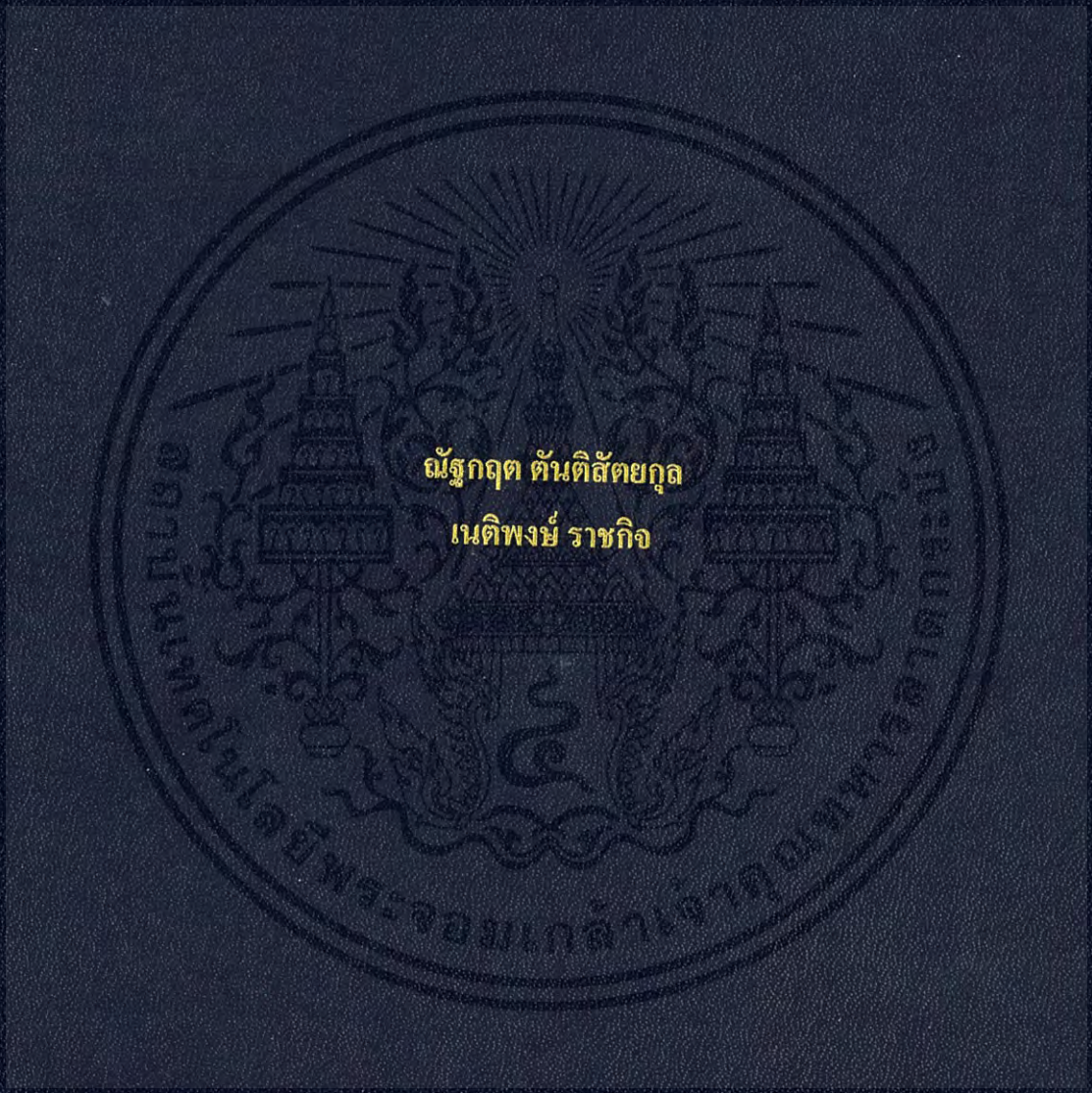
ระบบบริหารจัดการที่จอดรถกลางแจ้ง

OUTDOOR PARKING MANAGEMENT SYSTEM



ณัฐกฤต ตันติสัตยกุล

เนติพงษ์ ราชกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ระบบบริหารจัดการที่จอดรถกลางแจ้ง

OUTDOOR PARKING MANAGEMENT SYSTEM

ณัฐกฤต ตันติสัตยกุล

เนติพงษ์ ราชกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบบริหารจัดการที่จอดรถกลางแจ้ง

OUTDOOR PARKING MANAGEMENT SYSTEM

ผู้จัดทำ

1. นายณัฐกฤต ตันติสัตยกุล รหัสนักศึกษา 58010361

2. นายเนติพงษ์ ราชกิจ รหัสนักศึกษา 58010688

_____ อาจารย์ที่ปรึกษา

(อาจารย์จิระศักดิ์ สิทธิกร)

_____ อาจารย์ที่ปรึกษาร่วม

(รศ.ดร.อรฉัตร จิตต์โสภักตร์)

ระบบบริหารจัดการที่จอดรถกลางแจ้ง

นายณัฐกฤต ตันติสัตยกุล 58010361
นายเนติพงษ์ ราชกิจ 58010688
อาจารย์จิระศักดิ์ สิทธิกร อาจารย์ที่ปรึกษา
รศ.ดร.อรฉัตร จิตต์โสภักตร์ อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2561

บทคัดย่อ

การใช้งานที่จอดรถเป็นปัญหาที่เกิดขึ้นกับผู้ขับขี่รถตั้งแต่ระดับบุคคลทั่วไปที่ขับรถส่วนตัวจนถึงระดับธุรกิจ ไม่ว่าจะเป็นธุรกิจด้านห้างสรรพสินค้า ธุรกิจประเภทขนส่งธุรกิจประเภทโรงงานที่ต้องมีการลำเลียงสินค้า ปัญหาที่รองลงมาจากจำนวนที่จอดไม่เพียงพอคือกรณีที่ต้องใช้เวลาขับรถเพื่อวนหาที่จอดรถว่างทำให้เสียเวลาในการทำงาน หรือทำให้ธุรกิจที่ต้องนำเวลามาเป็นตัวชี้วัดการทำงานมีประสิทธิภาพลดลง

การลดปัญหาที่จอดรถกรณีดังกล่าวจำเป็นต้องมีระบบที่คอยช่วยเหลือ บริหารการจัดการที่จอดรถ เช่น แสดงจำนวนที่จอดรถที่วางในแต่ละส่วน การนับจำนวนของรถที่จอดไว้ รวมถึงการระบุตำแหน่งช่องจอดรถที่พร้อมให้บริการ

สำหรับระบบบริหารจัดการที่จอดรถโดยทั่วไปมักจะมีการติดตั้งเซ็นเซอร์อิลเท็กทรอนิกส์เหนือที่จอดรถแต่ละคันเพื่อตรวจสอบ และนำไปประมวลผล ซึ่งเหมาะกับที่จอดรถในที่ร่ม เนื่องจากสามารถติดตั้ง และดูแลรักษาได้ง่ายแต่หากนำไปใช้กับที่จอดรถกลางแจ้งจะมีต้นทุนในการติดตั้งสูงกว่าอีกทั้งทำให้อายุการใช้งานลดลง

โครงงานนี้นำเสนอระบบบริหารจัดการที่จอดรถกลางแจ้งที่สามารถแสดงข้อมูลที่จอดรถและแนะนำช่องจอดรถที่พร้อมให้บริการโดยใช้การประมวลผลภาพในการตรวจสอบตำแหน่งของรถที่จอดผ่านกล้องที่ติดตั้งไว้ที่มุมสูงของที่จอดรถเพื่อให้สามารถจับภาพได้กว้างแทนการใช้เซนเซอร์อิเล็กทรอนิกส์ที่ฝังอยู่บนเพดานในการประมวลผลภาพ

Outdoor Parking Management System

Mr.Nattakit	Tantisattayakun	58010361
Mr.Netipong	Ratchakit	58010688
Mr.Jirasak	Sittigorn	Advisor
Assoc.Prof.Dr.Orachat	Chitsobhuk	Co-Advisor

Academic year 2018

ABSTRACT

Parking is a problem for many driver from general to business. Whether the business of shopping mall or transportation of factory that have to conveying product. Minor problem from insufficient parking slot is in case that need to drive for half an hour to find parking slot cause to wastes a lot of time for work or reduce performance some business that have time is a measure of work

To reduce the problem of parking in such cases need a system that helps. Parking management, for example, shows the number of parking spaces placed in each section. Counting the number of cars parked including show the availability of parking spaces.

For parking management systems, there is usually an electronic sensor installed above each parking lot to check and to process. Suitable for indoor parking, as can be easy to install and maintain, but if used outdoors it will cost more to install and will reduce the service life.

This project present an outdoor parking management system that can display parking information. It also introduces the availability of parking spaces using image processing to monitor the parking position via cameras that mount at top view of parking to capture a wide range of images instead of using electronic sensors embedded in the ceiling in image processing

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์ที่ปรึกษา อาจารย์จิระศักดิ์ สิทธิกร และ อาจารย์ที่ปรึกษาร่วม รศ.ดร.อรฉัตร จิตต์โสภักตร์ ที่คอยเอาใจใส่ดูแล สอบถามความคืบหน้าคอยให้คำปรึกษา เสนอแนะแนวคิดในการแก้ปัญหา จนทำให้ปริญญานิพนธ์เล่มนี้สามารถจัดทำขึ้นมาได้สำเร็จลุล่วง

ขอขอบคุณอาจารย์ทุกท่านที่ได้อบรมสั่งสอนให้วิชาความรู้หลากหลายด้านที่ทำให้ทางคณะผู้จัดทำสามารถนำความรู้มาประยุกต์ใช้ในการจัดทำปริญญานิพนธ์เล่มนี้ได้

สุดท้ายนี้ ขอขอบพระคุณทุกท่านที่ช่วยส่งเสริม กระตุ้นเตือน และเป็นกำลังใจให้ตลอดมา รวมถึงผู้ที่มีส่วนเกี่ยวข้องที่ไม่ได้กล่าวถึงทุกท่าน

ณัฐกฤต ตันติสัตยกุล

เนติพงษ์ ราชกิจ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เนื่องจากปัจจุบันผู้ขับขี่รถยนต์มักประสบปัญหาในการหาที่จอดรถตามสถานที่ต่างๆได้อย่างลำบาก บางครั้งต้องเสียเวลากับการวนรถหาที่จอดรถ ทำให้ผู้ให้บริการสถานที่ต้องลงทุนติดตั้งอุปกรณ์ระบบบริหารจัดการที่จอดรถ สำหรับที่จอดรถในร่มนั้นจะมีการตรวจสอบจำนวนที่ว่างของที่จอดรถโดยใช้เซนเซอร์ตรวจจับวัตถุ ซึ่งมีค่าใช้จ่ายในการติดตั้งและค่าบำรุงรักษา และหากนำอุปกรณ์ดังกล่าวไปใช้กับที่จอดรถกลางแจ้งจะเกิดค่าใช้จ่ายในการติดตั้งและค่าบำรุงรักษาที่สูงมาก ทำให้เมื่อมีการเสียหายของอุปกรณ์ผู้ประกอบการต่องมีค่าใช้จ่ายเพิ่มขึ้น

ดังนั้นโครงงานนี้จึงนำเสนอระบบบริหารจัดการที่จอดรถกลางแจ้งที่สามารถแสดงข้อมูลที่จอดรถ และแนะนำช่องจอดรถที่พร้อมให้บริการโดยใช้การประมวลผลภาพ โดยเทคโนโลยีการประมวลผลด้วยภาพจะเข้ามาช่วยแก้ไขปัญหาในเรื่องค่าใช้จ่าย สาเหตุที่ค่าใช้จ่ายถูกลงเนื่องจากถ้าใช้เซนเซอร์เป็นส่วนตรวจสอบรถต้องใช้จำนวนมากเพราะอัตราการตรวจจับจะเป็นอัตรา 1 เซนเซอร์ : 1 ช่องจอดรถ แต่ถ้าใช้กล้อง สามารถตรวจจับได้หลายช่องที่จอดรถต่อกล้องเพียง 1 ตัว จึงประหยัดทั้งค่าติดตั้งและค่าบำรุงรักษา

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อประยุกต์ใช้ Machine Learning Model กับการตรวจสอบที่จอดรถ
2) เพื่อตรวจสอบจำนวนที่ว่างของที่จอดรถ และแสดงผลให้ผู้ขับรถได้ทราบ
3) พัฒนาระบบบริหารจัดการสำหรับระบบจอดรถทั่วไป

1.3 ขอบเขตของโครงงาน

1) สามารถตรวจจับภาพและจำแนกออกว่าเป็นรถ
2) สามารถตรวจสอบที่จอดรถว่าว่างอยู่หรือไม่
3) สามารถประมวลผลรวมจำนวนที่จอดรถที่ว่างไปแสดงผลออกทางจอได้
4) สามารถชี้แนะที่จอดรถตรงไหนว่างให้ผู้ที่ขับรถได้

1.4 ข้อจำกัดของโปรแกรม

1) หากรูปที่ได้มามืดเกินไป หรือเป็นรูปที่ถ่ายได้ในเวลากลางคืน ตัวโมเดลจะไม่สามารถตรวจจับได้ หรือตรวจจับได้ผิดพลาด

2) รถบางประเภทที่ไม่ได้อยู่ใน Dataset COCO ที่ใช้สำหรับเทรนโมเดล ตัวโมเดลจะไม่สามารถตรวจจับได้ หรือตรวจจับได้ยาก

3) รูปภาพที่จะใช้โมเดลตรวจจับว่ามียานพาหนะอยู่หรือไม่ ไม่สามารถใช้รูปลานจอดรถกว้างๆที่ครอบคลุมหลายๆช่องจอดรถได้ จำเป็นต้องแยกภาพเป็นแต่ละช่องจอดรถเดี่ยวๆแล้วนำไปเข้าโมเดล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ตัวระบบสามารถพัฒนาต่อยอดไปใช้ในสถานที่จริงได้

2) ตัวระบบสามารถตรวจสอบจำนวนที่ว่างของที่จอดรถได้

3) สามารถนำความรู้ที่ได้รับมาตลอดการทำงานนำไปต่อยอดใช้งานได้จริง

1.6 แผนการดำเนินงาน

ตาราง 1.1 แผนการดำเนินงานครึ่งแรก

ลำดับ	งานที่ทำ	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	ทำ Proposal และ ศึกษาเรื่อง Image Processing																
2	กำหนดขอบเขตโครงงาน																
3	ศึกษาการใช้ RaspberryPI และ ติดตั้ง Raspbian OS																
4	ทดลองอ่านภาพจากกล้องและประมวลผลภาพด้วย RaspberryPI																
5	ศึกษาทฤษฎีที่นำมาใช้ในโครงงาน																
6	จัดทำโมเดลลานจอดรถ																
7	ทดลองประมวลผลภาพจากโมเดลลานจอดรถและปรับให้คุณภาพดียิ่งขึ้น																

ตาราง 1.2 แผนการดำเนินงานครึ่งหลัง

ลำดับ	งานที่ทำ	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	ศึกษาวิธีตรวจจับช่องจอดรถโดยใช้ Machine Learning																
2	สร้าง Machine Learning Model สำหรับแบบจำลองลานจอดรถ																
3	ทดลอง Web Server สำหรับใช้งานระบบ																
4	ทดสอบและพัฒนาประสิทธิภาพการทำงานของระบบ																
5	จัดทำเอกสาร																

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 Machine Learning

Machine Learning เป็นศาสตร์แขนงหนึ่งที่ทำให้คอมพิวเตอร์มีความสามารถในการเรียนรู้ด้วยตนเอง การสร้างอัลกอริทึมที่สามารถเรียนรู้ข้อมูลและทำนายข้อมูลได้นั้นจะทำงานโดยอาศัยโมเดลที่สร้างมาจากชุดข้อมูลที่มีอยู่ เมื่อมีข้อมูลใหม่เข้ามาจะสามารถทำนายหรือตัดสินใจได้โดยปราศจากการทำงานตามลำดับคำสั่งโปรแกรม ซึ่งทางผู้จัดทำได้เลือกใช้ Machine Learning แบบ Supervised Learning

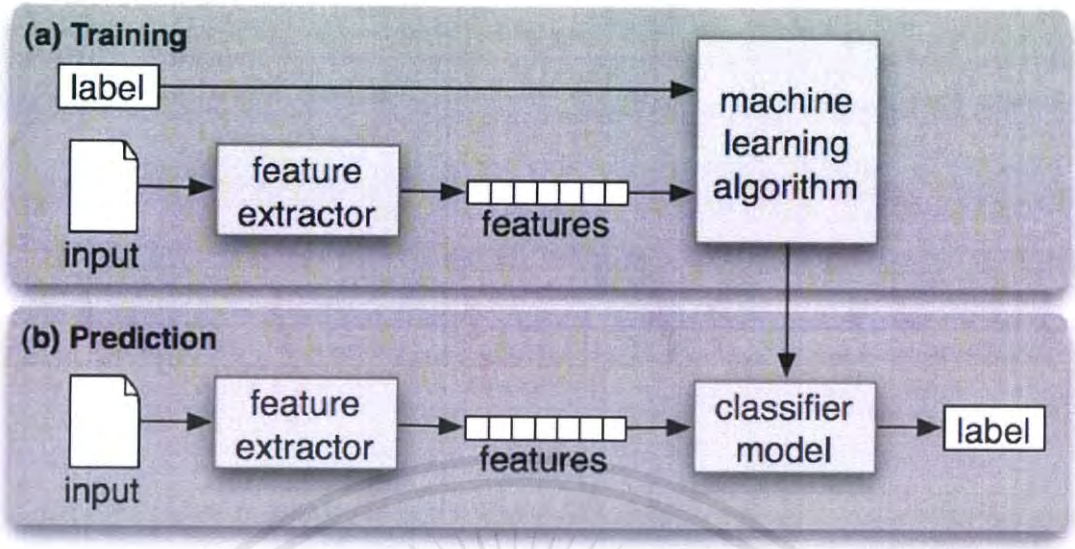
Supervised Learning

เป็นการเรียนรู้แบบมีผู้สอน ในการเรียนรู้แบบนี้คือ จะคอยสอน Machine ให้ว่า Input แบบนี้แล้วจะได้ Output แบบไหน แบ่งออกเป็น

- การแบ่งแยกประเภท (Classification) เป็นการจัดหมวดหมู่ให้ข้อมูลโดยนิยมใช้กับข้อมูลที่ไม่ต่อเนื่อง เช่น ต้องการทำนายเพศของลูกค้าซักคนหนึ่ง จะเริ่มเก็บข้อมูลตั้งแต่ ส่วนสูง น้ำหนัก อาชีพ เงินเดือน สินค้าที่ซื้อ และอื่นๆ จากฐานข้อมูลลูกค้า รวมถึงข้อมูลเพศของลูกค้าแต่ละคน มันสามารถตอบได้เพียงว่าลูกค้ารายนั้นเป็นผู้ชายหรือผู้หญิงเท่านั้น ผลลัพธ์จากตัวแบ่งแยกประเภท (Classifier) จะถูกกำหนดด้วยความน่าจะเป็นว่าเป็น ชาย หรือ หญิง โดยขึ้นอยู่กับข้อมูลลักษณะ (Feature) ที่เก็บได้เป็นหลัก

- การถดถอย (Regression) เป็นการใช้วิเคราะห์ข้อมูลที่มีความต่อเนื่อง เช่น นักวิเคราะห์ด้านการเงินคนหนึ่งต้องการทำนายมูลค่าของหุ้นโดยดูจาก Feature เช่น ส่วนได้ส่วนเสีย (Equity) สถานะของหุ้นก่อนหน้านี้ และดัชนีเศรษฐกิจมหภาคระบบถูกเทรนเพื่อประเมินราคาของหุ้นด้วยความผิดพลาดที่น้อยที่สุด

Supervised Learning สามารถหาคำตอบได้ตามที่เราสอนไว้ ซึ่งตรงกับความต้องการของตัวชิ้นงานที่ต้องการนำ Machine Learning มาช่วยตรวจจับยานพาหนะ ซึ่งตัว Machine Learning เองนั้นยังมีแบบอื่นอยู่อีกนอกจาก Supervised Learning ที่ไม่ได้นำมาเสนอในรายงานนี้

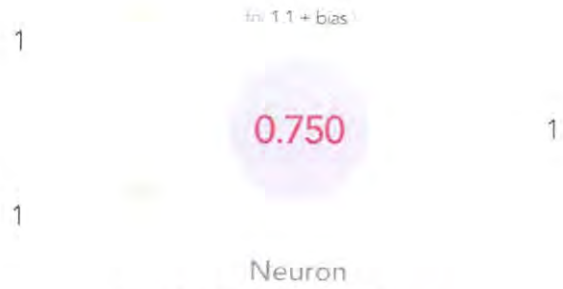


รูป 2.1 การทำงานของ Machine Learning

2.2 Artificial Neural Networks (ANN)

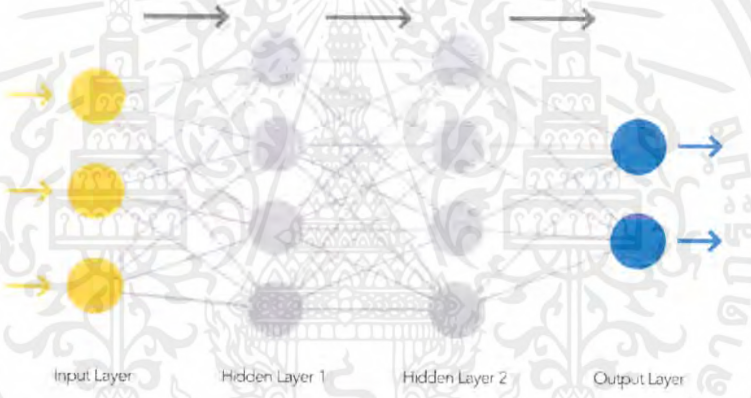
โครงข่ายใยประสาทเสมือน (Artificial Neural Networks) อาศัยแนวคิดและเทคนิคจากการทำงานของระบบโครงข่ายใยประสาทในระบบประสาทของมนุษย์โดยจำลองการทำงานเหมือนกลุ่มเซลล์ประสาทที่เชื่อมโยงกันเป็นระบบประสาทที่สามารถรับรู้หลายๆสิ่งได้ในเวลาเดียวกันด้วยการประมวลผลแบบขนาน (Parallel Network) ทำให้ระบบสามารถตัดสินใจได้ใกล้เคียงกับมนุษย์

การทำงานของ Neural Networks คือเมื่อมี Inputs เข้ามา จะนำ Inputs มาคูณกับ Weight ของแต่ละขา ผลที่ได้จาก Inputs ทุกขาของ Neuron จะนำมารวมกันแล้วมาเทียบกับ Threshold ที่กำหนดไว้ ซึ่งถ้าผลรวมมีค่ามากกว่า Threshold ตัว Neuron จะส่ง Output ออกไปยัง Inputs ของ Neuron อื่นที่เชื่อมกันใน Network ถ้าค่าน้อยกว่า Threshold ก็จะไม่เกิด Output สิ่งสำคัญคือต้องทราบค่า Weight และ Threshold สำหรับสิ่งที่ต้องการเพื่อให้คอมพิวเตอร์รู้จำ ซึ่งเป็นค่าที่ไม่แน่นอน แต่สามารถกำหนดให้คอมพิวเตอร์ปรับค่าเหล่านั้นได้โดยการสอนให้คอมพิวเตอร์ Pattern ของสิ่งที่ต้องการให้มันรู้จำ เรียกว่า "Back Propagation" ในการฝึก Feed-Forward Neural Network จะมีการใช้อัลกอริทึมแบบ Back-Propagation เพื่อใช้ในการปรับปรุง Network Weight หลังจากใส่รูปแบบข้อมูลสำหรับฝึกให้แก่เครือข่ายในแต่ละครั้งแล้ว Output จากเครือข่ายจะถูกนำไปเปรียบเทียบกับผลที่คาดหวังแล้วทำการคำนวณหาค่าความผิดพลาด ซึ่งค่าความผิดพลาดก็จะถูกส่งกลับเข้าสู่เครือข่ายเพื่อใช้แก้ไขค่า Weight ต่อไป



รูป 2.2 การทำงานของ Neuron

Feed-Forward Neural Network จะแบ่ง Neuron ออกเป็นกลุ่มๆ โดยแต่ละกลุ่มจะเรียกเป็น Layer โดยข้อมูลที่เข้ามาจะไหลไปในทิศทางเดียวไม่ไหลย้อนกลับจาก Layer สู่ Layer ถัดไป



รูป 2.3 Neural Network

ส่วนประกอบของ Neural Network

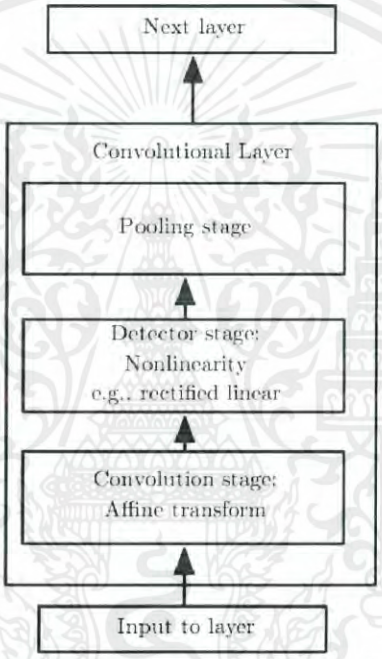
Input Layer ชั้นนี้จะเป็นข้อมูล Input จำนวนของโหนด ขึ้นอยู่กับจำนวนของ Input ว่ามีข้อมูลอะไรบ้างที่นำเข้ามาคิดในโมเดล (ปกติแล้วจะเรียกปัจจัยที่นำมาวิเคราะห์ว่า Feature)

Hidden Layer เป็นชั้นที่อยู่ระหว่าง Input Layer และ Output Layer ซึ่งมีผลอย่างมากต่อประสิทธิภาพในการเรียนรู้ของโมเดล ซึ่ง Hidden Layer จะมีกี่ชั้นก็ได้ แล้วแต่ละชั้นจะมีจำนวนของ Neuron เท่าไหร่ก็ได้เช่นกัน ในการเพิ่มจำนวนชั้นและจำนวน Neuron จะส่งผลต่อการทำงานของโมเดล

Output Layer เป็นชั้นที่จะนำเอาข้อมูลจากการคำนวณไปใช้ จำนวนของโหนดในชั้นนี้นั้นจะขึ้นอยู่กับรูปแบบของ Output ที่จะนำไปใช้

2.3 Convolutional Neural Network (CNN)

Convolutional Neural Network คือ Artificial Neural Network (ANN) ประเภทหนึ่งที่ใช้ Convolutional layers เพื่อกรองข้อมูลที่เป็นประโยชน์จาก Inputs การทำงานนั้นยังรวมถึงการรวมข้อมูลอินพุต (Feature map) เข้ากับ Convolution kernel (Filter) เพื่อสร้าง Feature Map ขึ้นมา โดย Filter ใน Convolutional Layers หรือ Conv Layers จะได้รับการปรับเปลี่ยนตามพารามิเตอร์เพื่อดึงข้อมูลที่เป็นประโยชน์ที่สุด Convolutional Networks จะปรับโดยอัตโนมัติเพื่อที่จะหาคุณลักษณะที่ดีที่สุด



รูป 2.4 องค์ประกอบของ Convolutional Neural Network layer

จากรูป จะเห็นได้ว่าใน 1 Layer ประกอบไปด้วย 3 ส่วนใหญ่ๆ คือ

1. Convolution Stage ในขั้นตอนนี้จะสร้าง Sliding Window (Filter) มาสแกนรูป Input เพื่อทำ Feature Map (ทำการสแกนรูปเพื่อแยกองค์ประกอบของรูป ออกมา เช่น ขอบ สี รูปทรง เป็นต้น)

2. Detector Stage ชั้นนี้จะทำหน้าที่รับ Output จาก Convolution Stage แปลงให้อยู่ในรูปของ Non-Linear โดยใช้ Activation เช่น Rectified Linear Units (ReLU)

3. Pooling Stage ทำหน้าที่ Resize ข้อมูลให้ขนาดเล็กลงโดยที่รายละเอียดของ Input ยังครบถ้วนเหมือนเดิม หลักการทำงานขั้นตอนนี้คล้ายกับ Convolution Stage แต่ต่างกันที่ Output ที่ได้จะมีขนาดเล็กลง โดย Pooling มีประโยชน์ในเรื่องเพิ่มความไวในการคำนวณ และแก้ปัญหา Overfitting ของโมเดล

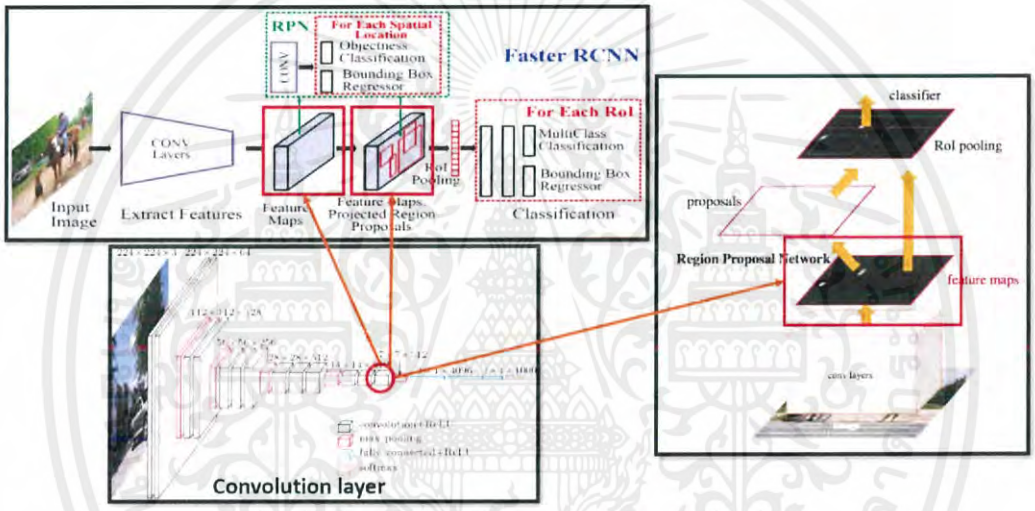
2.4 Faster R-CNN

Faster R-CNN คือการนำ Selective Search มารวมไว้ใน Neural Network เดียวกันซึ่งมีส่วนประกอบหลัก 3 ส่วนคือ

1) ส่วนฐานที่ทำหน้าที่สกัด Feature

2) ส่วน Region Proposal Network (RPN) ทำหน้าที่สกัดบริเวณที่น่าจะเป็นวัตถุจาก Feature Map

3) ส่วนจำแนกประเภท ที่นำ Feature Map และ Region ที่ได้จาก RPN มาประมวลผลโดยทำ ROI pooling เพื่อตอบว่าบริเวณใดของภาพมีวัตถุใดอยู่

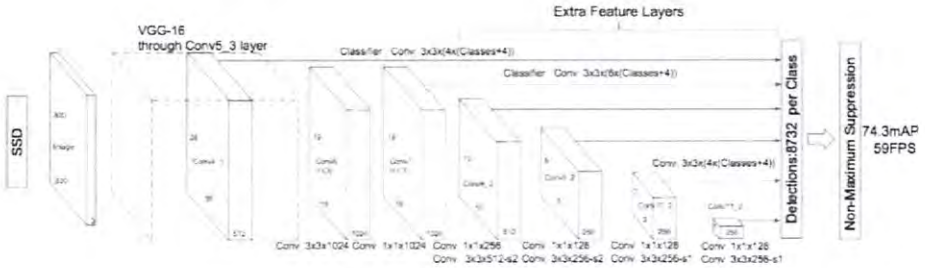


รูป 2.5 โครงสร้างของ Faster R-CNN

2.5 Single Shot multi-Box Detector (SSD)

SSD ถูกออกแบบมาเพื่อทำงานด้าน Object Detection แบบ Real-time เพราะมีขนาดที่เล็ก และมีประสิทธิภาพในระดับหนึ่ง

เหตุที่ทำให้ SSD ใช้เวลาในการประมวลผลน้อย คือ การกำจัด Bounding Box Proposal ออก และสามารถแชร์ Feature ระหว่าง Layer ได้โดยจะใช้ Predictor (filter) ใน Aspect Ratio ที่ แตกต่างกัน เพื่อทำ Feature Map ในหลายๆขนาดแทนการทำ Object Proposal หรือการหาพื้นที่ที่คาดว่าจะมีวัตถุอยู่ออก จะช่วยให้ทำงานกับภาพที่มีวัตถุขนาดต่างกันมากได้



รูป 2.6 โครงสร้างของ Shot multi-Box Detector (SSD)

2.6 TensorFlow

TensorFlow เป็น Open Source Library สำหรับใช้พัฒนา Machine Learning ซึ่งถูกพัฒนาโดย Googleที่ช่วยให้การเทรน และการสร้างโมเดลเป็นเรื่องที่ง่ายขึ้นสามารถทำงานผ่าน CPUs GPUs ได้หลายตัว ซึ่งช่วยลดระยะเวลาในการประมวลผลลงได้

TensorFlow Object Detection API

TensorFlow Object Detection API เป็นฟังก์ชันสำหรับตรวจจับวัตถุภายในภาพที่ถูกเพิ่มเข้ามาบนโปรเจ็ค TensorFlow ที่รองรับเครือข่ายนิวรอนหลายแบบพร้อมกับค่า Weight ที่ฝึกกับชุดข้อมูล COCO มาเรียบร้อยแล้ว (มีชุดข้อมูลอื่นเช่น Open Images ด้วย) ทำให้คนทั่วไปสามารถสร้างระบบตรวจจับวัตถุใช้เองได้ทันที หากต้องการเทรนโมเดลเพิ่มเติมเองก็สามารถทำได้เช่นกัน อีกทั้งยังมีตารางเทียบค่าความเร็ว และค่าความแม่นยำของแต่ละแบบให้ดูอีกด้วย

รูป 2.7 เปรียบเทียบ Pre-Trained Model

โดยผู้จัดทำเลือกใช้ faster_rcnn_inception_v2 สำหรับตรวจจับยานพาหนะ โดยเลือกจากความเร็วในการประมวลผลที่ไม่ช้าจนเกินไป และมีความแม่นยำ

บทที่ 3

การออกแบบและพัฒนา

3.1 ภาพรวมของระบบ

อุปกรณ์ทั้งหมดจะเชื่อมต่อกันผ่าน Wi-Fi Router เพื่อให้อยู่ในเครือข่ายเดียวกัน โดย Raspberry Pi จะทำการถ่ายวิดีโอผ่าน USB Web Camera ที่เชื่อมต่ออยู่ และทำการเชื่อมต่อ FTP Server โดยกรอก Username ,Password ,Host IP หากทั้ง 3 อย่างถูกต้องจะทำการอัพโหลดเฟรมภาพจากวิดีโอขึ้นไป ซึ่งไดเรคทอรีหลักของ FTP Server จะอยู่ในโฟลเดอร์ที่จะอัพขึ้นไปเป็น Web Server โดยทั้ง Web Server และ FTP Server ถูกให้บริการโดย PC ผ่านซอฟต์แวร์ Apache และ Fire Zilla ซึ่งตัวระบบสามารถตั้งค่าลานจอดรถผ่านบน Web Browser ได้ และสามารถรองรับต่อการขยายพื้นที่ลานจอดรถหรือเพิ่มจำนวนกล้องได้โดยง่าย

รูป 3.1 ภาพรวมของระบบ

3.1.1 อุปกรณ์ฮาร์ดแวร์ และหน้าที่การทำงาน

3.1.1.1 Wi-Fi Router

ทำหน้าที่ในการกระจายสัญญาณอินเทอร์เน็ตแบบไร้สาย หรือเชื่อมต่อสายแลนให้กับอุปกรณ์เพื่อให้อยู่ภายในเครือข่ายเดียวกัน ทำให้สามารถติดต่อส่งข้อมูลหากันได้

3.1.1.2 Web Server

ทำหน้าที่ในการกระจายสัญญาณอินเทอร์เน็ตแบบไร้สาย หรือเชื่อมต่อสายแลนให้กับอุปกรณ์ เพื่อให้อยู่ภายในเครือข่ายเดียวกันทำให้สามารถติดต่อส่งข้อมูลหากันได้

3.1.1.3 Raspberry Pi + USB Web Camera

ทำหน้าที่ในการถ่ายภาพและอัพโหลดภาพดังกล่าวไปยัง FTP Server

3.1.1.4 FTP Server

ทำหน้าที่ในการให้บริการ FTP (โปรโตคอลสำหรับถ่ายโอนไฟล์ระหว่างคอมพิวเตอร์) หากมีการเชื่อมต่อจาก Client เข้ามา

3.1.2 อุปกรณ์ฮาร์ดแวร์ และหน้าที่การทำงาน

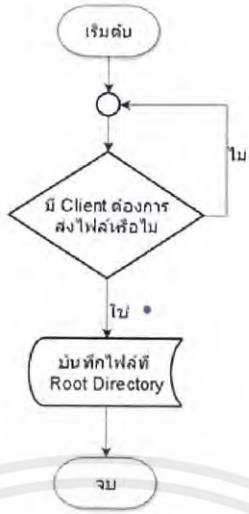
3.1.2.1 Raspberry Pi + USB Web Camera

-ทำหน้าที่ในการถ่ายวิดีโอและส่งเฟรมภาพไปยัง FTP Server

รูป 3.2 Flowchart การทำงาน RaspberryPi+USB Web Camera

3.1.2.2 FTP Server

-ทำหน้าที่ในการให้บริการ FTP แก่ Client ถ้าหากมี Client กรอก IP



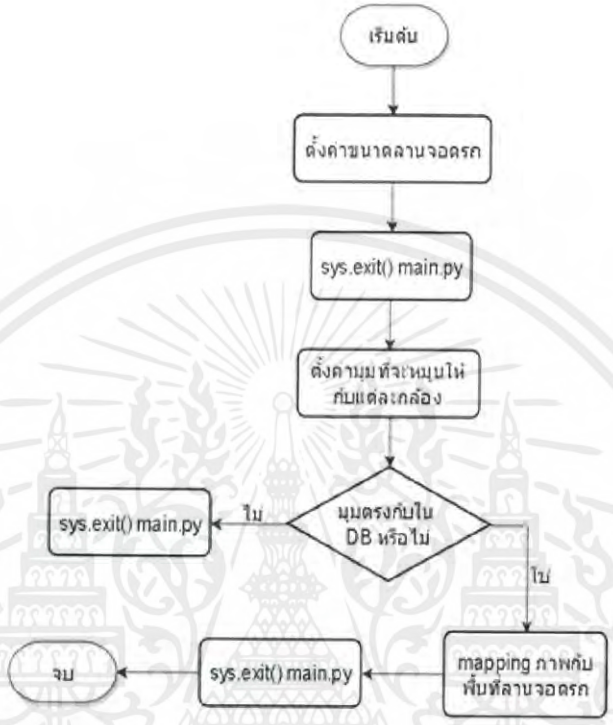
รูป 3.3 Flowchart การทำงาน FTP Server

3.1.2.3 Web Browser แบ่งตามหน้าที่ได้ 3 ประเภทดังนี้

-ทำหน้าที่ในการรันไฟล์ Python และหยุดการรันเมื่อมีการตั้งค่าลานจอดรถใหม่ซึ่งไฟล์ main.py จะเป็นการหาช่องจอดรถไหนว่างบ้าง โดยใช้ภาพที่รับมาจากกล้องในการประมวลผล และนำผลลัพธ์ที่ได้เก็บไว้ในฐานข้อมูล

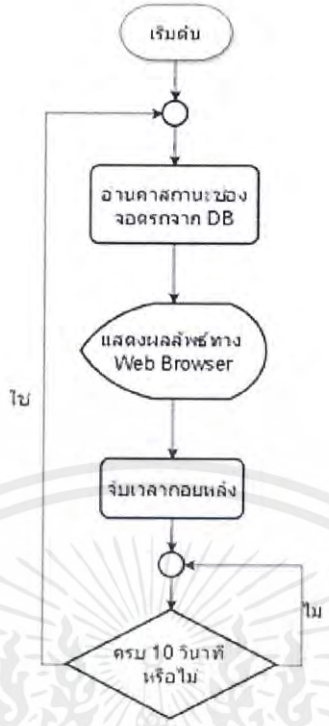
รูป 3.4 Flowchart การทำงาน WebBrowser ส่วนรัน Python

-เป็นหน้า UI สำหรับตั้งค่าลานจอดรถ และ Label ภาพกับช่องจอดรถ สามารถ กำหนดขนาดลานจอดรถโดยกำหนดขนาดแถวและคอลัมน์ กำหนดขนาดองศาที่ต้องการจะหมุนภาพ กำหนด Label ให้พื้นที่ช่องจอดรถ โดยค่าที่ตั้งค่าทั้งหมดจะถูกเก็บไว้ในฐานข้อมูล



รูป 3.5 Flowchart การทำงาน WebBrowser ส่วนตั้งค่าลานจอดรถ

-เป็นหน้า UI สำหรับแสดงสถานะช่องจอดรถ โดยจะดึงค่าสถานะช่องจอดรถมาจากฐานข้อมูล ที่ได้มาจากการประมวลผลภาพ และจะรีเฟรชหน้าทุก 10 วินาที



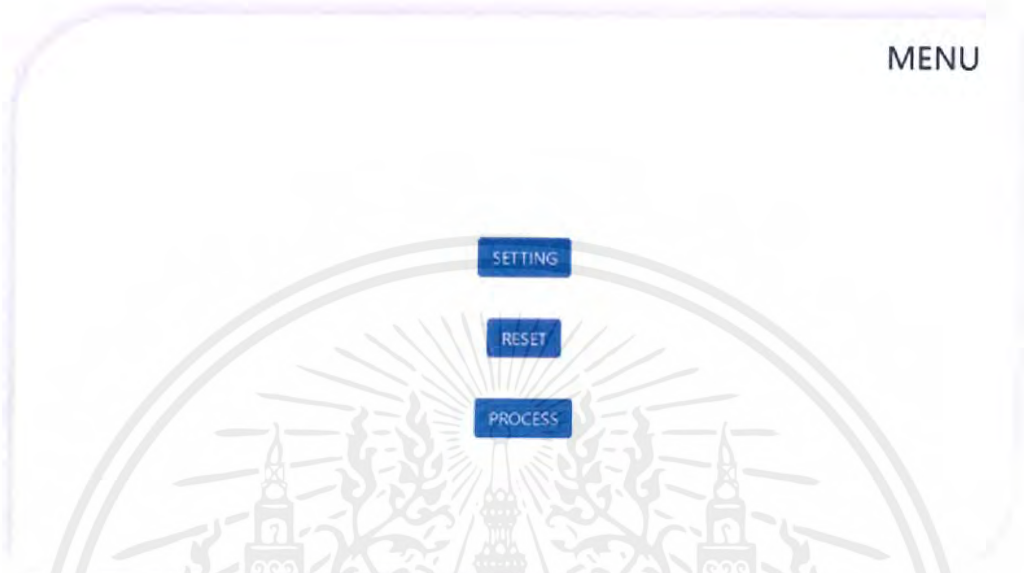
รูป 3.6 Flowchart การทำงาน WebBrowser ส่วนแสดงผลลัพธ์

3.1.3 การรองรับของระบบ

หากต้องการขยายพื้นที่ลานจอดรถสามารถทำได้โดยง่ายเพียงเพิ่มกล้องถ่ายภาพให้ครอบคลุมพื้นที่ที่เพิ่มเติมมาและทำการตั้งค่าขนาดลานจอดรถกำหนดพื้นที่ลานจอดรถกับช่องจอดรถแล้วกดปุ่มรันโปรแกรมอีกครั้งก็สามารถใช้งานได้เลย

3.2 UI

3.2.1 main.php

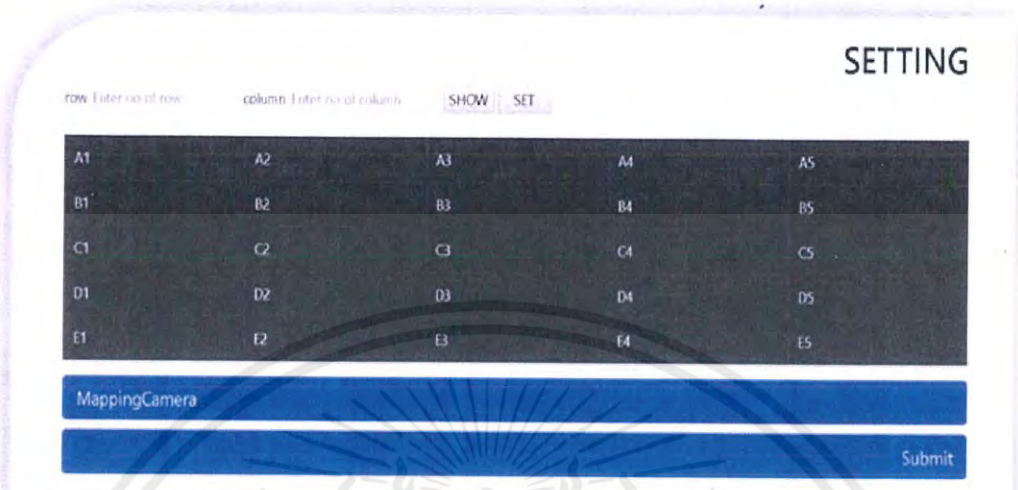


รูป 3.7 หน้าต่าง UI หน้า main.php

ในหน้า main.php จะมี 3 ปุ่มด้วยกัน โดยมีการทำงานดังนี้

1. Setting ถ้ากดปุ่มนี้จะเป็นการไปยังหน้า setting.php
2. Reset ถ้ากดปุ่มนี้จะเป็นการล้างการตั้งค่าทั้งหมดให้กลับมายังค่า Default
3. Process ถ้ากดปุ่มนี้จะเป็นการ รัน Script Python บน Web Server

3.2.2 setting.php

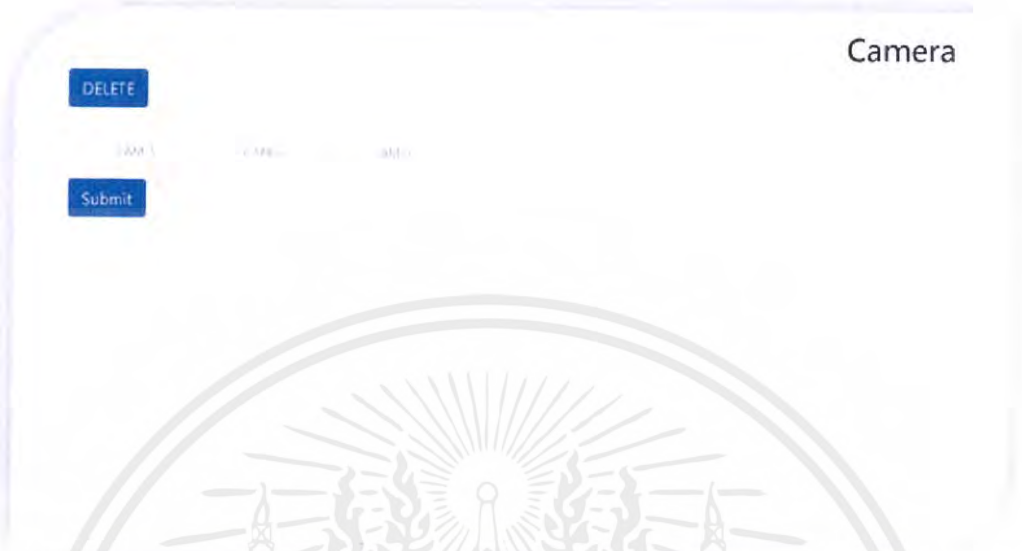


รูป 3.8 หน้าต่าง UI หน้า setting.php

ในหน้า setting.php จะเป็นการตั้งค่าขนาดลานจอดรถ โดยมีการทำงานดังนี้

1. Row ให้กรอกจำนวนแถวของลานจอดรถ (default = 5)
2. Column ให้กรอกจำนวนคอลัมน์ของลานจอดรถ (default = 5)
3. Show ถ้ากดปุ่มนี้จะเป็นการแสดง Table ขนาด Row * Column
4. Set ถ้ากดปุ่มนี้จะเป็นการเก็บค่า Row , Column ไปยังฐานข้อมูล
5. Mapping ถ้ากดปุ่มนี้จะเป็นการไปยังหน้า map.php
6. Submit ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า main.php

3.2.3 map.php



รูป 3.9 หน้าต่าง UI หน้า map.php

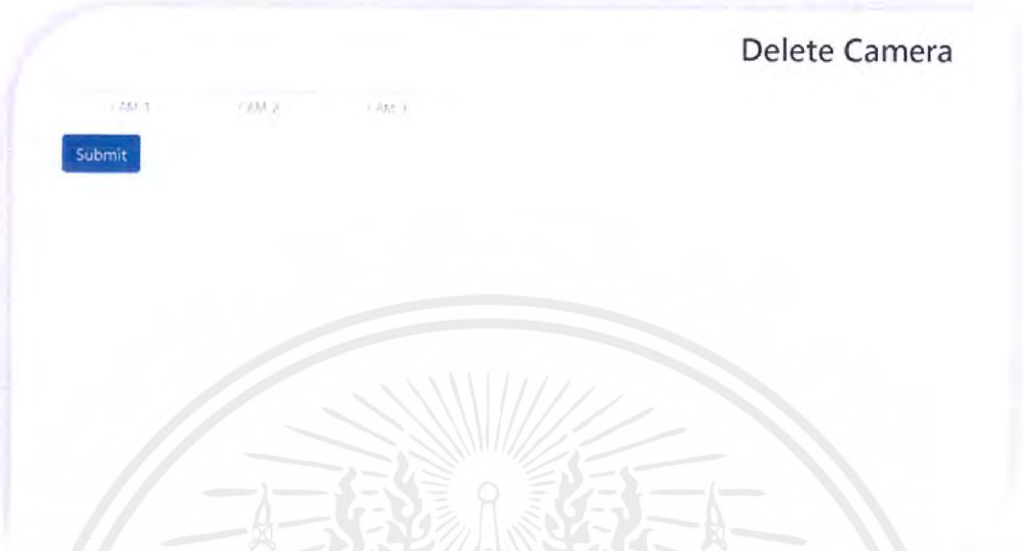
ในหน้า map.php จะเป็นการเลือกกล้องที่ต้องการตั้งค่า โดยมีการทำงานดังนี้

1.Delete ถ้ากดปุ่มนี้จะเป็นการไปยังหน้า del.php

2. Cam_N ถ้ากดปุ่มนี้จะเป็นการไปยังหน้า setimage.php โดยจะส่ง url parameter ตามเลข กล้องไปด้วย (ในส่วนนี้จะเป็นการเพิ่มเองอัตโนมัติ หากมีการเพิ่มกล้อง จำนวนปุ่มจะเพิ่มเองทันที)

3. Submit ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า setting.php

3.2.4 del.php



รูป 3.10 หน้าต่าง UI หน้า del.php

ในหน้า del.php จะเป็นการเลือกกล้องที่ต้องการตั้งค่า โดยมีการทำงานดังนี้

1. Cam_N ถ้ากดปุ่มนี้จะเป็นการลบภาพที่ N ออกและตัดการเชื่อมต่อ FTP กับกล้องที่ N
2. Submit ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า map.php

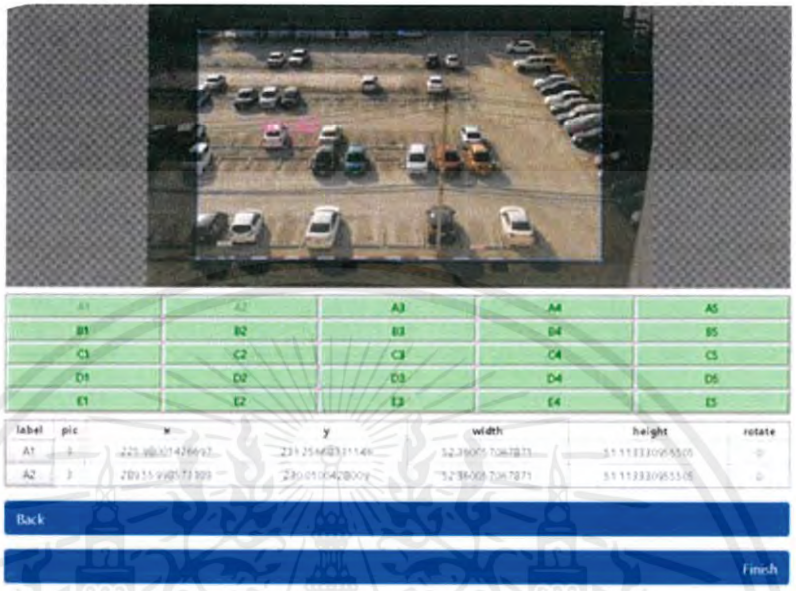
3.2.5 setimage.php

รูป 3.11 หน้าต่าง UI หน้า setimage.php

ในหน้า setimage.php จะเป็นการปรับมุมภาพให้ง่ายต่อการ Image Processing โดยมีการทำงานดังนี้

1. Rotation ให้กรอกเลของศาที่ต้องการจะหมุนภาพ
2. Set ถ้ากดปุ่มนี้จะเป็นการหมุนภาพตามเลของศาที่กรอก
3. Reset ถ้ากดปุ่มนี้จะเป็นการล้างมุมที่หมุนกลับมาเป็นภาพดังเดิม
4. Crop Image ถ้ากดปุ่มนี้จะเป็นการไปยังหน้า crop.php โดยจะส่ง Url Parameter ตามเลขกล้อง และองศาที่ต้องการหมุนไปด้วย
5. Back ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า map.php

3.2.6 crop.php



รูป 3.12 หน้าต่าง UI หน้า crop.php

ในหน้า crop.php จะเป็นการ Mapping ภาพกับช่องจอดรถ ถ้าหากในฐานข้อมูลมีข้อมูลการLabel ของภาพนี้แล้วจะนำมาแสดงให้เห็นด้วย โดยมีการทำงานดังนี้

1. Cropper ให้ลากสี่เหลี่ยมในตำแหน่งที่เราต้องการ
2. Label_Table ถ้ากดปุ่มนี้จะเป็นการเก็บค่า สี่เหลี่ยมที่เราลากโดยใช้ชื่อ Label ตามที่เราคลิกโดยจะเก็บข้อมูลไว้ในฐานข้อมูล
3. Label_Button ถ้ากดปุ่มนี้จะเป็นการลบตำแหน่งที่เราเคย Label ไว้ออกจากฐานข้อมูล
4. Back ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า setimage.php
5. Finish ถ้ากดปุ่มนี้จะเป็นการกลับไปยังหน้า main.php

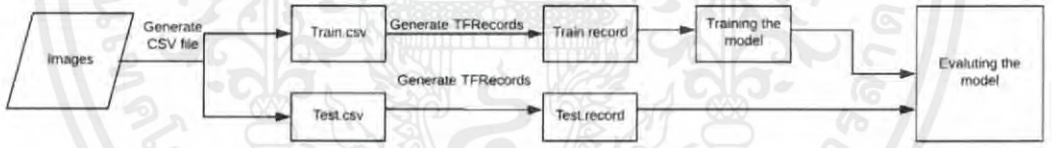
3.3 การพัฒนาโปรแกรมตรวจจับรถ

ตัวโปรแกรมตรวจจับรถเขียนด้วย TensorFlow โดยใช้ TensorFlow Object Detection API ซึ่งใช้ภาษา Python ในการทำ Machine Learning โดยการเลือกตัว Pre-trained Model จะมีไฟล์ config

เบื้องต้นมาให้ (สามารถปรับค่าต่างๆตามความเหมาะสมได้หรือจะใช้ค่าเริ่มต้นก็ได้) มาทำการเทรนด้วยชุดข้อมูลที่มีอยู่ เพื่อใช้ในการ Classification วัตถุที่เราต้องการ ซึ่งในที่นี้คือ รถ

เมื่อเลือกตัว Pre-trained Model ที่จะนำมาใช้ได้แล้ว(faster_rcnn_inception_v2) ต่อมาทำการเตรียม Dataset ที่จะใช้ในการสอนโมเดล โดยแบ่งข้อมูลออกเป็น 2 ส่วนคือ ข้อมูลสำหรับการเทรน (Train) ประมาณ 80% และข้อมูลสำหรับทดสอบ (Test) ประมาณ 20% โดยการเตรียมข้อมูลเพื่อนำไปใช้สอนโมเดลนั้นต้องทำให้อยู่ในรูปแบบที่ใช้กับ TensorFlow เริ่มแรกให้ระบุตำแหน่งของรถใน Dataset ด้วยโปรแกรม LabelImg เป็นการเก็บพิกัดของตำแหน่งที่ระบุในรูปภาพโดยทำกับทุกรูปที่จะใช้ จะได้ผลลัพธ์ออกมาเป็นไฟล์นามสกุล .XML ต่อมาต้องแปลงไฟล์นามสกุล .XML เป็นไฟล์นามสกุล .CSV เพื่อนำไปแปลงต่อให้เป็นไฟล์นามสกุล .record ซึ่งเป็น File Format ของ TensorFlow จึงจะนำไปใช้ได้

เริ่มทำการเทรนโมเดล โดยในระหว่างการเทรนเราสามารถตรวจสอบค่า Loss ต่าง ๆ เช่น Classification Loss, Localization Loss และ Total Loss ที่เป็นค่าความผิดพลาดได้ เพื่อใช้ดูว่าการเทรนโมเดลของเรานั้นเป็นไปในทิศทางใด โดยจะทำการเทรนจนกว่าค่า Total Loss จะเริ่มไม่มีการเปลี่ยนแปลง เมื่อเทรนเสร็จแล้วต่อมาต้องทำการบันทึกตัวโมเดล หรือเรียกว่าการ Freezing Graph เป็นการบันทึกค่าพารามิเตอร์สำคัญที่ถูกปรับค่าจากการเทรน เพื่อนำโมเดลไปใช้ต่อ



รูป 3.13 ภาพรวมการเทรนโมเดล

3.3.1 เตรียม Dataset

Dataset ที่นำมาใช้สอนโมเดลเป็นรูปลานจอดรถข้างโรงอาหารที่ถ่ายบนอาคารเรียนเพื่อให้ได้ภาพในมุมสูงใกล้เคียงกับการนำไปใช้จริง นำรูปไประบุตำแหน่งของรถด้วยโปรแกรม LabelImg โดยใน 1 ภาพของลานจอดรถสามารถแยกภาพรถที่จะนำไปใช้ในการสอนโมเดลได้หลายสิบภาพ ในที่นี้ทางผู้จัดทำได้แบ่งรูปรถในรูปลานจอดรถออกเป็น สำหรับใช้เทรน (Train) 50 รูป และใช้ทดสอบ (Test) อีก 10 รูป (สาเหตุที่ใช้รูปน้อย เพราะทางผู้จัดทำต้องการให้ตัวโมเดลที่สอนขึ้นมาสามารถใช้กับตัวแบบจำลองเดโมได้ และต้องการแสดงให้เห็นว่ามันสามารถทำได้ จึงใช้รูปในจำนวนที่น้อย)

หลังจากทำ LabelImg จะได้ไฟล์ของทั้ง Train และ Test เป็นไฟล์นามสกุล .XML ที่เก็บพิกัดตำแหน่งของรถในรูปลานจอดรถ นำ .XML ไปแปลงเป็นไฟล์นามสกุล .CSV แล้วนำไปแปลงเป็น File Format ของ TensorFlow คือ TFrecord โดยจะได้ไฟล์ Output คือ train.record สำหรับข้อมูลเทรน(Train) และ test.record สำหรับข้อมูลทดสอบ(Test)

รูป 3.14 ตัวอย่างรูปลานจอดรถ

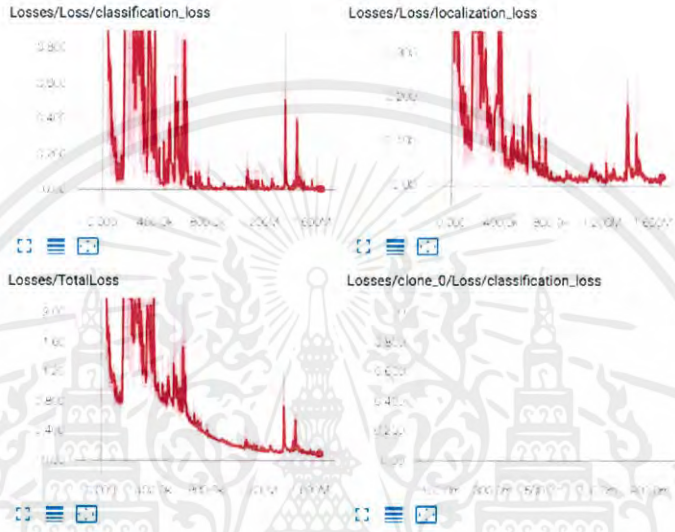
รูป 3.15 โปรแกรม LabelImg

3.3.2 เทรนโมเดล

แก้ไขค่าต่างๆในไฟล์ config สำหรับการเทรนโมเดล ค่าที่ต้องมีการปรับแก้ คือ Batch Size, Learning Rate และ Path ของโฟลเดอร์ Dataset รวมถึงโฟลเดอร์ที่เก็บไฟล์ .record

สร้างไฟล์สำหรับทำ Label Map ซึ่งเป็นไฟล์ที่ระบุคลาสทั้งหมดที่เราได้ทำ Label ไว้ตอนทำด้วย LabelImg สำหรับโครงงานนี้มีเพียงคลาสเดียว คือ car โดยจะเป็นไฟล์นามสกุล .pbtxt

เมื่อทำการแก้ไขไฟล์ config และสร้างไฟล์ Label Map เรียบร้อยแล้วก็สามารถเริ่มเทรนโมเดล และคอยตรวจสอบการเทรนผ่าน Tensorboard เพื่อดูค่า Loss ต่าง ๆ เช่น Classification Loss, Localization Loss และ Total Loss ที่เป็นค่าความผิดพลาดต่างๆ



รูป 3.16 ตัวอย่างค่าที่แสดงบน Tensorboard

3.3.3 บันทึกโมเดล

เมื่อเทรนโมเดลเสร็จเรียบร้อยแล้ว เราจะทำบันทึกโมเดลนั้น หรือที่เรียกว่า Freezing the Graph เพื่อบันทึกค่าน้ำหนักพารามิเตอร์ต่างๆที่โมเดลได้มีการปรับค่าเป็น TensorFlow Graph

3.3.4 นำโมเดลไปใช้งาน

นำไปใช้งานโดยเขียนโปรแกรมให้รับข้อมูลภาพจากกล้อง และนำข้อมูลที่ผู้ใช้งานได้ระบุพิกัดช่องจอดรถที่ต้องการตรวจสอบบนภาพนั้นๆไว้ มาทำงานร่วมกันในการส่งบางส่วนของภาพตามที่ระบุพิกัดไว้เข้าสู่โมเดลเพื่อทำการ Classification หากตรวจไม่พบรถบนส่วนของภาพนั้น โปรแกรมจะทำการบันทึกข้อมูลไปที่ช่องจอดรถนั้นเพื่อบอกว่าช่องจอดรถนั้นว่าง ทำไปจนครบตามที่ผู้ใช้งานระบุไว้ แล้ววนทำซ้ำไปเรื่อยๆจนกว่าจะมีการแก้ไขจากผู้ใช้งาน หรือสั่งหยุดโปรแกรม

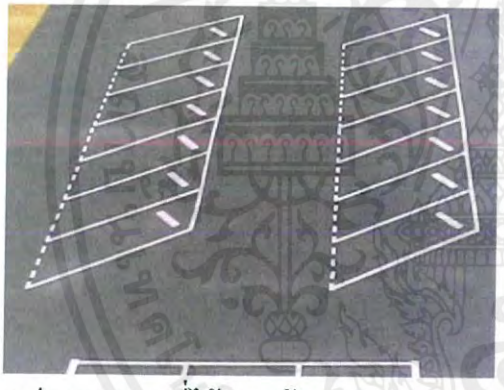
บทที่ 4

การทดลองและผลการทดลอง

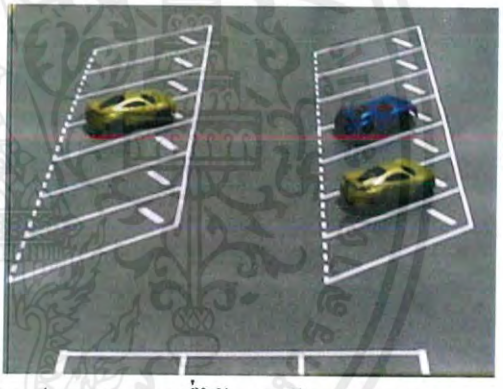
4.1 บันทึกภาพจากกล้อง

บันทึกภาพจากกล้อง USB Camera ที่เชื่อมต่ออยู่กับ Raspberry Pi โดยใช้ cv2.VideoCapture() ซึ่งเป็นฟังก์ชันหนึ่งใน OpenCV Library ตัวฟังก์ชันจะแสดงภาพที่ถ่ายได้แบบ Frame by Frame ตามระยะเวลาที่กำหนด ดังนั้นระหว่างที่ฟังก์ชันทำงานจึงสามารถบันทึกภาพที่แสดงอยู่ขณะนั้นได้ แล้วทำการบันทึกภาพนั้นลงโฟลเดอร์ที่เตรียมไว้เพื่อรอนำไปประมวลผลต่อไป

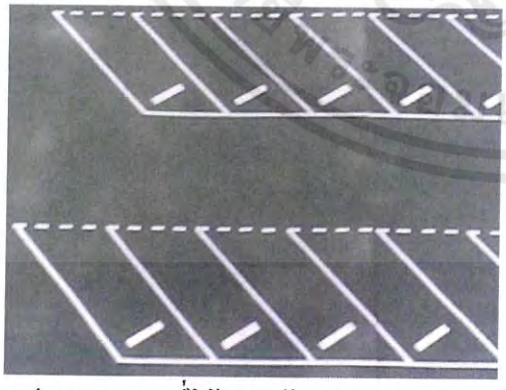
ผลการทดลอง ได้ภาพจากกล้องและสามารถบันทึกลงโฟลเดอร์ได้ ภาพที่ได้มีคุณภาพที่น่าพึงพอใจ (บางภาพไม่ชัดด้วยเหตุผลที่ว่ายังไม่มีตำแหน่งตั้งกล้องที่มั่นคง ถ้าสามารถตั้งกล้องให้นิ่งไม่เคลื่อนไหว จะได้ภาพที่ดีกว่านี้)



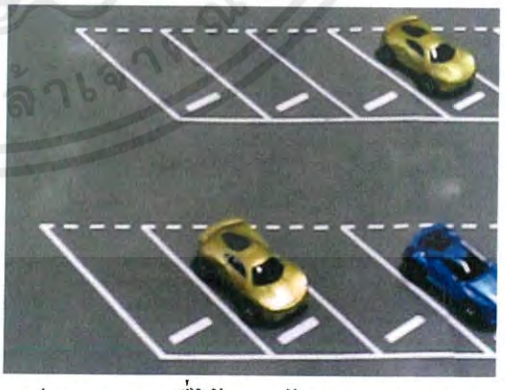
รูป 4.1 ก) ภาพที่ได้จากกล้อง USB Camera



รูป 4.1 ข) ภาพที่ได้จากกล้อง USB Camera



รูป 4.1 ค) ภาพที่ได้จากกล้อง USB Camera



รูป 4.1 ง) ภาพที่ได้จากกล้อง USB Camera

4.2 ส่งไฟล์ภาพผ่าน FTP Server

สิ่งที่ต้องการคือ Raspberry Pi ที่เชื่อมต่อกับกล้อง Megapixel และมี Server ทำหน้าที่ประมวลผล และแสดงผลลัพธ์ออกมา Raspberry Pi จะทำการถ่ายวิดีโอและทำการส่งไฟล์ภาพไปที่ FTP Server แล้วให้ Server ทำการดึงไฟล์ภาพจาก FTP Server มาประมวลผล และแสดงผลลัพธ์

<u>ผลการทดลอง</u> สามารถส่งไฟล์ภาพเข้า FTP Server ได้ และสามารถดึงไฟล์ภาพจาก FTP Server ได้

4.3 ทดลองหาผลลัพธ์ด้วย Machine Learning Model

เลือกใช้ TensorFlow ซึ่งเป็น Open Source Library สำหรับใช้พัฒนา Machine Learning ซึ่ง TensorFlow มีตัว Pre-Trained Model สำหรับทำ Object Detection เบื้องต้นให้อยู่แล้วสามารถนำไปใช้ทดสอบได้เลย

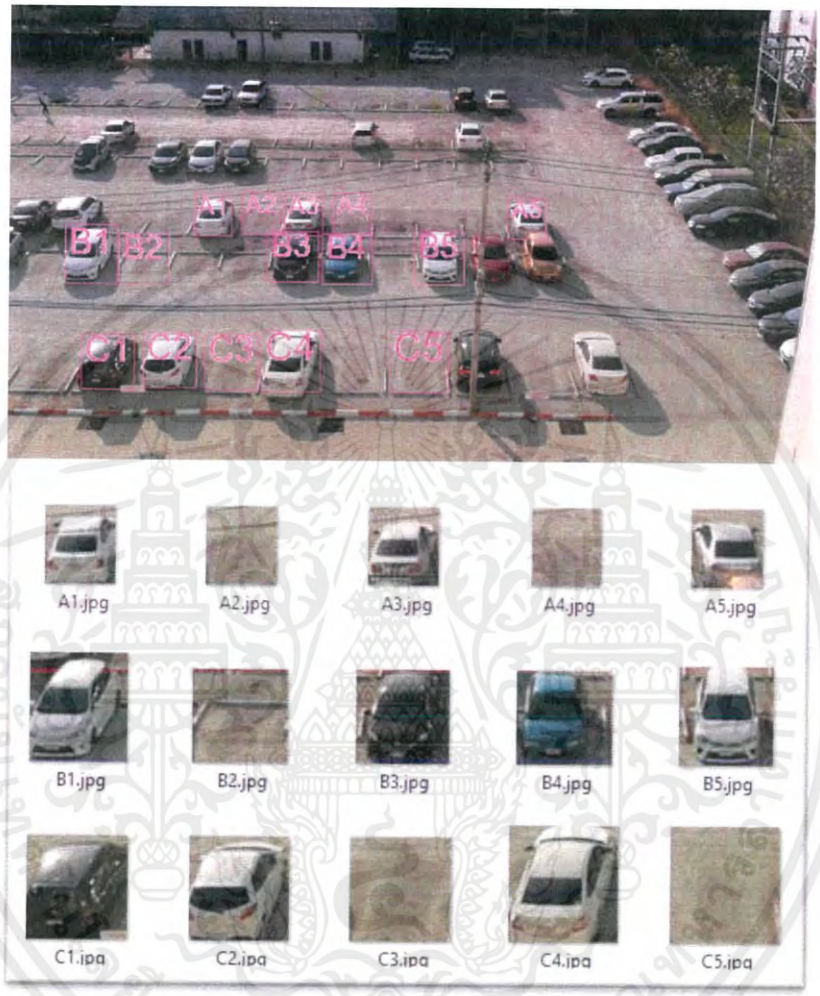
COCO-trained models

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

รูป 4.2 ค่าเปรียบเทียบ Pre-Trained Model

จากรูปที่ 4.2 ทางผู้จัดทำได้เลือกตัวโมเดลในกลุ่มของ SSD เนื่องจากมีความเร็วในการประมวลผลค่อนข้างเร็ว จึงเลือกมา2ตัว เพื่อใช้ในการเปรียบเทียบหาตัวที่ดีกว่า ได้แก่ โมเดลที่

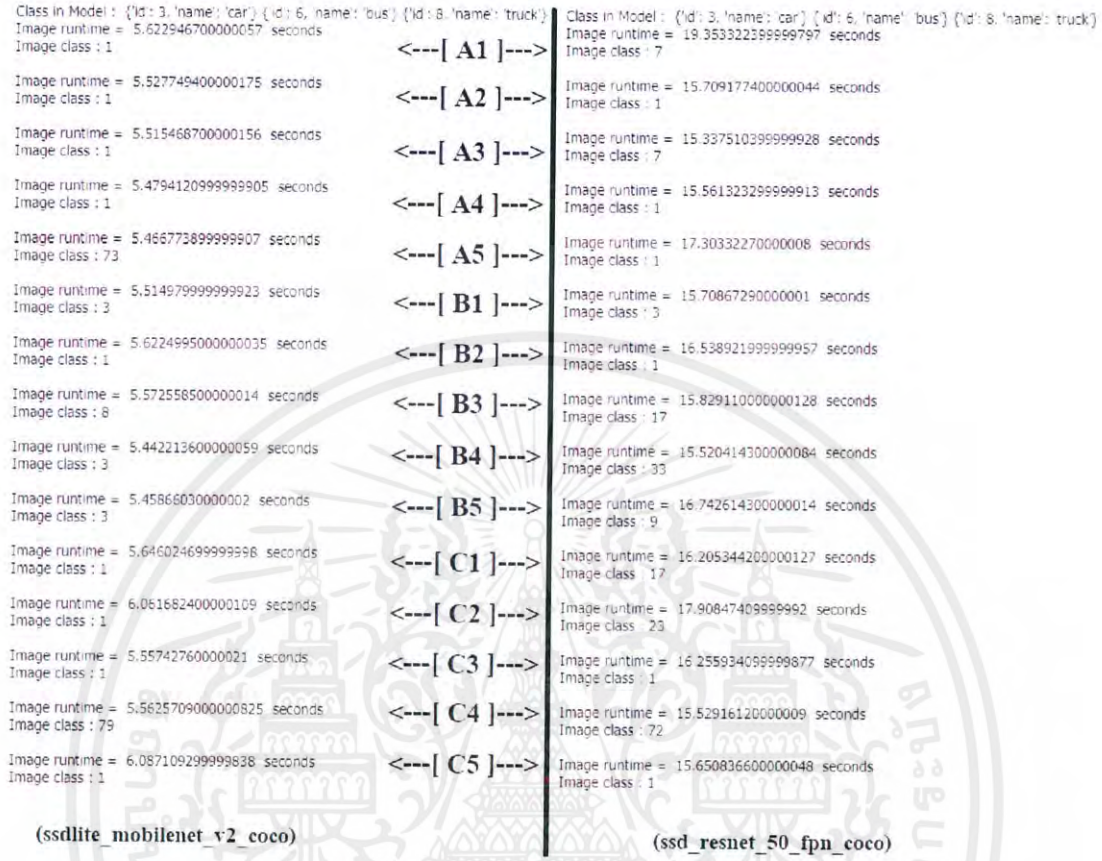
ประมวลผลได้เร็ว แต่ความแม่นยำต่ำ (ssdlite_mobilenet_v2_coco) และโมเดลที่ประมวลผลช้า แต่ความแม่นยำสูง (ssd_resnet_50_fpn_coco) ในการทดลองจะใช้ GPU : Nvidia Geforce GTX660M ในการประมวลผล



รูป 4.3 รูปภาพรถจริงสำหรับใช้ทดสอบโมเดล

รูปภาพที่จะนำไปใช้ทดสอบนำมาจากรูปจริงโดยตัดส่วนที่ต้องการออกมา นำเข้าโมเดลให้โมเดลตรวจจับรถที่อยู่ในรูป

ผลการทดลอง



รูป 4.4 ผลการรันเปรียบเทียบโมเดล SSD

จากผลการรัน ssdlite_mobilenet_v2_coco สามารถตรวจจับรถที่อยู่ในรูปได้ แต่ได้เฉพาะคันที่เห็นด้านหน้ารถเท่านั้น (B1 ,B3 ,B4 ,B5) และใช้เวลาในการประมวลผลต่อรูปอยู่ที่ 5-6 วินาที ในขณะที่ ssd_resnet_50_fpn_coco หาได้เพียงรูปเดียว (B1) และใช้เวลาประมวลผลต่อรูปอยู่ที่ 15-17 วินาที

<u>สรุป</u> ssdlite_mobilenet_v2_coco เหมาะแก่การนำไปใช้งานจริงทั้งในเรื่องของความเร็ว และความแม่นยำที่ยอมรับได้ แต่ต้องทำการเทรนตัวโมเดลเพิ่มเติมเพื่อให้สามารถตรวจจับรถคันที่เห็นด้านหลังรถได้

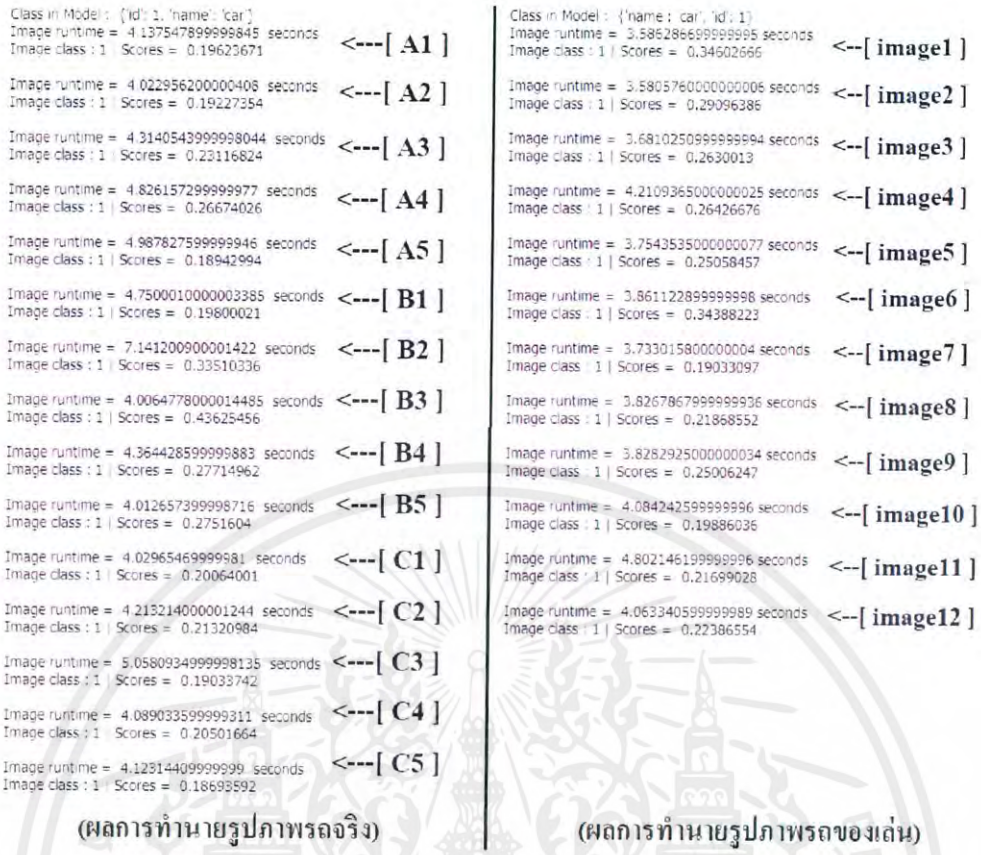
4.4 ทดลองสร้างโมเดล SSD

ไม่สามารถเทรนตัวโมเดล ssdlite_mobilenet_v2_coco ต่อจากของเดิมได้ เนื่องจากข้อมูลที่มีแตกต่างจากข้อมูล COCO ของทาง TensorFlow API ที่ใช้เทรนโมเดลมาก่อนค่อนข้างมาก

ดังนั้นทางผู้จัดทำจำเป็นต้องเทรนตัวโมเดล ssdlite_mobilenet_v2_coco ขึ้นมาใหม่ด้วยข้อมูลเท่าที่มี โดยเป้าหมายที่ต้องการให้โมเดลตรวจจับได้ คือ รูปรถจริงเหมือนกับการทดลองที่ 4.3 และรูปรถของเล่น (ทางผู้จัดทำต้องการให้โมเดลตรวจจับรถของเล่นได้ด้วย เพื่อใช้กับตัวเดโม) โดยใช้ GPU : Nvidia Geforce GTX660M ในการประมวลผล

รูป 4.5 ชุดข้อมูลที่ใช้เทรนโมเดล SSD

รูป 4.6 รถของเล่นสำหรับใช้ทดสอบโมเดล



รูป 4.7 ผลการรันโมเดล ssdlite_mobilenet_v2_coco

เนื่องจากโมเดลที่เทรนขึ้นมาใหม่ มีเพียงคลาสเดียว จึงใช้ค่า Scores ในการตัดสินใจว่ามีความเหมือนรถมากน้อยเท่าไหร่

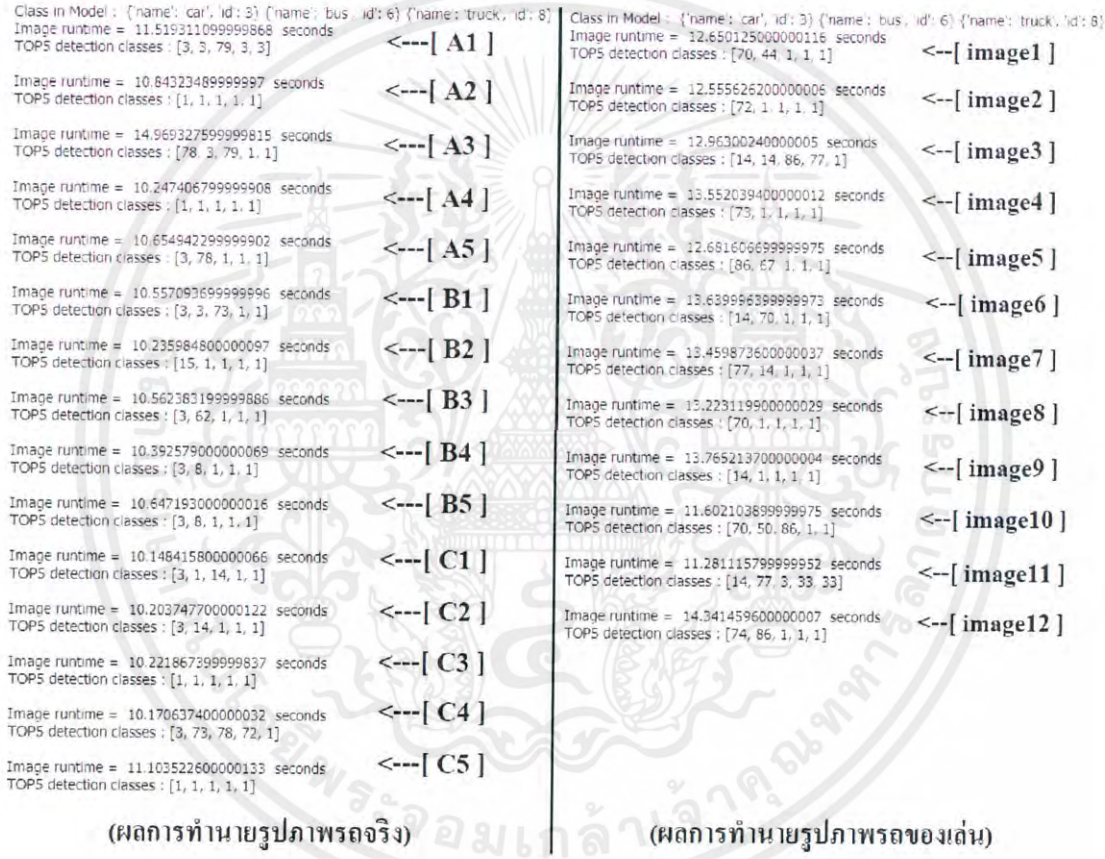
<u>ผลการทดลอง</u> จากการทดสอบตัวโมเดล ssdlite_mobilenet_v2_coco ที่เทรนขึ้นใหม่ จะเห็นได้ว่าทั้งรูปภาพรถจริง และรูปภาพรถของของเล่น ตัวโมเดลไม่สามารถแยกระหว่างมีรถกับไม่มีรถได้ จากรูป 4.7 จะเห็นได้ว่าค่าความเหมือนที่โมเดลทำนายออกมานั้นไม่สัมพันธ์กับรูปภาพเลย

<u>สรุป</u> ssdlite_mobilenet_v2_coco ที่เทรนขึ้นมาใหม่ไม่สามารถนำมาใช้ได้ เนื่องจากชุดข้อมูลที่ใช้เทรนโมเดลมีขนาดของรูปภาพไม่เหมาะสม เพราะตามโครงสร้างของตัวโมเดล SSD จะ Resize รูปภาพที่เข้าไปเทรน ให้มีขนาด 300x300 pixels เป็นอย่างน้อย แต่ชุดข้อมูลที่ทางผู้จัดทำใช้นั้น มีขนาดรูปภาพที่นำเข้าไปเทรนเฉลี่ยอยู่ที่ 150x150 pixels เมื่อถูกขยายเป็น 300x300 pixels ตัวรูปภาพจะสูญเสียรายละเอียดของตัวรูปภาพ ส่งผลให้ตัวโมเดลเรียนรู้ในสิ่งที่ไม่ถูกต้อง และทำนายได้ไม่แม่นยำ

4.5 ทดลองสร้างโมเดล Faster R-CNN

จากการทดลองที่ 4.4 จะเห็นว่าโมเดลจำพวกSSD ไม่สามารถนำมาใช้ได้ เนื่องจากข้อมูลที่มีอยู่นั้นไม่สามารถที่จะใช้เทรนตัวโมเดลSSDขึ้นมาใหม่ได้

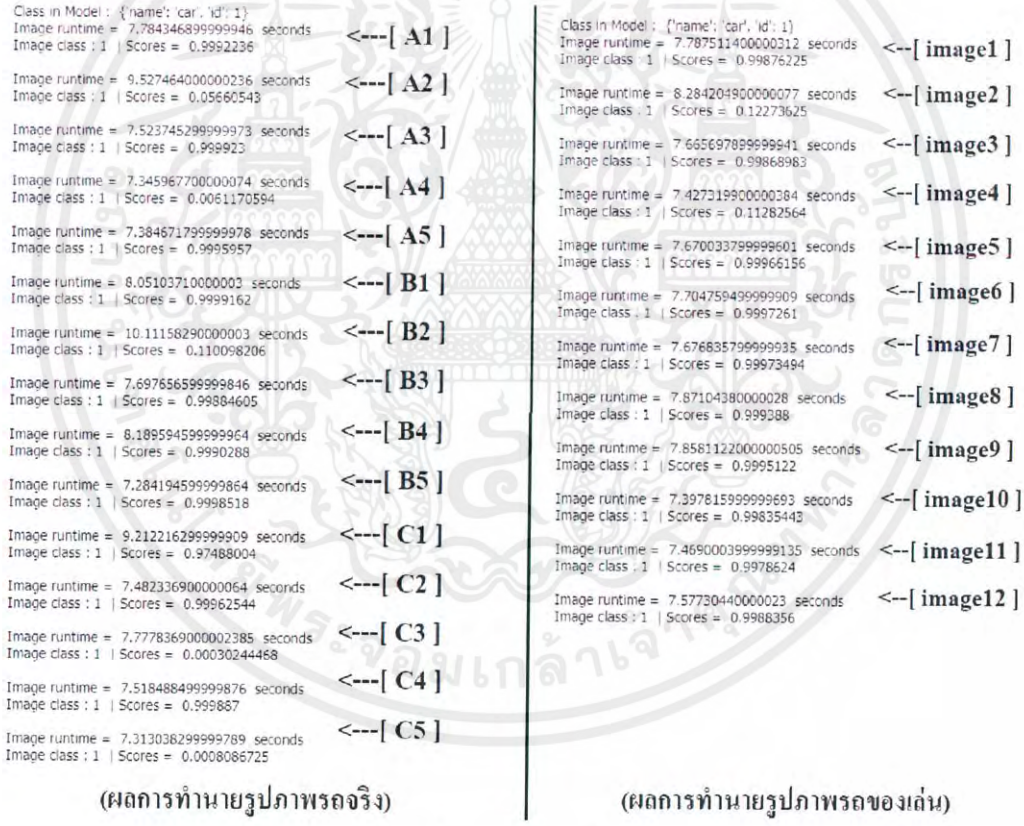
ดังนั้นทางผู้จัดทำจำเป็นต้องต้องหาโมเดลที่มีความเร็วและความแม่นยำใกล้เคียงกัน นั่นคือ faster_rcnn_inception_v2_coco โดยจะทำการทดลองด้วยข้อมูลชุดเดิมเหมือนกับการทดลองที่ 4.3 และข้อมูลที่เป็นรถของเล่นเหมือนกับการทดลองที่ 4.4 โดยใช้ GPU : Nvidia Geforce GTX660M ในการประมวลผล



รูป 4.8 ผลการรันโมเดล faster_rcnn_inception_v2_coco

จากการใช้ faster_rcnn_inception_v2_coco เห็นได้ว่าโมเดลมีความแม่นยำมาก แต่ก็ใช้เวลามากเช่นกัน และไม่สามารถตรวจจับรถของเล่นได้ ดังนั้นจึงต้องทำการเทรนโมเดลเองเพื่อให้สามารถตรวจจับรถของเล่นได้ และลดเวลาประมวลผล โดยการเทรนที่ Steps น้อยๆ

รูป 4.9 ชุดข้อมูลที่ใช้เทรนโมเดล Faster R-CNN



รูป 4.10 ผลการรันโมเดล faster_rcnn_inception_v2

<u>ผลการทดลอง</u> เนื่องจากโมเดลที่เทรนขึ้นมาใหม่ มีเพียงคลาสเดียว จึงใช้ค่า Scores ในการตัดสินใจว่ามีความเหมือนรถมากน้อยเท่าไหร่ จากรูป 4.8 จะเห็นได้ว่าตัวโมเดลที่เทรนขึ้นมาใหม่สามารถตรวจจับรถทั้ง2แบบได้ และใช้เวลาในการประมวลผลน้อยลง

<u>สรุป</u> faster_rcnn_inception_v2 ที่เทรนขึ้นมาใหม่สามารถนำมาใช้ในโครงงานนี้ได้ โดยใช้ช่วงค่า Scores > 0.9 ในการเลือกนำไปประมวลผลต่อไป

4.6 ทดสอบค่าความแม่นยำของโมเดล Faster R-CNN

หาค่าความแม่นยำของตัวโมลเดลในการตรวจจับรถในสภาพที่มีแสงเพียงพอ และในสภาพที่มีแสงน้อย โดยการทำ Confusion Matrix และ Two-class prediction (Positive=ไม่มีรถ ,Negative=มีรถ)

รูป 4.11 ลานจอดรถที่มีสภาพแสงเพียงพอ

จากรูป เลือกช่องจอดรถที่อยู่ใน 4 แถวแรกที่เห็นได้ชัด เป็นจำนวน 160 ภาพ เพื่อใช้ทดสอบโมเดล

ตาราง 4.1 Confusion Matrix ของโมเดล (ใช้รูปภาพที่มีแสงเพียงพอ)

		Predicted	
		ไม่มีรถ	มีรถ
Acture	ไม่มีรถ	54	3
	มีรถ	0	103

จากตารางที่ 4.1 จะได้ TF = 54, TN = 103, FP = 0 และ FN = 3 สามารถคำนวณหาค่าความแม่นยำ (Accuracy) ของตัวโมเดลได้จาก

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{4.1}$$

ค่าความแม่นยำ (Accuracy) ของโมเดลต่อรูปภาพที่มีแสงเพียงพอ จะอยู่ที่ 98.125%

รูป 4.12 ลานจอดรถที่มีสภาพแสงน้อย

เป็นรูปภาพเดิมที่นำไปปรับ Brightness ให้เสมือนเป็นช่วงเวลากลางคืน หรือเวลาที่ไม่มีแสง เลือกช่องจอดรถในตำแหน่งเดิม และจำนวนเท่าเดิม 160 ภาพ เพื่อใช้ทดสอบโมเดล

ตาราง 4.2 Confusion Matrix ของโมเดล (ใช้รูปภาพที่มีแสงน้อย)

		Predicted	
		ไม่มีรถ	มีรถ
Acture	ไม่มีรถ	54	82
	มีรถ	3	21

จากตารางที่ 4.1 จะได้ TF = 54, TN = 103, FP = 0 และ FN = 3 สามารถคำนวณหาค่าความแม่นยำ (Accuracy) ของตัวโมเดลได้จาก

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \tag{4.2}$$

ค่าความแม่นยำ (Accuracy) ของโมเดลต่อรูปภาพที่มีแสงน้อย จะอยู่ที่ 46.875%

<u>ผลการทดลอง</u> ค่าความแม่นยำของโมเดลต่อรูปภาพที่มีแสงเพียงพอ กับรูปภาพที่มีแสงน้อย มีค่าเท่ากับ 98.125% และ 46.875% ตามลำดับ

<u>สรุป</u> โมเดลจะทำงานได้ดีขึ้นอยู่กับสภาพแสง และรายละเอียดของวัตถุในรูปภาพที่ถ่ายได้

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผล

5.1.1 การทำงาน

ทำการเปิด RaspberryPi โดยจะมีการตั้งค่า Start Scipt เอาไว้เพื่อรันไฟล์ที่ใช้ถ่ายวิดีโอและทำการอัพโหลดเฟรมภาพไปยัง FTP Server

ทำการตั้งค่าบน Web Browser เช่น ขนาดลานจอดรถ กำหนดLabelให้ช่องจอดรถ แล้วทำการกดรัน Python สำหรับทำ Image Processing ซึ่งจะเป็น Infinity Loop โดยถ้าหากมีการเปลี่ยนแปลงการตั้งค่าจะหยุดการรันโดยอัตโนมัติ

ทำการเปิดดูผลลัพธ์สถานะที่จอดรถบน Web Browserได้ โดยจะอัพเดททุกๆ 10 วินาทีใช้วิธีการ Refresh หน้าแสดงผล

5.2 ปัญหาและแนวทางการแก้ไข

5.2.1 ปัญหาความแม่นยำในการตรวจจับของโมเดล

เหตุ - เนื่องจาก Dataset ที่ใช้เทรนมีจำนวนน้อย หากมีวัตถุอื่นที่ไม่ใช่ยานพาหนะที่เทรนไว้ ซึ่งมีขนาดพอๆกัน หรือเป็นยานพาหนะคนละประเภท จะส่งผลให้ตัวโมเดลเกิดความผิดพลาดได้

แก้ไข - หา Dataset มาเทรนตัวโมเดลเพิ่ม ยิ่ง Dataset มีมากเท่าไหร่โอกาศเกิดปัญหาก็จะน้อยลงไปเท่านั้น

5.2.2 ปัญหาด้านคุณภาพของภาพ

เหตุ – หากอุปกรณ์ถ่ายภาพมีความละเอียดต่ำเกินไป และนำไปใช้จับพื้นที่กว้าง จะทำให้ได้ภาพที่ความละเอียดต่ำ เมื่อทำการตัดภาพตามพื้นที่ที่ตั้งค่าไว้ แล้วนำเข้าโมเดล จะส่งผลให้ตัวโมเดลเกิดความผิดพลาดได้

แก้ไข – หากต้องการจับพื้นที่กว้างควรใช้อุปกรณ์จับภาพที่มีคุณภาพสูง หรือจับพื้นที่ให้มีขนาดพอเหมาะกับคุณภาพของอุปกรณ์

5.3 แนวทางการพัฒนาต่อ

พัฒนาระบบให้จำแนกยานพาหนะได้มากขึ้น เพื่อความหลากหลายของการใช้ง