การประมวลผลภาพอุบัติเหตุรถยนต์โดยใช้กระบวนการโครงข่ายประสาทแบบคอนโวลูชัน

Image Processing for Car Accident Using Convolutional Neural Network

ฐิติชยาน์ คุ้มวงษ์

THITICHAYA KHUMWONG

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง
คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2566

KMITL-2023-SC-M-017-035

Image Processing for Car Accident Using Convolutional Neural Network

THITICHAYA KHUMWONG

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF 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

2023

KMITL-2023-SC-M-017-035

หัวข้อการค้นคว้าอิสระ การประมวลผลภาพอุบัติเหตุรถยนต์โดยใช้กระบวนการโครงข่ายประสาทแบบคอนโวลูชัน

ชื่อนักศึกษา นางสาวฐิติชยาน์ คุ้มวงษ์

รหัสนักศึกษา 63605071

ปริญญา วิทยาศาสตรมหาบัณฑิต(วิทยาการข้อมูลและการวิเคราะห์) ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง

พ.ศ. 2566

อาจารย์ที่ปรึกษาการค้นคว้าอิสระ ผู้ช่วยศาสตราจารย์.ดร.บุษยมาส พิมพ์พรรณชาติ

บทคัดย่อ

การค้นคว้าอิสระนี้เป็นกาศึกษาการนำเทคนิคของทาง CNN สร้างโมเดลในการจำแนกรถ การจำแนกว่ารถยนต์ได้รับความเสียหายหรือไม่ การจำแนกตำแหน่งความเสียที่เกิดขึ้น ช่วยวิเคราะห์ความเสียหายจากรูปภาพของรถยนต์เบื้องต้น และเพื่อยืนยันตำแหน่งความเสียหายของรถยนต์กับใบเคลมโดยข้อมูลที่นำมาวิเคราะห์จากบริษัทประกันวินาศภัย

ผลการศึกษาพบว่าในการจำแนกตำแหน่งของรถโดยมีการเปรียบเทียบ CNN, VGG16 และ ResNet เมื่อนำมาเปรียบเทียบผลของทั้ง 3 Model พบว่า CNN Model นั้นมีการทำนายตำแหน่งรถยนต์ในแต่ละ Class ได้มีความถูกต้องมากกว่า และค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 81.72 ลำดับถัดมาเป็น VGG 16 Model ซึ่งทำนายแม่นยำในส่วนของSide class มากกว่า Class อื่นๆ และค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 59.29 และสุดท้ายคือ ResNet Model ที่มีการเกิด Overfitting จึงทำให้ผลทำนายเป็น Rear class ทั้งหมด และผลการศึกษาในการตรวจจับความเสียหายของรถยนต์ เมื่อนำผลวิเคราะห์ของR CNN, R CNN DC5 มาเปรียบเทียบกัน พบว่าการทำนายความเสียหายรถยนต์ทั้ง 2 Model โดย R CNN ค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 95.67 และ R CNN DC5 ค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 85.25 ทำนายผลได้ใกล้เคียงกัน แต่ในส่วนการทำนายรูปไม่มีความเสียหาย R CNN Model สามารถทำนายได้อย่างถูกมากกว่า R CNN DC5 Model จากศึกษาพบอีกว่าในกรณีที่เป็นข้อมูลรูปที่มีสีดำหรือมีการถ่ายความเสียหายจากระยะไกล Model ยังคงไม่สามารถตรวจจับได้

คำสำคัญ : ตำแหน่งของรถยนต์ ความเสียหายของรถยนต์ โครงข่ายประสาทแบบคอนโวลูชัน

Independent Study Title Image Processing for Car Accident Using Convolutional Neural Network
Student Name Thitichaya Khumwong
Student ID 63605071
Degree Master of Science (Data Science and Analytics)
KMITL-Digital Analytics and Intelligence Center
Year 2023
Independent Study Advisor Asst.Prof.Dr. Busayamas Pimpunchat

ABSTRACT

This independent study examines the application of CNN techniques to create a car classification model. Classifying whether the vehicle has been damaged or not Classification of the location of the damage that occurred. Helps analyze damage from preliminary images of cars. and to confirm the damage location of the car with the claim form The information analyzed by a non-life insurance company.

The study found that in classifying the car's position by comparing CNN, VGG16 and ResNet, when comparing the results of all 3 models, it was found that CNN Model predicted the car's position in each class more accurately. and the average of the Accuracy was 81.72%, followed by the VGG 16 Model, which predicted more accurately in the Side class than other classes and the average Accuracy was 59.29%, and the last one was the ResNet Model that caused overfitting. Make prediction results for all Rear class. And the results of a study on vehicle damage detection When the analysis results of R CNN, R CNN DC5 are compared. It was found that the prediction of damage to both models of cars by R CNN, the average Accuracy was 95.67 percent and R CNN DC5, the average Accuracy was 85.25 percent, the predictions were similar. But in terms of image prediction without damage, the R CNN model can predict more accurately than the R CNN DC5 model. The study also found that in the case of image data that has black color or is photographed with damage from a distance, the model still does not can detect.

Keywords: Identification of the location, Damage of the automobile accident, Convolutional Neural Network

กิตติกรรมประกาศ

การค้นคว้าอิสระฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี โดยได้รับความอนุเคราะห์จาก ผศ.ดร. บุษยมาส พิมพ์พรรณชาติ ที่ได้ให้คำแนะนำ คำปรึกษาและเสียสละเวลาในการให้คำปรึกษาในการค้นคว้าอิสระนี้ รวมทั้งการตรวจสอบและแก้ไขข้อบกพร่องต่างๆเป็นอย่างดีโดยตลอด

ขอขอบพระคุณ สมาคมประกันวินาศภัยไทย เป็นอย่างสูง ที่ได้ให้ข้อมูลเคสตัวอย่างอุบัติเหตุของรถยนต์ต่างๆ เป็นอย่างดี

ขอขอบพระคุณ ท่านอาจารย์ภาควิชาวิทยาการข้อมูลและการวิเคราะห์ ที่ได้ประสิทธิ์ประสาทวิชาพร้อมทั้งให้คำแนะนำด้านต่างๆ และขอขอบคุณเจ้าหน้าที่ทุกท่านที่ให้ความสะดวกและช่วยเหลือด้านต่างๆตลอดการทำงานวิจัย

ฐิติชยาน์ คุ้มวงษ์

สารบัญ

สารบัญ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัญหาการจราจรติดขัดในกรุงเทพมหานคร เป็นปัญหาที่มีความสำคัญลำดับต้นๆที่มีมากขึ้น โดยปัจจัยที่ทำให้เกิดปัญหาการจราจรติดขัดมีได้หลายสาเหตุ เช่น ระบบขนส่งสาธารณะและปริมาณถนนไม่เพียงพอต่อผู้ใช้ในท้องถนน ฝนตก น้ำท่วม การซ่อมแซมถนน และการเกิดอุบัติเหตุบนท้องถนน ในช่วงมกราคม - กรกฎาคม ปี พ.ศ. 2564 ได้มีสถิติรายงานผู้ประสบภัยจากรถจักยานยนต์และรถยนต์จากศูนย์ข้อมูลอุบัติเหตุเพื่อเสริมสร้างวัฒนธรรมความปลอดภัยทางถนน(https://www.thairsc.com) มียอดสะสมเป็นจำนวน 92,159 ราย

เมื่อพิจารณาปัจจัยที่ก่อให้เกิดอุบัติเหตุ ได้แก่ ปัจจัยด้านบุคคล ยานพาหนะ และสิ่งแวดล้อม เมื่อเกิดอุบัติเหตุทางบกแล้ว เพื่อให้ลดความเสียหายจึงมีกฎหมายกำหนดให้ผู้ครอบครองรถยนต์ทุกคันต้องทำประกันภัยรถยนต์ภาคบังคับหรือที่เรียกว่าประกันภัยรถยนต์ภาคบังคับที่จะให้ความคุ้มครองในด้านชีวิตและทรัพย์สินผู้ประสบภัย โดยบริษัทผู้รับประกันภัยจ่ายค่ารักษาพยาบาล และค่าสินไหมทดแทนตามวงเงินคุ้มครองที่ระบุไว้ตามเงื่อนไขของกรมธรรม์ และดังนั้นการประกันภัยรถยนต์ภาคสมัครใจมีความสำคัญอย่างมากต่อเจ้าของและผู้ครอบครองรถยนต์เพื่อช่วยลดความเสี่ยงในการรับภาระค่าใช้จ่ายเมื่อเกิดอุบัติเหตุ

การชดเชยเยียวยาจากอุบัติเหตุในกรณีของผู้ประสบอุบัติเหตุ มักประสบปัญหาในการได้รับความช่วยเหลือหลายประการ ทั้งส่วนของการขาดความรู้เรื่องสิทธิของผู้โดยสารที่ประสบเหตุพึงได้รับ การชดเชยเยียวยาที่ไม่เป็นธรรมจากบริษัทประกันภัย และผู้ประกอบการขนส่งรถโดยสารสาธารณะ รวมถึงการขาด คำแนะนำในกรณีที่เป็นคดีความฟ้องร้องในชั้นศาล และประเด็นต้องพิจารณาคือการชดเชยเยียวยาเมื่อเกิดอุบัติเหตุในปัจจุบันมีความเหมาะสมหรือไม่ ทั้งในส่วนของค่าใช้จ่ายเกี่ยวกับค่ารักษาพยาบาล ค่าสินไหมทดแทนการขาดรายได้ และค่าชดเชยอื่น ๆ ซึ่งสิ่งเหล่านี้ผู้ประสบอุบัติเหตุมักจะต้องรอตัวแทนจากบริษัทประกันภัยของตนเพื่อมาวิเคราะห์ความเสียหายที่เกิด ซึ่งทำบางครั้งจะต้องจอดรถไว้ที่เกิดเหตุ ทำให้เป็นกีดขวางการจราจร

ดังนั้นการค้นคว้าอิสระนี้ จึงนำเสนอ การนำเทคนิคของทาง CNN สร้างโมเดลในการจำแนกรถ การจำแนกว่ารถยนต์ได้รับความเสียหายหรือไม่ การจำแนกตำแหน่งความเสียที่เกิดขึ้น ช่วยวิเคราะห์ความเสียหายจากรูปภาพของรถยนต์เบื้องต้น และเพื่อยืนยันตำแหน่งความเสียหายของรถยนต์กับใบเคลม

1.2 วัตถุประสงค์ของงานวิจัย

1. มีอัลกอริทึมในการสร้างโมเดลการจำแนกรถ การจำแนกว่ารถยนต์ได้รับความเสียหายหรือไม่ การจำแนกตำแหน่งความเสียที่เกิดขึ้น
2. เพื่อยืนยันตำแหน่งความเสียหายของรถยนต์กับใบเคลม

1.3 ขอบเขตงานวิจัย

1. ด้านประชากรและพื้นที่
   รูปภาพอุบัติเหตุรถยนต์ที่จากบริษัทประกันภัย
2. ด้านขอบเขตเนื้อหา
   งานวิจัยฉบับนี้ได้เลือกใช้รูปภาพเพื่อนำมาวิเคราะห์ความเสียหายที่เกิดจากอุบัติเหตุรถยนต์ โดยใช้เทคนิคของ Machine learning เป็นตัวช่วยในการวิเคราะห์
3. งานวิจัยนี้จะไม่ได้พิจารณาในส่วนของความลึกของความเสียหาย และไม่สามารถแยกความเสียหายใหม่หรือเก่าได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เพื่อให้มีอัลกอริทึมที่สามารถช่วยวิเคราะห์ความเสียหายของรถยนต์
2. เพื่อเสนอแนวทางในการใช้เทคนิคของ machine learning ในการวิเคราะห์รูปภาพ
3. เพื่อเสนอแนวทางการนำไปใช้ต่อในการพิจารณาพระราชบัญญัติจราจรทางบก หมวดรถวิ่งตามกัน/ในทิศทางเดียวกัน

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

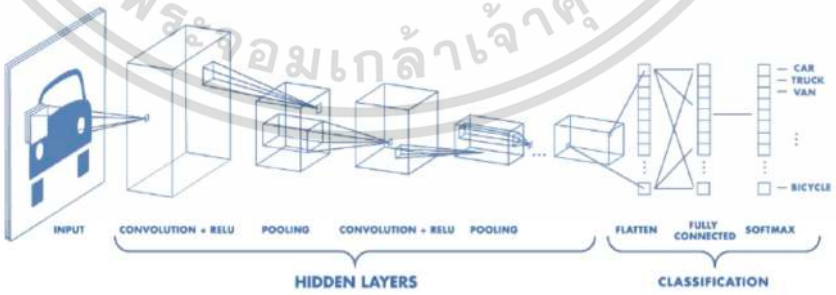
ในส่วนของทฤษฎีและงานวิจัยที่เกี่ยวข้องจะทำการนำเสนอความรู้พื้นฐานของที่เกี่ยวข้องกับ CNN, Convolutional Auto Encoders, ResNet, Faster-RCNN และ Transfer Learning และงานวิจัยที่เกี่ยวข้อง

2.1 โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network)

Convolutional Neural Network (CNN) หรือ โครงข่ายประสาทแบบคอนโวลูชันเป็นโครงข่ายประสาทเทียมหนึ่งในกลุ่ม bio-inspired โดยที่ CNN จะจำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อย ๆ และนำกลุ่มของพื้นที่ย่อย ๆมาผสานกัน เพื่อดูว่าสิ่งที่เห็นอยู่คือสิ่งใด

การมองพื้นที่ย่อยของมนุษย์จะมีการแยกคุณลักษณะ (feature) ของพื้นที่ย่อยนั้น เช่น ลายเส้นและการตัดกันของสี ซึ่งการที่มนุษย์รู้ว่าพื้นที่ตรงนี้เป็นเส้นตรงหรือสีตัดกัน เพราะมนุษย์ดูทั้งจุดที่สนใจและบริเวณรอบ ๆ ประกอบกัน

เครือข่ายประสาทแบบคอนโวลูชันจัดว่าเป็นการเรียนรู้เชิงลึก (Deep learning) ซึ่งมีความแตกต่างไปจากการเรียนรู้ของเครื่องจักร (Machine learning) ทั่วๆไป ที่ผู้ใช้งานจะต้องทำการสกัดลักษณะเด่นด้วยตนเองก่อนจะป้อนเป็นอินพุตให้กับเครือข่ายประสาทเทียมใช้ในการเรียนรู้ ส่วนการเรียนรู้เชิงลึกนั้นมีการใช้เครือข่าย ANN ที่มีชั้นซ่อนเร้น (Hidden layers) หลายชั้นทั้งนี้เพื่อเพิ่มความสามารถในการคิดที่มากกว่าปกติทำให้สามารถทำการคำนวณโจทย์ปัญหาซับซ้อนได้สามารถใช้เทคนิคต่าง ๆได้มากขึ้น และที่สำคัญที่สุดคือ CNN สามารถทำการคิดอย่างเป็นขั้นเป็นตอนได้ซึ่งสามารถลอกเลียนแบบการทำงานของสมองมนุษย์ได้ดีขึ้น แผนภาพตัวอย่างสถาปัตยกรรมของเครือข่ายประสาทแบบคอนโวลูชัน แสดงในรูปที่ 2.1



รูปที่ 2.1 สถาปัตยกรรมของ Convolutional Neural Network
หมายเหตุ.จาก https://medium.com/.

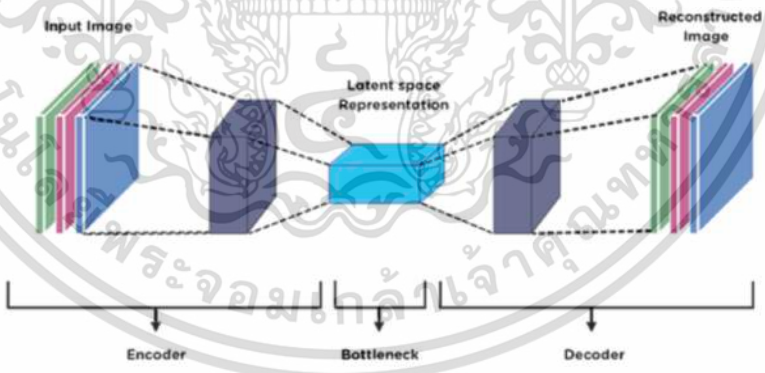
2.2 ตัวเข้ารหัสแบบคอนโวลูชัน (Convolutional Auto Encoders)

ลักษณะโครงสร้างของ Autoencoder จะประกอบไปด้วยส่วนของ Encoder, Latent layer และส่วนของ Decoder

Encoder จะประกอบไปด้วยชั้น convolutional layer ตามด้วย max pooling และขนาดของข้อมูลจะลดลงเรื่อย ๆเมื่อผ่านไปหลายๆชั้น โดยที่ชั้นแรกมีข้อมูลเท่ากับจำนวนพิกเซลของภาพ input จากนั้นเมื่อผ่านชั้น max pooling จะทำให้ขนาดของภาพเล็กลง จนนำไปสู่ชั้นที่เรียกว่า Latent layer

Latent layer หรือมีอีกชื่อว่า Code ก็คือแก่นใจความที่ถูกย่อมาจาก Encoder จะประกอบไปด้วย node เพียงชั้นเดียวที่สามารถเซ็ตค่าได้ว่าให้มีจำนวนเท่าไหร่ และจำนวน node นี้ก็เปรียบเสมือนจำนวนกระดาษที่อนุญาตให้ใช้ในการเขียนสรุป ยิ่งมีมากก็จะมีแนวโน้มที่จะได้ข้อมูลที่เป็นน้ำมากขึ้น แต่หากมีน้อยไปก็อาจจะสรุปได้ไม่ครบใจความสำคัญ

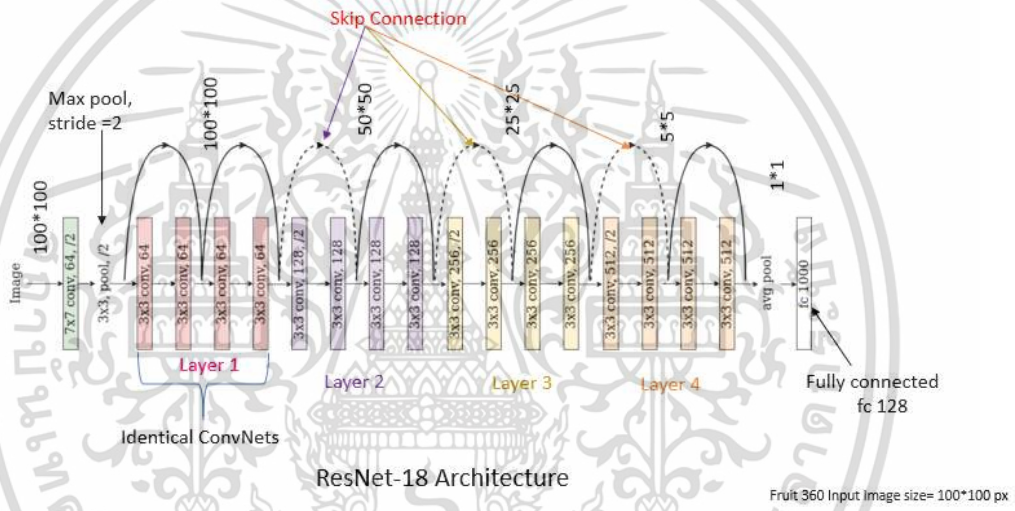
Decoder จะประกอบไปด้วยชั้น deconvolution หรือสามารถเลือกให้เป็น convolution ตามด้วย up sampling ก็ได้ ในส่วนนี้จะทำการขยายใจความจาก Code ที่ถูกย่อลงแล้ว ที่ได้ให้มีความสมบูรณ์มากที่สุด แต่แน่นอนว่าจะไม่มีทางเป็นเหมือนต้นฉบับแน่นอน อย่างไรก็ตามใจความที่ถอดออกมาได้ก็จะมีส่วนที่เป็นแก่นเสมอ แต่ส่วนที่เป็นน้ำนั้นก็มีสิทธิที่จะถูกหลงลืมหล่นหายไป



รูปที่ 2.2 สถาปัตยกรรมของ Convolutional Auto Encoders
หมายเหตุ. จาก https://beeying.medium.com/.

2.3 Deep Residual Network

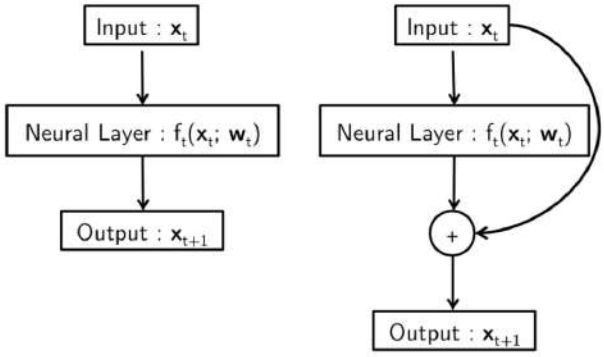
ResNet หรือ Residual Network ใช้การเรียนรู้ที่เหลือแทนที่จะพยายามเรียนรู้คุณสมบัติบางอย่าง ส่วนที่เหลือสามารถเข้าใจได้ง่ายๆ ว่าเป็นการลบคุณสมบัติที่เรียนรู้จากการป้อนข้อมูลของเลเยอร์นั้น แนวคิดหลักของ ResNet คือการนำเสนอการเชื่อมต่อทางลัดที่ข้ามเลเยอร์อย่างน้อยหนึ่งชั้น ตัวโครงข่ายนี้จะประกอบด้วย 4 บล็อกใหญ่ (stage) จำนวนเลเยอร์ที่มีพารามิเตอร์สำหรับการฝึกทั้งหมดจะกลายเป็นจำนวนชั้นที่ใช้ เนื่องจากโครงสร้างของ ResNet นั้นจัดให้เป็นคอนโวลูชันสองชั้นติดและมีเส้นเชื่อมกระโดดข้าม ดังนั้นเวลาบอกจำนวนในแต่ละบล็อกใหญ่ มักจะใช้เป็นเลขฐานสองแทน



รูปที่ 2.3 สถาปัตยกรรมของ ResNet

หมายเหตุ. จาก https://www.pluralsight.com/guides/introduction-to-resnet

โครงสร้างของ neural network และแนวคิดที่ใช้ในการพัฒนา deep neural network ให้มีจำนวน network layer เยอะมากๆ จนเป็นอนันต์ได้หนึ่งในตัวแบบโครงสร้างของ neural network ที่ต้องกล่าวถึง ที่ทำให้เราสามารถขยายจำนวน network layer (และยังสามารถ train ได้) จากหลักสิบไปถึงหลักพันคือการใช้โครงสร้างของ residual network[1] ซึ่งในโครงสร้างของ residual network นี้ output ของแต่ละ layer จะถูกบวกด้วย input ของ layer นั้นๆ ก่อนจะถูกส่งต่อให้กับ layer ถัดไป Figure 1 เปรียบเทียบความแตกต่างระหว่าง layer ใน neural network ทั่วไปและ layer ใน residual network (หรือนิยมเรียกว่า residual block)

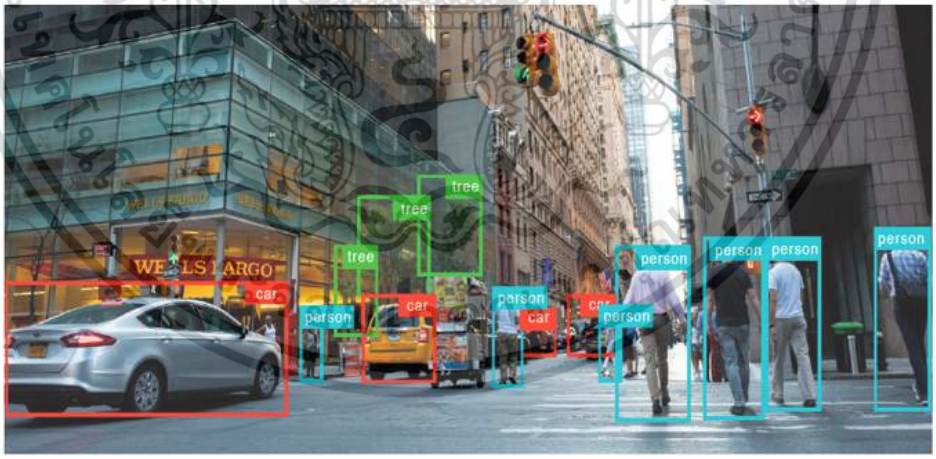


รูปที่ 2.3 การคำนวณ residual network

หมายเหตุ. จาก https://insight.avantis.finance/neural-networks-with-infinite-layers-5cd57b9da79c

2.4 Faster-RCNN

RCNN เป็นโครงข่ายประสาทเทียมที่เสนอพื้นที่ที่สนใจขึ้นมา โดยมีจุดเริ่มต้นจากการฝึกฝนและปรับแต่งตัวแบบจาก CNN โดยนำตัว Support Vector Machine มาช่วยในการแยกคลาส เพื่อให้สามารถตีกรอบพื้นที่ที่สนใจได้อย่างแม่นยำมากขึ้น

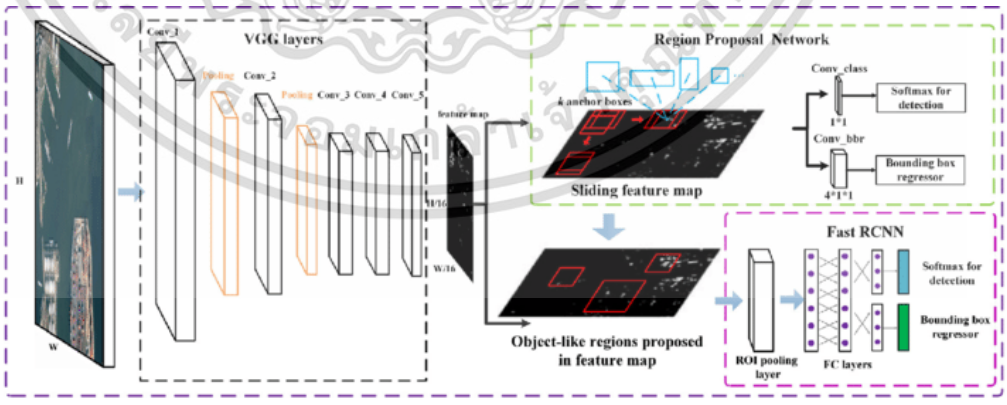


รูปที่ 2.4 การตรวจจับวัตถุโดยเสนอพื้นที่ที่สนใจ

หมายเหตุ. จาก https://medium.com/

ซึ่งในงานวิจัยนี้ได้เลือก Faster-RCNN มาช่วยในการวิเคราะห์ข้อมูลรูปภาพรถยนต์ เนื่องว่าตัว Faster-RCNN ช่วยในการกำจัดจุดอ่อนของ RCNN ในเรื่องของการที่ต้องใช้พื้นที่ย่อยในการเรียนรู้ทุกครั้ง และยังต้องนำพื้นที่ย่อยเหล่านั้นมาคำนวณ CNN ใหม่ทุกครั้ง ขั้นตอนการทำงานของ Faster-RCNN มีรายละเอียดดังนี้

- ภาพนำเข้าก็นำไปผ่านตัว Selective Search เพื่อหา proposed regions
- นำภาพทั้งภาพไปผ่าน CNN Barebone ตัวที่เลือกไว้ จะ VGG-16, Resnet, Inception แต่ network ที่เลือกมานั้นจะตัด fully connected layerทิ้งไปทั้งหมด จะใช้ผล feature map จาก convolution layer สุดท้ายเท่านั้น
- แต่ละ proposed region ที่ได้จาก Selective Search ที่เป็นพิกัดตำแหน่งบนภาพนำเข้าจะถูกนำมาหาพิกัดตำแหน่งที่สัมพันธ์กันกับขนาดของ feature map จาก convolution layer สุดท้าย (เพราะผ่าน CNN มา feature mapจะมีขนาดเล็กลง)
- จากพิกัดตำแหน่งบน feature map convolution layer สุดท้ายที่ได้ นำ region ตรงนั้นไปผ่าน ROI Pooling Layer เพื่อให้ได้ feature vector ที่มีขนาดคงที่ ซึ่งจะถูกส่งไปที่ fully connected layer กับ softmax เพื่อทำ classification ว่า proposed region feature นั้นจะตอบว่าเป็นคลาสอะไร (ไม่ได้ใช้ SVM มาเป็น classifier แล้ว ต่างจาก R-CNN เพราะ SVM train ยากและธรรมชาติของมันเองไม่ได้เป็น multiclass classifier)
- จากพิกัดตำแหน่งบนภาพนำเข้าจริงของ proposed region ก็จะถูกส่งเข้าไปที่ regressor network ร่วมกับ feature vector ที่ได้จาก ROI Pooling เพื่อปรับแก้พิกัดของวัตถุที่จะตอบออกไป



รูปที่ 2.5 สถาปัตยกรรมของ Faster-RCNN

หมายเหตุ.จาก https://medium.com/

2.5 ทฤษฎีการ Transfer Learning

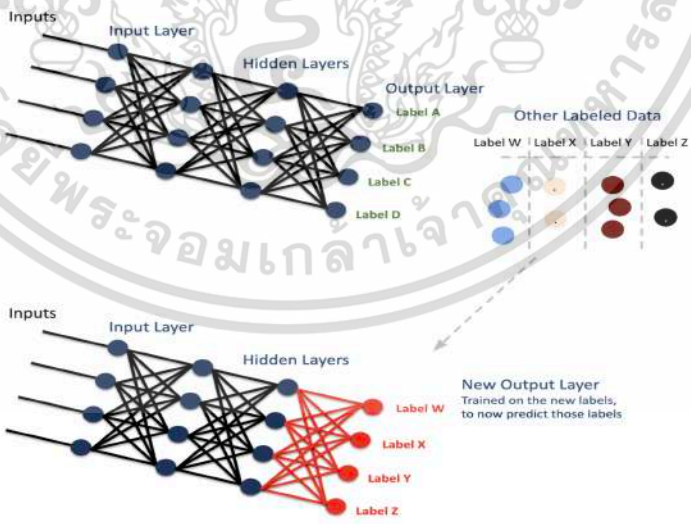
Transfer Learning คือ เทคนิคที่ช่วยลดเวลาการเทรนโมเดล Deep Learning ด้วยการนำบางส่วนของโมเดลที่เทรนเรียบร้อยแล้ว กับงานที่ใกล้เคียงกัน มาใช้เป็นส่วนหนึ่งของโมเดลใหม่

การใช้ Transfer Learning ส่วนใหญ่ แบ่งเป็น 3 แบบดังนี้

- ใช้ ConvNet เป็น Fixed Feature Extractor นำ ConvNet มาลบ Dense Layer สุดท้ายออกไป เราจะได้ Feature Extractor ที่เราสามารถสร้าง Linear Classifier (Head) เทรนให้ Classify Feature เหล่านี้สำหรับงานใหม่กับชุดข้อมูล Dataset ใหม่ที่มีขนาดเล็กกว่ามาก

- Fine-tuning โมเดล ConvNet แทนที่เราจะเทรนเฉพาะ Head เราสามารถ Fine-tuning ทั้งโมเดล ConvNet ทุก Layer เพื่อให้ได้ประสิทธิภาพที่ดีขึ้นกับงานใหม่และ Dataset ใหม่

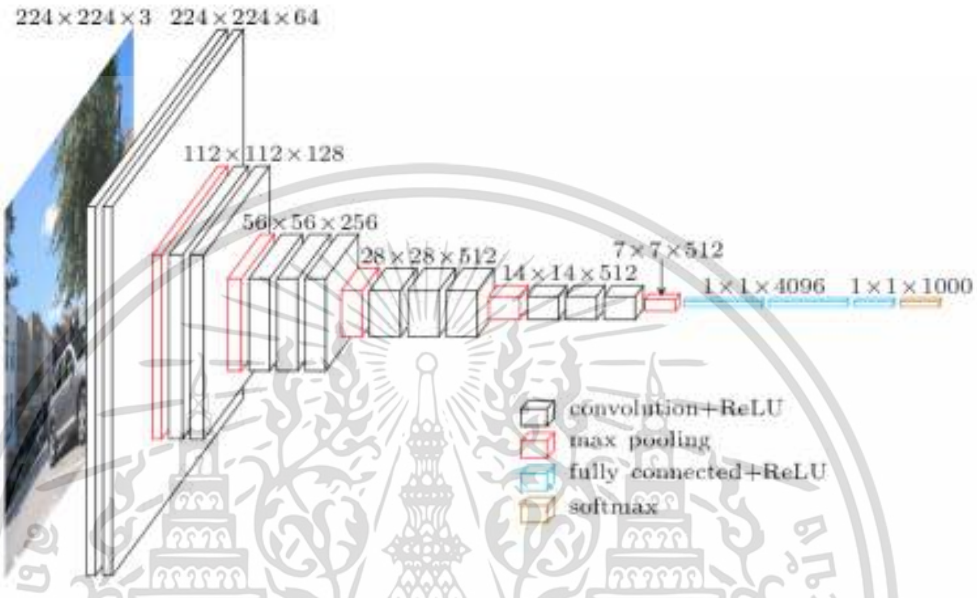
- Pretrained models เนื่องจาก ConvNet สมัยใหม่ ต้องใช้เวลาเทรนที่ยาวนานประมาณ 2-3 สัปดาห์ บนเครื่อง Server ความเร็วสูง ที่มีหลาย GPU จึงมีผู้นำ Pretrained models โมเดลที่เทรนเรียบร้อยแล้ว มาแชร์กันในอินเตอร์เน็ต ให้ผู้อื่นได้ใช้ เรียกว่า Model Zoo



รูปที่ 2.6 การทำงานของวิธี transfer learning
หมายเหตุ.จาก https://www.bualabs.com/

2.6 VGG16

ส่วน Architech ของ VGG 16 ลักษณะมี Hidden Layer 16 ชั้น จึงถูกเรียกว่า VGG 16 โดย Model VGG16 มาจากการแข่ง ImageNet ซึ่งเป็น Dataset ที่มีข้อมูลมากกว่า 14 ล้านรูป และมีภาพถึง 1000 classes. และแน่นอน Model VGG16 สามารถติด 1 ใน 5 ความแม่นยำในชุดข้อมูล Test (92.7%)



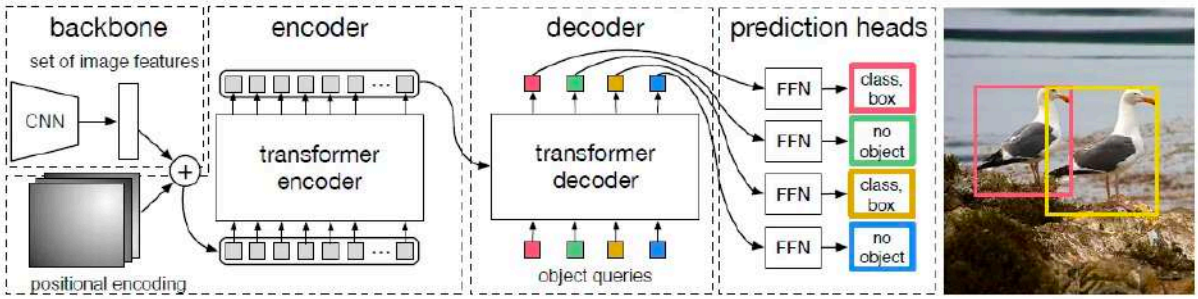
รูปที่ 2.7 การทำงานของวิธี transfer learning

หมายเหตุ.จาก https://www.cs.toronto.edu/~frossard/post/vgg16/

2.7 RCNN DC5

DC5 (Dilated-C5): ใช้แกนหลัก ResNet conv5 พร้อมการขยายใน conv5 และหัว conv มาตรฐาน และ FC สำหรับการทำนายหน้ากากและกล่องตามลำดับ สิ่งนี้ใช้โดยกระดาษ ConvNet ที่เปลี่ยนรูปได้

DC5 - Dilated convolution เป็นเวอร์ชันแก้ไขของอัลกอริทึม convolution เริ่มต้นที่การคำนวณบางจุดจะถูกข้ามไปตามชุดพารามิเตอร์การขยายที่กำหนด เป็นผลให้มีฟิลด์รับที่ใหญ่กว่าเลเยอร์คอนโวลูชั่นมาตรฐาน ในการขยายจะใช้กับชั้นการม้วนลอนที่ห้าของสถาปัตยกรรม ResNet และวิธีนี้เรียกว่า DC5



รูปที่ 2.8 สถาปัตยกรรมของ RCNN DC5

หมายเหตุ.จาก https://sh-tsang.medium.com/review-detr-end-to-end-object-detection-with-transformers-c64977be4b8e

- DETR ใช้แกนหลัก CNN ทั่วไปเพื่อเรียนรู้การแสดงภาพอินพุตแบบ 2 มิติ โดยทั่วไปเอาต์พุตจะมีหมายเลขช่อง C=2048 และ H,W=H0/32, W0/32 โดยที่ H0 และ W0 คือความสูงและความกว้างของภาพ
- แกนหลักคือ ResNet-50 หรือ ResNet-101 ที่ ImageNet ได้รับการฝึกฝนล่วงหน้าพร้อมชุดมาตรฐานแบบแช่แข็ง รุ่นที่เกี่ยวข้องเรียกว่า DETR และ DETR-R101 ตามลำดับ
- ความละเอียดของคุณลักษณะสามารถเพิ่มได้โดยการเพิ่มการขยายไปยังระยะสุดท้ายของกระดูกสันหลัง รุ่นที่เกี่ยวข้องเรียกว่า DETR-DC5 และ DETR-DC5-R101 (สเตจ C5 แบบขยาย) ตามลำดับ
- ใช้โมเดลที่มีตัวเข้ารหัส 6 ตัวและตัวถอดรหัส 6 ตัวที่มีความกว้าง 256 พร้อมหัวจับ 8 ตัว
- จากนั้นตัวถอดรหัส Transformer จะใช้การฝังตำแหน่งที่เรียนรู้ในจำนวนคงที่จำนวนเล็กน้อยที่เรียกว่าการสืบค้นวัตถุและเข้าร่วมเพิ่มเติมกับเอาต์พุตตัวเข้ารหัส
- การฝังตัวถอดรหัสเอาต์พุตแต่ละรายการจะถูกส่งผ่านไปยังเครือข่ายฟีดส่งต่อที่ใช้ร่วมกัน (FFN) (perceptron 3 ชั้นที่มีฟังก์ชันการเปิดใช้งาน ReLU และมิติที่ซ่อนอยู่ d) ที่คาดการณ์การตรวจจับ (คลาสและกล่องขอบเขต) หรือคลาส "ไม่มีวัตถุ" เช่น พิกัดกึ่งกลางมาตรฐาน ความสูงและความกว้างของกล่อง w.r.t. รูปภาพอินพุตและป้ายกำกับคลาสโดยใช้ฟังก์ชัน softmax

2.8 FPN for Fast RCNN

การใช้งาน FPN ใน RCNN ที่รวดเร็วนั้นง่ายมาก RCNN ที่รวดเร็วใช้เทคนิคข้อเสนอภูมิภาค เช่น การค้นหาแบบเลือกเพื่อสร้าง ROI และใช้การรวม ROI บนแผนที่คุณลักษณะมาตราส่วนเพื่อให้

ได้ผลลัพธ์สุดท้าย ด้วยการใช้ FPN เราจะมีแผนที่คุณลักษณะหลายรายการที่มีขนาดต่างกัน และเราต้องการกลยุทธ์ในการกำหนด ROI ที่กำหนดให้กับแผนที่คุณลักษณะ (ตอนนี้เรามีแผนที่คุณลักษณะหลายรายการ ซึ่งคุณลักษณะแผนที่ที่จะใช้สำหรับ ROI ที่กำหนด)เดียว

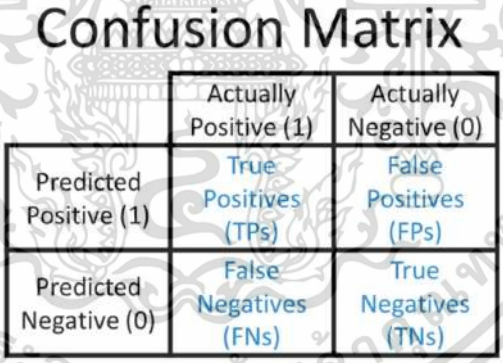
สูตรคำนวณ

$$k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor.$$

ที่นี่ 224 คือขนาดการฝึกอบรมของรูปภาพในชุดข้อมูล imagenet (resnet ที่ใช้จะถูกเก็บไว้ใน imagenet) k0 คือแผนผังคุณลักษณะที่กำหนด ROI ขนาด 224 ให้ w และ h คือความกว้างและความสูงของ ROI ส่วนหัวมีพารามิเตอร์ที่ใช้ร่วมกันสำหรับแต่ละแผนผังคุณลักษณะ

2.9 Confusion Matrix

Confusion Matrix ถือเป็นเครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนาย หรือ Prediction ที่ทำนายจากModelที่เราสร้างขึ้น ในMachine learning โดยมีไอเดียจากการวัดว่า สิ่งที่เราคิด (Model ทำนาย) กับ สิ่งที่เกิดขึ้นจริง มีสัดส่วนเป็นอย่างไร



รูปที่ 2.9 Confusion Matrix

หมายเหตุ.จาก https://medium.com/@pagongatchalee/confusion-matrix

- True Positive (TP)= สิ่งที่ทำนาย ตรงกับสิ่งที่เกิดขึ้นจริง ในกรณี ทำนายว่าจริง และสิ่งที่เกิดขึ้น ก็คือ จริง
- True Negative (TN)= สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณี ทำนายว่า ไม่จริง และสิ่งที่เกิดขึ้น ก็คือ ไม่จริง
- False Positive (FP)= สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง

- False Negative (FN)= สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้นคือ จริง

สามารถใช้ Confusion Matrix มาคำนวณ การประเมินประสิทธิภาพของการทำนายด้วย Model โดยใช้ค่าต่างๆ ดังนี้

- Precision : บอกว่าจากผลที่โมเดลทำนายเป็น positive ทั้งหมด โมเดลเราทาย positive ถูกจริงๆเท่าไหร่

สูตรคำนวณ $$Precicion = \frac{TP}{(TP + FP)}$$

- Recall : ทางสถิติเรียกว่า Sensitivity คือค่าที่บอกว่าโมเดลเราเจอ actual positive class ที่เราสนใจมากหรือน้อยเพียงใด

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

- F1 score : การหาค่าเฉลี่ยของ precision และ recall ไม่ได้บวกกันหารสองตรงๆ แต่เป็นค่าเฉลี่ยแบบ harmonic mean

สูตรคำนวณ $$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

2.10 งานวิจัยที่เกี่ยวข้อง

R.Rajkumar et al.(2020) ผู้วิจัยมุ่งเน้นการทำงานอัตโนมัติของการประมวลผลการประกันภัยรถยนต์โดยใช้ Deep Convolution Networks เนื่องจากข้อมูลที่จำกัด เราพบว่าการใช้การเรียนรู้การถ่ายโอนและการใช้ตัวเข้ารหัสอัตโนมัติแบบแปรผันเพื่อค้นหาคุณสมบัติทำงานได้ดี เราได้สร้าง 4 วิธีการจำแนกประเภทรถยนต์ การจำแนกประเภทว่ารถได้รับความเสียหายหรือไม่ การจำแนกประเภทความเสียหายที่เกิดขึ้นและความรุนแรงของความเสียหายตามลำดับ ผู้วิจัยแสดงวิธีการต่าง ๆ ที่สามารถนำมาใช้ในการวิเคราะห์ความเสียหายของรถยนต์ และบทความนี้ไม่ใช่การค้นหาอย่างละเอียดถี่ถ้วนในการประกันภัยความเสียหายของรถยนต์ทั้งโดเมนและการดำเนินการเรียกร้องค่าสินไหมทดแทน เอกสารนี้บันทึกเทคนิคต่าง ๆ ที่เราใช้เพื่อวิเคราะห์ความเสียหายของรถ

โดยผู้วิจัยได้เลือกงานวิจัยมาใช้อ้างอิงในการออกประเภทของ model ที่จะใช้วิเคราะห์ภาพอุบัติเหตุของรถยนต์ และอ้างอิงถึงหลักการใช้ Deep convolution network

Kalpesh Patil et al.(2017) การประมวลผลประกันภัยรถยนต์ตามภาพเป็นพื้นที่สำคัญที่มีขอบเขตขนาดใหญ่สำหรับระบบอัตโนมัติ ในบทความนี้ เราจะพิจารณาถึงปัญหาการจำแนกประเภท

ความเสียหายของรถยนต์ ซึ่งบางหมวดหมู่อาจมีความละเอียดที่ละเอียด เราสำรวจเทคนิคการเรียนรู้เชิงลึกเพื่อจุดประสงค์นี้ ในขั้นต้น เราพยายามฝึกอบรม CNN โดยตรง อย่างไรก็ตาม เนื่องจากข้อมูลที่มีป้ายกำกับชุดเล็ก จึงไม่สามารถใช้งานได้ดี จากนั้น เราจะสำรวจผลของการฝึกอบรมล่วงหน้าเฉพาะโดเมน ตามด้วยการปรับแต่งอย่างละเอียด สุดท้าย เราทดลองด้วยการถ่ายโอนการเรียนรู้และการเรียนรู้ทั้งมวล ผลการทดลองแสดงให้เห็นว่าการเรียนรู้แบบถ่ายโอนทำงานได้ดีกว่าการปรับแต่งเฉพาะโดเมน เรามีความแม่นยำถึง 89.5% ด้วยการผสมผสานระหว่างการถ่ายโอนและการเรียนรู้ทั้งมวล

ผู้วิจัยได้เลือกงานวิจัยนี้มาเพื่ออ้างอิงถึงวิธีการจำแนกประเภทความเสียหายของรถยนต์ เทคนิคการเรียนรู้เชิงลึก และเทคนิคการโอนถ่ายข้อมูลเพื่อช่วยในการสร้าง model ตามในวัตถุประสงค์ที่ตั้งไว้ของงานวิจัยนี้

Javier O. Pinzon Arenas et al.(2018) บทความนี้นำเสนอการใช้งานการจำลอง mobile robotic arm ซึ่งมีหน้าที่เรียงวัตถุแบบสุ่มในพื้นที่ทำงาน ซึ่งสำหรับการพัฒนาการบทความนี้ได้มีการปรับใช้ Faster R-CNN ซึ่งจะสามารถระบุและค้นหาองค์ประกอบที่ไม่ได้ถูกจัดเรียงไว้ได้แม่นยำถึง 99% และสามารถทดสอบในเวลาแบบตามจริง (real-time) ได้แบบ 100% กล่าวคือ หุ่นยนต์สามารถดักจับและค้นหาวัตถุทั้งหมดให้สามารถจัดเรียงได้ โดยคำนึงถึงว่า virtual environment ต้องมีการควบคุม รวมถึงเรื่องของขนาดของภาพ input ที่ได้รับจาก workplace ที่จะเข้า network จำเป็นต้องมีขนาด 700x525 พิกเซล

ผู้วิจัยได้เลือกงานวิจัยนี้มาศึกษาเนื่องจากต้องที่จะศึกษาทฤษฎีและวิธีการใช้งานของ Faster-RCNN เพื่อที่ผู้วิจัยจะนำมาใช้เป็นแนวทางในการวิเคราะห์รูปภาพอุบัติเหตุของรถยนต์

บทที่ 3
วิธีการดำเนินงานวิจัย

การค้นคว้าอิสระครั้งนี้ได้ศึกษาการประมวลผลภาพอุบัติเหตุรถยนต์โดยใช้กระบวนการโครงข่ายประสาทแบบคอนโวลูชันเพื่อมีอัลกอริทึมในการสร้างโมเดลการจำแนกรถ การจำแนกว่ารถยนต์ได้รับความเสียหายหรือไม่ การจำแนกตำแหน่งความเสียที่เกิดขึ้น ช่วยแนะนำความเสียหายที่เกิดขึ้นว่าตรงกับพระราชบัญญัติจราจรทางบกในหมวดรถวิ่งตามทางเดียวกัน

ซึ่งงานวิจัยนี้มีขั้นตอนการดำเนินงานดังนี้

3.1 วิธีการและขั้นตอนการดำเนินงาน

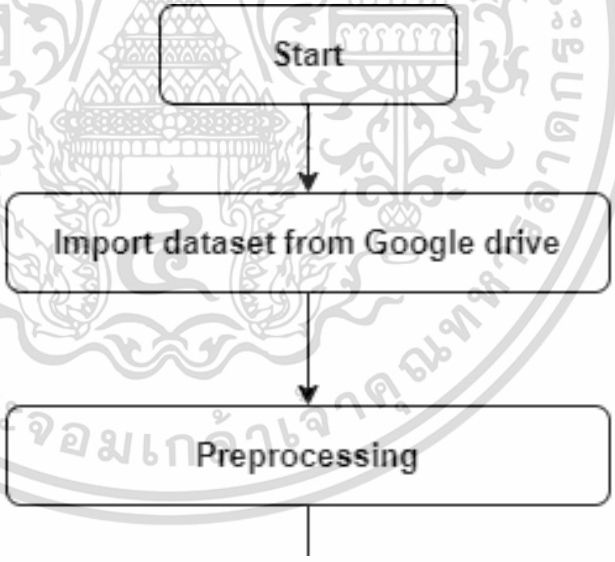
3.2 เครื่องที่ใช้ในการวิจัย

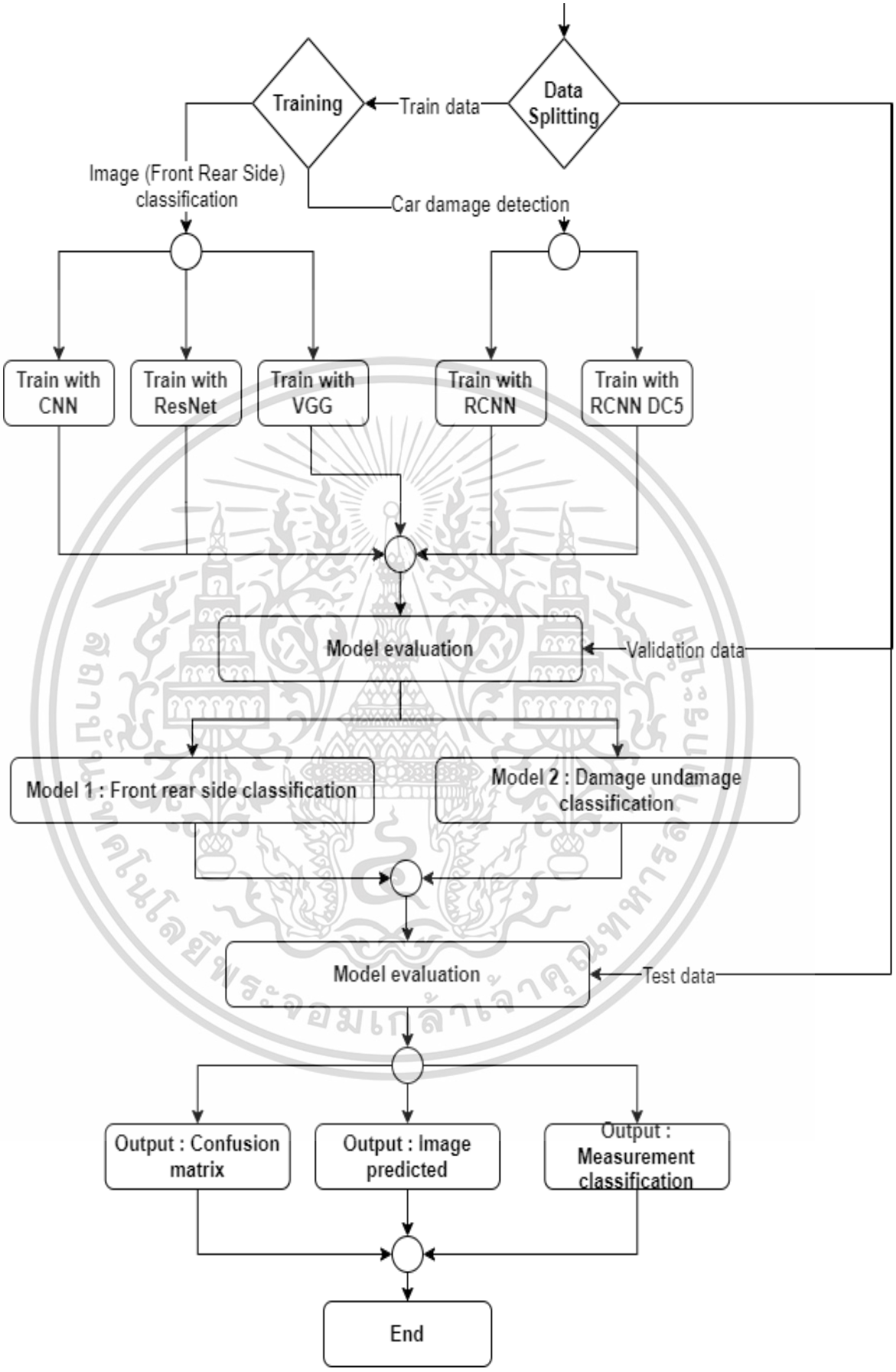
3.3 การนำเข้าข้อมูล

3.4 การวิเคราะห์ข้อมูล

3.1 วิธีการและขั้นตอนการดำเนินงาน

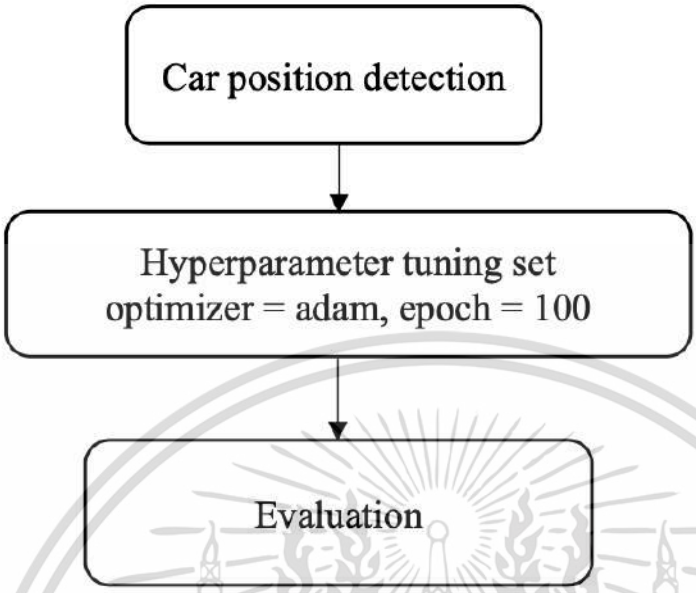
ขั้นตอนที่ 1 ขั้นตอนวิเคราะห์ภาพอุบัติเหตุรถยนต์



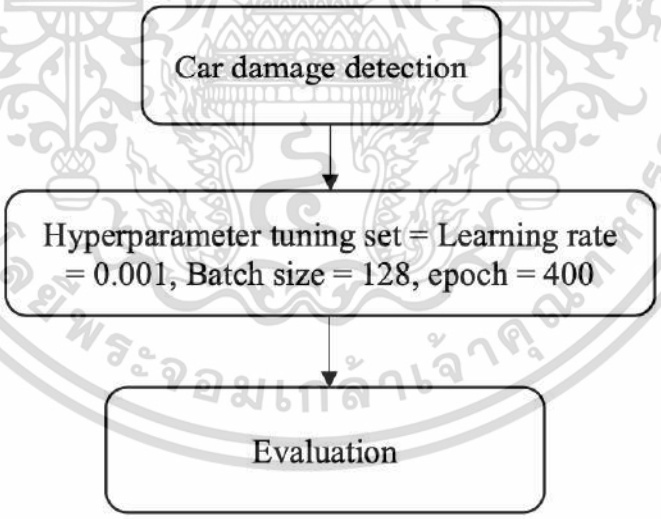


ขั้นตอนที่ 2 การเข้า Model เพื่อจำแนกข้อมูลในส่วนต่างๆ

- Model 1 : การตรวจสอบตำแหน่งของตัวรถที่ได้รับความเสียหาย



- Model 2 : การตรวจสอบว่ามีความเสียหายที่เกิดขึ้นกับตัวรถ



ขั้นตอนที่ 3 การ Prediction โดยข้อมูลที่เป็นชุด data test ที่เก็บเข้า google drive

3.2 เครื่องมือที่ใช้ในการวิจัย

3.2.1 ใบบันทึกข้อมูลแจ้งอุบัติเหตุรถยนต์ ของสมาคมประกันวินาศภัย โดยใบบันทึกแจ้งอุบัติเหตุรถยนต์จะมีรายละเอียดในการบันทึก ได้แก่ ข้อมูลผู้เอาประกัน วันที่เกิดเหตุ ข้อมูลผู้ขับขี่รถประกัน ลักษณะการเกิดเหตุ และแผนที่เกิดเหตุ

3.2.2 รูปภาพของอุบัติเหตุรถยนต์ที่เกิดขึ้น

ตารางที่ 3.1 ตัวอย่างความเสียหายที่เกิดจากอุบัติเหตุรถยนต์การชนด้านหน้ารถ

ความเสียหายที่กระโปรงหน้ารถ	
ความเสียหายไฟด้านหน้าซ้าย/ขวา	

ตารางที่ 3.2 ตัวอย่างความเสียหายที่เกิดจากอุบัติเหตุรถยนต์การชนด้านข้างขวา

ความเสียหายประตูด้านขวาหน้า/หลัง	
ความเสียหายกระจกข้างด้านขวา	

ตารางที่ 3.3 ตัวอย่างความเสียหายที่เกิดจากอุบัติเหตุรถยนต์การชนด้านข้างซ้าย

ความเสียหายประตูด้านซ้ายหน้า/หลัง	
ความเสียหายกระจกข้างด้านซ้าย	

ตารางที่ 3.4 ตัวอย่างความเสียหายที่เกิดจากอุบัติเหตุรถยนต์การชนด้านหลัง

ความเสียหายไฟท้ายซ้าย/ขวา	
ความเสียหายกระโปรงท้าย	

3.2.3 กระบวนการ classification เพื่อนำมา classification ตัวรถยนต์ว่าเป็นส่วนของรถยนต์ได้แก่ หน้ารถ ด้านข้างรถ และหลังรถ โดยใช้ Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

3.2.4 กระบวนการที่ใช้ในการวิเคราะห์รูปภาพในส่วนที่ใช้ทำนายความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ โดยใช้ Method R CNN และ Method R CNN DC5

3.2.5 โปรแกรม google colab ใช้ในการวิเคราะห์ข้อมูล

3.3 การนำเข้าข้อมูล

ในการเก็บรวบรวมข้อมูลรูปภาพอุบัติเหตุรถยนต์ ได้ทำการเก็บรูปที่ใช้สำหรับ train model และ test model โดยรูปภาพที่ใช้trainเป็นรูปข้อมูลอุบัติเหตุรถยนต์ ที่เป็น open data และข้อมูลที่ใช้ test จะเป็นภาพถ่ายอุบัติเหตุรถยนต์จากบริษัทประกันภัย ซึ่งได้เก็บรวมข้อมูลไว้ที่ google drive

3.4 การวิเคราะห์ข้อมูล

3.4.1 การแบ่งกลุ่มตัวรถยนต์ว่าเป็นส่วนของรถยนต์ ได้แก่ หน้ารถ ด้านข้างรถ และหลังรถ โดยใช้ Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

- การ normalize data เพื่อให้สามารถนำข้อมูลเข้า model
- วิเคราะห์ค่า accuracy ของการ train model
- นำข้อมูลมา prediction
- วิเคราะห์ผล Confusion Matrix

3.4.2 การวิเคราะห์ความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ โดยใช้ Method R CNN และ Method R CNN DC5

- train และข้อมูลสำหรับการ validate หลังจากได้นั้นได้นำข้อมูลรูปและ meta data มาประกอบกันเพื่อได้ label data
- วิเคราะห์ค่า accuracy ของการ train model R CNN
- นำข้อมูลมา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย
- วิเคราะห์ผล Confusion Matrix

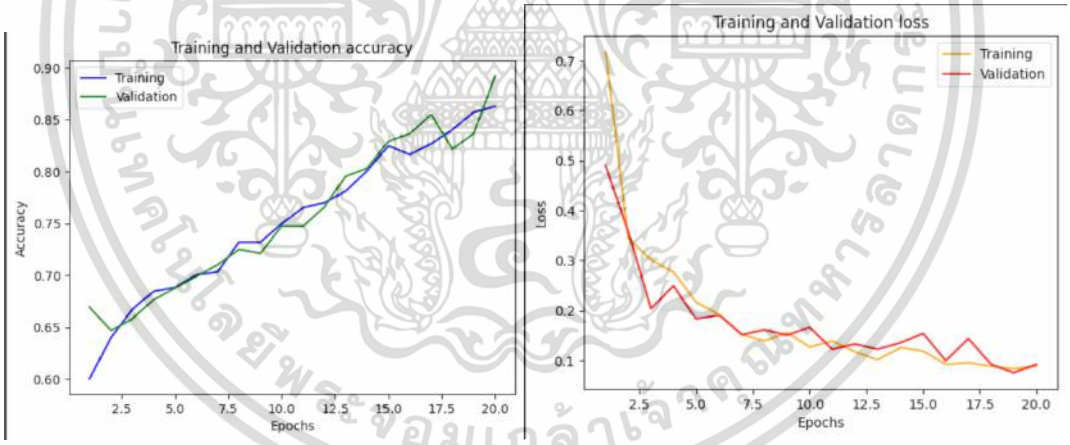
บทที่ 4
ผลวิจัยและอภิปรายผล

จากการนำข้อมูลที่ใช้ในการ train และ test โดยข้อมูลที่ใช้ในการ test นั้น เป็นข้อมูลจากสมาคมประกันวินาศภัย เมื่อนำมาพิจารณาจะได้ผลลัพธ์ข้อมูลดังนี้

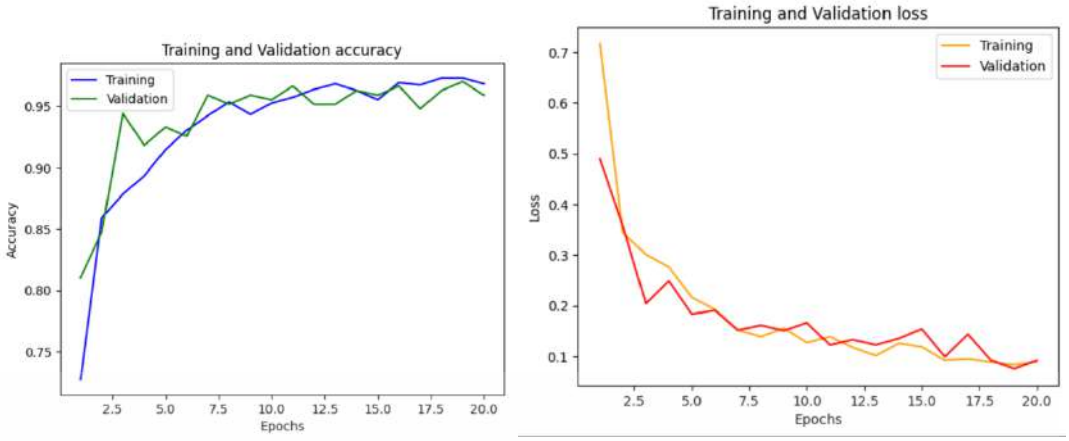
4.1 ผลการแบ่งกลุ่มตัวรถยนต์ว่าเป็นส่วนของรถยนต์ ได้แก่ หน้ารถ ด้านข้างรถ และหลังรถ โดยใช้ Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

4.1.1 การนำข้อมูล train และ test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้จากการทำนายรูปภาพว่าเป็นส่วนใดของรถยนต์ โดยเริ่มจากการการ normalize date เพื่อให้สามารถนำข้อมูลเข้า model ได้ง่ายขึ้น ขั้นตอนต่อมาได้สร้าง model CNN , Method VGG16 และ ResNet โดยใช้ optimizer เป็น Adam

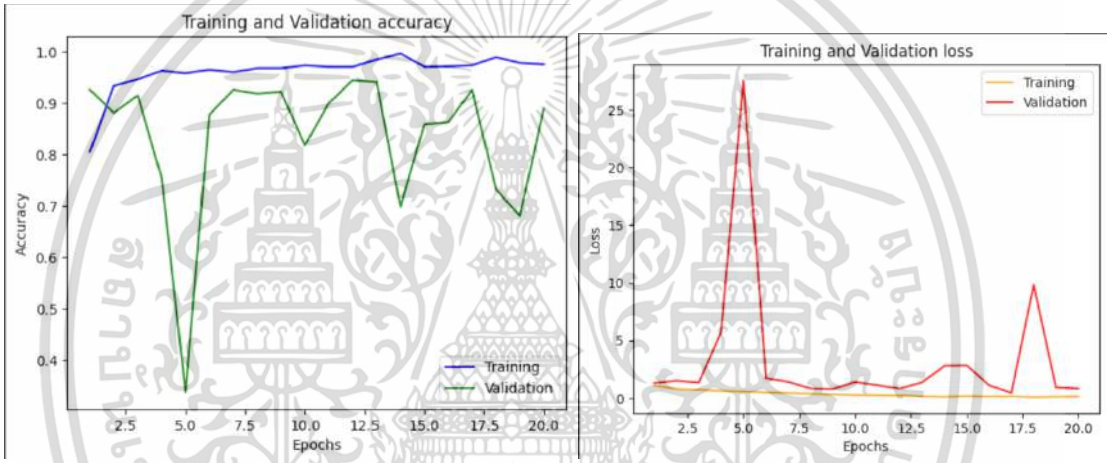
เมื่อทำการ train model จากขั้นตอนด้านบน จะเห็นได้ว่ากราฟที่ได้ออกมาตามภาพด้านว่า เมื่อยิ่งทำการ train มาเท่าไหร่ ยิ่งทำให้ค่า accuracy สูงมากขึ้น(กราฟด้านซ้ายมือ) ซึ่งผกผันกับค่า loss ที่ลดลง (กราฟด้านขวามือ) เมื่อทำการ train model มากยิ่งขึ้น



รูปที่ 4.3 กราฟค่า Accuracy และ Loss ของ CNN Model กับ Training ,Validation Dataset



รูปที่ 4.4 กราฟค่า Accuracy และ Loss ของ VGG 16 Model กับ Training ,Validation Dataset



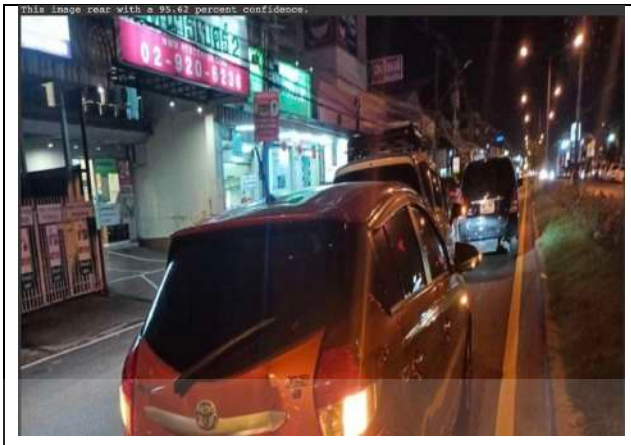
รูปที่ 4.5 กราฟค่า Accuracy และ Loss ของ ResNet Model กับ Training ,Validation Dataset

จากรูปที่ 4.3-4.4 ค่า Accuracy และค่า Loss ของ CNN และ VGG 16 กับ Training ,Validation Dataset มีลักษณะคล้ายกัน คือ เมื่อมี epochs ที่มากขึ้น ค่า Accuracy ก็จะสูงขึ้นตาม โดยมีค่า Accuracy ประมาณ 0.9 ส่วนค่า Loss นั้นจะสวนทางกันโดยมี epochs ที่มากขึ้น ค่า Loss ก็จะยิ่งน้อยลงอยู่ที่ประมาณ 0.1

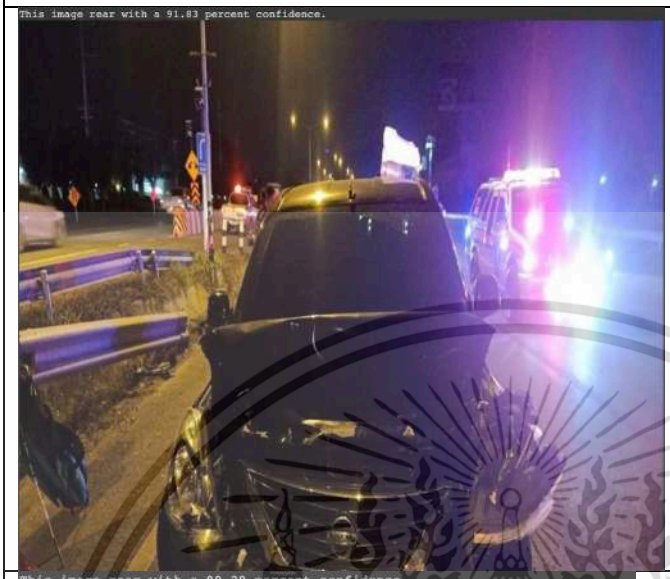
จากรูปที่ 4.5 ค่า Accuracy และค่า Loss ของ ResNet Model กับ Training ,Validation Dataset มีลักษณะกราฟของค่า Accuracy และค่า Loss ที่ training dataset เมื่อมี epochs ที่มากขึ้น ค่า Accuracy ก็จะสูงขึ้นตาม ส่วน Validation Dataset จะมีการแปรผันค่อนข้างสูง

ตารางที่ 4.1 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method CNN มี optimizer เป็น adam

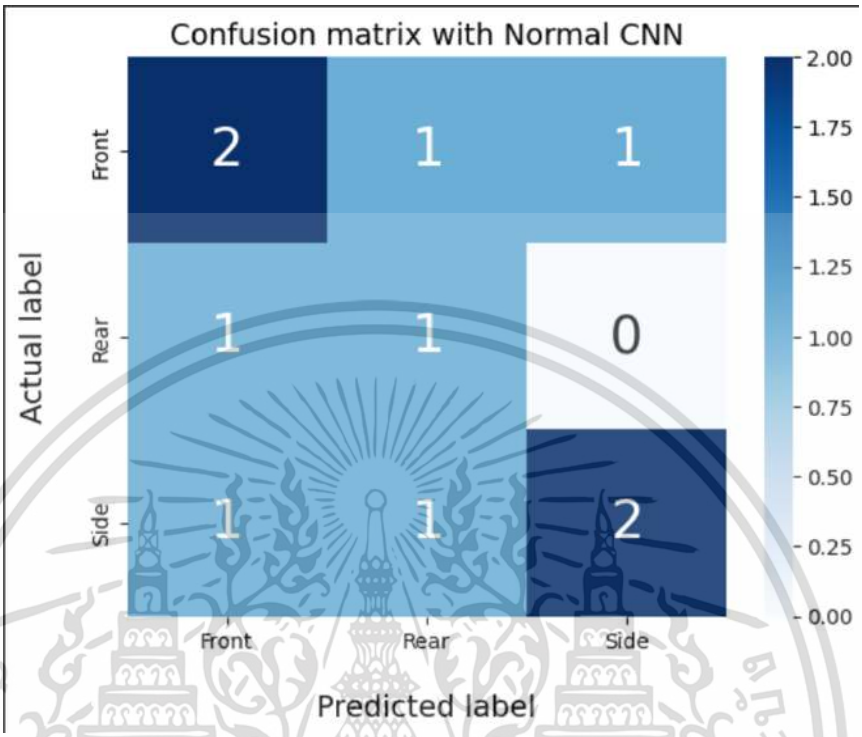
รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านข้างของรถยนต์ด้วย accuracy 100%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านข้างของรถยนต์ด้วย accuracy 43.71%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 100%

	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านข้างของรถยนต์ด้วย accuracy 95.20%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 100%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 100%

ตารางที่ 4.1 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method CNN มี optimizer เป็น adam **(ต่อ)**

รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 91.83%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านข้างของรถยนต์ด้วย accuracy 99.29%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 87.23%

การที่จะนำ Model ไปใช้งานนั้น จำเป็นต้องมีการวัดประสิทธิภาพ Model ก่อนว่า Model นั้นมีประสิทธิภาพเพียงพอที่จะนำมาพัฒนาหรือนำไปใช้งานด้านต่างๆ ซึ่งการวัดประสิทธิภาพนั้นส่วนใหญ่จะวัดค่าข้อมูลโดย Confusion Matrix



รูปที่ 4.6 ภาพแสดงตาราง confusion matrix ของ CNN Model ของการทำนายตำแหน่งรถยนต์

ในรูปที่ 4.6 ผลลัพธ์ที่ทำนายพบว่าจากจากรูปภาพที่เป็นหน้ารถทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้ารถ 2 รูป ทำนายผิดเป็นหลังรถ 1 รูป ทำนายผิดเป็นด้านข้าง 1 รูป และจากรูปภาพที่เป็นหลังรถทั้งหมด 2 รูป มีการทำนายถูกเป็นหลังรถ 1 รูป ทำนายผิดเป็นหน้ารถ 1 รูป และรูปภาพด้านข้างทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้าข้าง 2 รูป มีการทำนายเป็นหลังรถ 1 รูป ทำนายผิดเป็นหน้ารถ 1 รูป

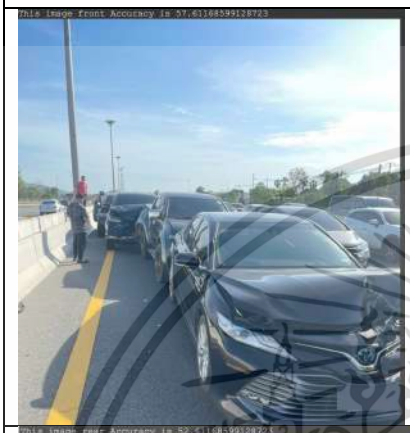
ตารางที่ 4.2 ตารางผลลัพธ์ Precision, Recall ของ CNN Model ของการทำนายตำแหน่งรถยนต์

Class	Precision	Recall	F1-score
Side	0.50	0.50	0.50
Rear	0.33	0.50	0.40
Front	0.67	0.50	0.57

จากตารางที่ 4.2 จะได้ว่า ค่า Precision ของ Side, Rear และ Front มีค่า0.50, 0.33 และ 0.67 ตามลำดับ ซึ่งทำให้เห็นได้ว่า CNN Model มีความแม่นยำในการทำนาย Class Side และ Front

มากกว่า Rear เมื่อนำผลในตารางที่ 4.1 ตาราง Confusion matrix และค่า Precision, Recall มาวิเคราะห์รวมกันจะเห็นได้ว่า Model สามารถทำนายในส่วนของ Front และ Side ได้ดีกว่า

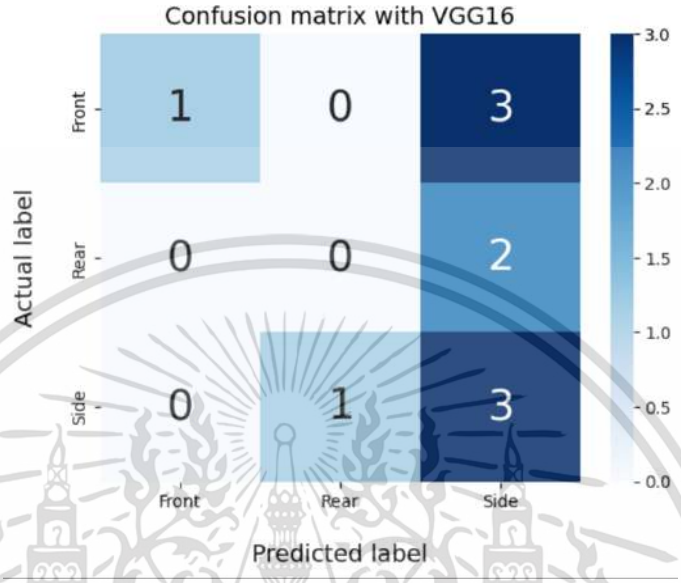
ตารางที่ 4.3 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method VGG16

รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 57.61%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านข้างของรถยนต์ด้วย accuracy 52.61%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 65.81%

ตารางที่ 4.3 (ต่อ) ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method VGG16

รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 60.19%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 52.01%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 67.51%

การที่จะนำ Model ไปใช้งานนั้น จำเป็นต้องมีการวัดประสิทธิภาพ Model ก่อนว่า Model นั้นมีประสิทธิภาพเพียงพอที่จะนำมาพัฒนาหรือนำไปใช้งานด้านต่างๆ ซึ่งการวัดประสิทธิภาพนั้นส่วนใหญ่จะวัดค่าข้อมูลโดย Confusion Matrix



รูปที่ 4.7 ภาพแสดงตาราง confusion matrix ของ VGG16 Model ของการทำนายตำแหน่งรถยนต์

ในรูปที่ 4.7 ผลลัพธ์ที่ทำนายพบว่าจากจากรูปภาพที่เป็นหน้ารถทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้ารถ 1 รูป ทำนายผิดเป็นด้านข้าง 3 รูป และจากรูปภาพที่เป็นหลังรถทั้งหมด 2 รูป มีการทำนายเป็นหลังรถ 0 รูป ทำนายผิดเป็นด้างข้าง 2 รูป และรูปภาพด้านข้างทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้าด้าน 3 รูป ทำนายผิดเป็นหลังรถ 1 รูป

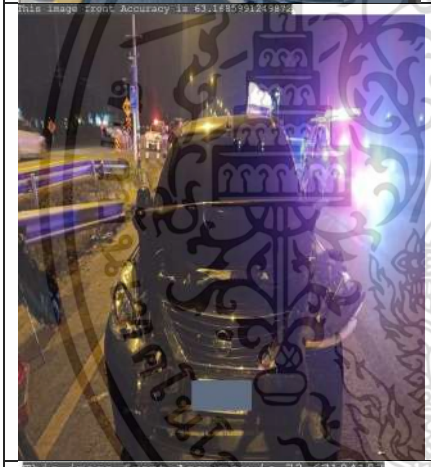
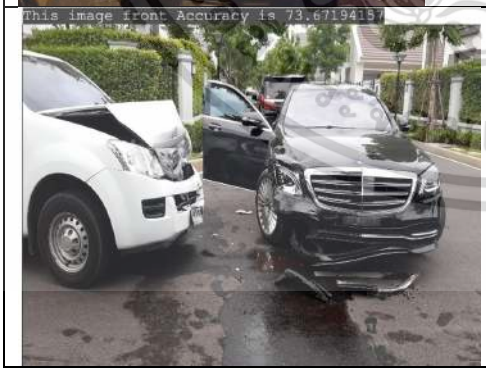
ตารางที่ 4.4 ตารางผลลัพธ์ Precision, Recall ของ VGG16 Model ของการทำนายตำแหน่งรถยนต์

Class	Precision	Recall	F1-score
Side	1.00	0.25	0.40
Rear	0.00	0.00	0.00
Front	0.38	0.75	0.50

จากตารางที่ 4.4 จะได้ว่า ค่า Precision ของ Side, Rear และ Front มีค่า 1.00,0.00 และ 0.38ตามลำดับ ซึ่งทำให้เห็นได้ว่า VGG16 Model มีความแม่นยำในการทำนาย Class Side มากกว่า Front และมากกว่า Rear เมื่อนำผลในตารางที่ 4.3 ตาราง Confusion matrix และค่า Precision,

Recall มาวิเคราะห์รวมกันจะเห็นได้ว่า Model สามารถทำนายในส่วนของ Side ได้ดีกว่า Front และ Rear

ตารางที่ 4.5 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method ResNet

รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 58.61%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 63.16%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy73.67%

ตารางที่ 4.5 (ต่อ) ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยใช้ Method ResNet

รูปภาพตัวรถยนต์	ผลลัพธ์
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 64.21%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 50.62%
	ทำนายส่วนของรถยนต์ทำนายว่าเป็นด้านหน้าของรถยนต์ด้วย accuracy 63.59%

รูปที่ 4.8 ภาพแสดงตาราง confusion matrix ของ ResNet Model ของการทำนายตำแหน่งรถยนต์

ในรูปที่ 4.8 ผลลัพธ์ที่ทำนายพบว่าจากจากรูปภาพที่เป็นหน้ารถทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้ารถ 0 รูป ทำนายผิดเป็นหลังรถทั้งหมด 4 รูป และจากรูปภาพที่เป็นหลังรถ 2 รูป ทำนายผิดเป็นหลังรถทั้งหมด 2 รูป มีการทำนายเป็นหลังรถ 4 รูป และรูปภาพด้านข้างทั้งหมด 4 รูป มีการทำนายถูกต้องว่าเป็นหน้าด้าน 0 รูป ทำนายผิดเป็นหลังรถทั้งหมด 4 รูป ซึ่งจากที่ได้ผลทำนายดังภาพนี้อาจจะเกิดจาก Overfitting

ตารางที่ 4.6 ตารางผลลัพธ์ Precision, Recall ของ ResNet Model ของการทำนายตำแหน่งรถยนต์

Class	Precision	Recall	F1-score
Side	0.00	0.00	0.00
Rear	0.20	1.00	0.33
Front	0.00	0.00	0.00

จากตารางที่ 4.6 จะได้ว่า ค่า Precision ของ Side, Rear และ Front มีค่า 0.00, 0.20 และ 0.00 ตามลำดับ ซึ่งทำให้เห็นได้ว่า VGG16 Model มีความแม่นยำในการทำนาย Class Rear มากกว่า Front และSide เมื่อนำผลในตารางที่ 4.5 ตาราง Confusion matrix และค่า Precision, Recall มาวิเคราะห์รวมกันจะเห็นได้ว่า Model ทำนายเป็น Rear ทั้งหมด ซึ่งการผลในลักษณะนี้อาจจะเกิด Model มีการ Overfitting

เมื่อนำมาเปรียบเทียบผลของทั้ง 3 Model พบว่า CNN Model นั้นมีการทำนายตำแหน่งรถยนต์ในแต่ละ Class ได้มีความถูกต้องมากกว่า และลำดับถัดมาเป็น VGG 16 Model ซึ่งทำนาย

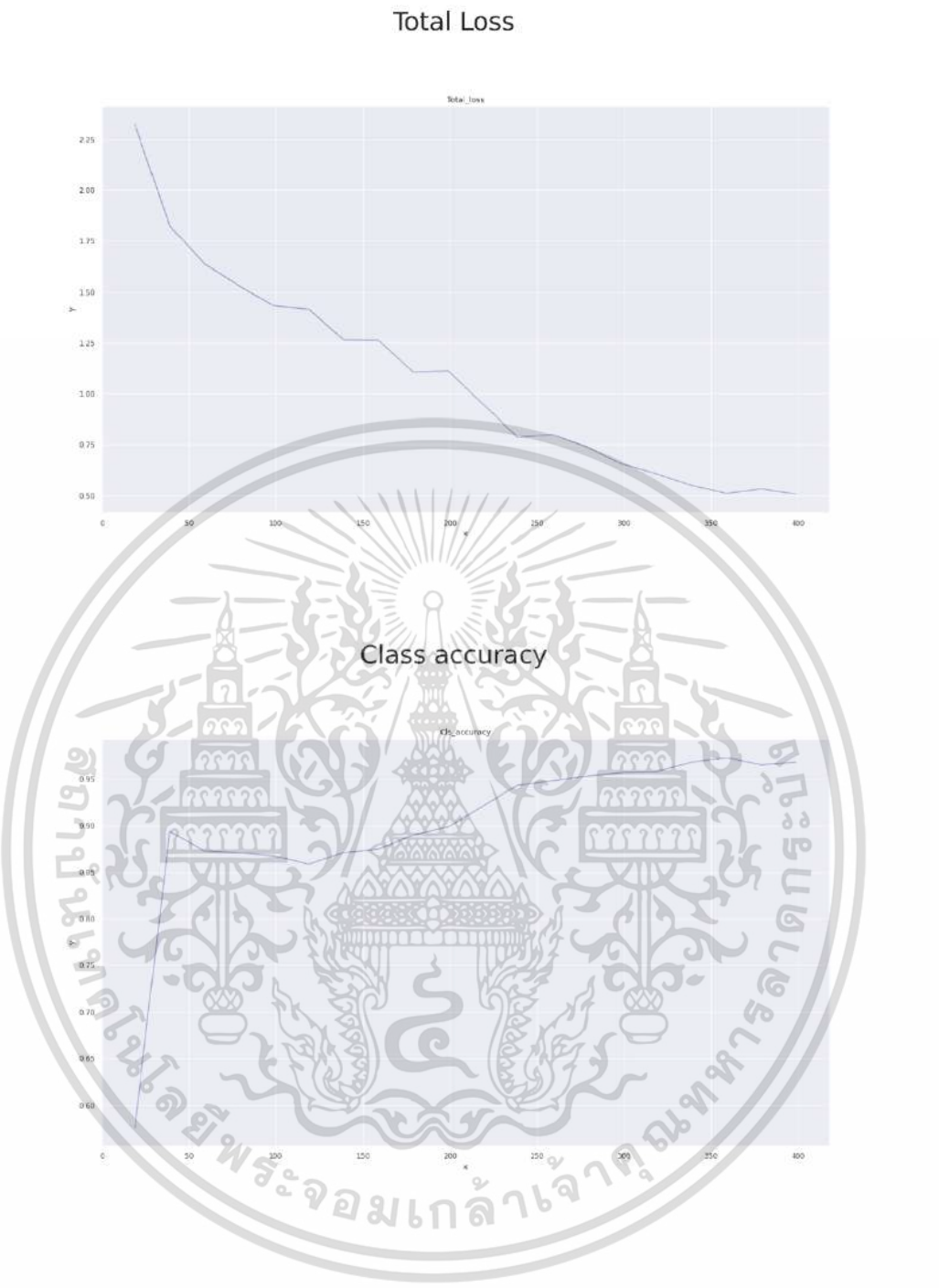
แม่นยำในส่วนของSide class มากกว่า Class อื่นๆ และสุดท้ายคือ ResNet Model ที่มีการเกิด Overfitting จึงทำให้ผลทำนายเป็น Rear class ทั้งหมด

4.2 ผลการวิเคราะห์ความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ โดยใช้ Method R CNN และ Method R CNN DC5

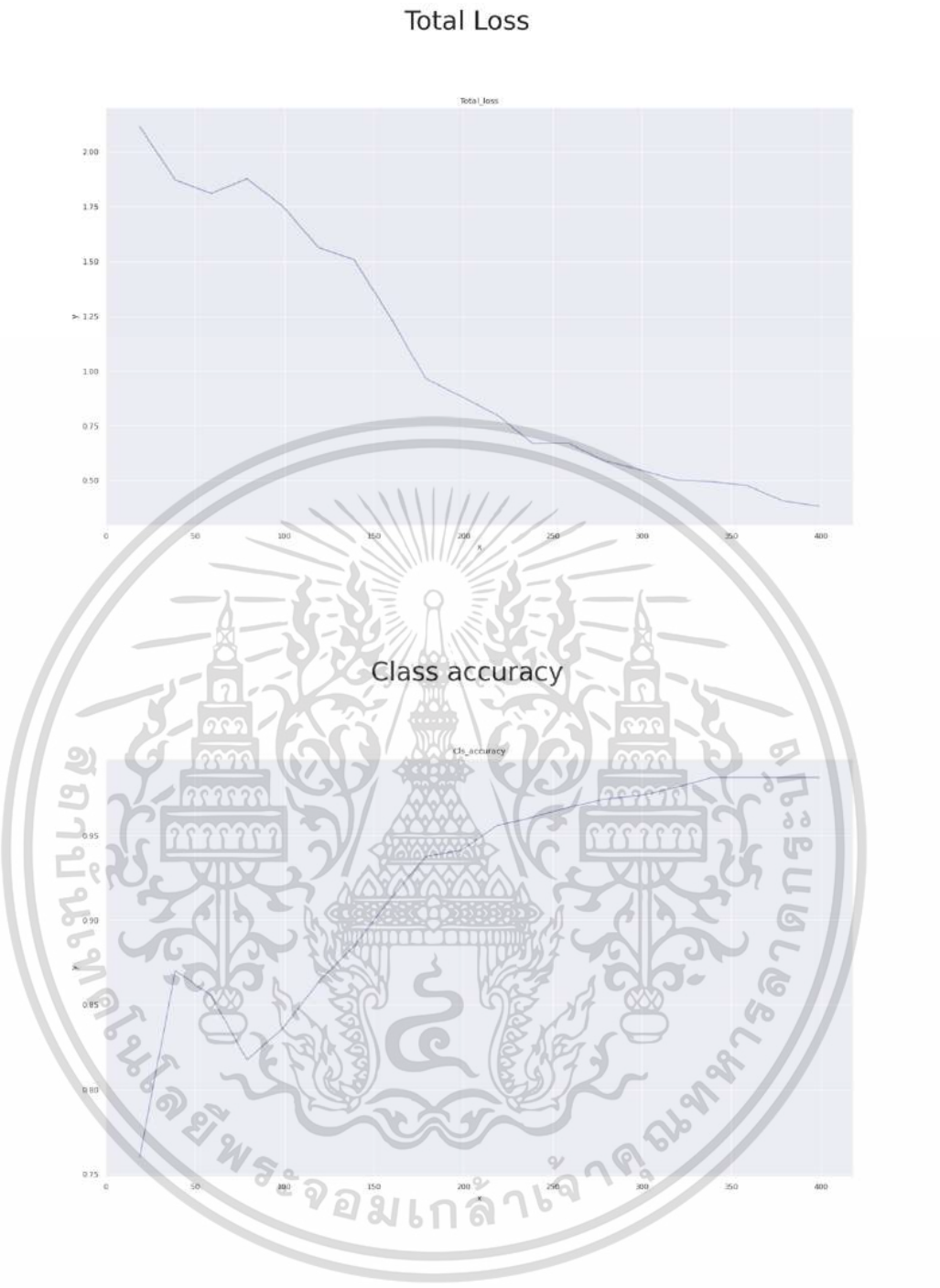
4.2.1 การนำข้อมูล train และ test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหาย โดยทำการเรียกข้อมูลที่อยู่ทั้งในส่วนของข้อมูลที่ใช้สำหรับการ train และข้อมูลสำหรับการ validate หลังจากได้นั้นได้นำข้อมูลรูปและ meta data มาประกอบกันเพื่อได้ label data

เมื่อได้ข้อมูล label แล้ว ได้ทำการกำหนด Hyperparameter โดย กำหนด learning rate =0.01 , Batch size=128 และ epochs=400 เมื่อทำการ train ข้อมูลตามค่าต่างๆที่กำหนด จะได้กราฟค่า loss จากการ train model ทั้งหมด 400 รอบ ดังนี้

จากรูปที่ 4.9 เห็นได้ว่าในช่วงเริ่มต้นในการ train model จะมีค่า loss ที่มาก เมื่อ train จนครบ 400 ค่า loss จะลดลงตามกราฟ และในส่วนของกราฟ accuracy เมื่อทำการtrain ไปจนครบ 400 รอบ จะเห็นได้กราฟมีแนวโน้มเพิ่มขึ้น หมายว่าค่าเมื่อจำนวนรอบที่ train มากขึ้น ค่า accuracy ก็มีการเพิ่มขึ้น



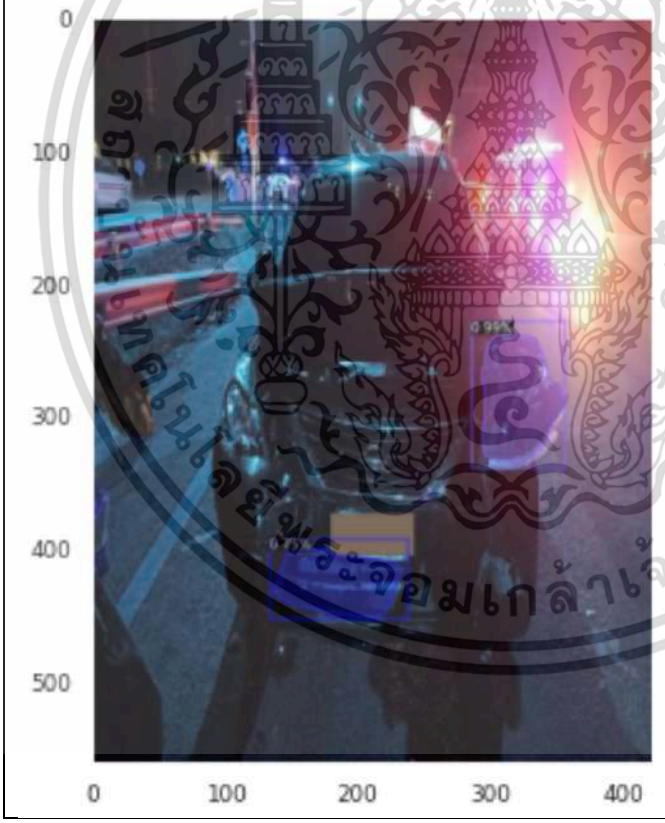
รูปที่ 4.9 กราฟค่า loss และค่า Accuracy จากการ R CNN model



รูปที่ 4.10 กราฟค่า loss จากการ R CNN DC5 model

จากรูปที่ 4.10 เห็นได้ว่าในช่วงเริ่มต้นในการ train model จะมีค่า loss ที่มาก เมื่อ train จนครบ 400 ค่า loss จะลดลงตามกราฟ และในส่วนของกราฟ accuracy เมื่อทำการtrain ไปจนครบ 400 รอบ จะเห็นได้กราฟมีแนวโน้มเพิ่มขึ้น ซึ่งทำให้เห็นได้ว่า R CNN และ R CNN DC5 มีลักษณะของกราฟทั้งสองที่เหมือนกัน

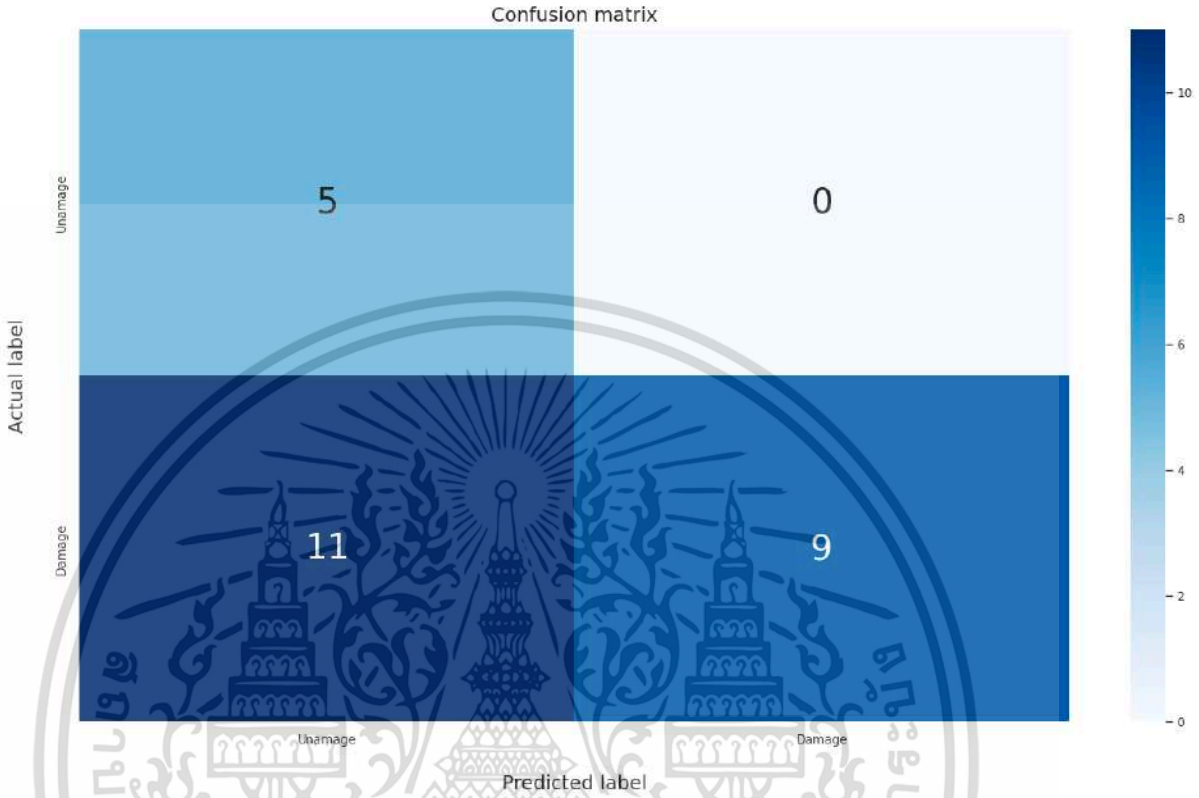
ตารางที่ 4.7 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายโดยใช้ Method R CNN

ภาพความเสียหาย	ผลลัพธ์
	ภาพความเสียหายของรถยนต์ได้ 3 จุด ในส่วนหน้ารถยนต์สีดำ 2 จุด โดยมีค่า accuracy 98%, 76% ตามลำดับ และรถยนต์คันสีขาว 1 จุด โดยมีค่า accuracy 72%
	ภาพความเสียหายของรถยนต์ได้ 2 จุด โดยมีค่า accuracy 99%, 75% ตามลำดับ

ตารางที่ 4.7 (ต่อ) ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายโดยใช้ Method R CNN

ภาพความเสียหาย	ผลลัพธ์
	ภาพความเสียหายของรถยนต์ด้านรถยนต์ของรถบรรทุก โดยมี accuracy 78% 76% ตามลำดับ
	ภาพความเสียหายของรถยนต์โดยจับความเสียหายของรถยนต์คันหน้าสุด คันกลาง และ คันหลังสุด โดยคันหน้าจับความเสียหายได้ที่บริเวณหน้ารถยนต์ ที่กระโปรงหน้ารถ ไฟหน้า และล้อรถ โดยมี accuracy 62% 67% ตามลำดับ คันกลางมี accuracy 60% และคันหลังสุด accuracy 61%

การที่จะนำ Model ไปใช้งานนั้น จำเป็นต้องมีการวัดประสิทธิภาพ Model ก่อนว่า Model นั้นมีประสิทธิภาพเพียงพอที่จะนำมาพัฒนาหรือนำไปใช้งานด้านต่างๆ ซึ่งการวัดประสิทธิภาพนั้นส่วนใหญ่จะวัดค่าข้อมูลโดย Confusion Matrix



รูปที่ 4.11 ภาพแสดงตาราง confusion matrix ของ R CNN Model ของการทำนายความเสีย

ในรูปที่ 4.11 ผลลัพธ์ที่ทำนายพบว่าจากจากรูปภาพที่มีความเสียหายทั้งหมด 20 รูป มีการทำนายถูกต้องว่ามีความเสียหาย 9 รูป ทำนายผิดเป็นไม่มีความเสียหาย 11 รูป และจากรูปภาพที่ไม่มีความเสีย 5 รูป มีการทำนายถูกทั้งหมดเป็นไม่มีความเสียหาย 5 รูป

ตารางที่ 4.8 ตารางผลลัพธ์ Precision, Recall ของ R CNN Model ของการทำนายความเสีย

Class	Precision	Recall	F1-score
Undamage	0.45	1.00	0.62
Damage	1.00	0.70	0.82

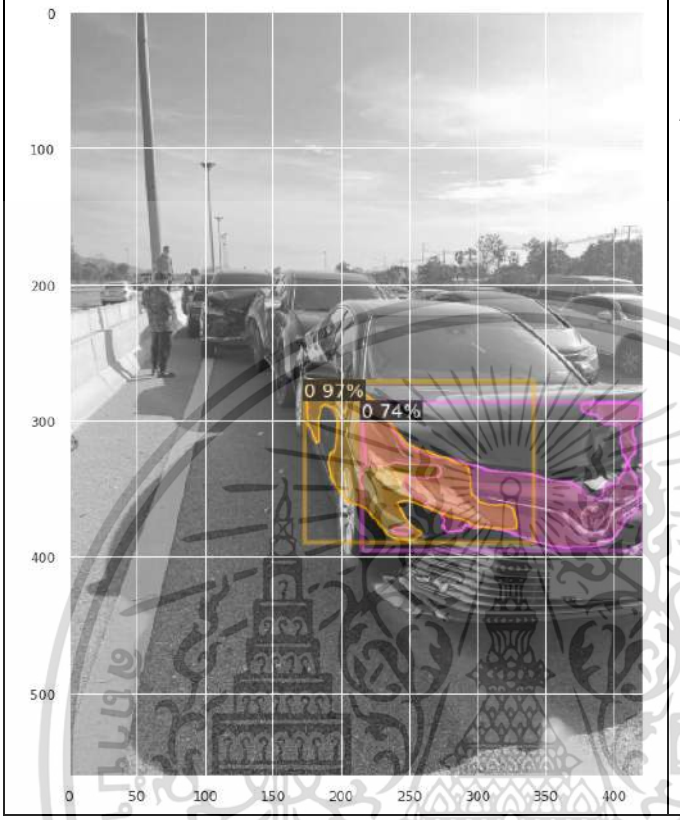
จากตารางที่ 4.9 จะได้ว่า ค่า Precision ของ Undamage, Damage มีค่า 0.45 , 1.00 ตามลำดับ ซึ่งทำให้เห็นว่า R CNN Model มีความแม่นยำในการทำนาย Class Damage มากกว่า Undamage และค่า Recall ของ Undamage เท่ากับ 1 อาจเนื่องมาจากข้อมูลที่ใช้ Test น้อยเกินไป

เมื่อนำผลในตารางที่ 4.7 ตาราง Confusion matrix และค่า Precision, Recall มาวิเคราะห์รวมกันจะเห็นได้ว่า Model สามารถทำนายส่วนที่เป็นความเสียหายของรถยนต์ได้เป็นอย่างดี

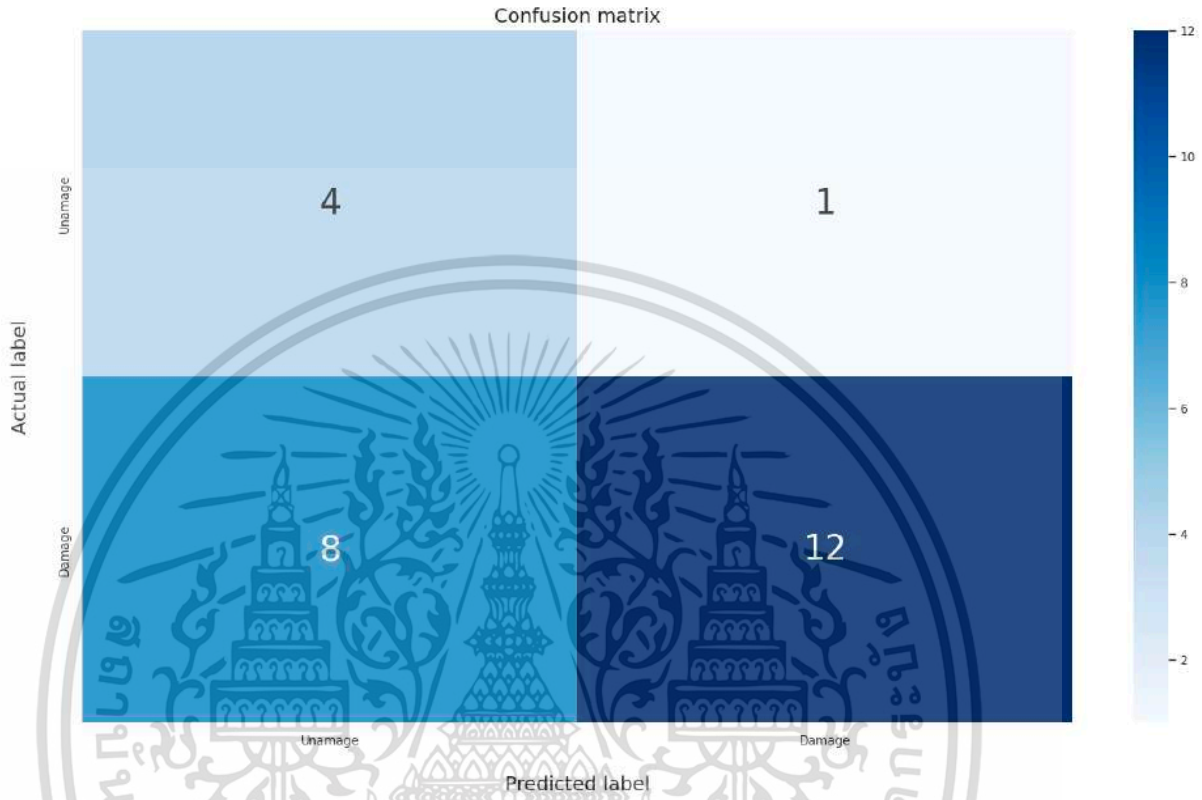
ตารางที่ 4.9 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายโดยใช้ Method R CNN DC5

ภาพความเสียหาย	ผลลัพธ์
	ภาพความเสียหายของรถยนต์ได้ 3 จุด ในส่วนหน้ารถยนต์สีดำ โดยมีค่า accuracy 72%, 89% และ 84%
	ภาพความเสียหายของรถยนต์ในส่วนหน้ารถยนต์สีดำ โดยมีค่า accuracy 88% และ 97%
	ภาพความเสียของรถยนต์โดยจับภาพความเสียหายได้ในบริเวณหน้ารถบรรทุก โดยมีค่า accuracy 84% และ 72% ซึ่งอีกบริเวณที่สามารถจับภาพความเสียหายได้ในบริเวณท้ายรถเก๋ง โดยมีค่า accuracy77% 95% และ 97%

ตารางที่ 4.9 (ต่อ) ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายโดยใช้ Method R CNN DC5

ภาพความเสียหาย	ผลลัพธ์
	ภาพความเสียหายของรถยนต์โดยจับความเสียหายของรถยนต์คันหน้าสุด โดยมีค่า accuracy74% และ 97%

การที่จะนำ Model ไปใช้งานนั้น จำเป็นต้องมีการวัดประสิทธิภาพ Model ก่อนว่า Model นั้นมีประสิทธิภาพเพียงพอที่จะนำมาพัฒนาหรือนำไปใช้งานด้านต่างๆ ซึ่งการวัดประสิทธิภาพนั้นส่วนใหญ่จะวัดค่าข้อมูลโดย Confusion Matrix



รูปที่ 4.11 ภาพแสดงตาราง confusion matrix ของ R CNN DC 5 Model ของการทำนายความเสีย

ในรูปที่ 4.11 ผลลัพธ์ที่ทำนายพบว่าจากจากรูปภาพที่มีความเสียหายทั้งหมด 20 รูป มีการทำนายถูกต้องว่ามีความเสียหาย 12 รูป ทำนายผิดเป็นไม่มีความเสียหาย 8 รูป และจากรูปภาพที่ไม่มีความเสียหาย 5 รูป มีการทำนายเป็นไม่มีความเสีย 4 รูป ทำนายผิดเป็นมีความเสียหาย 1 รูป

ตารางที่ 4.10 ตารางผลลัพธ์ Precision, Recall ของ R CNN Model ของการทำนายความเสีย

Class	Precision	Recall	F1-score
Undamage	0.44	0.80	0.57
Damage	0.94	0.85	0.83

จากตารางที่ 4.10 จะได้ว่า ค่า Precision ของ Undamage, Damage มีค่า 0.45 , 1.00 ตามลำดับ ซึ่งทำให้เห็นว่า R CNN DC5 Model มีความแม่นยำในการทำนาย Class Damage มากกว่า Undamage

นำผลในตารางที่ 4.9 ตาราง Confusion matrix และค่า Precision, Recall มาวิเคราะห์รวมกันจะเห็นได้ว่า Model สามารถทำนายส่วนที่เป็นความเสียหายของรถยนต์ได้เป็นอย่างดี

เมื่อนำผลวิเคราะห์ของทั้ง 2 Model มาเปรียบเทียบกัน พบว่าการทำนายความเสียหายรถยนต์ทั้ง 2 Model ทำนายผลได้ใกล้เคียงกัน แต่ในส่วนการทำนายรูปไม่มีความเสียหาย R CNN Model สามารถทำนายได้อย่างถูกมากกว่า R CNN DC5 Model

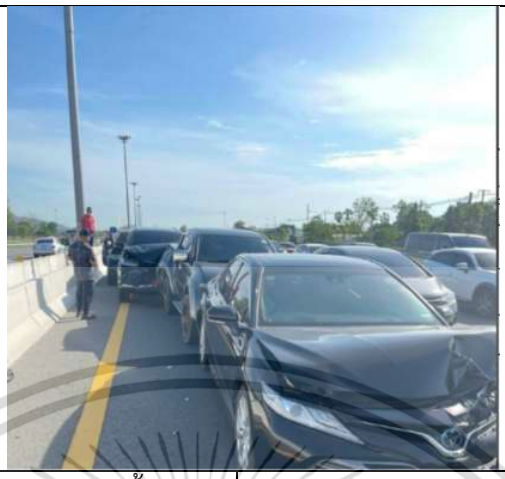
4.3 ผลการวิเคราะห์ข้อมูลรูปภาพและตำแหน่งจากความเสียหายจากข้อมูลการเกิดเหตุเพื่อวิเคราะห์ความเสียหายที่สอดคล้องกับใบเคลม

4.3.1 การวิเคราะห์ข้อมูลข้อมูลรูปภาพและการเลือกตำแหน่งจากความเสียหายจากข้อมูลการเกิดเหตุ เพื่อวิเคราะห์ความเสียหายที่สอดคล้องกับใบเคลม โดยนำข้อมูลในการวิเคราะห์ ในข้อ 4.1 และ 4.2 มาวิเคราะห์กับตำแหน่งความสียหายที่ได้ถูกบันทึกในใบเคลม ดังนี้

ตารางที่ 4.11 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหน้าของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

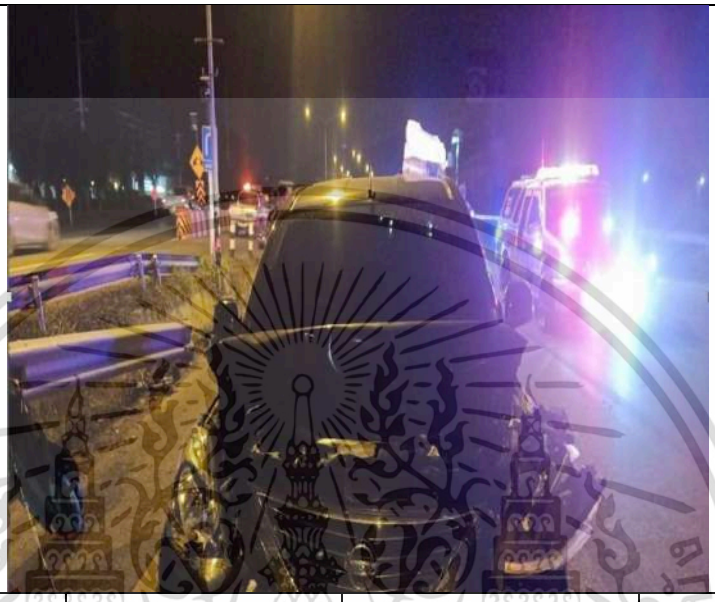
รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Front	Front	95.09%
VGG16	Front	Front	57.61%
ResNet	Front	Front	58.61%

ตารางที่ 4.12 ข้อมูลที่ถูกบันทึกในใบเคลม

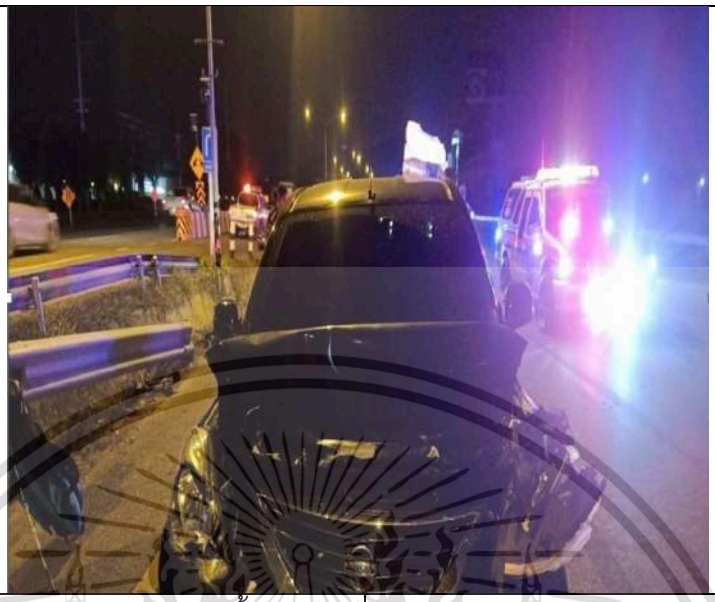
รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า กระโปรงท้าย
ข้อมูลจริงจากบริษัทประกันภัย	จุดเกิดเหตุที่ชนท้ายรถต่อๆกัน

จากตารางที่ 4.11 – 4.12 เมื่อนำผลมาเทียบกับพบว่าทั้ง 3 Model ได้ให้พบการทำนายเป็นด้านหน้ารถทั้ง 3 Model และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งหน้ารถยนต์ และหลังรถยนต์ ซึ่งมีผลตรงกันในการระบุเสียหน้ารถยนต์ แต่ในส่วนท้ายรถยนต์นั้นทั้ง 3 Model ไม่สามารถทำนายได้ เนื่องจากภาพที่นำมาเข้า Model นั้นเป็นเพียงด้านหน้ารถยนต์

ตารางที่ 4.13 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหน้าของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Front	Front	95.81%
VGG16	Front	Side	52.61%
ResNet	Front	Front	63.16%

ตารางที่ 4.14 ข้อมูลที่ถูกบันทึกในใบเคลม

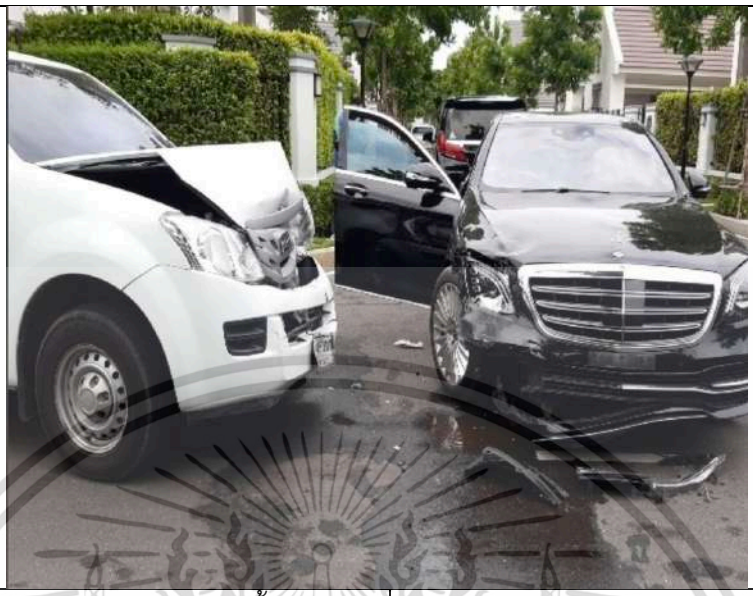
รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันชนกับด้านหน้ารถคู่กรณี

จากตารางที่ 4.13 – 4.14 เมื่อนำผลมาเทียบกับพบว่าทั้ง 2 Model ได้ให้พบการทำนายเป็นด้านหน้ารถ ยกเว้น VGG 16 Model ทำนายเป็นด้านข้าง และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งหน้ารถยนต์ ซึ่งมีผลตรงกันกับ CNN, ResNet Modelในการระบุเสียหน้ารถยนต์

ตารางที่ 4.15 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหน้าของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Front	Front	100%
VGG16	Front	Front	65.81%
ResNet	Front	Front	73.67%

ตารางที่ 4.16 ข้อมูลที่ถูกบันทึกในใบเคลม

รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันชนกับด้านหน้ารถคู่กรณี

จากตารางที่ 4.15 – 4.16 เมื่อนำผลมาเทียบกับพบว่าทั้ง 3 Model ได้ให้พบการทำนายเป็นด้านหน้ารถ และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งหน้ารถยนต์ ซึ่งมีผลตรงกันในการระบุเสียหน้ารถยนต์

ตารางที่ 4.17 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหน้าของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Front	Side	99.29%
VGG16	Front	Front	60.19%
ResNet	Front	Front	64.21%

ตารางที่ 4.18 ข้อมูลที่ถูกบันทึกในใบเคลม

รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า กระโปรงท้าย ไฟหน้าซ้าย
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันถูกคู่กรณีชนด้านท้ายรถ

จากตารางที่ 4.17 – 4.18 เมื่อนำผลมาเทียบกับพบว่าทั้ง 2 Model ได้ให้พบการทำนายเป็นด้านหน้ารถ ยกเว้น CNN Model ทำนายเป็นด้านข้าง และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งหน้ารถยนต์ และหลังรถยนต์ ซึ่งมีผลตรงกันกับ VGG16, ResNet Modelในการระบุเสียหน้ารถยนต์ แต่ในส่วนท้ายรถยนต์นั้นทั้ง 3 Model ไม่สามารถทำนายได้ เนื่องจากภาพที่นำมาเข้า Model นั้นเป็นเพียงด้านหน้ารถยนต์

ตารางที่ 4.19 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหน้าของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Front	Front	100%
VGG16	Front	Front	67.51%
ResNet	Front	Front	63.59%

ตารางที่ 4.20 ข้อมูลที่ถูกบันทึกในใบเคลม

รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า กระโปรงท้าย
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันถูกคู่กรณีชนด้านท้ายรถ

จากตารางที่ 4.19 – 4.20 เมื่อนำผลมาเทียบกับพบว่าทั้ง 3 Model ได้ให้พบการทำนายเป็นด้านหน้ารถทั้ง 3 Model และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งหน้ารถยนต์ และหลังรถยนต์ ซึ่งมีผลตรงกันในการระบุเสียหน้ารถยนต์ แต่ในส่วนท้ายรถยนต์นั้นทั้ง 3 Model ไม่สามารถทำนายได้ เนื่องจากภาพที่นำมาเข้า Model นั้นเป็นเพียงด้านหน้ารถยนต์

ตารางที่ 4.21 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามจุดความเสียหายที่เป็นส่วนด้านหลังของรถยนต์โดย Method CNN, Method VGG16 และ ResNet มี optimizer เป็น adam

รูปภาพตัวรถยนต์			
			
Model	Actual	Prediction	Accuracy
CNN	Rear	Side	95.62%
VGG16	Rear	Front	52.01%
ResNet	Rear	Front	50.62%

ตารางที่ 4.22 ข้อมูลที่ถูกบันทึกในใบเคลม

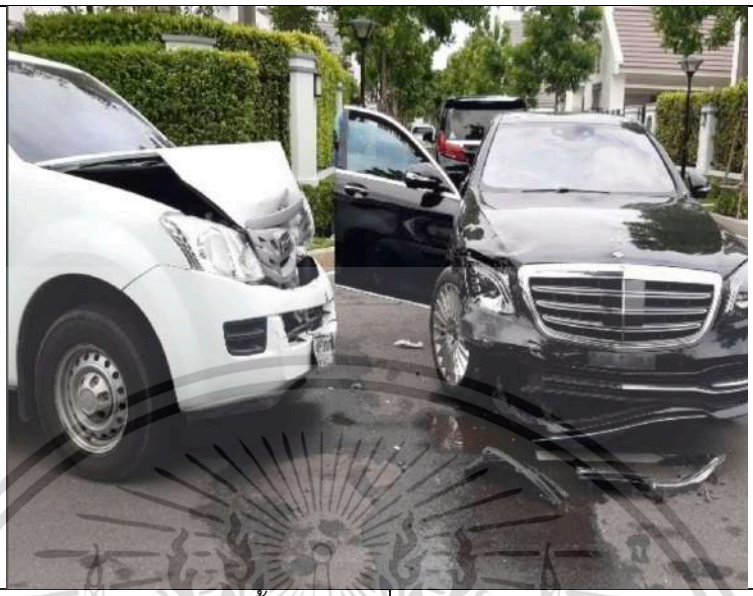
รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงท้าย
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันถูกคู่กรณีชนด้านท้ายรถ

จากตารางที่ 4.21 – 4.22 เมื่อนำผลมาเทียบกับพบว่าทั้ง CNN, ResNet Model ได้ให้พบการทำนายเป็นด้านหน้ารถ ส่วน CNN Model ได้ทำนายเป็นด้านข้าง และตำแหน่งความเสียหายที่ได้ระบุในใบเคลมได้มีการระบุเป็นตำแหน่งด้านหลังรถ ซึ่งมีผลไม่ตรงกันกับทั้ง 3 Model ซึ่งเมื่ออ้างอิงจากผลวิเคราะห์ในข้อ 4.1 จะเห็นได้ทั้ง 3 Model ยังทำนายในส่วนด้านหลังรถยนต์ไม่ค่อยดีนัก

ตารางที่ 4.23 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ที่เป็นส่วนด้านหน้าของรถยนต์ โดยใช้ Method R CNN และ R CNN DC5

<table>
<tr><td colspan="4">รูปภาพตัวรถยนต์</td></tr>
<tr><td colspan="2">R CNN</td><td colspan="2">R CNN DC5</td></tr>
<tr><td colspan="2">  </td><td colspan="2">  </td></tr>
<tr><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>3 จุด</td><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>3 จุด</td></tr>
<tr><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีฟ้า) = 98%
ตำแหน่งจุดที่ 2 (สีฟ้า) = 76%
ตำแหน่งจุดที่ 3 (สีม่วง) = 72%</td><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีเขียว) = 72%
ตำแหน่งจุดที่ 2 (สีเขียว) = 89%
ตำแหน่งจุดที่ 3 (สีแดง) = 84%</td></tr>
</table>

ตารางที่ 4.24 ข้อมูลที่ถูกบันทึกในใบเคลม

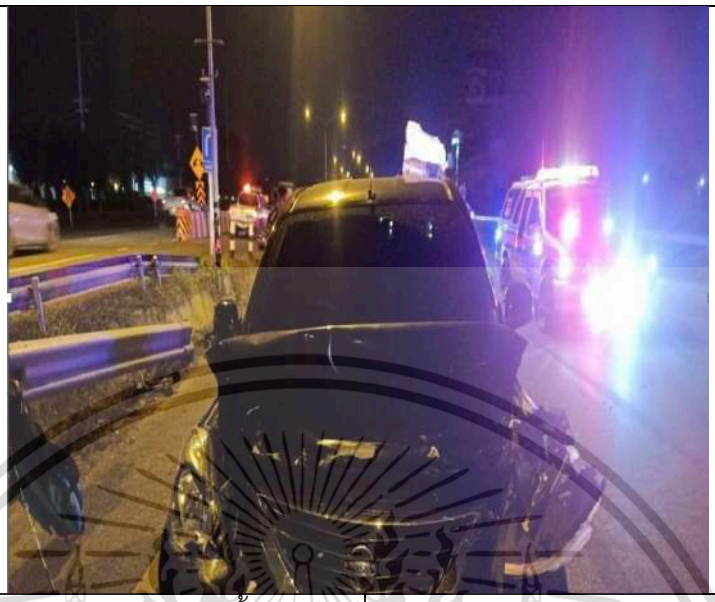
รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันชนกับด้านหน้ารถคู่กรณี

จากตารางที่ 4.23 – 4.24 เมื่อนำผลมาเทียบกับพบว่า R CNN Model สามารถทำการ Detection รูปได้ครอบคลุมกว่า R CNN DC5 Model และในส่วนที่มีการ Detection ได้ของทั้ง 2 Model นั้นเป็นในส่วนของบริเวณหน้ารถยนต์ ซึ่งตรงกับผลบันทึกในใบเคลม

ตารางที่ 4.25 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ที่เป็นส่วนด้านหน้าของรถยนต์ โดยใช้ Method R CNN และ R CNN DC5

<table>
<tr><td colspan="4">รูปภาพตัวรถยนต์</td></tr>
<tr><td colspan="2">R CNN</td><td colspan="2">R CNN DC5</td></tr>
<tr><td colspan="2">  </td><td colspan="2">  </td></tr>
<tr><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>2 จุด</td><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>2 จุด</td></tr>
<tr><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีน้ำเงิน) = 75%
ตำแหน่งจุดที่ 2 (สีม่วง) = 99%</td><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีแดง) = 88%
ตำแหน่งจุดที่ 2 (สีแดงเข้ม) = 97%</td></tr>
</table>

ตารางที่ 4.26 ข้อมูลที่ถูกบันทึกในใบเคลม

รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันชนกับด้านหน้ารถคู่กรณี

จากตารางที่ 4.25 – 4.26 เมื่อนำผลมาเทียบกับพบว่า R CNN Model และ R CNN DC5 Model สามารถทำการ Detection ได้จุดผลลัพธ์เท่ากัน โดยที่ทั้ง 2 Model นั้นสามารถจับความเสียหายที่ไฟหน้ารถได้ทั้งคู่ ส่วนอีกจุดเป็นคนละจุดกัน และในส่วนที่มีการ Detection ได้ของทั้ง 2 Model นั้นเป็นในส่วนของบริเวณหน้ารถยนต์ ซึ่งตรงกับผลบันทึกในใบเคลม

ตารางที่ 4.27 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ที่เป็นส่วนด้านหน้าของรถยนต์ โดยใช้ Method R CNN และ R CNN DC5

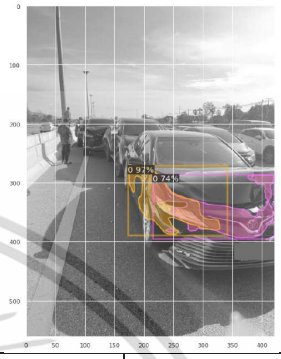
<table>
<tr><td colspan="4">รูปภาพตัวรถยนต์</td></tr>
<tr><td colspan="2">R CNN</td><td colspan="2">R CNN DC5</td></tr>
<tr><td colspan="2">  </td><td colspan="2">  </td></tr>
<tr><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>2 จุด</td><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>5 จุด</td></tr>
<tr><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีแดง) = 76%
ตำแหน่งจุดที่ 2 (สีเขียว) = 78%</td><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีแดง) = 95%
ตำแหน่งจุดที่ 2 (สีเขียวอ่อน) = 84%
ตำแหน่งจุดที่ 3 (สีขาว) = 72%
ตำแหน่งจุดที่ 4 (สีม่วง) = 97%
ตำแหน่งจุดที่ 5 (สีเขียว) = 77%</td></tr>
</table>

ตารางที่ 4.20 ข้อมูลที่ถูกบันทึกในใบเคลม

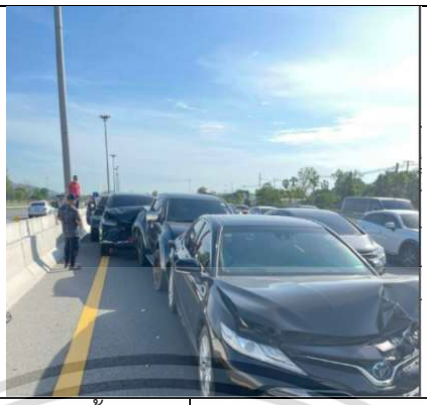
รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า กระโปรงท้าย
ข้อมูลจริงจากบริษัทประกันภัย	รถประกันถูกคู่กรณีชนด้านท้ายรถ

จากตารางที่ 4.27 – 4.28 เมื่อนำผลมาเทียบกับพบว่า R CNN Model สามารถทำการ Detection จุดได้น้อยกว่า R CNN DC5 Model และเมื่อดูจุดที่มีการ Detection พบว่า R CNN DC5 Model ได้จับจุดที่เกิดความเสียหายได้ดีกว่า ซึ่งผลการทำนายนั้นสอดคล้องกับใบเคลมที่ระบุความเสียที่เกิดทั้งหน้ารถและหลังรถ

ตารางที่ 4.29 ผลลัพธ์ข้อมูล test มา prediction และทำการเปรียบเทียบกับค่า accuracy ที่ได้ตามความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ที่เป็นส่วนด้านหน้าของรถยนต์ โดยใช้ Method R CNN และ R CNN DC5

<table>
<tr><td colspan="4">รูปภาพตัวรถยนต์</td></tr>
<tr><td colspan="2">R CNN</td><td colspan="2">R CNN DC5</td></tr>
<tr><td colspan="2">  </td><td colspan="2">  </td></tr>
<tr><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>7 จุด</td><td>ผลลัพธ์ตำแหน่งจุดเสียหายที่ทำนายได้</td><td>2 จุด</td></tr>
<tr><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีม่วง) = 67%
ตำแหน่งจุดที่ 2 (สีชมพู) = 62%
ตำแหน่งจุดที่ 3 (สีเหลือง) = 51%
ตำแหน่งจุดที่ 4 (สีส้ม) = 60%
ตำแหน่งจุดที่ 5 (สีเขียว) = 61%
ตำแหน่งจุดที่ 6 (สีชมพู) = 58%
ตำแหน่งจุดที่ 7 (สีชมพู) = 57%</td><td>Accuracy</td><td>ตำแหน่งจุดที่ 1 (สีชมพู) = 97%
ตำแหน่งจุดที่ 2 (สีส้ม) = 44%</td></tr>
</table>

ตารางที่ 4.30 ข้อมูลที่ถูกบันทึกในใบเคลม

รูปภาพตัวรถยนต์	
	
ตำแหน่งความเสียหายจากการเกิดเหตุขึ้น	กระโปรงหน้า กระโปรงท้าย
ข้อมูลจริงจากบริษัทประกันภัย	จุดเกิดเหตุที่ชนท้ายรถต่อๆกัน

จากตารางที่ 4.29 – 4.30 เมื่อนำผลมาเทียบกับพบว่า R CNN Model สามารถทำการ Detection จุดได้มากกว่า R CNN DC5 Model และเมื่อดูจุดที่มีการ Detection พบว่า R CNN Model ได้จับจุดที่เกิดความเสียหายได้ดีกว่า ซึ่งผลการทำนายนั้นสอดคล้องกับใบเคลมที่ระบุความเสียที่เกิดทั้งหน้ารถและหลังรถ

บทที่ 5
สรุปผลการศึกษาและข้อเสนอแนะ

จากการศึกษาครั้งนี้เป็นการศึกษาการประมวลผลภาพอุบัติเหตุรถยนต์โดยใช้กระบวนการโครงข่ายประสาทแบบคอนโวลูชันมีวัตถุประสงค์สร้างอัลกอริทึมในการจำแนกรถ การจำแนกว่ารถยนต์ได้รับความเสียหายหรือไม่ การจำแนกตำแหน่งความเสียที่เกิดขึ้น และเพื่อยืนยันตำแหน่งความเสียหายของรถยนต์กับใบเคลม

ซึ่งได้นำข้อมูลจากบริษัทประกันวินาศภัย เมื่อนำมาพิจารณาจะได้ผลลัพธ์ข้อมูลตามขั้นตอนดังนี้

1. ผลการแบ่งกลุ่มตัวรถยนต์ว่าเป็นส่วนของรถยนต์ ได้แก่ หน้ารถ ด้านข้างรถ และหลังรถ โดยใช้ Method CNN ,Method VGG16 และ ResNet มี optimizer เป็น adam,
2. ผลการวิเคราะห์ความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์ โดยใช้ Method R CNN และ Method R CNN DC5
3. เพื่อยืนยันตำแหน่งความเสียหายของรถยนต์กับใบเคลม

5.1 สรุปผลการวิจัย

5.1.1 การแบ่งกลุ่มตัวรถยนต์ว่าเป็นส่วนใดของรถยนต์

ในการจำแนกตำแหน่งของรถ พบว่าในการจำแนกรูปตำแหน่งของรถ Method CNN มี สามารถทำนายตำแหน่งรถได้เป็นอย่างดี โดยมีค่า Precision ของ Side, Rear และ Front มีค่า 0.50, 0.33 และ 0.67 ตามลำดับ ซึ่งทำให้เห็นได้ว่า CNN Model มีความแม่นยำในการทำนาย Class Side และ Front มากกว่า Rear และค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 81.72 และผลวิเคราะห์ข้อมูลพบว่าจากการนำรายงานจากผู้ปฏิบัติหน้าที่มาเปรียบเทียบกับผลวิเคราะห์จาก Model พบว่าผลการทำนายของ Model ถูกต้องตามรายงาน ร้อยละ 66.67

การจำแนกรูปตำแหน่งของรถ Method VGG16 สามารถทำนายตำแหน่งรถได้ดี โดยค่า Precision ของ Side, Rear และ Front มีค่า 1.00, 0.00 และ 0.38ตามลำดับ ซึ่งทำให้เห็นได้ว่า VGG16 Model มีความแม่นยำในการทำนาย Class Side มากกว่า Front และมากกว่า Rear ซึ่งค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 59.29 และผลวิเคราะห์ข้อมูลพบว่าจากการนำรายงานจากผู้ปฏิบัติหน้าที่มาเปรียบเทียบกับผลวิเคราะห์จาก Model พบว่าผลการทำนายของ Model ถูกต้องตามรายงาน ร้อยละ 66.67

การจำแนกรูปตำแหน่งของรถ Method ResNet สามารถทำนายตำแหน่งรถไม่ดีนัก โดยค่า Precision ของ Side, Rear และ Front มีค่า 0.00, 0.20 และ 0.00 ตามลำดับ ซึ่งทำให้เห็นได้ว่า VGG16 Model มีความแม่นยำในการทำนาย Class Rear มากกว่า Front และ Side ซึ่งค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 62.31 และผลวิเคราะห์ข้อมูลพบว่าจากการนำ

รายงานจากผู้ปฏิบัติหน้าที่มาเปรียบเทียบกับผลวิเคราะห์จาก Model พบว่าผลการทำนายของ Model ถูกต้องตามรายงาน ร้อยละ 83.3

เมื่อนำมาเปรียบเทียบผลของทั้ง 3 Model พบว่า CNN Model นั้นมีการทำนายตำแหน่งรถยนต์ในแต่ละ Class ได้มีความถูกต้องมากกว่า และลำดับถัดมาเป็น VGG 16 Model ซึ่งทำนายแม่นยำในส่วนของSide class มากกว่า Class อื่นๆ และสุดท้ายคือ ResNet Model ที่มีการเกิด Overfitting จึงทำให้ผลทำนายเป็น Rear class ทั้งหมด

5.1.2 การวิเคราะห์ความเสียหายของรูปถ่ายอุบัติเหตุรถยนต์

ในการตรวจจับความเสียหายของรถจาก Method R CNN พบว่ามีความแม่นยำในการตรวจจับความเสียหายได้เป็นอย่างดี โดยค่า Precision ของ Undamage, Damage มีค่า 0.45 , 1.00 ตามลำดับ ซึ่งทำให้เห็นว่า R CNN Model มีความแม่นยำในการทำนาย Class Damage มากกว่า Undamage และค่า Recall ของ Undamage เท่ากับ 1 อาจเนื่องมาจากข้อมูลที่ใช้ Test น้อยเกินไป และค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 95.67 จากผลการใช้ข้อมูลของทางผู้วิจัย และ ยังพบอีกว่าในกรณีที่เป็นข้อมูลรูปที่มีสีดำหรือมีการถ่ายความเสียหายจากระยะไกล Model ยังคงไม่สามารถตรวจจับได้

การตรวจจับความเสียหายของรถจาก Method R CNN DC5 พบว่ามีความแม่นยำในการตรวจจับความเสียหาย โดยค่า Precision ของ Undamage, Damage มีค่า 0.45 , 1.00 ตามลำดับ ซึ่งทำให้เห็นว่า R CNN DC5 Model มีความแม่นยำในการทำนาย Class Damage มากกว่า Undamage และค่าเฉลี่ยของ Accuracy อยู่ที่ร้อยละ 85.25

เมื่อนำผลวิเคราะห์ของทั้ง 2 Model มาเปรียบเทียบกัน พบว่าการทำนายความเสียหายรถยนต์ทั้ง 2 Model ทำนายผลได้ใกล้เคียงกัน แต่ในส่วนการทำนายรูปไม่มีความเสียหาย R CNN Model สามารถทำนายได้อย่างถูกมากกว่า R CNN DC5 Model

5.2 ข้อเสนอแนะ

1. รูปภาพที่จะใช้ในการเข้า model นั้นควรเป็นรูปภาพมุมแคบ สามารถเห็นจุดเสียหายที่จัดเจนและรอบจุดตัวรถยนต์ เพื่อที่จะให้ model สามารถบอกจุดความเสียหายและจุดเสียหายได้อย่างแม่นยำขึ้น

2. ความสว่างของรูปภาพเนื่องจากตัว model จะไม่สามารถเรียนรู้ภาพที่มีสีดำได้ดีนัก ทำให้ตัว model จะต้องทำงานปรับประสิทธิภาพเพิ่มเติมในการเรียนรู้ภาพสีดำให้มากขึ้น

3. สำหรับ Model ที่ใช้ทำนายตำแหน่งของรถยนต์ ผู้วิจัยเสนอว่าควรที่จะต้องทำการ detection รูปภาพเพิ่มเติมว่ากำลังทำนายในส่วนใดของรถยนต์

4. เสนอแนะให้ทำ Method อื่นๆเพิ่มเติมของ Model ของการทำนายตำแหน่งรถยนต์ เช่น ResNet101, ResNet152

5. เสนอแนะให้ทำ Method อื่นๆเพิ่มเติมของ Model ของการทำนายความเสียหายของรถยนต์ เช่น VGG16, ResNet50, ResNet101, ResNet152

เอกสารอ้างอิง

ศูนย์ข้อมูลอุบัติเหตุเพื่อเสริมสร้างวัฒนธรรมความปลอดภัยทางถนน. 2564. **รายงานสถิติผู้ประสบอุบัติเหตุรถทั่วประเทศ**. [Online]. Available : https://www.thairsc.com/.

Ratchakitcha.soc.go.th. 2565. **พระราชบัญญัติ จราจรทางบก (ฉบับที่ ๑๓) พ.ศ. ๒๕๖๕**. [Online]. Available : https://www.ratchakitcha.soc.go.th/DATA/PDF/2565/A/028/T_0005.PDF

คณะกรรมการพัฒนาธุรกิจและวิชาการประกันภัยสมาคมประกันวินาศภัยไทย. 2564. **คู่มือประกันวินาศภัยไทย.** กรุงเทพฯ : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย

Medium. 2018. **Convolutional Neural Network**. [Online]. Available : https://medium.com/.

Medium. 2020. **Convolutional Auto Encoders**. [Online]. Available : https://beeying.medium.com/.

สุริยะ ชยะธรรมกูล. 2563. “การจำแนกผลึกน้ำตาลการเรียนรู้เชิงลึก.” สารนิพนธ์, สาขาวิชาวิศวกรรมซอฟต์แวร์, ภาควิชาวิศวกรรมคอมพิวเตอร์, คณะวิศวกรรมศาสตร์, จุฬาลงกรณ์มหาวิทยาลัย.

Medium. 2019. **ย้อนรอย Object Detection และเจาะลึก RetinaNet**. [Online]. Available : https://medium.com/.

Medium. 2018. **Object Detection (Part 2)**. [Online]. Available : https://medium.com/.

BUA Labs. 2020. **Transfer Learning**. [Online]. Available : https://www.bualabs.com/.

ICHI.PRO. 2019. **การจำแนกภาพด้วย ResNet50 Convolution Neural Network (CNN) บนการถ่ายภาพรังสี Covid-19**. [Online]. Available : https://ichi.pro/th/.

R.Rajkumar. Ronhit Neema. N.Chaitanyanathreddy. G.Varun. K.Govinda. 2020. “Automobile Insurance Processing using Deep Convolutional Networks.” School of Computer Science and Engineering, Vellore Institute of Techcnology, Vellore. India.

Kalpesh Patil. Mandar Kulkarni. Anand Sriraman. Shirish Karande. 2017. “Deep Learning Based Car Damage Classification.” TCS Innovation Labs, Pune, India.

Javier O. Pinzón Arenas. Robinson Jiménez. Paula C. 2018. “Faster R-CNN for object location in a Virtual Environment for sorting task.” Useche Murillo Mechatronics Engineering Program, Faculty of Engineering, Nueva Granada Military University, Bogotá, Colombia.

ภาคผนวก

ภาคผนวก ก

1.Car damage model

RCNN

```
!pip3 install 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

%matplotlib inline
from pycocotools.coco import COCO
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import pylab
import random
pylab.rcParams['figure.figsize'] = (8.0, 10.0)# Import Libraries

# For visualization
import os
import seaborn as sns
from matplotlib import colors
from tensorboard.backend.event_processing import event_accumulator as ea
from PIL import Image
from google.colab import drive

drive.mount('/content/gdrive')

getDataPath = '/content/gdrive/MyDrive/Project_Ayok/dataset/car_damage'

dataDir=getDataPath+'/val'
dataType='COCO_val_annos'
mul_dataType='COCO_mul_val_annos'
annFile='{}/{}.json'.format(dataDir,dataType)
mul_annFile='{}/{}.json'.format(dataDir,mul_dataType)
```

```
img_dir = getDataPath+"/img"

coco=COCO(annFile)
mul_coco=COCO(mul_annFile)

cats = coco.loadCats(coco.getCatIds())
nms=[cat['name'] for cat in cats]
print('COCO categories for damages: \n{}\n'.format(', '.join(nms)))

nms = set([cat['supercategory'] for cat in cats])
print('COCO supercategories for damages: \n{}\n'.format(', '.join(nms)))

#Multi Class #Parts dataset

mul_cats = mul_coco.loadCats(mul_coco.getCatIds())
mul_nms=[cat['name'] for cat in mul_cats]
print('COCO categories for parts: \n{}\n'.format(', '.join(mul_nms)))

mul_nms = set([mul_cat['supercategory'] for mul_cat in mul_cats])
print('COCO supercategories for parts: \n{}\n'.format(', '.join(mul_nms)))

catIds = coco.getCatIds(catNms=['damage']);
imgIds = coco.getImgIds(catIds=catIds );
random_img_id = random.choice(imgIds)
print("{} image id was selected at random from the {} list".format(random_img_id,
imgIds))

imgId = coco.getImgIds(imgIds = [random_img_id])
img = coco.loadImgs(imgId)[0]
print("Image details \n",img)

I = io.imread(img_dir + '/25'+'.jpg')
plt.axis('off')
```

```
plt.imshow(I)
plt.show()

annIds = coco.getAnnIds(imgIds=imgId,iscrowd=None)
anns = coco.loadAnns(annIds)

plt.imshow(I)
plt.axis('on')
coco.showAnns(anns, draw_bbox=True )

mul_annIds = mul_coco.getAnnIds(imgIds=imgId,iscrowd=None)
mul_anns = mul_coco.loadAnns(mul_annIds)

category_map = dict()

for ele in list(mul_coco.cats.values()):
    category_map.update({ele['id']:ele['name']})

parts = []
for region in mul_anns:
    parts.append(category_map[region['category_id']])

print("Parts are:", parts)

#Plot Parts
I = io.imread(img_dir + '/' + img['file_name'])
plt.imshow(I)
plt.axis('on')
mul_coco.showAnns(mul_anns, draw_bbox=True )

#get parts annotations
mul_annIds = mul_coco.getAnnIds(imgIds=imgId,iscrowd=None)
mul_anns = mul_coco.loadAnns(mul_annIds)
```

```
# Create a dictionary between category_id and category name
category_map = dict()

for ele in list(mul_coco.cats.values()):
    category_map.update({ele['id']:ele['name']})

#Create a list of parts in the image
parts = []
for region in mul_anns:
    parts.append(category_map[region['category_id']])

print("Parts are:", parts)

#Plot Parts
I = io.imread(img_dir + '/' + img['file_name'])
plt.imshow(I)
plt.axis('on')
mul_coco.showAnns(mul_anns, draw_bbox=True )

!python3 -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
!git clone https://github.com/facebookresearch/detectron2
!python3 /content/detectron2/setup.py install

!python3  -m pip install detectron2 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu92/torch1.7/index.html
!pip3 install torch==1.7.0 torchvision -f
https://download.pytorch.org/whl/torch_stable.html

import torch, torchvision
print(torch.__version__, torch.cuda.is_available())

!python3 --version
```

```
!nvidia-smi

# import some common libraries
import numpy as np
import os, json, cv2, random
import matplotlib.pyplot as plt
import skimage.io as io
# /content/detectron2
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# # import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.engine import DefaultTrainer
from detectron2.utils.visualizer import ColorMode
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader

# Set base params
plt.rcParams["figure.figsize"] = [16,9]
print(detectron2.__version__)

!python3 -m detectron2.utils.collect_env

dataset_dir = "/content/gdrive/MyDrive/Project_Ayok/dataset/car_damage"
img_dir = "img/"
train_dir = "train/"
val_dir = "val/"
```

```
from detectron2.data.datasets import register_coco_instances
register_coco_instances("car_dataset_train", {},
os.path.join(dataset_dir,train_dir,"COCO_train_annos.json"),
os.path.join(dataset_dir,img_dir))
register_coco_instances("car_dataset_val", {},
os.path.join(dataset_dir,val_dir,"COCO_val_annos.json"),
os.path.join(dataset_dir,img_dir))

dataset_dicts = DatasetCatalog.get("car_dataset_train")
metadata_dicts = MetadataCatalog.get("car_dataset_train")

#Implementing my own Trainer Module here to use the COCO validation evaluation
during training
# TODO: add data custom augmentation
class CocoTrainer(DefaultTrainer):

  @classmethod
  def build_evaluator(cls, cfg, dataset_name, output_folder=None):

    if output_folder is None:
        os.makedirs("coco_eval", exist_ok=True)
        output_folder = "coco_eval"

    return COCOEvaluator(dataset_name, cfg, False, output_folder)
# In detectron2, epoch is MAX_ITER * BATCH_SIZE / TOTAL_NUM_IMAGES

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-
InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("car_dataset_train",)
cfg.DATASETS.TEST = ("car_dataset_val",)
```

```
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let training initialize from
model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001  # pick a good LR
cfg.SOLVER.WARMUP_ITERS = 700
cfg.SOLVER.MAX_ITER = 400 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (600, 800)
cfg.SOLVER.GAMMA = 0.05
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128   # faster, and good enough for
this  dataset (default: 512)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2  # only has one class (damage) + 1
cfg.MODEL.RETINANET.NUM_CLASSES = 2 # only has one class (damage) + 1
cfg.TEST.EVAL_PERIOD = 400



# Clear any logs from previous runs
#TODO add timestamp to logs
!rm -rf cfg.OUTPUT_DIR



os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()

# plots = plot(logdir= './output', savedir= './')
for root, dirs, files in os.walk('./output'):
  print(root)
  print(dirs)
  print(files)
```

```
def smooth(scalars, weight=0.6):
    """
    Reference: https://github.com/plotly/dash-live-model-
training/blob/master/app.py#L163
    """
    last = scalars[0]
    smoothed = list()
    for point in scalars:
        smoothed_val = last * weight + (1 - weight) * point
        smoothed.append(smoothed_val)
        last = smoothed_val
    return smoothed


def plot(logdir: str, savedir: str, smoothing: float = 0.6, no_title=False, no_legend=False,
no_axis_labels=False):
    """ re-draw the tf summary events plots  using seaborn
    :param logdir: Path to the directory having event logs
    :param savedir: Path to save the seaborn graphs
    :param smoothing: smoothing window space for the plots
    """
    assert 0 <= smoothing <= 1, 'Smoothing value should be in [0,1]'


    plots = []


    sns.set(style="darkgrid")
    sns.set_context("paper")


    # Collect data
    # we recognize all files which have tfevents
    scalars_info = {}
    for root, dirs, files in os.walk(logdir):
```

```
    for event_file in [x for x in files if 'tfevents' in x]:
        event_path = os.path.join(root, event_file)

        acc = ea.EventAccumulator(event_path)
        acc.Reload()

        # only support scalar now
        scalar_list = acc.Tags()['scalars']
        for tag in scalar_list:
            x = [s.step for s in acc.Scalars(tag)]
            y = [s.value for s in acc.Scalars(tag)]
            data = {'x': x, 'y': y, 'legend': root.split(logdir)[1][1:] if root != logdir else None}
            if tag not in scalars_info:
                scalars_info[tag] = [data]
            else:
                scalars_info[tag].append(data)

# We recognize groups assuming each group name has /
# And, each group is saved in a separate directory
for tag, tag_data in scalars_info.items():
    _split = tag.split('/')
    if len(_split) <= 1:
        _path = os.path.join(savedir, 'seaborn')
        _name = _split[0]
    else:
        _path = os.path.join(savedir, 'seaborn', _split[0])
        _name = ''.join(_split[1:])

    os.makedirs(_path, exist_ok=True)

    color_list = list(sns.color_palette(palette='dark', n_colors=len(tag_data)))[::-1]
    for data in tag_data:
        x, y = data['x'], data['y']
```

```
            y_smooth = smooth(y, weight=smoothing)
            current_color = color_list.pop()
            _plt = sns.lineplot(x = x, y = y, color=colors.to_rgba(current_color, alpha=0.4))
            _legend = data['legend'] if not no_legend else None
            # _plt = sns.lineplot(x, y_smooth, label=data['legend'], color=current_color)

        if not no_axis_labels:
            _plt.set(xlabel='x', ylabel='y')
        if not no_title:
            _plt.set_title(_name.capitalize())

        plots.append(os.path.join(_path, _name + '.png'))
        plt.savefig(os.path.join(_path, _name + '.png'))
        plt.clf()
    return plots

plots = plot(logdir= './output', savedir= './')

my_dpi = 1000
fig, ax = plt.subplots(2,1, figsize = (12,10), dpi=my_dpi)


ax[0].set_title('Total Loss', fontsize=12)
ax[0].set_xticks([])
ax[0].set_yticks([])
ax[0].imshow(Image.open('./seaborn/total_loss.png'))

ax[1].set_title('Class accuracy', fontsize=12)
ax[1].set_xticks([])
ax[1].set_yticks([])
ax[1].imshow(Image.open('./seaborn/fast_rcnn/cls_accuracy.png'))

evaluator = COCOEvaluator("car_dataset_val", cfg, False, output_dir="./output/")
```

```
val_loader = build_detection_test_loader(cfg, "car_dataset_val")
print(inference_on_dataset(trainer.model, val_loader, evaluator))

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
for this model
cfg.DATASETS.TEST = ("car_dataset_val", )
predictor = DefaultPredictor(cfg)
print(cfg)

from sklearn.metrics import accuracy_score, confusion_matrix
tdata = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash/'
def preprocess_imgs(path, img_size):
    set_new = []
    for value in os.listdir(path):
        for img in os.listdir(path + value):
            print(img)
            img = cv2.imread(path + value + "/" + img)
            outputs = predictor(img)
            # If null is undamge
            # If not null is damage
            if len(outputs['instances'].pred_classes) != 0 :
                set_new.append(1)
            else :
                set_new.append(0)
    # return np.array(set_new)
    return set_new

test_data = preprocess_imgs(tdata, img_size=(150,150))

reality = []
for value in os.listdir(tdata):
    for img in os.listdir(tdata + value):
```

```
        # 1 = damage 0 = undamage
        reality.append(1) if value.lower() == "damage" else reality.append(0)

accuracy = accuracy_score(reality, test_data)
print("Test Accuracy:", accuracy)
print(test_data)
print(reality)

import seaborn as sns
from sklearn.metrics import classification_report

confusion_mtx = confusion_matrix(reality, test_data)
print(classification_report(reality, test_data))

ax = plt.axes()
sns.heatmap(confusion_mtx, annot=True,annot_kws={"size": 25}, cmap="Blues", ax =
ax)
ax.set_xlabel("Predicted Diagnosis", fontsize=14, labelpad=20)
ax.xaxis.set_ticklabels(['Unamage', 'Damage'])

# set y-axis label and ticks
ax.set_ylabel("Actual Diagnosis", fontsize=14, labelpad=20)
ax.yaxis.set_ticklabels(['Unamage', 'Damage'])
ax.set_title('Confusion matrix', size=14)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

fig, ax = plt.subplots(4, 4, figsize =(20,16))
indices=[ax[0][0],ax[1][0],ax[2][0],ax[3][0],
         ax[0][1],ax[1][1],ax[2][1],ax[3][1],
         ax[0][2],ax[1][2],ax[2][2],ax[3][2],
         ax[0][3],ax[1][3],ax[2][3],ax[3][3] ]
```

```
i=-1
test_real_data = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash/damage/'
for d in os.listdir(test_real_data):
    i=i+1
     im = cv2.imread(os.path.join(test_real_data,d))
     outputs = predictor(im)
     v = Visualizer(im[:, :, ::-1],

                    scale=0.5,
                    instance_mode=ColorMode.IMAGE_BW   # remove the colors of
unsegmented pixels. This option is only available for segmentation models
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    indices[i].grid(False)
    indices[i].imshow(out.get_image()[:, :, ::-1])
```

RCNN DC5

```
!pip3 install 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

%matplotlib inline
from pycocotools.coco import COCO
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import pylab
import random
pylab.rcParams['figure.figsize'] = (8.0, 10.0)# Import Libraries

# For visualization
import os
```

```
import seaborn as sns
from matplotlib import colors
from tensorboard.backend.event_processing import event_accumulator as ea
from PIL import Image
from google.colab import drive

drive.mount('/content/gdrive')

getDataPath = '/content/gdrive/MyDrive/Project_Ayok/dataset/car_damage'

dataDir=getDataPath+'/val'
dataType='COCO_val_annos'
mul_dataType='COCO_mul_val_annos'
annFile='{}/{}.json'.format(dataDir,dataType)
mul_annFile='{}/{}.json'.format(dataDir,mul_dataType)
img_dir = getDataPath+"/img"

coco=COCO(annFile)
mul_coco=COCO(mul_annFile)

cats = coco.loadCats(coco.getCatIds())
nms=[cat['name'] for cat in cats]
print('COCO categories for damages: \n{}\n'.format(', '.join(nms)))

nms = set([cat['supercategory'] for cat in cats])
print('COCO supercategories for damages: \n{}\n'.format(', '.join(nms)))

#Multi Class #Parts dataset

mul_cats = mul_coco.loadCats(mul_coco.getCatIds())
mul_nms=[cat['name'] for cat in mul_cats]
print('COCO categories for parts: \n{}\n'.format(', '.join(mul_nms)))
```

```
mul_nms = set([mul_cat['supercategory'] for mul_cat in mul_cats])
print('COCO supercategories for parts: \n{}\n'.format(', '.join(mul_nms)))

catIds = coco.getCatIds(catNms=['damage']);
imgIds = coco.getImgIds(catIds=catIds );
random_img_id = random.choice(imgIds)
print("{} image id was selected at random from the {} list".format(random_img_id,
imgIds))

imgId = coco.getImgIds(imgIds = [random_img_id])
img = coco.loadImgs(imgId)[0]
print("Image details \n",img)

I = io.imread(img_dir + '/25'+'.jpg')
plt.axis('off')
plt.imshow(I)
plt.show()

annIds = coco.getAnnIds(imgIds=imgId,iscrowd=None)
anns = coco.loadAnns(annIds)

plt.imshow(I)
plt.axis('on')
coco.showAnns(anns, draw_bbox=True )

mul_annIds = mul_coco.getAnnIds(imgIds=imgId,iscrowd=None)
mul_anns = mul_coco.loadAnns(mul_annIds)

category_map = dict()

for ele in list(mul_coco.cats.values()):
    category_map.update({ele['id']:ele['name']})
```

```
parts = []
for region in mul_anns:
    parts.append(category_map[region['category_id']])

print("Parts are:", parts)

#Plot Parts
I = io.imread(img_dir + '/' + img['file_name'])
plt.imshow(I)
plt.axis('on')
mul_coco.showAnns(mul_anns, draw_bbox=True )

#get parts annotations
mul_annIds = mul_coco.getAnnIds(imgIds=imgId,iscrowd=None)
mul_anns = mul_coco.loadAnns(mul_annIds)

# Create a dictionary between category_id and category name
category_map = dict()

for ele in list(mul_coco.cats.values()):
    category_map.update({ele['id']:ele['name']})

#Create a list of parts in the image
parts = []
for region in mul_anns:
    parts.append(category_map[region['category_id']])

print("Parts are:", parts)

#Plot Parts
I = io.imread(img_dir + '/' + img['file_name'])
plt.imshow(I)
plt.axis('on')
```

```
mul_coco.showAnns(mul_anns, draw_bbox=True )

!python3 -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
!git clone https://github.com/facebookresearch/detectron2
!python3 /content/detectron2/setup.py install

!python3  -m pip install detectron2 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu92/torch1.7/index.html
!pip3 install torch==1.7.0 torchvision -f
https://download.pytorch.org/whl/torch_stable.html

import torch, torchvision
print(torch.__version__, torch.cuda.is_available())

!python3 --version
!nvidia-smi

# import some common libraries
import numpy as np
import os, json, cv2, random
import matplotlib.pyplot as plt
import skimage.io as io
# /content/detectron2
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# # import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
```

```
from detectron2.engine import DefaultTrainer
from detectron2.utils.visualizer import ColorMode
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader

# Set base params
plt.rcParams["figure.figsize"] = [16,9]
print(detectron2.__version__)

!python3 -m detectron2.utils.collect_env

dataset_dir = "/content/gdrive/MyDrive/Project_Ayok/dataset/car_damage"
img_dir = "img/"
train_dir = "train/"
val_dir = "val/"


from detectron2.data.datasets import register_coco_instances
register_coco_instances("car_dataset_train", {},
os.path.join(dataset_dir,train_dir,"COCO_train_annos.json"),
os.path.join(dataset_dir,img_dir))
register_coco_instances("car_dataset_val", {},
os.path.join(dataset_dir,val_dir,"COCO_val_annos.json"),
os.path.join(dataset_dir,img_dir))

dataset_dicts = DatasetCatalog.get("car_dataset_train")
metadata_dicts = MetadataCatalog.get("car_dataset_train")

#Implementing my own Trainer Module here to use the COCO validation evaluation
during training
# TODO: add data custom augmentation
class CocoTrainer(DefaultTrainer):
```

```
  @classmethod
  def build_evaluator(cls, cfg, dataset_name, output_folder=None):

    if output_folder is None:
        os.makedirs("coco_eval", exist_ok=True)
        output_folder = "coco_eval"

    return COCOEvaluator(dataset_name, cfg, False, output_folder)
# In detectron2, epoch is MAX_ITER * BATCH_SIZE / TOTAL_NUM_IMAGES

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-
InstanceSegmentation/mask_rcnn_R_101_DC5_3x.yaml"))
cfg.DATASETS.TRAIN = ("car_dataset_train",)
cfg.DATASETS.TEST = ("car_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
InstanceSegmentation/mask_rcnn_R_101_DC5_3x.yaml")  # Let training initialize from
model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001  # pick a good LR
cfg.SOLVER.WARMUP_ITERS = 700
cfg.SOLVER.MAX_ITER = 400 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (600, 800)
cfg.SOLVER.GAMMA = 0.05
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128   # faster, and good enough for
this  dataset (default: 512)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # only has one class (damage) + 1
cfg.MODEL.RETINANET.NUM_CLASSES = 1 # only has one class (damage) + 1
cfg.TEST.EVAL_PERIOD = 600
```

```
# Clear any logs from previous runs
#TODO add timestamp to logs
# !rm -rf cfg.OUTPUT_DIR


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()



# plots = plot(logdir= './output', savedir= './')
for root, dirs, files in os.walk('./output'):
  print(root)
  print(dirs)
  print(files)

def smooth(scalars, weight=0.6):
    """
    Reference: https://github.com/plotly/dash-live-model-
training/blob/master/app.py#L163
    """
    last = scalars[0]
    smoothed = list()
    for point in scalars:
        smoothed_val = last * weight + (1 - weight) * point
        smoothed.append(smoothed_val)
        last = smoothed_val
    return smoothed


def plot(logdir: str, savedir: str, smoothing: float = 0.6, no_title=False, no_legend=False,
no_axis_labels=False):
```

```
""" re-draw the tf summary events plots  using seaborn
:param logdir: Path to the directory having event logs
:param savedir: Path to save the seaborn graphs
:param smoothing: smoothing window space for the plots
"""
assert 0 <= smoothing <= 1, 'Smoothing value should be in [0,1]'

plots = []

sns.set(style="darkgrid")
sns.set_context("paper")

# Collect data
# we recognize all files which have tfevents
scalars_info = {}
for root, dirs, files in os.walk(logdir):
    for event_file in [x for x in files if 'tfevents' in x]:
        event_path = os.path.join(root, event_file)

        acc = ea.EventAccumulator(event_path)
        acc.Reload()

        # only support scalar now
        scalar_list = acc.Tags()['scalars']
        for tag in scalar_list:
            x = [s.step for s in acc.Scalars(tag)]
            y = [s.value for s in acc.Scalars(tag)]
            data = {'x': x, 'y': y, 'legend': root.split(logdir)[1][1:] if root != logdir else None}
            if tag not in scalars_info:
                scalars_info[tag] = [data]
            else:
                scalars_info[tag].append(data)
```

```
    # We recognize groups assuming each group name has /
    # And, each group is saved in a separate directory
    for tag, tag_data in scalars_info.items():
        _split = tag.split('/')
        if len(_split) <= 1:
            _path = os.path.join(savedir, 'seaborn')
            _name = _split[0]
        else:
            _path = os.path.join(savedir, 'seaborn', _split[0])
            _name = ''.join(_split[1:])

        os.makedirs(_path, exist_ok=True)

        color_list = list(sns.color_palette(palette='dark', n_colors=len(tag_data)))[::-1]
        for data in tag_data:
            x, y = data['x'], data['y']
            y_smooth = smooth(y, weight=smoothing)
            current_color = color_list.pop()
            _plt = sns.lineplot(x = x, y = y, color=colors.to_rgba(current_color, alpha=0.4))
            _legend = data['legend'] if not no_legend else None
            # _plt = sns.lineplot(x, y_smooth, label=data['legend'], color=current_color)

        if not no_axis_labels:
            _plt.set(xlabel='x', ylabel='y')
        if not no_title:
            _plt.set_title(_name.capitalize())

        plots.append(os.path.join(_path, _name + '.png'))
        plt.savefig(os.path.join(_path, _name + '.png'))
        plt.clf()
    return plots

plots = plot(logdir= './output', savedir= './')
```

```
my_dpi = 1000
fig, ax = plt.subplots(2,1, figsize = (12,10), dpi=my_dpi)


ax[0].set_title('Total Loss', fontsize=12)
ax[0].set_xticks([])
ax[0].set_yticks([])
ax[0].imshow(Image.open('./seaborn/total_loss.png'))

ax[1].set_title('Class accuracy', fontsize=12)
ax[1].set_xticks([])
ax[1].set_yticks([])
ax[1].imshow(Image.open('./seaborn/fast_rcnn/cls_accuracy.png'))

evaluator = COCOEvaluator("car_dataset_val", cfg, False, output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "car_dataset_val")
print(inference_on_dataset(trainer.model, val_loader, evaluator))

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set a custom testing threshold
for this model
cfg.DATASETS.TEST = ("car_dataset_val", )
predictor = DefaultPredictor(cfg)
print(cfg)

from sklearn.metrics import accuracy_score, confusion_matrix
tdata = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash/'
def preprocess_imgs(path, img_size):
    set_new = []
    for value in os.listdir(path):
        for img in os.listdir(path + value):
          print(img)
```

```
            img = cv2.imread(path + value + "/" + img)
            outputs = predictor(img)
            # If null is undamge
            # If not null is damage
            if len(outputs['instances'].pred_classes) != 0 :
              set_new.append(1)
            else :
              set_new.append(0)
    # return np.array(set_new)
    return set_new

test_data = preprocess_imgs(tdata, img_size=(150,150))

reality = []
for value in os.listdir(tdata):
    for img in os.listdir(tdata + value):
        # 1 = damage 0 = undamage
        reality.append(1) if value.lower() == "damage" else reality.append(0)

accuracy = accuracy_score(reality, test_data)
print("Test Accuracy:", accuracy)
print(test_data)
print(reality)

import seaborn as sns
from sklearn.metrics import classification_report

confusion_mtx = confusion_matrix(reality, test_data)
print(classification_report(reality, test_data))

ax = plt.axes()
sns.heatmap(confusion_mtx, annot=True,annot_kws={"size": 25}, cmap="Blues", ax =
ax)
```

```
ax.set_xlabel("Predicted Diagnosis", fontsize=14, labelpad=20)
ax.xaxis.set_ticklabels(['Unamage', 'Damage'])

# set y-axis label and ticks
ax.set_ylabel("Actual Diagnosis", fontsize=14, labelpad=20)
ax.yaxis.set_ticklabels(['Unamage', 'Damage'])
ax.set_title('Confusion matrix', size=14)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

fig, ax = plt.subplots(4, 4, figsize =(20,16))
indices=[ax[0][0],ax[1][0],ax[2][0],ax[3][0],
         ax[0][1],ax[1][1],ax[2][1],ax[3][1],
         ax[0][2],ax[1][2],ax[2][2],ax[3][2],
         ax[0][3],ax[1][3],ax[2][3],ax[3][3] ]
i=-1
test_real_data = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash/damage/'
for d in os.listdir(test_real_data):
    i=i+1
     im = cv2.imread(os.path.join(test_real_data,d))
     outputs = predictor(im)
     v = Visualizer(im[:, :, ::-1],

                    scale=0.5,
                    instance_mode=ColorMode.IMAGE_BW
    )
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    indices[i].grid(False)
    indices[i].imshow(out.get_image()[:, :, ::-1])
```

VGG

```
!nvidia-smi

import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
import pandas as pd

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D
from google.colab import drive
from keras.preprocessing.image import ImageDataGenerator

from sklearn.model_selection import train_test_split

drive.mount('/content/gdrive')

getPath_data_train =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/train'
getPath_data_all =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/all'

all_categories = []
all_filename = []

images = os.listdir(getPath_data_train)
for index,r in enumerate(images):
 if r != 'dataset_for_classification':
    read = images[index]
```

```
    read = getPath_data_train+'/'+read
    print(read)
    print(r)
    for i,file in enumerate(os.listdir(read)):
     if r == 'front':
       all_filename.append(file)
       all_categories.append(2)
     elif(r=='rear'):
       all_filename.append(file)
       all_categories.append(1)
     else:
       all_filename.append(file)
       all_categories.append(0)
else:
     None

df =pd.DataFrame({
     'image': all_filename,
     'category':all_categories
     })

df['category'] = df['category'].replace({2:'front',1:'rear',0:'side'})

x_train,x_val = train_test_split(df ,random_state=42,shuffle=True,test_size=0.2)

x_train = x_train.reset_index(drop=True)
x_val = x_val.reset_index(drop=True)
print('Train Images shape is : ',x_train.shape)
print('Validation Images shape is : ',x_val.shape)

batch_size = 128
train_ds = ImageDataGenerator(rescale = 1.0/255,
    shear_range = 0.2,
```

```
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_train,
    getPath_data_all,
    x_col='image',
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
)



val_ds = ImageDataGenerator(rescale = 1.0/255,
  shear_range = 0.2,
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_val,
    getPath_data_all,
    x_col='image',
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
```

```
    batch_size=batch_size
)

from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import  ModelCheckpoint
from tensorflow.keras import optimizers
# from tensorflow.keras.losses import CategoricalFocalCrossentropy
from tensorflow.keras.losses import CategoricalCrossentropy



from tensorflow.keras.layers import Input

def non_trainable(model):
    for i in range(len(model.layers)):
      model.layers[i].trainable = False
    return model

def create_model(n_classes,output_activation):
    os.environ['PYTHONHASHSEED'] = '0'
    tf.keras.backend.clear_session()

    ## Set the random seed values to regenerate the model.
    np.random.seed(0)

    #Input layer
    input_layer = Input(shape=(256,256,3),name='Input_Layer')

    #Adding pretrained model
    vgg_model = tf.keras.applications.vgg16.VGG16(include_top = False,weights =
'imagenet',input_tensor = input_layer)
    vgg_model = non_trainable(vgg_model)

    #Flatten
```

```
    flatten = Flatten(data_format='channels_last',name='Flatten')(vgg_model.output)

    #FC layer
    FC1 = Dense(units=512,activation='relu',name='FC1')(flatten)

    #FC layer
    FC2 = Dense(units=256,activation='relu',name='FC2')(FC1)

    #Dropout layer
    droput1 = Dropout(0.5)(FC2)

    #output layer
    Out = Dense(units=n_classes,activation=output_activation,name='Output')(droput1)

    #Creating the Model
    model = Model(inputs=input_layer,outputs=Out)

    return model


# vgg_model.summary()
model = create_model(3,'softmax')

file_path = "vgg16/vgg16_stage2_fc-{val_accuracy:.3f}.hdf5"

checkpoint = ModelCheckpoint(filepath=file_path,
save_best_only=True,monitor='val_loss',verbose=0,mode='auto')


model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizers.Adam(learning_rate=0.0001),
    metrics=['accuracy'])
```

```
n_training_samples = len(train_ds)
n_validation_samples = len(val_ds)

history = model.fit(train_ds
                    ,epochs=20
                    ,validation_data=val_ds
                    ,validation_steps=n_validation_samples//batch_size
                    ,steps_per_epoch =n_training_samples
                    )

loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, acc, color='blue', label='Training')
plt.plot(epochs, val_acc, color='green', label='Validation')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(epochs, loss, color='orange', label='Training')
plt.plot(epochs, val_loss, color='red', label='Validation')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
test_path = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash_classification'
#get image path
test_images = os.listdir(test_path)
 #creat data frame
df_test =pd.DataFrame({
 'image': test_images,
})

test_data_gen = ImageDataGenerator( rescale = 1./255 ).flow_from_dataframe(
df_test,
test_path,
x_col='image',
y_col= None,
target_size=(256,256),
class_mode=None,
shuffle=True,
batch_size=batch_size)

predection = model.predict(test_data_gen)
predection
tf.nn.softmax(predection[0])

from google.colab.patches import cv2_imshow
import cv2

def get_Label(number):
  labels = {0:'side', 1:'rear',2:'front'}
  return labels[number]


from sklearn.metrics import confusion_matrix,classification_report
get_val_vali = [1,2,1,2,2,2,0,0,0,0]
```

```
y_pred = []
y_get_df = []
for i in os.listdir(test_path):
  pic_predict = os.path.join(test_path,i)
  img = tf.keras.utils.load_img(
  pic_predict, target_size=(256, 256)
  )
  img_array = tf.keras.utils.img_to_array(img)
  img_array = tf.expand_dims(img_array, 0)
  predictions = model.predict(img_array)
  score = tf.nn.softmax(predictions[0])
  cv2.imread(pic_predict)


  print(score)
  print("Pic's name :"+i+" This image {}".format(get_Label(int(np.argmax(score)))) + "
Accuracy is "+str(100 * np.max(score)))
  y_pred.append(int(np.argmax(score)))
  # img_show = cv2.imread(pic_predict)
  # cv2_imshow(img_show)
print("-----------------------------------Report----------------------------------------------------")
# tn,fp,fn,tp = confusion_matrix(get_val_vali,y_pred).ravel()
# print("True negative {} False positive {} False negative {} True positive
{}".format(tn,fp,fn,tp))
print(confusion_matrix(get_val_vali,y_pred))
print(classification_report(get_val_vali,y_pred))


import seaborn as sns
ax = plt.axes()
sns.heatmap(confusion_matrix(get_val_vali,y_pred), annot=True,annot_kws={"size": 25},
cmap="Blues", ax = ax)
ax.set_xlabel("Predicted Front Rear Side Confusion matrix", fontsize=14, labelpad=20)
ax.xaxis.set_ticklabels(['Front', 'Rear','Side'])
```

```
# set y-axis label and ticks
ax.set_ylabel("Actual", fontsize=14, labelpad=20)
ax.yaxis.set_ticklabels(['Front', 'Rear','Side'])
ax.set_title('Confusion matrix with VGG', size=14)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

CNN
!nvidia-smi

import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
import pandas as pd

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D
from google.colab import drive
from keras.preprocessing.image import ImageDataGenerator

from sklearn.model_selection import train_test_split

drive.mount('/content/gdrive')

getPath_data_train =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/train'
```

```
getPath_data_all =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/all'

all_categories = []
all_filename = []

images = os.listdir(getPath_data_train)
for index,r in enumerate(images):
 if r != 'dataset_for_classification':
   read = images[index]
   read = getPath_data_train+'/'+read
   print(read)
   print(r)
   for i,file in enumerate(os.listdir(read)):
    if r == 'front':
      all_filename.append(file)
      all_categories.append(2)
    elif(r=='rear'):
      all_filename.append(file)
      all_categories.append(1)
    else:
      all_filename.append(file)
      all_categories.append(0)
else:
    None

df =pd.DataFrame({
    'image': all_filename,
    'category':all_categories
    })

df['category'] = df['category'].replace({2:'front',1:'rear',0:'side'})
```

```
x_train,x_val = train_test_split(df ,random_state=42,shuffle=True,test_size=0.2)

x_train = x_train.reset_index(drop=True)
x_val = x_val.reset_index(drop=True)
print('Train Images shape is : ',x_train.shape)
print('Validation Images shape is : ',x_val.shape)

batch_size = 128
train_ds = ImageDataGenerator(rescale = 1.0/255,
  shear_range = 0.2,
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_train,
    getPath_data_all,
    x_col='image',
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
 )


val_ds = ImageDataGenerator(rescale = 1.0/255,
  shear_range = 0.2,
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
```

```
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_val,
    getPath_data_all,
    x_col='image',
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
)



from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import  ModelCheckpoint
from tensorflow.keras import optimizers
# from tensorflow.keras.losses import CategoricalFocalCrossentropy
from tensorflow.keras.losses import CategoricalCrossentropy

def create_model():
    model = Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(3, activation='softmax'))

    return model
```

```
model = create_model()

model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizers.Adam(learning_rate=0.0001),
    metrics=['accuracy'])

n_training_samples = len(train_ds)
n_validation_samples = len(val_ds)

history = model.fit(train_ds
                    ,epochs=20
                    ,validation_data=val_ds
                    ,validation_steps=n_validation_samples//batch_size
                    ,steps_per_epoch =n_training_samples
                    )

loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, acc, color='blue', label='Training')
plt.plot(epochs, val_acc, color='green', label='Validation')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(epochs, loss, color='orange', label='Training')
plt.plot(epochs, val_loss, color='red', label='Validation')
```

```
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

test_path = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash_classification'
#get image path
test_images = os.listdir(test_path)
 #creat data frame
df_test =pd.DataFrame({
 'image': test_images,
})

test_data_gen = ImageDataGenerator( rescale = 1./255 ).flow_from_dataframe(
df_test,
test_path,
x_col='image',
y_col= None,
target_size=(256,256),
class_mode=None,
shuffle=True,
batch_size=batch_size)

predection = model.predict(test_data_gen)
predection
tf.nn.softmax(predection[0])

from google.colab.patches import cv2_imshow
import cv2

def get_Label(number):
  labels = {0:'side', 1:'rear',2:'front'}
```

```
  return labels[number]


from sklearn.metrics import confusion_matrix,classification_report
get_val_vali = [1,2,1,2,2,2,0,0,0,0]

y_pred = []
y_get_df = []
for i in os.listdir(test_path):
  pic_predict = os.path.join(test_path,i)
  img = tf.keras.utils.load_img(
  pic_predict, target_size=(256, 256)
  )
  img_array = tf.keras.utils.img_to_array(img)
  img_array = tf.expand_dims(img_array, 0)
  predictions = model.predict(img_array)
  score = tf.nn.softmax(predictions[0])
  cv2.imread(pic_predict)


  print(score)
  print("Pic's name :"+i+" This image {}".format(get_Label(int(np.argmax(score)))) + "
Accuracy is "+str(100 * np.max(score)))
  y_pred.append(int(np.argmax(score)))
  # img_show = cv2.imread(pic_predict)
  # cv2_imshow(img_show)
print("-----------------------------------Report-----------------------------------------------")
# tn,fp,fn,tp = confusion_matrix(get_val_vali,y_pred).ravel()
# print("True negative {} False positive {} False negative {} True positive
{}".format(tn,fp,fn,tp))
print(confusion_matrix(get_val_vali,y_pred))
print(classification_report(get_val_vali,y_pred))


import seaborn as sns
```

```
ax = plt.axes()
sns.heatmap(confusion_matrix(get_val_vali,y_pred), annot=True,annot_kws={"size": 25},
cmap="Blues", ax = ax)
ax.set_xlabel("Predicted Front Rear Side Confusion matrix", fontsize=14, labelpad=20)
ax.xaxis.set_ticklabels(['Front', 'Rear','Side'])

# set y-axis label and ticks
ax.set_ylabel("Actual", fontsize=14, labelpad=20)
ax.yaxis.set_ticklabels(['Front', 'Rear','Side'])
ax.set_title('Confusion matrix with Normal CNN', size=14)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

ภาคผนวก ข

2.Front rear classification

CNN

```
!nvidia-smi

import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
import pandas as pd

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D
from google.colab import drive
from keras.preprocessing.image import ImageDataGenerator

from sklearn.model_selection import train_test_split

drive.mount('/content/gdrive')

getPath_data_train =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/train'
getPath_data_all =
'/content/gdrive/MyDrive/Project_Ayok/dataset_for_classification/all'

all_categories = []
all_filename = []
```

```
images = os.listdir(getPath_data_train)
for index,r in enumerate(images):
 if r != 'dataset_for_classification':
   read = images[index]
   read = getPath_data_train+'/'+read
   print(read)
   print(r)
   for i,file in enumerate(os.listdir(read)):
    if r == 'front':
      all_filename.append(file)
      all_categories.append(2)
    elif(r=='rear'):
      all_filename.append(file)
      all_categories.append(1)
    else:
      all_filename.append(file)
      all_categories.append(0)
else:
    None

df =pd.DataFrame({
    'image': all_filename,
    'category':all_categories
    })

df['category'] = df['category'].replace({2:'front',1:'rear',0:'side'})

x_train,x_val = train_test_split(df ,random_state=42,shuffle=True,test_size=0.2)

x_train = x_train.reset_index(drop=True)
x_val = x_val.reset_index(drop=True)
print('Train Images shape is : ',x_train.shape)
print('Validation Images shape is : ',x_val.shape)
```

```
batch_size = 128
train_ds = ImageDataGenerator(rescale = 1.0/255,
  shear_range = 0.2,
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_train,
    getPath_data_all,
    x_col='image',
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
 )



val_ds = ImageDataGenerator(rescale = 1.0/255,
  shear_range = 0.2,
  zoom_range = 0.2,
  height_shift_range=0.2,
  width_shift_range=0.2,
  fill_mode='nearest',
  horizontal_flip=True,
  rotation_range = 40).flow_from_dataframe(
    x_val,
    getPath_data_all,
    x_col='image',
```

```
    y_col='category',
    target_size=(256,256),
    class_mode='categorical',
    shuffle=True,
    batch_size=batch_size
)
```

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import  ModelCheckpoint
from tensorflow.keras import optimizers
# from tensorflow.keras.losses import CategoricalFocalCrossentropy
from tensorflow.keras.losses import CategoricalCrossentropy

from tensorflow.keras.applications import ResNet50
from keras.regularizers import l2
from keras.layers import (Activation, Dropout, Flatten, Dense, GlobalMaxPooling2D,
                          BatchNormalization, Input, Conv2D, GlobalAveragePooling2D)

base_model = ResNet50(input_shape = (150, 150, 3),  include_top = False, weights =
'imagenet')
#
base_model.load_weights('../input/resnet50/resnet50_weights_tf_dim_ordering_tf_kern
els_notop.h5')
x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu', kernel_regularizer=l2(5e-4))(x)
x = Dropout(0.5)(x)
x = Dense(3, activation='sigmoid')(x)
model= Model( base_model.input, x)

model.summary()
```

```
model.compile(
    loss='categorical_crossentropy',
    optimizer=optimizers.Adam(learning_rate=0.0001),
    metrics=['accuracy'])

filepath="resnet/resnet_stage1_fc-{val_accuracy:.3f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath,monitor='val_loss',
save_best_only=True,verbose=0,mode='auto')



n_training_samples = len(train_ds)
n_validation_samples = len(val_ds)

history = model.fit(train_ds
                    ,epochs=20
                    ,validation_data=val_ds
                    ,validation_steps=n_validation_samples//batch_size
                    ,steps_per_epoch =n_training_samples
                    )

loss = history.history['loss']
val_loss = history.history['val_loss']
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(1, len(loss) + 1)

plt.plot(epochs, acc, color='blue', label='Training')
plt.plot(epochs, val_acc, color='green', label='Validation')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.show()

plt.plot(epochs, loss, color='orange', label='Training')
plt.plot(epochs, val_loss, color='red', label='Validation')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

test_path = '/content/gdrive/MyDrive/Project_Ayok/datareal_crash_classification'
#get image path
test_images = os.listdir(test_path)
 #creat data frame
df_test =pd.DataFrame({
 'image': test_images,
})

test_data_gen = ImageDataGenerator( rescale = 1./255 ).flow_from_dataframe(
df_test,
test_path,
x_col='image',
y_col= None,
target_size=(256,256),
class_mode=None,
shuffle=True,
batch_size=batch_size)

predection = model.predict(test_data_gen)
predection
tf.nn.softmax(predection[0])

from google.colab.patches import cv2_imshow
```

```
import cv2

def get_Label(number):
  labels = {0:'side', 1:'rear',2:'front'}
  return labels[number]


from sklearn.metrics import confusion_matrix,classification_report
get_val_vali = [1,2,1,2,2,2,0,0,0,0]

y_pred = []
y_get_df = []
for i in os.listdir(test_path):
  pic_predict = os.path.join(test_path,i)
  img = tf.keras.utils.load_img(
  pic_predict, target_size=(256, 256)
  )
  img_array = tf.keras.utils.img_to_array(img)
  img_array = tf.expand_dims(img_array, 0)
  predictions = model.predict(img_array)
  score = tf.nn.softmax(predictions[0])
  cv2.imread(pic_predict)


  print(score)
  print("Pic's name :"+i+" This image {}".format(get_Label(int(np.argmax(score)))) + "
Accuracy is "+str(100 * np.max(score)))
  y_pred.append(int(np.argmax(score)))
  # img_show = cv2.imread(pic_predict)
  # cv2_imshow(img_show)
print("-----------------------------------Report-----------------------------------------------")
# tn,fp,fn,tp = confusion_matrix(get_val_vali,y_pred).ravel()
# print("True negative {} False positive {} False negative {} True positive
{}".format(tn,fp,fn,tp))
```

```
print(confusion_matrix(get_val_vali,y_pred))
print(classification_report(get_val_vali,y_pred))

import seaborn as sns
ax = plt.axes()
sns.heatmap(confusion_matrix(get_val_vali,y_pred), annot=True,annot_kws={"size": 25},
cmap="Blues", ax = ax)
ax.set_xlabel("Predicted Front Rear Side Confusion matrix", fontsize=14, labelpad=20)
ax.xaxis.set_ticklabels(['Front', 'Rear','Side'])

# set y-axis label and ticks
ax.set_ylabel("Actual", fontsize=14, labelpad=20)
ax.yaxis.set_ticklabels(['Front', 'Rear','Side'])
ax.set_title('Confusion matrix with ResNet', size=14)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

ภาคผนวก ค

3.API Model

```
!pip3 install 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'

# Commented out IPython magic to ensure Python compatibility.
# %matplotlib inline
from pycocotools.coco import COCO
import numpy as np
import skimage.io as io
import matplotlib.pyplot as plt
import pylab
import random
pylab.rcParams['figure.figsize'] = (8.0, 10.0)# Import Libraries

# For visualization
import os
import seaborn as sns
from matplotlib import colors
from tensorboard.backend.event_processing import event_accumulator as ea
from PIL import Image
from google.colab import drive

drive.mount('/content/gdrive')

from keras.models import load_model
import os
import tensorflow as tf
from tensorflow import keras
import numpy as np
import cv2

def modelFRS(getImg):
```

```
  model =
load_model('/content/gdrive/MyDrive/Project_Ayok/model/models/frsmodels/FRS-
model.h5')
  class_names = ['front', 'rear', 'side']
  img = tf.keras.utils.load_img(
        getImg, target_size=(180, 180)
  )
  img_array = tf.keras.utils.img_to_array(img)
  img_array = tf.expand_dims(img_array, 0)
  predict_picture = model.predict(img_array)
  result = []
  score = tf.nn.softmax(predict_picture[0])
  result_class = class_names[np.argmax(score)]
  result_accuracy = 100 * np.max(score)
  # result.append(result_class,result_accuracy)
  return result_class,result_accuracy


getImg = '/content/gdrive/MyDrive/Project_Ayok/model/upload_pic/9j4AAQSkZJ.png'

modelFRS(getImg)

# !cat /usr/local/cuda/version.txt
# The user requested torch==1.7.0
#     torchvision 0.8.2+cu92 depends on torch==1.7.1
#     The user requested torch==1.7.0
#     torchvision 0.8.2+cu110 depends on torch==1.7.1
#     The user requested torch==1.7.0
#     torchvision 0.8.2+cu101 depends on torch==1.7.1
#     The user requested torch==1.7.0
#     torchvision 0.8.2+cpu depends on torch==1.7.1
#     The user requested torch==1.7.0
#     torchvision 0.8.2 depends on torch==1.7.1
```

```
!python3  -m pip install detectron2 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu92/torch1.7/index.html
!pip3 install torch==1.7.0 torchvision -f
https://download.pytorch.org/whl/torch_stable.html

# !python3  -m pip install detectron2 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.8/index.html
# !pip3 install torch==1.8.0 torchvision -f
https://download.pytorch.org/whl/torch_stable.html
# !pip3 install torch torchvision -f
https://download.pytorch.org/whl/cu100/torch_stable.html
# !conda install pytorch torchvision cudatoolkit=10.0 -c pytorch
# pip3 install torch torchvision

!python3 -m detectron2.utils.collect_env

import torch, torchvision
print(torch.__version__, torch.cuda.is_available())

# !python3 -m pip install 'git+https://github.com/facebookresearch/detectron2.git'
!git clone https://github.com/facebookresearch/detectron2
!python3 /content/detectron2/setup.py install


!pip3 install flask-ngrok
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.tgz
!tar -xvf /content/ngrok-stable-linux-amd64.tgz
!./ngrok authtoken 2DLf6JIdJfJz3CHkIFlExgyN97M_2VVdEEEZNtshtiifBq3qa

# https://github.com/onnx/onnx/issues/582#issuecomment-824263936
!pip install onnx==1.8.1

assert torch.__version__.startswith("1.7")
```

```
# import some common libraries
import numpy as np
import os, json, cv2, random
import matplotlib.pyplot as plt
import skimage.io as io
# /content/detectron2
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# # import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
from detectron2.engine import DefaultTrainer
from detectron2.utils.visualizer import ColorMode
from detectron2.evaluation import COCOEvaluator, inference_on_dataset
from detectron2.data import build_detection_test_loader
import argparse
import os
import onnx

from detectron2.checkpoint import DetectionCheckpointer
from detectron2.data import build_detection_test_loader
from detectron2.modeling import build_model
from detectron2.export import Caffe2Tracer
```

```
# Set base params
plt.rcParams["figure.figsize"] = [16,9]
print(detectron2.__version__)

!nvidia-smi
!kill process_id

from detectron2.checkpoint.detection_checkpoint import pickle
from detectron2.utils.events import EventStorage
from detectron2.config import get_cfg
from detectron2.engine import DefaultPredictor
from detectron2.checkpoint import DetectionCheckpointer
# from detectron2.data import build_detection_test_loader
from detectron2.modeling import build_model

from detectron2.export import Caffe2Model


with open('/content/gdrive/MyDrive/Project_Ayok/model/models/config.yaml','rb') as f:
  cfg = pickle.load(f)


cfg.MODEL.WEIGHTS =
"/content/gdrive/MyDrive/Project_Ayok/model/models/output/model_final.pth" # path
for final model

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
predictor = DefaultPredictor(cfg)

img =
io.imread("/content/gdrive/MyDrive/Project_Ayok/datatest_real/LINE_ALBUM_Case1_22
0608.jpg")
image = torch.as_tensor(img.astype("float32").transpose(2, 0, 1))
```

```
outputs = predictor(img)

v = Visualizer(img[:, :, ::-1],
               #  metadata=val_metadata_dicts,
                scale=0.5,
                instance_mode=ColorMode.IMAGE_BW)


out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
plt.imshow(out.get_image()[:, :, ::-1])


cfg = None

def compulsaryLogic(positionFisrtList):
  if "กระโปรงหน้า" in  positionFisrtList or "กระโปรงท้าย" in  positionFisrtList :
    return "พ.ร.บ. 1 หรือ พ.ร.บ. 4 หรือ พ.ร.บ. 5"
  elif "ประตูหน้าด้านขวา" in  positionFisrtList or "ประตูหลังด้านขวา" in  positionFisrtList:
    return "พ.ร.บ. 1 หรือ พ.ร.บ. 6 หรือ พ.ร.บ. 11 หรือ พ.ร.บ. 17"
  elif "กระจกข้างด้านซ้าย" in  positionFisrtList or "ไฟด้านหน้าซ้าย" in  positionFisrtList or
"ประตูหน้าด้านซ้าย" in  positionFisrtList or "ประตูหลังด้านซ้าย" in  positionFisrtList:
    return "พ.ร.บ. 8 หรือ พ.ร.บ. 19 หรือ พ.ร.บ. 11 หรือ พ.ร.บ. 17 หรือ พ.ร.บ. 9"
  #elif "กระจกข้างด้านซ้าย" in  positionFisrtList and "ไฟด้านหน้าซ้าย" in  positionFisrtList
and "ประตูหน้าด้านซ้าย" in  positionFisrtList and "ประตูหลังด้านซ้าย" in  positionFisrtList:
  #  return "พ.ร.บ. 11 หรือ พ.ร.บ. 17"
  #elif "ไฟด้านหน้าซ้าย" in  positionFisrtList:
  #  return "พ.ร.บ. 9"
  elif "ไฟด้านหลังขวา" in  positionFisrtList:
    return "พ.ร.บ. 13"
  elif "ไฟด้านหน้าขวา" in  positionFisrtList or "กระจกข้างด้านขวา" in positionFisrtList:
    return "พ.ร.บ. 2 หรือ พ.ร.บ. 7 หรือ พ.ร.บ. 12 หรือ พ.ร.บ. 14 หรือ พ.ร.บ. 15"


# def compulsaryLogic(positionFisrtList,positionSecondList):
```

```
#   if "กระโปรงหน้า" in  positionFisrtList:
#     if "ประตูหน้าด้านขวา" in  positionSecondList or "ประตูหลังด้านขวา" in
positionSecondList:
#       return "พ.ร.บ. 1"
#     elif "กระโปรงท้าย" in  positionSecondList:
#       return "พ.ร.บ. 4 หรือ พ.ร.บ. 5"
#   elif "ประตูหน้าด้านขวา" in  positionFisrtList:
#     if "ไฟด้านหน้าซ้าย" in  positionSecondList:
#       return "พ.ร.บ. 6"
#   elif "กระจกข้างด้านซ้าย" in  positionFisrtList or "ไฟด้านหน้าซ้าย" in  positionFisrtList or
"ประตูหน้าด้านซ้าย" in  positionFisrtList or "ประตูหลังด้านซ้าย" in  positionFisrtList:
#     if "ไฟด้านหน้าขวา" in  positionSecondList:
#       return "พ.ร.บ. 8 หรือ พ.ร.บ. 19"
#   elif "กระจกข้างด้านซ้าย" in  positionFisrtList and "ไฟด้านหน้าซ้าย" in  positionFisrtList
and "ประตูหน้าด้านซ้าย" in  positionFisrtList and "ประตูหลังด้านซ้าย" in  positionFisrtList:
#     if "กระจกข้างด้านขวา" in  positionSecondList and "ไฟด้านหน้าขวา" in
positionSecondList and "ประตูหน้าด้านขวา" in  positionSecondList and "ประตูหลังด้านขวา"
in  positionSecondList:
#       return "พ.ร.บ. 11 หรือ พ.ร.บ. 17"
#   elif "ไฟด้านหน้าซ้าย" in  positionFisrtList:
#     if "ไฟด้านหลังขวา" in  positionSecondList:
#       return "พ.ร.บ. 9"
#     elif "ไฟด้านหน้าขวา" in positionSecondList:
#       return "พ.ร.บ. 18"
#   elif "ไฟด้านหลังขวา" in  positionFisrtList:
#     if "ไฟด้านหน้าซ้าย" in  positionSecondList:
#       return "พ.ร.บ. 13"
#   elif "ไฟด้านหน้าขวา" in  positionFisrtList or "กระจกข้างด้านขวา" in positionFisrtList:
#     if "กระจกข้างด้านซ้าย" in  positionSecondList  or "ไฟด้านหน้าซ้าย" in
positionSecondList or "ประตูหน้าด้านซ้าย" in  positionSecondList or "ประตูหลังด้านซ้าย" in
positionSecondList:
#       return "พ.ร.บ. 2 หรือ พ.ร.บ. 7 หรือ พ.ร.บ. 12 หรือ พ.ร.บ. 14 หรือ พ.ร.บ. 15"
```

```
from traitlets.traitlets import Enum
from flask import Flask,send_file
from flask_ngrok import run_with_ngrok
from flask import Flask, request, jsonify,make_response
app = Flask(__name__)
run_with_ngrok(app)
from werkzeug.utils import secure_filename
import base64



UPLOAD_FOLDER = '/content/gdrive/MyDrive/Project_Ayok/model/upload_pic'
PREDICT_FOLDER = '/content/gdrive/MyDrive/Project_Ayok/model/predict_pic'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['PREDICT_FOLDER'] = PREDICT_FOLDER



with open('/content/gdrive/MyDrive/Project_Ayok/model/models/config.yaml','rb') as f:
  cfg = pickle.load(f)



cfg.MODEL.WEIGHTS =
"/content/gdrive/MyDrive/Project_Ayok/model/models/output/model_final.pth" # path
for final model

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
predictor = DefaultPredictor(cfg)

def allowed_file(filename):
    return '.' in filename and \
```

```
        filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route("/predictDamage",methods=['POST'])
def predictDamage():
    request_data = request.get_json()
    image_list = None
    pre_predict_list = []
    if request_data:
      if 'image_list' in request_data:
        image_list = request_data['image_list']
        pre_predict_list.append(image_list)
        result_return = []
        get_list_file = []

      for i,j in enumerate(pre_predict_list[0]):
        get_base_64 = j['image_64']
        name = get_base_64.replace('/','')
        filename = str(name[:10])+'.png'
        base64_img_bytes = get_base_64.encode('utf-8')
        getImg = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        with open(getImg, 'wb') as file_to_save:
          decoded_image_data = base64.decodebytes(base64_img_bytes)
          file_to_save.write(decoded_image_data)
          file_to_save.close()
        get_list_file.append(getImg)


    # predictor = DefaultPredictor(model)
    image = torch.as_tensor(img.astype("float32").transpose(2, 0, 1))
    outputs = predictor(img)
    # im = first_batch[0]['image']
```

```
    v = Visualizer(img[:, :, ::-1],
                   #  metadata=val_metadata_dicts,
                   scale=0.5,
                   instance_mode=ColorMode.IMAGE_BW)
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    plt.imshow(out.get_image()[:, :, ::-1])

    exportResult = os.path.join(app.config['PREDICT_FOLDER'], filename)
    plt.savefig(exportResult)

    with open(exportResult, 'rb') as binary_file:
      binary_file_data = binary_file.read()
      base64_encoded_data = base64.b64encode(binary_file_data)
      base64_message = base64_encoded_data.decode('utf-8')
    return jsonify(result_image_base64=base64_message)
@app.route("/predictFRS",methods=['POST'])
# Parameter picture.jpg
def predictFRS():
  if request.method == "POST":
    request_data = request.get_json()
    image_list = None
    pre_predict_list = []
    if request_data:
      if 'image_list' in request_data:
        image_list = request_data['image_list']
        pre_predict_list.append(image_list)
        json_list = []

        result_compul_list = []

      for i,j in enumerate(pre_predict_list[0]):
        get_base_64 = j['image_64']
        get_damage_id = j['damage_id']
```

```
        get_first_car = j['position']
        # get_second_car = j['positionSecondList']
        result_compul = compulsaryLogic(get_first_car)
        name = get_base_64.replace('/','')
        filename = str(name[:10])+'.png'
        base64_img_bytes = get_base_64.encode('utf-8')
        getImg = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        with open(getImg, 'wb') as file_to_save:
          decoded_image_data = base64.decodebytes(base64_img_bytes)
          file_to_save.write(decoded_image_data)
          file_to_save.close()
        model_result_acc,model_result_class =  modelFRS(getImg)
        json_data = {
           "accuracy":model_result_class,
           "position_result":model_result_acc,
           "compul":result_compul,
           "damage_id":get_damage_id
        }
        json_list.append(json_data)

    return jsonify(result_image=json_list)
app.run()
with open('/content/S__32965035.jpg', 'rb') as binary_file:
      binary_file_data = binary_file.read()
      base64_encoded_data = base64.b64encode(binary_file_data)
      base64_message = base64_encoded_data.decode('utf-8')
print(base64_message)
```

ประวัติผู้เขียน

ชื่อ	นางสาว ฐิติชยาน์ คุ้มวงษ์
วัน เดือน ปีเกิด	01 กุมภาพันธ์ พ.ศ. 2541
ที่อยู่ปัจจุบัน	492/4 หมู่ 13 ตำบล บัวขาว อำเภอ กุฉินารายณ์ จังหวัด กาฬสินธุ์
ประวัติการศึกษา	(2563) วิทยาศาสตรบัณฑิต สาขา สถิติประยุกต์ เกรดเฉลี่ย 3.11 (สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง)