
รูปที่ 4.12 Classification Report ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation

4.3 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)

จากการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล มีการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation) ทำการเปรียบเทียบผลลัพธ์ประสิทธิภาพโมเดล ได้ผลการทดลองตามดังนี้

4.3.1 ผลลัพธ์ประสิทธิภาพของโมเดล VGG16 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.87 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.32 ตามรูปที่ 4.13 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.87 Precision เท่ากับ 0.90 Recall เท่ากับ 0.87 และ F1-Score เท่ากับ 0.88 ตามรูปที่ 4.14



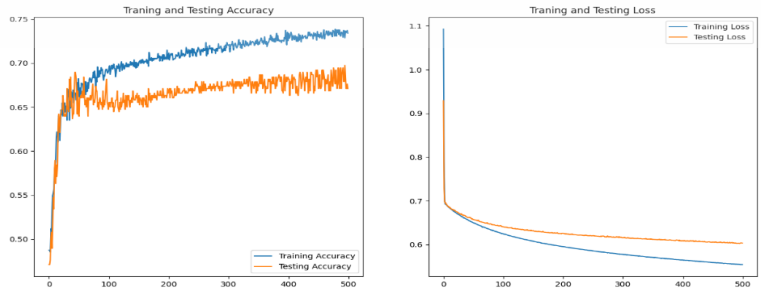
รูปที่ 4.13 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

Classification_report	precision	recall	f1-score	support
Nodamage	0.88	0.84	0.86	179
Damage	0.87	0.90	0.88	201
accuracy			0.87	380
macro avg	0.87	0.87	0.87	380
weighted avg	0.87	0.87	0.87	380

รูปที่ 4.14 Classification Report ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

4.3.2 ผลลัพธ์ประสิทธิภาพของโมเดล ResNet50 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.67 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.60 ตามรูปที่ 4.15 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.67 Precision เท่ากับ 0.71 Recall เท่ากับ 0.68 และ F1-Score เท่ากับ 0.69 ตามรูปที่ 4.16



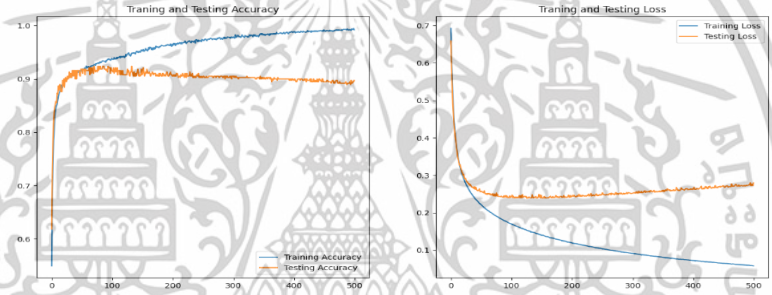
รูปที่ 4.15 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.66	0.63	0.64	179		
Damage	0.68	0.71	0.69	201		
accuracy			0.67	380		
macro avg	0.67	0.67	0.67	380		
weighted avg	0.67	0.67	0.67	380		

รูปที่ 4.16 Classification Report ของโมเดล ResNet50
ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

4.3.3 ผลลัพธ์ประสิทธิภาพของโมเดล InceptionV3 Without Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.90 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.28 ตามรูปที่ 4.17 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.90 Precision เท่ากับ 0.88 Recall เท่ากับ 0.93 และ F1-Score เท่ากับ 0.90 ตามรูปที่ 4.18



รูปที่ 4.17 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.87	0.92	0.89	179		
Damage	0.93	0.88	0.90	201		
accuracy			0.90	380		
macro avg	0.90	0.90	0.90	380		
weighted avg	0.90	0.90	0.90	380		

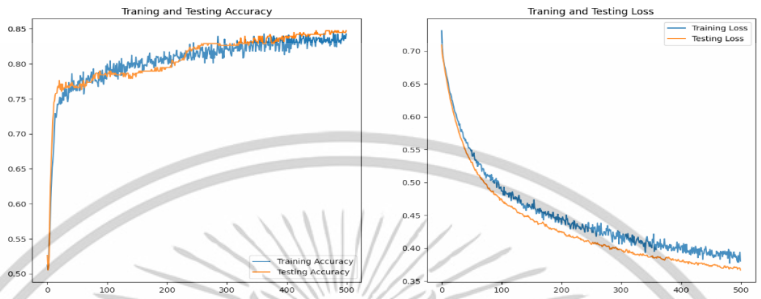
รูปที่ 4.18 Classification Report ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation

4.4 ผลลัพธ์ประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และแปลงข้อมูลต้นฉบับ (With Data Augmentation)

จากการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล มีการเปลี่ยนชั้นประมวลผลเป็น Average Pooling และในขั้นตอน Preprocessing ได้ เพิ่มขั้นตอนการทำแปลงข้อมูลต้นฉบับ (With Data Augmentation) กับชุดข้อมูลฝึกสอน (Training Set) ทำการเปรียบเทียบผลลัพธ์ประสิทธิภาพโมเดลได้ผลการทดลองตามดังนี้

4.4.1 ผลลัพธ์ประสิทธิภาพของโมเดล VGG16 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.84 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.37 ตามรูปที่ 4.19 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.85 Precision เท่ากับ 0.80 Recall เท่ากับ 0.90 และ F1-Score เท่ากับ 0.85 ตามรูปที่ 4.20



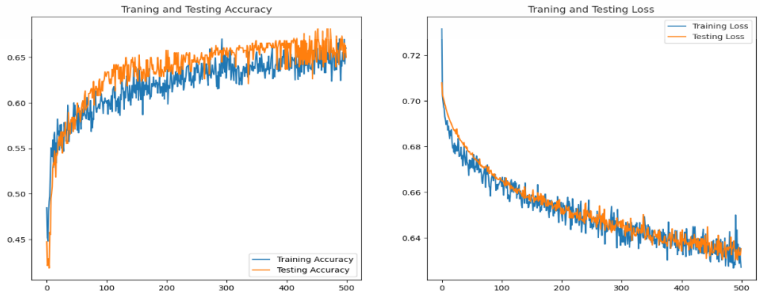
รูปที่ 4.19 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

Classification_report	precision	recall	f1-score	support
Nodamage	0.80	0.90	0.85	179
Damage	0.90	0.80	0.85	201
accuracy			0.85	380
macro avg	0.85	0.85	0.85	380
weighted avg	0.85	0.85	0.85	380

รูปที่ 4.20 Classification Report ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

4.4.2 ผลลัพธ์ประสิทธิภาพของโมเดล ResNet50 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.65 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.64 ตามรูปที่ 4.21 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.65 Precision เท่ากับ 0.56 Recall เท่ากับ 0.72 และ F1-Score เท่ากับ 0.63 ตามรูปที่ 4.22



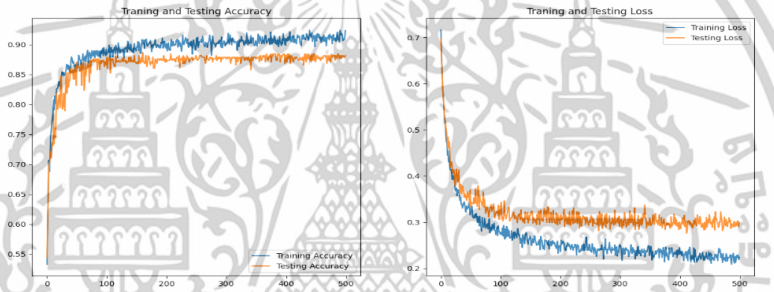
รูปที่ 4.21 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.60	0.75	0.67	179		
Damage	0.72	0.56	0.63	201		
accuracy			0.65	380		
macro avg	0.66	0.66	0.65	380		
weighted avg	0.66	0.65	0.65	380		

รูปที่ 4.22 Classification Report ของโมเดล ResNet50
ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

4.4.3 ผลลัพธ์ประสิทธิภาพของโมเดล InceptionV3 With Data Augmentation

จากกราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss กับ epochs ได้ผลลัพธ์ค่าความแม่นยำ (Accuracy Validation) เท่ากับ 0.88 ค่าความผิดพลาด (Loss Validation) เท่ากับ 0.30 ตามรูปที่ 4.23 และส่วนค่า Accuracy ที่ได้เท่ากับ 0.88 Precision เท่ากับ 0.84 Recall เท่ากับ 0.93 และ F1-Score เท่ากับ 0.88 ตามรูปที่ 4.24



รูปที่ 4.23 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

Classification_report			precision	recall	f1-score	support
Nodamage	0.83	0.93	0.88	179		
Damage	0.93	0.84	0.88	201		
accuracy			0.88	380		
macro avg	0.88	0.88	0.88	380		
weighted avg	0.89	0.88	0.88	380		

รูปที่ 4.24 Classification Report ของโมเดล InceptionV3
ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation

4.5 เปรียบเทียบประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Max Pooling

จากผลการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling และใช้ข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) สามารถสรุปผลลัพธ์จากตาราง Confusion Matrix ในแต่ละโมเดลตามภาพรูปที่ 4.25 เพื่อนำไปเปรียบเทียบผลทดลองได้ดังนี้

โมเดล VGG16

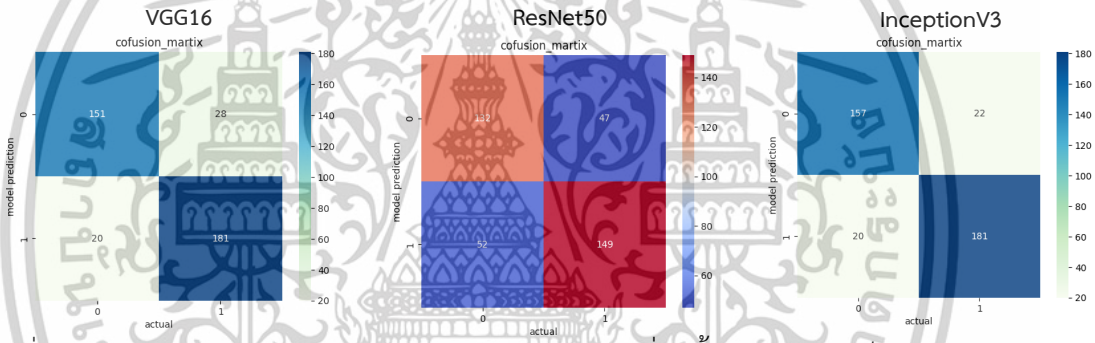
- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 181 คัน ทายผิด 20 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 151 คัน ทายผิด 28 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 332 คัน ทายผิด 48 คัน

โมเดล ResNet50

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 149 คัน ทายผิด 52 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 132 คัน ทายผิด 47 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 271 คัน ทายผิด 99 คัน

โมเดล InceptionV3

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 181 คัน ทายผิด 20 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 157 คัน ทายผิด 22 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 338 คัน ทายผิด 42 คัน



รูปที่ 4.25 ตาราง Confusion Matrix จากผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation ของ 3 โมเดล

จากผลการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling และใช้ข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) สามารถสรุปผลลัพธ์จากตาราง Confusion Matrix ในแต่ละโมเดลตามรูปที่ 4.26 เพื่อนำไปเปรียบเทียบผลทดลองได้ ดังนี้

โมเดล VGG16

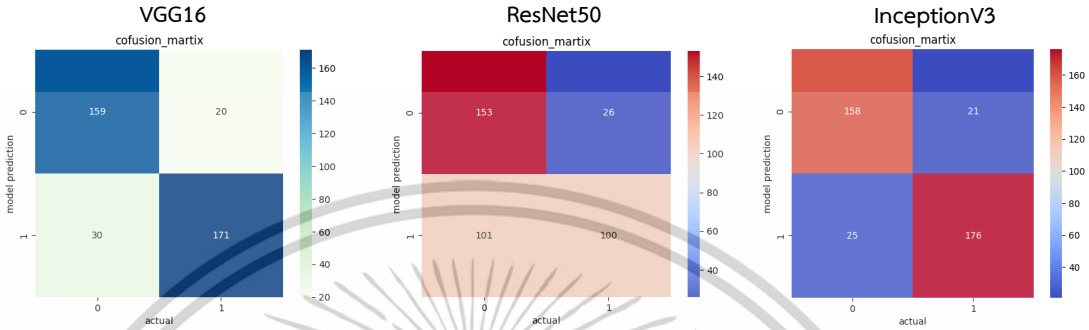
- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 171 คัน ทายผิด 30 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 159 คัน ทายผิด 20 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 330 คัน ทายผิด 50 คัน

โมเดล ResNet50

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 101 คัน ทายผิด 100 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 153 คัน ทายผิด 26 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 254 คัน ทายผิด 126 คัน

โมเดล InceptionV3

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 176 คัน ทายผิด 25 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 158 คัน ทายผิด 21 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 334 คัน ทายผิด 46 คัน



รูปที่ 4.26 ตาราง Confusion Matrix จากผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation ของ 3 โมเดล

จากตาราง Confusion Matrix ของแต่ละโมเดล โดยการทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling และทดลองใช้ข้อมูลที่ไม่มีแปลงข้อมูลต้นฉบับ (Without Data Augmentation) และใช้ข้อมูลที่มีแปลงข้อมูลต้นฉบับ (With Data Augmentation) สามารถสรุปค่า Accuracy , Precision และ Recall เพื่อเปรียบเทียบประสิทธิภาพได้ตามตารางที่ 4.1

ตารางที่ 4.1 ตารางเปรียบเทียบผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling ของ 3 โมเดล

Model	Without Data Augmentation			
	Accuracy	Precision	Recall	F1-Score
VGG16	0.8737	0.9005	0.8660	0.8829
ResNet50	0.7395	0.7413	0.7602	0.7506
InceptionV3	0.8895	0.9005	0.8916	0.8960
Model	**With Data Augmentation**			
	Accuracy	Precision	Recall	F1-Score
VGG16	0.8684	0.8507	0.8953	0.8724
ResNet50	0.6658	0.4975	0.7937	0.6116
InceptionV3	0.8789	0.8756	0.8934	0.8844

จากตารางที่ 4.1 หากทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling และทดลองใช้ข้อมูลที่ไม่มีแปลงข้อมูลต้นฉบับ (Without Data Augmentation) นั้นได้แสดงให้เห็นว่า โมเดล InceptionV3 มีประสิทธิภาพดีที่สุดสำหรับข้อมูล ชุดนี้ และรองลงมาคือ VGG16 และ Resnet50 ตามลำดับ โดยมี Accuracy 0.8895, 0.8737 และ 0.7395 ตามลำดับ และสำหรับชุดข้อมูลที่เพิ่มขั้นตอนการทำแปลงข้อมูลต้นฉบับ (With Data Augmentation) ในขั้นตอน Preprocessing กับชุดข้อมูลฝึกสอน (Training Set) โมเดลที่มีประสิทธิภาพดีที่สุดคือ โมเดล InceptionV3 , VGG16 และ ResNet50 ตามลำดับ โดยมี Accuracy 0.8789, 0.8684 และ 0.6658

4.6 เปรียบเทียบประสิทธิภาพของโมเดล โดยการเปลี่ยนชั้นประมวลผลเป็น Average Pooling

จากผลการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling และใช้ข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) สามารถสรุปผลลัพธ์จากตาราง Confusion Matrix ในแต่ละโมเดลตามรูปที่ 4.27 เพื่อนำไปเปรียบเทียบผลทดลองได้ ดังนี้

โมเดล VGG16

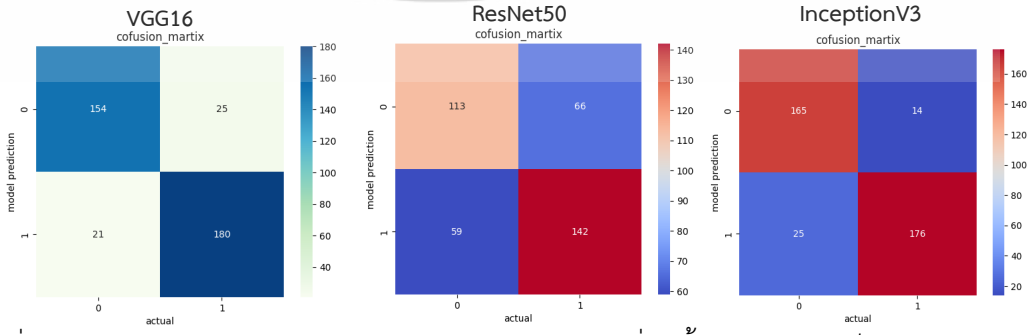
- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 180 คัน ทายผิด 21 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 154 คัน ทายผิด 25 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 334 คัน ทายผิด 46 คัน

โมเดล ResNet50

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 142 คัน ทายผิด 59 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 113 คัน ทายผิด 66 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 255 คัน ทายผิด 125 คัน

โมเดล InceptionV3

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 176 คัน ทายผิด 25 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 165 คัน ทายผิด 14 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 341 คัน ทายผิด 39 คัน



รูปที่ 4.27 ตาราง Confusion Matrix จากผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation ของ 3 โมเดล

จากผลการทดลองฝึกสอนโมเดลทั้ง 3 โมเดล ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling และใช้ข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) สามารถสรุปผลลัพธ์จากตาราง Confusion Matrix ในแต่ละโมเดลตามรูปที่ 4.28 เพื่อนำไปเปรียบเทียบผลทดลองได้ ดังนี้

โมเดล VGG16

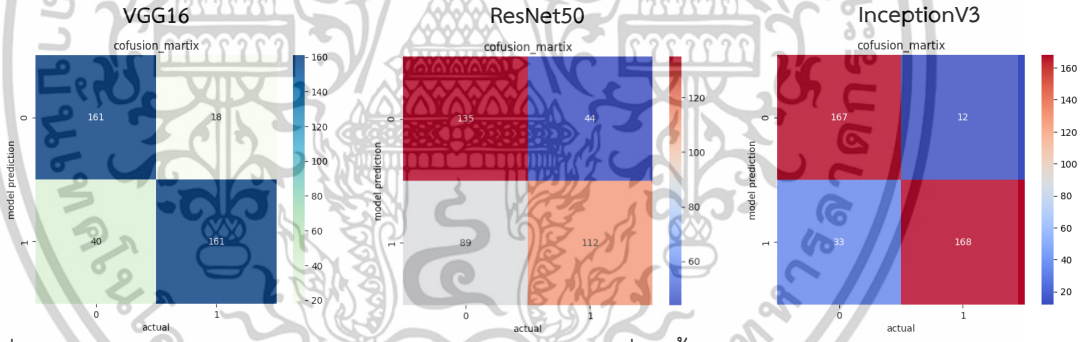
- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 161 คัน ทายผิด 40 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 161 คัน ทายผิด 18 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 322 คัน ทายผิด 58 คัน

โมเดล ResNet50

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 112 คัน ทายผิด 89 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 135 คัน ทายผิด 44 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 247 คัน ทายผิด 133 คัน

โมเดล InceptionV3

- ทายภาพถ่ายของรถยนต์ที่มีความเสียหายถูกต้อง 168 คัน ทายผิด 33 คัน
- ทายภาพถ่ายของรถยนต์ที่ไม่มีความเสียหายถูกต้อง 167 คัน ทายผิด 12 คัน
- รวมทายภาพถ่ายของรถยนต์ถูกต้องทั้งหมด 335 คัน ทายผิด 45 คัน



รูปที่ 4.28 ตาราง Confusion Matrix จากผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation ของ 3 โมเดล

จากตาราง Confusion Matrix ของแต่ละโมเดล โดยการทดลองเปลี่ยนชั้นประมวลผลเป็น Average Pooling และทดลองใช้ข้อมูลที่ไม่มีแปลงข้อมูลต้นฉบับ (Without Data Augmentation) และใช้ข้อมูลที่มีแปลงข้อมูลต้นฉบับ (With Data Augmentation) สามารถสรุปค่า Accuracy , Precision และ Recall เพื่อเปรียบเทียบประสิทธิภาพได้ตามตารางที่ 4.2

ตารางที่ 4.2 ตารางเปรียบเทียบผลการทดลองเปลี่ยนชั้นประมวลผลเป็น Average Pooling ของ 3 โมเดล

Model	Without Data Augmentation			
	Accuracy	Precision	Recall	F1-Score
VGG16	0.8789	0.8955	0.8780	0.8867
ResNet50	0.6711	0.7065	0.6827	0.6944
InceptionV3	0.8974	0.8756	0.9263	0.9003
Model	**With Data Augmentation**			
	Accuracy	Precision	Recall	F1-Score
VGG16	0.8474	0.8010	0.8994	0.8474
ResNet50	0.6500	0.5572	0.7179	0.6275
InceptionV3	0.8816	0.8358	0.9333	0.8819

จากตารางที่ 4.2 หากทดลองเปลี่ยนชั้นประมวลผลเป็น Average Pooling และทดลองใช้ข้อมูลที่ไม่มีแปลงข้อมูลต้นฉบับ (Without Data Augmentation) นั้นได้แสดงให้เห็นว่า โมเดล InceptionV3 มีประสิทธิภาพดีที่สุด และรองลงมาคือ VGG16 และ Resnet50 ตามลำดับ โดยมี Accuracy 0.8974, 0.8789 และ 0.6711 ตามลำดับ และสำหรับชุดข้อมูลที่เพิ่มขั้นตอนการทำแปลงข้อมูลต้นฉบับ (With Data Augmentation) ในขั้นตอน Preprocessing กับชุดข้อมูลฝึกสอน (Training Set) โมเดลที่มีประสิทธิภาพดีที่สุดคือ โมเดล InceptionV3 , VGG16 และ ResNet50 ตามลำดับ โดยมี Accuracy 0.8816, 0.8474 และ 0.6500

4.7 เปรียบเทียบประสิทธิภาพของโมเดล หลังจาก Fine Tune

ผลการทดลองการเพิ่มประสิทธิภาพโมเดลด้วยวิธีการ Fine Tune โดยใช้คำสั่ง Early Stopping ให้สิ้นสุดการฝึกสอนโดยกำหนดหากพบค่า Loss ไม่เปลี่ยนแปลง 3 epoch โมเดลจะหยุดการฝึกสอนทันที

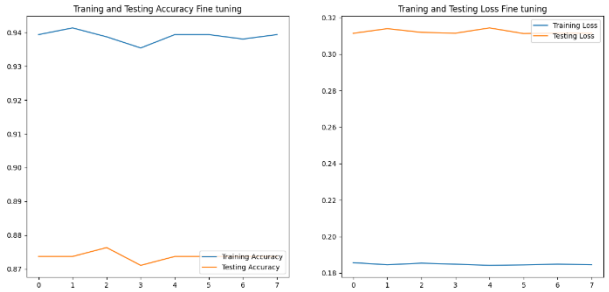
จากตารางที่ 4.3 นั้นได้แสดงให้เห็นว่าจากการทดลองกับชุดข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) พบว่าโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling และรอบการฝึกสอนจำนวน 10 epoch มีประสิทธิภาพดีที่สุด รองลงมาคือ โมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling และรอบการฝึกสอนจำนวน 4 epoch และอันดับสามคือ โมเดล VGG16 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling และรอบการฝึกสอนจำนวน 447 epoch โดยมี Accuracy 0.8921, 0.8895 และ 0.8789 ตามลำดับ

สำหรับการทดลองกับชุดข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) พบว่าโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling และรอบการฝึกสอนจำนวน 8 epoch มีประสิทธิภาพดีที่สุด รองลงมาคือ โมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling และรอบการฝึกสอนจำนวน 5 epoch และอันดับสามคือ โมเดล VGG16 ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling และรอบการฝึกสอนจำนวน 6 epoch โดยมี Accuracy 0.8816, 0.8763 และ 0.8684 ตามลำดับ

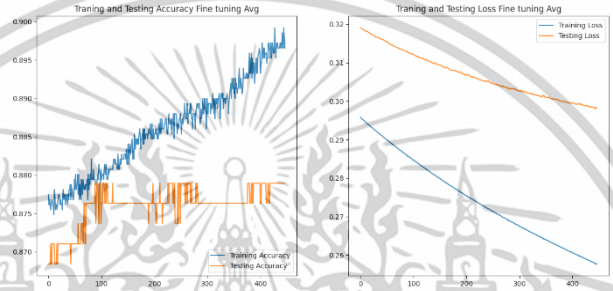
ตารางที่ 4.3 ตารางเปรียบเทียบผลการทดลองเพิ่มประสิทธิภาพ Fine Tune ของ 3 โมเดล

Model	Custom head	Without Data Augmentation				
		Accuracy	Precision	Recall	F1-Score	Epochs
VGG16	Max	0.8737	0.8905	0.8732	0.8818	8
	Average	0.8789	0.8955	0.8780	0.8867	447
ResNet50	Max	0.7474	0.7612	0.7612	0.7612	12
	Average	0.6842	0.7413	0.6866	0.7129	6
InceptionV3	Max	0.8895	0.8905	0.8995	0.8950	4
	Average	0.8921	0.8856	0.9082	0.8967	10
Model	**Custom head**	**With Data Augmentation**				
		Accuracy	Precision	Recall	F1-Score	Epochs
VGG16	Max	0.8684	0.8507	0.8953	0.8724	6
	Average	0.8475	0.7960	0.8989	0.8443	5
ResNet50	Max	0.6868	0.5572	0.7887	0.6531	6
	Average	0.6421	0.5323	0.7181	0.6114	5
InceptionV3	Max	0.8763	0.8209	0.9375	0.8753	5
	Average	0.8816	0.8358	0.9333	0.8819	8

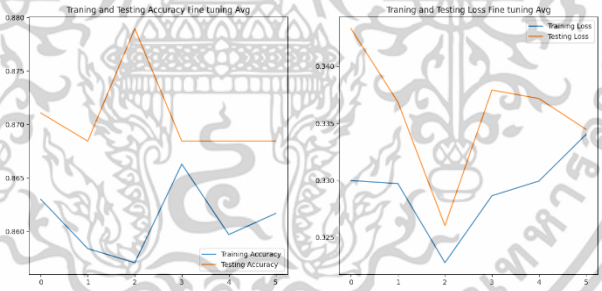
จากการทดลองปรับแต่ง Fine Tune โมเดลด้วยวิธีการเลือกชั้นที่จะฝึกสอน (Selective Layer Training) และได้มีการตรวจสอบกราฟที่แสดงความสัมพันธ์ระหว่างค่า Accuracy และค่า Loss ต่อจำนวนรอบการฝึก (epochs) ของแต่ละโมเดล ตามรูปที่ 4.29 - 4.40 ผลที่ได้จากการตรวจสอบพบว่ากราฟการเรียนรู้ของโมเดลมีความผันผวนและเกิดปัญหาการฝึกเกินจำเป็น (Overfitting) โดยเหตุผลหลักมาจากปริมาณข้อมูลในชุดฝึกสอน (Train Set) ที่มีน้อยเมื่อเปรียบเทียบกับฐานข้อมูลขนาดใหญ่อย่าง ImageNet ซึ่งถูกใช้ในการฝึกสอนโมเดลต้นแบบอย่าง VGG16, ResNet50 และ InceptionV3 ที่มีการฝึกสอนมาอย่างดีเยี่ยมแล้ว



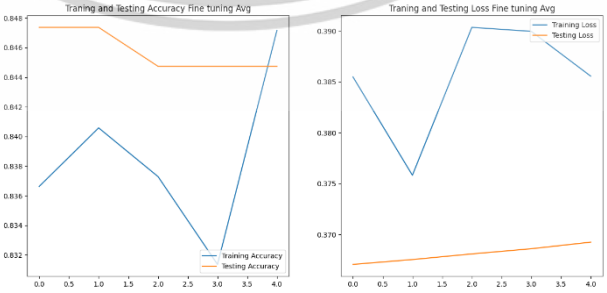
รูปที่ 4.29 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation หลังจาก Fine Tune



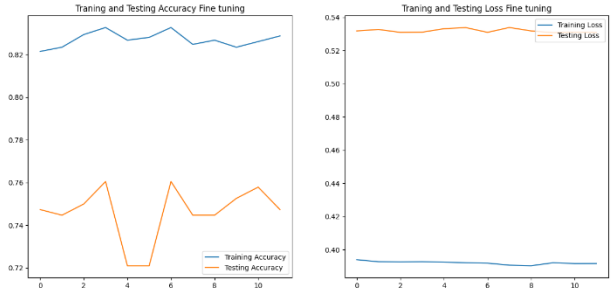
รูปที่ 4.30 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation หลังจาก Fine Tune



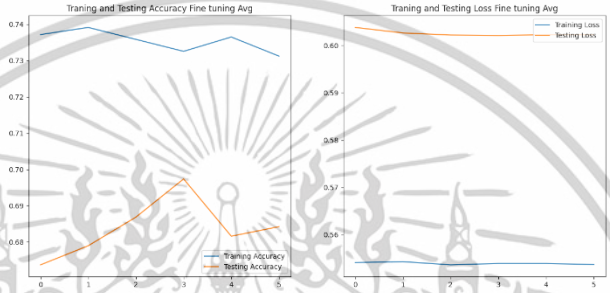
รูปที่ 4.31 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation หลังจาก Fine Tune



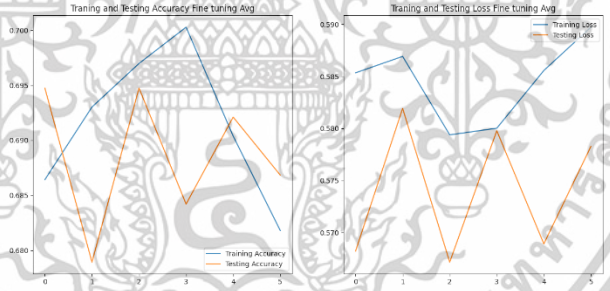
รูปที่ 4.32 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล VGG16 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation หลังจาก Fine Tune



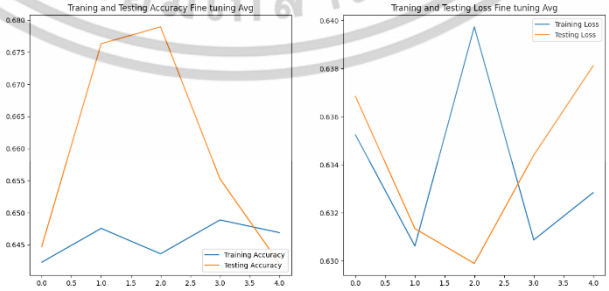
รูปที่ 4.33 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation หลังจาก Fine Tune



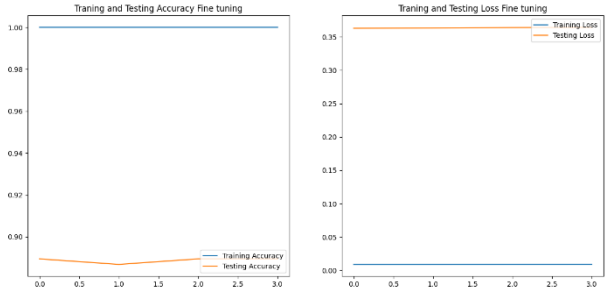
รูปที่ 4.34 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation หลังจาก Fine Tune



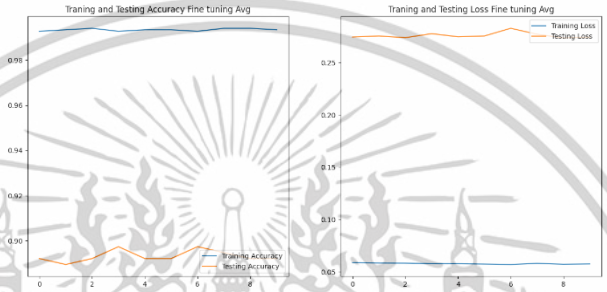
รูปที่ 4.35 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation หลังจาก Fine Tune



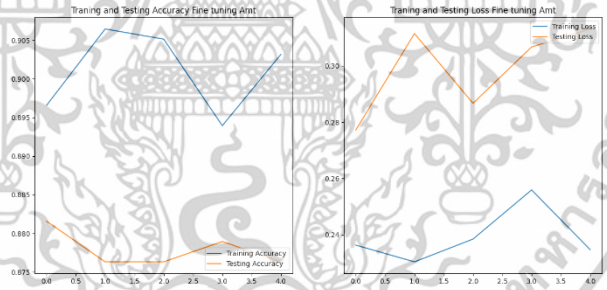
รูปที่ 4.36 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล ResNet50 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation หลังจาก Fine Tune



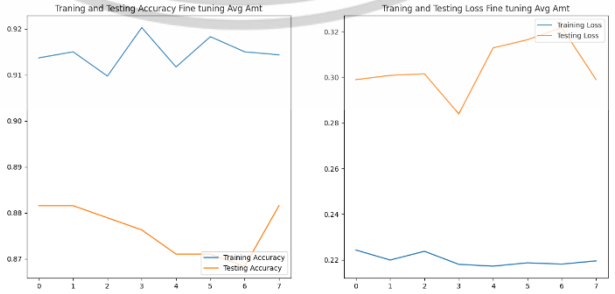
รูปที่ 4.37 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3 ชั้นประมวลผลเป็น Max Pooling และ Without Data Augmentation หลังจาก Fine Tune



รูปที่ 4.38 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3 ชั้นประมวลผลเป็น Average Pooling และ Without Data Augmentation หลังจาก Fine Tune



รูปที่ 4.39 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3 ชั้นประมวลผลเป็น Max Pooling และ With Data Augmentation หลังจาก Fine Tune



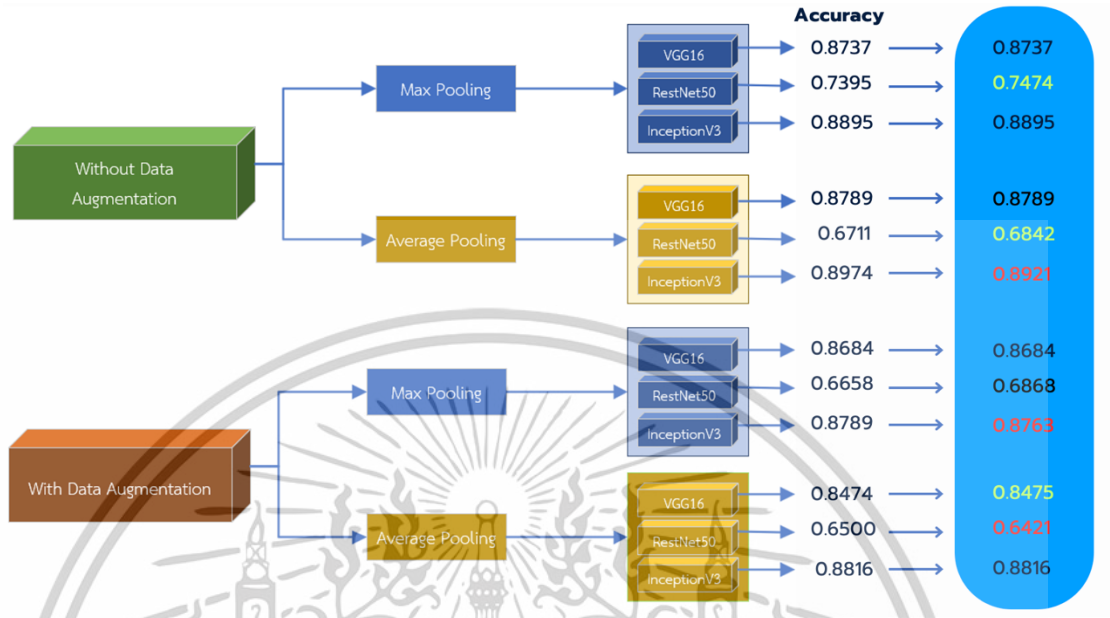
รูปที่ 4.40 กราฟแสดงความสัมพันธ์ระหว่าง accuracy และ loss ของโมเดล InceptionV3 ชั้นประมวลผลเป็น Average Pooling และ With Data Augmentation หลังจาก Fine Tune

หลังจากการทดลองทำการ Fine Tune พบว่าโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling ใช้ทดลองกับชุดข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) ที่มีค่า Accuracy สูงที่สุด พบว่าเกิดปัญหา ค่า Accuracy, Recall และ F1-Score กลับลดลงน้อยกว่าก่อนทำการปรับแต่งด้วย Fine Tune โดยโมเดล มีค่าตัวชี้วัดลดลง ดังนี้ ค่า Accuracy จาก 0.8974 เหลือ 0.8921 Recall จาก 0.9263 เหลือ 0.9082 และค่า F1-Score จาก 0.9003 เหลือ 0.8967 ส่วนค่า Precision กลับสูงขึ้นเพียงค่าเดียว จาก 0.8756 เป็น 0.8856 โดยปัญหาที่เกิดขึ้นมาจากปัญหา Overfitting ซึ่งสามารถสังเกตได้จากกราฟ Loss ของโมเดล InceptionV3 ก่อนการ Fine Tune ตามรูปที่ 4.17 ซึ่งแสดงให้เห็นว่าเมื่อมีการ Train โมเดลมากขึ้น ค่า Training Loss ลดลงเรื่อย ๆ แต่ค่า Validation Loss กลับเพิ่มขึ้นหลังจากจุดหนึ่ง เป็นสัญญาณว่ามีการเรียนรู้ที่ดีเกินไปจากชุดข้อมูลฝึกสอน (Train Set) ทำให้โมเดลไม่สามารถทำงานได้ดีในชุดข้อมูลทดสอบ (Test Set) และทำให้โมเดลขาดความสามารถในการ Generalize และเพิ่มโอกาสในการทำนายผิดพลาดกับชุดข้อมูลใหม่

ส่วนการทดลองโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling ใช้ทดลองกับชุดข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) โดยถึงแม้ว่ามีการแปลงชุดข้อมูลต้นฉบับ แต่ยังเกิดปัญหาค่า Accuracy, Precision และ F1-Score ลดลงหลังจากทำการ Fine Tune เช่นกัน ค่า Accuracy จาก 0.8789 เหลือ 0.8763 Precision จาก 0.8756 เหลือ 0.8209 และค่า F1-Score จาก 0.8844 เหลือ 0.8753 ส่วนค่า Recall กลับสูงขึ้นเพียงค่าเดียว จาก 0.8934 เป็น 0.9375 โดยปัญหาเกิดจาก Overfitting เช่นกัน เมื่อลองพิจารณาจาก Accuracy ของโมเดลก่อนการ Fine Tune ตามรูปที่ 4.11 ซึ่งแสดงให้เห็นถึงการแกว่งของ Validation Accuracy แสดงถึงการขาดความสม่ำเสมอและเสถียรภาพ ซึ่งอาจบ่งบอกว่าโมเดลมีปัญหาด้านการ Overfitting คือ โมเดลเรียนรู้รายละเอียดของข้อมูลการฝึกมากเกินไป แต่ไม่สามารถทั่วไปหรือทำนายได้ดีเมื่อเจอข้อมูลใหม่ ส่วนกราฟ Loss แสดงให้เห็นว่าค่า Training Loss ลดลงอย่างชัดเจนและค่อยๆ เข้าสู่ค่าที่ต่ำในช่วงประมาณ 0.2 - 0.3 ซึ่งเป็นสัญญาณว่าโมเดลเรียนรู้จากข้อมูลการฝึกได้ดีแต่ Validation Loss ก็ลดลงในช่วงแรก แต่เริ่มคงที่และมีความผันผวนมากขึ้นในช่วงหลัง ซึ่งสอดคล้องกับ Validation Accuracy ที่ไม่ค่อยเสถียร

ส่วนโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็นทั้ง Max Pooling ใช้ทดลองกับชุดข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) พบ Overfitting เช่นกัน ซึ่งสามารถสังเกตได้จากกราฟ Loss ของโมเดลก่อนการ Fine Tune ตามรูปที่ 4.5 และส่วนการทดลองโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling ใช้ทดลองกับชุดข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) จากกราฟแสดงความสัมพันธ์ของ Accuracy และ Loss ตามรูปที่ 4.23 เมื่อมีการ Train โมเดลมากขึ้น ค่า Loss จะมีแนวโน้มลดลง และ Accuracy มีแนวโน้มเพิ่มขึ้นแต่จะมีช่องว่างระหว่าง Training Loss กับ Validation Loss รวมทั้งช่องว่างระหว่าง Training Accuracy กับ Validation Accuracy สูง ซึ่งแสดงถึงความไม่เพียงพอของชุดข้อมูลฝึกสอน

(Train Set) ในการฝึกโมเดล หรือที่เรียกว่า Unrepresentative Train Dataset โดยสามารถดูกราฟ Loss และกราฟ Accuracy



รูปที่ 4.41 ผลลัพธ์ค่า Accuracy หลังปรับแต่ง Fine Tune ของ 3 โมเดล

หลังจากการทดลองทำการ Fine Tune พร้อมทั้งพิจารณากราฟแสดงความสัมพันธ์ของ Accuracy กับ Loss แล้ว โมเดล InceptionV3 ทั้งหมดถึงแม้จะให้ค่าตัวชี้วัดทั้ง 4 ค่าที่สูง แต่กลับพบปัญหาการเกิด Overfitting ดังนั้นเมื่อพิจารณาจากรูปที่ 4.41 โมเดลที่ให้ความแม่นยำ (Accuracy) รองลงมาจาก โมเดล InceptionV3 ทั้งหมดคือโมเดล VGG16 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling ใช้ทดลองกับชุดข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) ที่มีค่า Accuracy เท่ากับ 0.8789 รวมถึงค่าอื่นๆ ที่สังเกตุได้จากตารางที่ 4.3 ที่ให้ค่า Precision, Recall และ F1-Score เท่ากับ 0.8955, 0.8780, 0.8867 ตามลำดับ

บทที่ 5
สรุปผลลัพธ์ และข้อเสนอแนะ

การศึกษาค้นคว้าอิสระครั้งนี้ได้ศึกษาเกี่ยวกับการตรวจจับความเสียหายของรถยนต์ด้วยวิธีการเรียนรู้ของเครื่อง (Machine Learning) เพื่อจำแนกภาพถ่ายของรถยนต์ที่เกิดความเสียหายและที่ไม่เกิดความเสียหาย โดยใช้โครงสร้างประสาทเทียมแบบคอนโวลูชัน (Convolution Neural Network) ใช้เทคนิคการถ่ายโอน 3 โครงสร้าง (Transfer Learning) ประกอบด้วย โครงสร้างการถ่ายโอนโมเดล VGG16 , โครงสร้างการถ่ายโอนโมเดล ResNet50 และโครงสร้างการถ่ายโอนโมเดล InceptionV3 โดยหลังจากการทดลองสามารถสรุปผลลัพธ์และมีข้อเสนอแนะ ดังนี้

5.1 สรุปผลลัพธ์

จากการทดลองเพื่อศึกษาประสิทธิภาพของโมเดลการเรียนรู้เชิงลึกโดยใช้โครงสร้างประสาทเทียมแบบคอนโวลูชัน (Convolutional Neural Networks, CNN) ซึ่งเป็นการประยุกต์ใช้เทคนิคการถ่ายโอนความรู้ (Transfer Learning) จากโมเดลที่ฝึกสอนด้วยข้อมูลจาก ImageNet ซึ่งประกอบไปด้วยโมเดล VGG16, ResNet50 และ InceptionV3 โดยมุ่งเน้นการจำแนกภาพถ่ายของรถยนต์ที่เกิดความเสียหายและไม่เกิดความเสียหาย ในการทดลองนี้ ได้มีการเปรียบเทียบผลการใช้ชุดข้อมูลที่แปลง (With Data Augmentation) และไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation) และยังได้ปรับปรุงโมเดลด้วยการเพิ่มส่วนหัวที่กำหนดเอง (Custom Head) ซึ่งประกอบด้วย Max Pooling Layer และ Average Pooling รวมถึงการปรับแต่งโมเดลเพิ่มเติมด้วยการฝึกสอนชั้นที่เลือก (Selective Layer Training)

ผลลัพธ์ที่ดีที่สุดได้มาจากโมเดล VGG16 ที่ผ่านการปรับแต่งด้วย Fine Tune และใช้ชุดข้อมูลที่ไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation) โดยใช้ Custom Head แบบ Average Pooling ผลการทดลองนี้แสดงให้เห็นว่าโมเดลมีค่า Accuracy เท่ากับ 0.8789 Precision เท่ากับ 0.8955 Recall เท่ากับ 0.8780 และ F1-Score เท่ากับ 0.8867 พร้อมทั้งพิจารณากราฟ Loss และกราฟ Accuracy พบว่าโมเดลมีการเรียนรู้ที่ดี สามารถนำไปทำนายชุดข้อมูลที่ไม่เคยพบเห็นมาก่อนได้อย่างแม่นยำ ถึงแม้ว่าจะมีค่า Accuracy, Precision, Recall และ F1-Score น้อยกว่าโมเดล InceptionV3 ทั้งก่อนและหลังจากการปรับแต่งด้วย Fine Tune เนื่องจากที่ไม่ได้เลือกโมเดล InceptionV3 เพราะว่าโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็นทั้ง Max Pooling และ Average Pooling ที่ใช้ทดลองกับชุดข้อมูลที่ไม่ได้แปลงข้อมูลต้นฉบับ (Without Data Augmentation) และรวมถึงการทดลองโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Max Pooling ที่ใช้ทดลองกับชุดข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) เกิดปัญหามีการเรียนรู้ที่ดีเกินไปจากชุดข้อมูลฝึกสอน (Train Set) หรือเรียกว่า Overfitting หลังจากที่มีการ

พิจารณากราฟ Loss และกราฟ Accuracy แล้ว และส่วนของโมเดล InceptionV3 ที่เปลี่ยนชั้นประมวลผลเป็น Average Pooling ใช้ทดลองกับชุดข้อมูลที่แปลงข้อมูลต้นฉบับ (With Data Augmentation) เกิดปัญหาชุดข้อมูลฝึกสอน (Train Set) ไม่เพียงพอ ในการฝึกโมเดล หรือที่เรียกว่า Unrepresentative Train Dataset หลังจากที่มีการพิจารณากราฟ Loss และกราฟ Accuracy แล้วเช่นกัน

การศึกษานี้ยืนยันถึงความสามารถของโมเดล VGG16 ในการจำแนกภาพถ่ายของรถยนต์ที่มีความเสียหายและไม่มีความเสียหายได้อย่างมีประสิทธิภาพสูง แสดงถึงศักยภาพในการประยุกต์ใช้ในการประเมินความเสียหายของรถยนต์แบบอัตโนมัติในอนาคตได้เป็นอย่างดี ซึ่งจะช่วยเพิ่มประสิทธิภาพในกระบวนการทำงานของพนักงานประกันภัย ลดภาระงานและเวลาในการตรวจสอบด้วยตนเอง และเสริมความเร็วในการบริการเคลมของลูกค้าประกันภัย ส่งผลให้ลูกค้าได้รับการบริการที่รวดเร็วและมีประสิทธิภาพมากยิ่งขึ้น

5.2 ข้อเสนอแนะ

หลังจากการทดลองศึกษาค้นคว้าอิสระครั้งนี้ มีข้อเสนอแนะ ดังนี้

5.2.1 ควรศึกษาและทดสอบโมเดลอื่นที่มีประสิทธิภาพสูง เช่น EfficientNet หรือ Vision Transformers เพื่อนำมาเปรียบเทียบประสิทธิภาพ ซึ่งอาจให้ผลลัพธ์ที่ดีและแม่นยำมากขึ้นในการจำแนกความเสียหายของรถยนต์จากภาพถ่าย

5.2.2 ทดลองปรับจูนพารามิเตอร์รูปแบบอื่นๆ เพื่อเพิ่มประสิทธิภาพให้กับโมเดล เช่นใช้เทคนิค Cross Validation และ Grid Search เพื่อหาค่าพารามิเตอร์ที่เหมาะสมที่สุด รวมถึงเทคนิค Regularization เพื่อลดปัญหา Overfitting

5.2.3 ทดลองใช้เทคนิค Ensemble Learning ซึ่งเป็นการนำหลายๆ โมเดลมาผสมผสานกันเพื่อให้ได้ผลลัพธ์ที่ดีขึ้น การใช้เทคนิคนี้ช่วยเพิ่มความแม่นยำและลดความผันผวนของผลลัพธ์ในการทำนาย

5.3 ข้อจำกัดในการศึกษา

หลังจากการทดลองศึกษาค้นคว้าอิสระครั้งนี้ มีข้อจำกัดในการศึกษา ดังนี้

5.3.1 เกิดปัญหาไม่สามารถพัฒนาโมเดลใช้งานบน Google Colab ได้อย่างต่อเนื่องทำให้เกิดการสะดุดและส่งผลให้การพัฒนาโมเดลล่าช้า เนื่องจากข้อจำกัดของ time-usage limit ของ runtime โดยการทดลองศึกษาค้นคว้าอิสระครั้งนี้ได้ใช้วิธีการแก้ไขคือการสมัครบริการ Google Colab Pro ที่มีค่าใช้จ่ายเพื่อเพิ่มทรัพยากรและเวลาในการใช้งาน โดยบริการนี้ช่วยให้ผู้ใช้สามารถเข้าถึง GPU ที่มีประสิทธิภาพสูงขึ้นและเวลาการใช้งานที่ยาวนานขึ้น ส่งผลให้สามารถพัฒนาและทดสอบโมเดลได้อย่างต่อเนื่อง

5.3.2 ข้อจำกัดในด้านข้อมูลเกี่ยวกับภาพถ่ายของรถยนต์ที่ใช้ในการศึกษา พบว่าชุดข้อมูลที่มีการใช้งานอาจมีความหลากหลายไม่เพียงพอหรือมีคุณภาพไม่สูงเท่าที่ควร ซึ่งอาจส่งผลกระทบต่อประสิทธิภาพของโมเดลที่พัฒนา โดยเฉพาะเมื่อมีการทำการจำแนกประเภทหรือวิเคราะห์คุณสมบัติต่าง ๆ ของรถยนต์ หากข้อมูลภาพถ่ายที่มีการสะสมมีปริมาณไม่เพียงพอหรือมีความคล้ายคลึงกันมาก อาจทำให้โมเดลเกิด Overfitting อติเช่นที่พบปัญหาในการทดสอบกับโมเดล InceptionV3 ในการศึกษาค้นคว้าอิสระในครั้งนี้

5.4 ส่วนเพิ่มเติมที่ต้องการศึกษาในอนาคต

ต้องการพัฒนาโมเดลให้สามารถทำนายตำแหน่งที่เกิดความเสียหายบนรถยนต์ได้อย่างแม่นยำ รวมถึงการประเมินระดับความรุนแรงของความเสียหายนั้นด้วย การพัฒนานี้จะช่วยเพิ่มศักยภาพของโมเดลในการวิเคราะห์และประเมินสภาพรถยนต์ที่ได้รับความเสียหายได้อย่างละเอียดและมีประสิทธิภาพมากขึ้น โดยโมเดลที่พัฒนาจะต้องสามารถระบุพื้นที่ที่เกิดความเสียหายบนตัวรถ เช่น กันชน ประตู หรือไฟหน้า และประเมินระดับความรุนแรงของความเสียหายตั้งแต่เล็กน้อยไปจนถึงรุนแรงมาก การปรับปรุงดังกล่าวจะช่วยให้สามารถใช้โมเดลในการประเมินความเสียหายได้ในหลากหลายบริบท เช่น การประกันภัยรถยนต์ การซ่อมแซมรถยนต์ และการตรวจสภาพรถยนต์หลังเกิดอุบัติเหตุ ซึ่งจะช่วยลดความจำเป็นในการพึ่งพาผู้เชี่ยวชาญและเพิ่มความแม่นยำและรวดเร็วในการประเมินสภาพรถยนต์ที่เสียหายได้

เอกสารอ้างอิง

[1] Phyu Mar K., และ Kuntpon Woraratpanya. (2020). Car Damage Detection and Classification. Proceedings of International Conference on Advances in Information Technology (IAIT2020), 1-6. doi: 10.1145/3406601.3406651

[2] Kalpesh Patil, Mandar Kulkarni, Anand Sriraman และ Shirish Karande. (2017). Deep Learning Based Car Damage Classification. Conference on 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 1-6. doi: 10.1109/ICMLA.2017.0-179

[3] Sarath P., Soorya M., Shaik Abdul Rahman A., S. Suresh Kumar, และ K. Devaki. (2020). Assessing Car Damage using Mask R-CNN. International Journal of Engineering and Advanced Technology (IJEAT), 1-4. Retrieved from https://www.researchgate.net/publication/341039799_Assessing_Car_Damage_using_Mask_R-CNN

[4] Girish N., และ Mohammed Aqeel Arshad. (2021). Car Damage Detection using Machine Learning. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE). doi: 10.17148/IJARCCE.2021.10808

[5] Jihad Qaddour, และ Syeda Ayesha Siddiqa. (2023). Automatic Damaged Vehicle Estimator Using Enhanced Deep Learning Algorithm. Intelligent Systems with Applications. Doi: 10.1016/j.iswa.2023.200192

[6] สุริยะ ชยะธรรมกูล. (2020). การจำแนกผลึกน้ำตาลด้วยวิธีการเรียนรู้เชิงลึก. สารนิพนธ์ วท.ม. (สาขาวิชาวิศวกรรมซอฟต์แวร์) กรุงเทพฯ: บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย.

[7] ธนัช เบญจอนุอาชา. (2021). การจำแนกความเสียหายรถยนต์โดยการเรียนรู้เชิงลึก. สารนิพนธ์ วท.ม. (สาขาวิชาวิทยาการข้อมูล) กรุงเทพฯ: บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ.

[8] สำนักงานคณะกรรมการกำกับและส่งเสริมการประกอบธุรกิจประกันภัย. (2023), เบี้ยประกันวินาศภัยสะสม ปี 2024. สืบค้น 8 พฤศจิกายน 2023. จาก https://www.oic.or.th/th

ภาคผนวก

ขั้นตอนและคำสั่งที่ใช้เขียน Python ใน Google Colab

1) Import Library

```
import os
import cv2
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, GlobalAveragePooling2D, Dropout, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam

from keras.models import load_model

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

2) เชื่อมต่อ Google Colab กับ Google Drive เพื่อเข้าถึงไฟล์และโฟลเดอร์ที่อยู่ใน Google Drive

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

3) นำเข้าข้อมูล และแบ่งชุดข้อมูลเป็นชุดในการฝึกสอน (Train Set) และ ชุดในการทดสอบ (Test Set)

```
import pandas as pd
import os
from sklearn.model_selection import train_test_split

# Set the path of the folder containing the images
image_folder = '/content/drive/MyDrive/IS/Car Photo'

# Load data from Excel
label = pd.read_excel('/content/drive/MyDrive/IS/Data_Car.xlsx')

# Convert 'damage' variable
label['damage'] = label['damage'].replace('Damage', '1').replace('NoDamage', '0')

# Ensure 'damage' column is of type string, this is critical for class_mode="binary"
label['damage'] = label['damage'].astype(str)

# Drop the unwanted 'class' column
label.drop(['class'], axis=1, inplace=True)

# Create a new column in the DataFrame for the image file paths
label['image_path'] = label['image'].apply(lambda x: os.path.join(image_folder, x))

# Split the data into training and testing sets
train_set, test_set = train_test_split(label, test_size=0.2, random_state=42)

# Print basic information about the datasets
print("Training Set:")
print(train_set.head())  # Displays the first few rows of the training set
print("Number of training images:", len(train_set))
print("\nTesting Set:")
print(test_set.head())  # Displays the first few rows of the testing set
print("Number of testing images:", len(test_set))
```

4) นำชุดข้อในการฝึกสอน (Train Set) แบ่งออกเป็น 2 แบบ ดังนี้

4.1 ไม่แปลงข้อมูลต้นฉบับ (Without Data Augmentation)

```
# Create an instance of ImageDataGenerator
data_gen_train = ImageDataGenerator(rescale=1./255)
data_gen_test = ImageDataGenerator(rescale=1./255)
```

4.2 แปลงข้อมูลต้นฉบับ (With Data Augmentation)

```
# Create an instance of ImageDataGenerator with data augmentation parameters
data_gen_train = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

data_gen_test = ImageDataGenerator(
    rescale=1./255
)
```

5) สร้างตัวกำหนดข้อมูลภาพที่จะใช้สำหรับการฝึกสอนและทดสอบโมเดลโดยใช้ข้อมูลภาพตามที่ระบุใน DataFrame

```
# Use the flow_from_dataframe method
train_generator = data_gen_train.flow_from_dataframe(
    dataframe=train_set,
    directory=None,
    x_col='image_path',
    y_col='damage',
    target_size=(224, 224),
    class_mode='binary',
    batch_size=32,
    shuffle=False
)

test_generator = data_gen_test.flow_from_dataframe(
    dataframe=test_set,
    directory=None,
    x_col='image_path',
    y_col='damage',
    target_size=(224, 224),
    class_mode='binary',
    batch_size=32,
    shuffle=False
)
```

6) Import และกำหนด Model ในการ Training

```
# Import the VGG16 model
from tensorflow.keras.applications import VGG16

# Create an instance of the VGG16 model
base_model = VGG16(input_shape=(224, 224, 3), include_top=False, weights="imagenet")
```

7) ตั้งค่าให้ base_model ไม่สามารถปรับปรุง Weight ของโมเดลนี้ในระหว่าง Training

```
base_model.trainable = False
```

8) ตั้งค่าและปรับแต่งโมเดลโดยการสร้างเพิ่มชั้น (Layers) เพื่อลดมิติของข้อมูล โดยมีการเปรียบเทียบระหว่างการเพิ่มแบบ Max Pooling กับ Average Pooling พร้อมทั้งเพิ่มชั้น Dense Layer ที่มีขนาดหนึ่งหน่วยและใช้ฟังก์ชัน sigmoid เพื่อแปลงเอาต์พุตเป็นความน่าจะเป็นในช่วง 0 ถึง 1

```
max_pooling_layer = tf.keras.layers.GlobalMaxPooling2D()(base_model.output)
prediction_layer = tf.keras.layers.Dense(units=1, activation='sigmoid')(max_pooling_layer)
```

```
# เพิ่มเลเยอร์ GlobalAveragePooling2D
avg_pooling_layer = tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
prediction_layer = tf.keras.layers.Dense(units=1, activation='sigmoid')(avg_pooling_layer)
```

```
model = tf.keras.models.Model(inputs=base_model.input, outputs=prediction_layer)
```

9) เตรียมโมเดลให้พร้อมสำหรับการ Training โดยกำหนดให้ใช้ RMSprop เป็นวิธีการปรับปรุงน้ำหนักของโมเดล โดยตั้งค่าอัตราการเรียนรู้เป็น 0.0001,กำหนด Loss Function ใช้ binary_crossentropy เป็นฟังก์ชันสูญเสียสำหรับการจำแนกประเภทแบบไบนารี, และกำหนด Metrics ตั้งค่าให้ติดตามค่า 'accuracy' (ความแม่นยำ) เพื่อประเมินประสิทธิภาพของโมเดลระหว่างการฝึกสอนและการทดสอบ

```
#Tarin model Optimizers RMSprop learning=0.0001
import tensorflow as tf

model.compile(
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001),
    loss="binary_crossentropy",
    metrics=["accuracy"]  # Corrected the spelling of `metrics`
)
```

10) Train Model ที่เลือกใช้ในการทดสอบ

```
modelVGG16 = model.fit_generator(train_generator, epochs=500, validation_data=test_generator)
```

11) แสดงกราฟ Accuracy และ Loss ระหว่างค่าในการฝึกสอนและทดสอบ

```
acc = modelVGG16.history['accuracy']
val_acc = modelVGG16.history['val_accuracy']
loss = modelVGG16.history['loss']
val_loss = modelVGG16.history['val_loss']

epochs_range = range(500)

plt.figure(figsize=(15,15))
plt.subplot(2,2,1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Testing Accuracy')
plt.legend(loc='lower right')
plt.title('Traning and Testing Accuracy')

plt.subplot(2,2,2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Testing Loss')
plt.legend(loc='upper right')
plt.title('Traning and Testing Loss')
plt.show()
```

12) บันทึกและดาวน์โหลดโมเดลเพื่อนำไปใช้ในขั้นตอนถัดไป

```
model.save('./model/VGG16data2_max_batch128.h5')

loaded_model_VGG16 = load_model('./model/VGG16data2_max_batch128.h5')
```

13) วัดประสิทธิภาพของโมเดล

```
#print evaluation score
score = loaded_model_VGG16.evaluate(test_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

#predict Test Set
pred = loaded_model_VGG16.predict(test_generator)

#true y and Predicted y
y_true = test_generator.classes
y_pred = (pred.flatten() > 0.5).astype('int32')

print('Accuracy score:\n', accuracy_score(y_pred,y_true)) #vgg16
```

14) สร้างตาราง Confusion Matrix

```
mat = confusion_matrix(y_true, y_pred)

axes = sns.heatmap(mat, square=True, annot=True,
                   fmt='d', cbar=True, cmap=plt.cm.GnBu)

axes.set_xlabel('actual')
axes.set_ylabel('model prediction')

axes.set_title('cofusion_martix')
```

15) สร้าง Classification Report เพื่อดูค่า Accuracy, Precision, Recall

```
target_names = ['Nodamage', 'Damage']
print('Classification_report', classification_report(y_true, y_pred, target_names=target_names))
```

16) นำ Model ที่ได้มาทำการ Fine Tune

16.1 ตั้งค่าให้ base_model สามารถปรับปรุง Weight ของโมเดลนี้ในระหว่าง Training

```
#คำสั่งกำหนดให้โมเดลสามารถฝึกฝนข้อมูลของเราได้
base_model.trainable = True
```

16.2 กำหนด layer ให้โมเดลชั้นแรกถึง 15 ไม่สามารถปรับปรุง Weight ของโมเดลด้ คำสั่งนี้จะทำให้โมเดลเทรนเฉพาะ ชั้นที่ 16 ถึงชั้น 21 ชั้นสุดท้าย (ขึ้นอยู่กับโมเดลในการฝึกสอน)

```
#กำหนด layer ให้โมเดลชั้นแรกถึง 15 ไม่สามารถฝึกฝนได้
#คำสั่งนี้จะทำให้โมเดลเทรนเฉพาะ ชั้นที่ 16 ถึงชั้น 21 ชั้นสุดท้าย
for layer in loaded_model_VGG16.layers[:15]:
    layer.trainable = False
```

16.3 กำหนดให้โมเดลหยุดฝึกสอนหาก loss ไม่เปลี่ยนแปลงภายใน 3 epoch

```
#คำสั่งกำหนดให้โมเดลหยุดฝึกฝนหาก loss ไม่เปลี่ยนแปลงใน 3 epoch
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

16.4 กำหนดให้โมเดลสำหรับการ Training ในขั้นตอน Fine Tune โดยกำหนดให้ใช้ RMSprop เป็นวิธีการปรับปรุงน้ำหนักของโมเดล โดยตั้งค่าอัตราการเรียนรู้เป็น 0.0001,กำหนด Loss Function ใช้ binary_crossentropy เป็นฟังก์ชันสูญเสียสำหรับการจำแนกประเภทแบบไบนารี, และกำหนด Metrics ตั้งค่าให้ติดตามค่า 'accuracy' (ความแม่นยำ) เพื่อประเมินประสิทธิภาพของโมเดลระหว่างการฝึกสอนและการทดสอบ

```
#Tarin model Optimizers RMSprop learning=0.0001
loaded_model_VGG16.compile(
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001),  # Corrected from `lr` to `learning_rate`
    loss="binary_crossentropy",
    metrics=["accuracy"]  # Corrected the spelling of `metrics`
)
```

16.5 Train Model ที่เลือกใช้ในการทดสอบ Fine Tune

```
model_ft_VGG16=loaded_model_VGG16.fit_generator(train_generator, epochs=500, validation_data=test_generator,callbacks=[callback])
```

16.6 แสดงกราฟ Accuracy และ Loss ระหว่างค่าในการฝึกสอนและทดสอบ Fine Tune

```
acc_ft = model_ft_VGG16.history['accuracy']
val_acc_ft = model_ft_VGG16.history['val_accuracy']
loss_ft = model_ft_VGG16.history['loss']
val_loss_ft = model_ft_VGG16.history['val_loss']

epochs_range = range(len( model_ft_VGG16.history['accuracy']))

plt.figure(figsize=(15,15))
plt.subplot(2,2,1)
plt.plot(epochs_range, acc_ft, label='Training Accuracy')
plt.plot(epochs_range, val_acc_ft, label='Testing Accuracy')
plt.legend(loc='lower right')
plt.title('Traning and Testing Accuracy Fine tuning')

plt.subplot(2,2,2)
plt.plot(epochs_range, loss_ft, label='Training Loss')
plt.plot(epochs_range, val_loss_ft, label='Testing Loss')
plt.legend(loc='upper right')
plt.title('Traning and Testing Loss Fine tuning')
plt.show()
```

16.7 วัดประสิทธิภาพของโมเดลในการทดสอบ Fine Tune

```
#print evaluation score
score = loaded_model_VGG16.evaluate(test_generator)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

#predict Test Set
pred = loaded_model_VGG16.predict(test_generator)

#true y and Predicted y
y_true = test_generator.classes
y_pred = (pred.flatten() > 0.5).astype('int32')

print('Accuracy score:\n', accuracy_score(y_pred,y_true)) #vgg16_ft
```

16.8 สร้างตาราง Confusion Matrix ในการทดสอบ Fine Tune

```
mat = confusion_matrix(y_true, y_pred)

axes = sns.heatmap(mat, square=True, annot=True,
                   fmt='d', cbar=True, cmap=plt.cm.GnBu)

axes.set_xlabel('actual')
axes.set_ylabel('model prediction')

axes.set_title('cofusion_martix')
```

16.9 สร้าง Classification Report เพื่อดูค่า Accuracy, Precision, Recall ในการทดสอบ Fine Tune

```
target_names = ['Nodamage', 'Damage']
print('Classification_report', classification_report(y_true, y_pred, target_names=target_names))
```

ทั้งหมดนี้จะเป็นขั้นตอนการฝึกสอนโมเดลพร้อมทั้งเปรียบเทียบประสิทธิภาพที่ได้ในแต่ละโมเดล

ประวัติผู้เขียน

ชื่อ	นายธุวานนท์ พัฒนนวพงศ์
วัน เดือน ปีเกิด	24 มิถุนายน 2541
ที่อยู่ปัจจุบัน	219/250 หมู่บ้านราณี 4 ซอย 2 ถนนแจ้งวัฒนะ 6 แขวงตลาดบางเขน เขตหลักสี่ จังหวัดกรุงเทพมหานคร
ประวัติการศึกษา	(2563) วิทยาศาสตรบัณฑิต สาขาสถิติธุรกิจและการประกันภัย เกรดเฉลี่ย 3.45 (มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ)
	(2567) วิทยาศาสตรมหาบัณฑิต สาขาวิทยาการข้อมูลและการวิเคราะห์ (สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง)