
รายงานการวิจัยฉบับสมบูรณ์

เรื่อง

# การประยุกต์เทคโนโลยีภูมิสารสนเทศกับปัญญาประดิษฐ์พัฒนาระบบตรวจจับจำแนกวัตถุเชิงพื้นที่สำหรับตรวจสอบการจราจร บริเวณเทศบาลเมืองแสนสุข

# Application of geoinformatics technology and artificial intelligence to develop a spatial object classification and detection system for traffic inspection in the Saensuk municipality area.

กฤษณะ อิ่มสวาสดิ์

งานวิจัยนี้ได้รับทุนสนับสนุนจากงบประมาณเงินรายได้ ประจำปีงบประมาณ 2566

คณะภูมิสารสนเทศศาสตร์ มหาวิทยาลัยบูรพา

กันยายน 2567

การวิจัยนี้ได้รับทุนสนับสนุนจากงบประมาณเงินรายได้

คณะภูมิสารสนเทศศาสตร์ มหาวิทยาลัยบูรพา

ประจำปีงบประมาณ พ.ศ.2566

# บทคัดย่อ

การพัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้องวงจรปิด โดยใช้เทคนิคการเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชันและสร้างแบบจำลองแผนที่การจราจรบนถนน ในเขตเทศบาลเมืองแสนสุข จังหวัดชลบุรี ปัญญาประดิษฐ์มีบทบาทสำคัญในระบบขนส่งอัจฉริยะ (ITS) และการเฝ้าระวังการจราจรที่มีอัตราการเติบโตของยานพาหนะเพิ่มขึ้นอย่างรวดเร็ว โครงสร้างพื้นฐานการเฝ้าระวังการตรวจจับจากกล้องวงจรปิด ในวิจัยนี้นำโครงข่ายประสาทเทียมเชิงลึก YOLOV8 มาใช้วิเคราะห์การเรียนรู้ประเภทยานพาหนะทั้ง 12 ประเภท การจำแนกได้เปรียบเทียบสัดส่วนค่าความถูกต้องของการจำแนกภาพจากการคาดการณ์และภาพจริง Confusion Matrix ของการจำแนกยานพาหนะมีค่าความถูกต้องอยู่ในช่วงระหว่าง 0.68 –1 โดยประเภทรถอเนกประสงค์ประเภทครอบครัว (SUV), รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ (PPV) และรถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 คน (MPV) มีค่าระหว่าง 0.68 - 0.78 หรือประมาณ 68 – 78 เปอร์เซ็นต์ ส่วนรถมอเตอร์ไซด์พ่วงข้าง (Sidecar), รถบรรทุก (truck), รถกระบะ (pick-up), รถยนต์นั่งส่วนบุคคล 4 ที่นั่ง (sedan), รถมอเตอร์ไซด์ (Motorcycle), รถตู้ (van) และ รถประจำทาง (Bus) มีค่าในช่วง 0.87 – 0.90 หรือประมาณ 87 – 90 เปอร์เซ็นต์ ประเภทรถกระบะขนส่งตู้ทึบ (Solidbox pick-up) และรถสองแถว (Songthaew) มีค่าระหว่าง 0.91 – 1.00 หรือประมาณ 91 – 100 เปอร์เซ็นต์ โดยการจำแนกประเภทยานพาหนะมีระดับความแม่นยำเฉลี่ยประมาณ 80 เปอร์เซ็นต์ ค่า mAP50 เฉลี่ยรวมทุกประเภทยานพาหนะอยู่ที่ประมาณ 86 เปอร์เซ็นต์ และค่า mAP50-95 เฉลี่ยรวมทุกประเภทยานพาหนะอยู่ที่ประมาณ 70 เปอร์เซ็นต์ การสร้างแพลตฟอร์มสำหรับการจำแนกและติดตามยานพาหนะบนถนนมีรูปแบบที่แตกต่างกัน เพื่อทดสอบการจำแนกความถูกต้องแม่นยำของแบบจำลองการจำแนกยานพาหนะ เพื่อติดตามนับจำนวนยานพาหนะบนการจราจรแบบตามเวลาจริงจากกล้องวงจรปิด และพัฒนาวิธีการทำแผนที่มุมมองย้อนกลับแบบผสมผสานจากภาพถ่ายจากดาวเทียมและภาพจากกล้องวงจรปิด เพื่อพัฒนาแผนที่แบบจำลองการจราจรบนถนนบูรณาการเทคโนโลยีปัญญาประดิษฐ์สำหรับฝึกสอนข้อมูลให้สามารถจดจำ เรียนรู้ประเภทของยานพาหนะเพื่อสร้างแบบจำลองการตรวจจับ การจำแนกและประยุกต์บนข้อมูลภูมิสารสนเทศในรูปแบบแผนที่ โดยค่าความถูกต้องของการจำแนกและค่าความแม่นยำของตำแหน่งยานพาหนะมีค่าเฉลี่ย 85 – 90 เปอร์เซ็นต์ ของแผนที่แบบจำลองการจราจรขึ้นอยู่กับมุมมองกล้องวงจรปิด ความสูง รายละเอียดภาพ และ การวางจุดอ้างอิงบนภาพ

คำสำคัญ : ตรวจจับจำนวนยานพาหนะ, การนับจำนวนยานพาหนะ, การเฝ้าระวังการจราจร, แบบจำลองแผนที่การจราจรบนถนน, โครงข่ายประสาทเทียมเชิงลึก, คอนโวลูชั่น

# Abstract

Development of a Vehicle Detection, Counting, and Classification System from CCTV Using Deep Convolutional Neural Networks and a replica of the traffic map in Saensuk Municipality, Chon Buri province, Thailand. Artificial intelligence plays a crucial role in Intelligent Transportation Systems (ITS) and traffic surveillance amidst rapidly growing vehicle populations. This research employs the YOLOV8 deep neural network to analyze and classify 12 vehicle types from CCTV footage. Classification accuracy was evaluated using confusion matrices, with accuracy rates ranging from 0.68 to 1.00. SUVs, PPVs, and MPVs showed accuracies between 68-78%, while sidecars, trucks, pick-ups, sedans, motorcycles, vans, and buses achieved 87-90% accuracy. Solidbox pick-ups and songthaews demonstrated the highest accuracy at 91-100%. Overall classification accuracy averaged 80%, with mAP50 at 86% and mAP50-95 at 70% across all vehicle types. The study developed various platforms for real-time vehicle classification, tracking, and counting from CCTV feeds. It also created an integrated reverse-view mapping method combining satellite imagery and CCTV footage to produce an AI-enhanced replica of the traffic map. The system's ability to detect, classify, and apply geospatial data to mapping achieved an average accuracy of 85-90% for vehicle classification and position precision, with performance dependent on CCTV viewpoint, height, image resolution, and reference point placement.

Keywords: Vehicle detection, Vehicle counting, Traffic surveillance,
replica of the traffic map, Deep neural networks, Convolution

# ประกาศคุณูปการ

งานวิจัยนี้ได้รับทุนสนับสนุนการวิจัยจากงบประมาณเงินรายได้ คณะภูมิสารสนเทศศาสตร์ มหาวิทยาลัยบูรพา ประจำปีงบประมาณ พ.ศ. 2566 ซึ่งคณะกรรมการส่งเสริมการวิจัยและบริการวิชาการประจำคณะภูมิสารสนเทศศาสตร์ ได้สนับสนุนให้ดำเนินการจัดทำวิจัยเรื่อง “การประยุกต์เทคโนโลยีภูมิสารสนเทศกับปัญญาประดิษฐ์พัฒนาระบบตรวจจับจำแนกวัตถุเชิงพื้นที่สำหรับตรวจสอบการจราจร บริเวณเทศบาลเมืองแสนสุข” เลขที่สัญญา 1/2566 โดยใช้เทคโนโลยีภูมิสารสนเทศบูรณาการกับเทคโนโลยีปัญญาประดิษฐ์ในการพัฒนาระบบติดตามและจำแนกของยานพาหนะต้นแบบ รองรับเมืองแห่ง smart city และเป็นส่วนประกอบหนึ่งของการก้าวเข้าสู่ Digital twins ในด้านการจราจร ประกอบกับเทศบาลเมืองแสนสุขเป็นเมืองท่องเที่ยวที่ได้รับความนิยมสูงจากนักท่องเที่ยวทั้งชาวไทยและต่างประเทศ ทางผู้วิจัยจึงยกประเด็นของการจราจรที่เพิ่มขึ้นทุก ๆ ปี โดยเฉพาะอย่างยิ่งการสนับสนุนจากเจ้าหน้าที่ของเทศบาลเมืองแสนสุขที่ให้ความร่วมมือด้านข้อมูลกล้องวงจรปิด การตรวจสอบการทำงานของกล้องวงจรปิดให้สมบูรณ์ และทำให้งานวิจัยเรื่องดังกล่าวสำเร็จลุล่วงเป็นอย่างดี ผู้วิจัยรู้สึกซาบซึ้งเป็นอย่างยิ่ง จึงขอขอบพระคุณไว้ ณ โอกาสนี้ที่กรุณาสนับสนุนงานวิจัยและให้คำแนะนำในแนวทางที่ถูกต้อง ตลอดจนแก้ไขข้อบกพร่องต่าง ๆ

ผู้วิจัยขอกราบระลึกถึงพระคุณของคุณครูบาอาจารย์ ที่ได้อบรมสั่งสอน และประสาทวิชาความรู้แก่ผู้วิจัย ซึ่งผู้วิจัยจะได้แสวงหาความรู้เพื่อเป็นตัวอย่างแก่ลูกศิษย์ บุตร ธิดาและคนรุ่นหลังต่อไป นอกจากนี้ขอขอบคุณ คุณธนิษตา ไทยเกิดศรี (ภรรยา) ที่ประสานงาน สนับสนุน และเป็นกำลังใจให้ความสำเร็จอันเกิดจากการศึกษาวิจัยนี้ รวมถึงบุคคลท่านอื่นที่ไม่ได้กล่าวถึงในการช่วยเหลือให้งานวิจัยฉบับนี้สำเร็จ ขอมอบเป็นสิ่งทดแทนคุณ และขอกราบขอบพระคุณมา ณ ที่นี้เป็นอย่างสูง

กฤษณะ อิ่มสวาสดิ์

กันยายน 2567

# สารบัญ

# สารบัญ (ต่อ)

# สารบัญตาราง

**ตารางที่** **หน้า**

# สารบัญภาพ

**ภาพที่** **หน้า**

# สารบัญภาพ (ต่อ)

**ภาพที่** **หน้า**

# บทที่ 1

# บทนำ

## ความสำคัญและที่มาของปัญหา

เทคโนโลยีด้านภูมิสารสนเทศได้มีส่วนสำคัญในทุกกิจกรรมของมนุษย์ตั้งแต่การวางแผน ติดตามผลผลิตทางการเกษตร ความมั่นคงทางอาหาร จัดการสิ่งแวดล้อม ติดตามสถานการณ์การเปลี่ยนแปลงภูมิอากาศ การจัดการภัยพิบัติทางธรรมชาติ การวางแผนระบบการขนส่ง และการวิเคราะห์อุบัติเหตุจากการจราจรในเขตพื้นที่เมือง ซึ่งสามารถวิเคราะห์เป็นข้อมูลเชิงพื้นที่และตัดสินใจวางแผนได้อย่างมีประสิทธิภาพ อีกทั้งระบบการตรวจวัดจากเซ็นเซอร์ ทำให้เกิดข้อมูลเชิงพื้นที่ขนาดใหญ่ (GeoSpatial Big Data) ข้อมูลปริมาณมหาศาล กำลังผลักดันความก้าวหน้าของโครงสร้างพื้นฐานการคมนาคมและกลไกของเมืองอัจฉริยะ เช่น แนวโน้มที่ประชาชนสามารถใช้ระบบขนส่งที่หลากหลายเดินทางเข้าออกและระหว่างสถานที่ต่างๆ ในเขตเมือง ข้อมูลเก็บ รวบรวมจากเซ็นเซอร์ที่เพิ่มจำนวนมากขึ้นและจุดรับข้อมูลอื่นๆ ผสมผสานกับเทคโนโลยี GPS สามารถจัดทำเป็นแผนที่ตาม เวลาจริง (Real-time) เพื่อช่วยจัดการการสัญจรภายในเมืองให้คล่องตัวยิ่งขึ้น (สำนักงานส่งเสริมเศรษฐกิจดิจิทัล, 2565)

ประเทศจีนเป็นแนวหน้าในการนำปัญญาประดิษฐ์มาใช้ในแผนงานและการจัดการเมือง โดยเมืองหังโจว (Hangzhou) พัฒนาเทคโนโลยีที่ชื่อว่า ซิตี้เบรน (City Brain) เพื่อใช้ในการเก็บข้อมูลจากเซ็นเซอร์และกล้องวงจรปิดมา ประมวลผล นอกจากนี้ เมืองซูโจว (Suzhou) และประเทศอื่นๆ ในอาเซียนนำปัญญาประดิษฐ์มาใช้ในการจัดการจราจร (สำนักงานส่งเสริมเศรษฐกิจดิจิทัล, 2565)

ปัจจุบันเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence: AI) เป็นเทคโนโลยีที่ได้รับความนิยมและถูกพัฒนาอย่างต่อเนื่องในการสอนให้มีความสามรถทางสติปัญญาและการเรียนรู้เหมือนมนุษย์ คือ การเรียนรู้ของเครื่องจักร (Machine Learning: ML) หมายถึง ศาสตร์ที่ทำให้คอมพิวเตอร์หรือเครื่องจักรสามารถเรียนรู้ที่จะทำความเข้าใจความสัมพันธ์ของข้อมูลที่ถูกป้อนเข้า (Input) และสร้างผลลัพธ์การตอบสนองต่อข้อมูล (Output) ขึ้นมาได้เองโดยไม่ต้องถูกโปรแกรมหรือได้รับการป้อนคำสั่งเข้าไปใหม่ทุกครั้งที่คอมพิวเตอร์หรือเครื่องจักรได้รับข้อมูลใหม่เป็นการนำศาสตร์ด้านคณิตศาสตร์และสถิติขั้นสูงมาประยุกต์เข้ากับความรู้ด้านการจัดการข้อมูล และการเขียนโปรแกรม โดยมีหลักการ คือ การสร้างองค์ความรู้ในเชิงโมเดลทางคณิตศาสตร์จากข้อมูลป้อนเข้าด้วยตัวเครื่องจักร โดยโมเดลที่ถูกสร้างขึ้นมีความยืดหยุ่นและสามารถที่จะปรับตัวเองเข้ากับข้อมูลใหม่ ๆ ที่ได้รับป้อนเข้าไป ดังนั้น การเรียนรู้ของเครื่องจักร จึงเปรียบเสมือนความคิดระบบหนึ่งจากหลาย ๆ ระบบที่อยู่ในสมองของ AI (สำนักงานส่งเสริมเศรษฐกิจดิจิทัล, 2564)

การผสมผสานระหว่างภูมิศาสตร์และปัญญาประดิษฐ์ (GeoAI) สำหรับการจัดการปัญหาภูมิสารสนเทศที่หลากหลายในสภาพแวดล้อมทางธรรมชาติและสังคมมนุษย์เกิดขึ้นจากการรวมนวัตกรรมในวิทยาศาสตร์เชิงพื้นที่เข้ากับการเติบโตอย่างรวดเร็วของวิธีการในปัญญาประดิษฐ์ (AI) โดยเฉพาะอย่างยิ่งการเรียนรู้ของเครื่อง เช่น การเรียนรู้เชิงลึก การทำเหมืองข้อมูล และการคำนวณประสิทธิภาพเพื่อรวบรวมข้อมูลที่มีความหมายจากข้อมูลขนาดใหญ่เชิงพื้นที่ เป็นสหวิทยาการสูง โดยเชื่อมโยงสาขาวิทยาศาสตร์มากมาย เช่น ภูมิศาสตร์ วิทยาการคอมพิวเตอร์ วิศวกรรม สถิติ และวิทยาศาสตร์เชิงพื้นที่ เพื่อนำไปวิเคราะห์การจำแนกภาพ การตรวจจับวัตถุ การแบ่งส่วนฉาก การจำลองและการแก้ไข การดึงข้อมูลและการตอบคำถาม การรวมข้อมูลแบบทันทีทันใด และการเพิ่มมูลค่าทางภูมิศาสตร์ (University of Florida, 2022)

เทศบาลเมืองแสนสุขเป็นอีกเมืองที่ส่งเสริมเป็นเมืองอัจฉริยะ (Smart City) ที่เรียกว่า แสนสุขสมาร์ทซิตี้ โดยมุ่งเน้นเป็นต้นแบบเมืองสุขภาพดี ใช้เทคโนโลยีดูแลประชาชน และเป็นประเภทเมืองอัจฉริยะด้าน Smart Environment Smart Governance และ Smart Living โดยการขับเคลื่อนเมืองอัจฉริยะด้วยการสร้างแพลตฟอร์มข้อมูลเมืองอัฉริยะ (City data platform) เพื่อรองรับและสนับสนุนข้อมูลของเมือง เช่น ข้อมูลกล้องวงจรปิด CCTV ที่สามารถเข้าถึงได้ทุกเวลา

บางแสนเป็นแหล่งท่องเที่ยวที่ได้รับความนิยมอย่างมากที่อยู่ในพื้นที่เทศบาลเมืองแสนสุข จากสถิติการท่องเที่ยวบางแสน ปี 2562 มีจำนวนนักท่องเที่ยวทั้งชาวไทยและชาวต่างชาติทั้งหมด 5,740,108 คน เมื่อเทียบกับนักท่องเที่ยวปี 2561 มีจำนวนนักท่องเที่ยวทั้งชาวไทยและชาวต่างชาติทั้งหมด 5,692,678 คน ซึ่งมีอัตราการเพิ่มขึ้นของนักท่องเที่ยวประมาณ 47,430 คน ข้อมูลสถิติจาก (เทศบาลเมืองแสนสุข, 2562) เป็นดัชนีชี้วัดของปัญหาเรื่องการจราจรที่คับคั่งเพิ่มมากขึ้นโดยเฉพาะวันหยุด ซึ่งการตรวจสอบจำนวนปริมาณยานพาหนะในปัจจุบันนี้ใช้อากาศยานไร้คนขับ (Drone) เพื่อติดตามปริมาณการจราจรสะสม (ผู้จัดการออนไลน์, 2564) แต่ปัญหาของอากาศยานไร้คนขับคือเรื่องของพลังงานส่งผลต่อพิสัยและระยะการบินที่ไม่เพียงพอกับการเฝ้าติดตามอย่างต่อเนื่อง ดังนั้นโครงการวิจัยนี้มีแนวคิดการนำข้อมูลภาพเคลื่อนไหวจากกล้อง CCTV ในเขตเทศบาลเมืองแสนสุขมาจำแนกแยกประเภท ยานพาหนะ โดยนำเทคนิคการเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) ตรวจนับจำนวนยานพาหนะแยกแต่ละประเภทโดยใช้ OpenCV จากกล้องวงจรปิด CCTV ได้

## วัตถุประสงค์ของการวิจัย

พัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV ในเขตเทศบาลเมืองแสนสุข โดยใช้เทคนิคการเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) และสร้างแบบจำลองแผนที่การจราจรบนถนน (Digital replica of the road mapping)

## ประโยชน์ที่คาดว่าจะได้รับ

สามารถตรวจจับและแยกประเภทยานพาหนะในเขตเทศบาลเมืองแสนสุขแบบทันทีทันใดจากกล้องวงจรปิด (CCTV)

ได้แผนที่จำลองการจราจรบนถนนบริเวณเทศบาลเมืองแสนสุข

สามารถสนับสนุนข้อมูลของงานวิจัยให้กับหน่วยงานเทศบาลเมืองแสนสุขและสภ.เมืองแสนสุขเพื่อประกอบการตัดสินใจ

## ขอบเขตของการวิจัย

ขอบเขตของการวิจัยสามารถแบ่งออกได้ดังนี้

ขอบเขตเชิงเนื้อหา

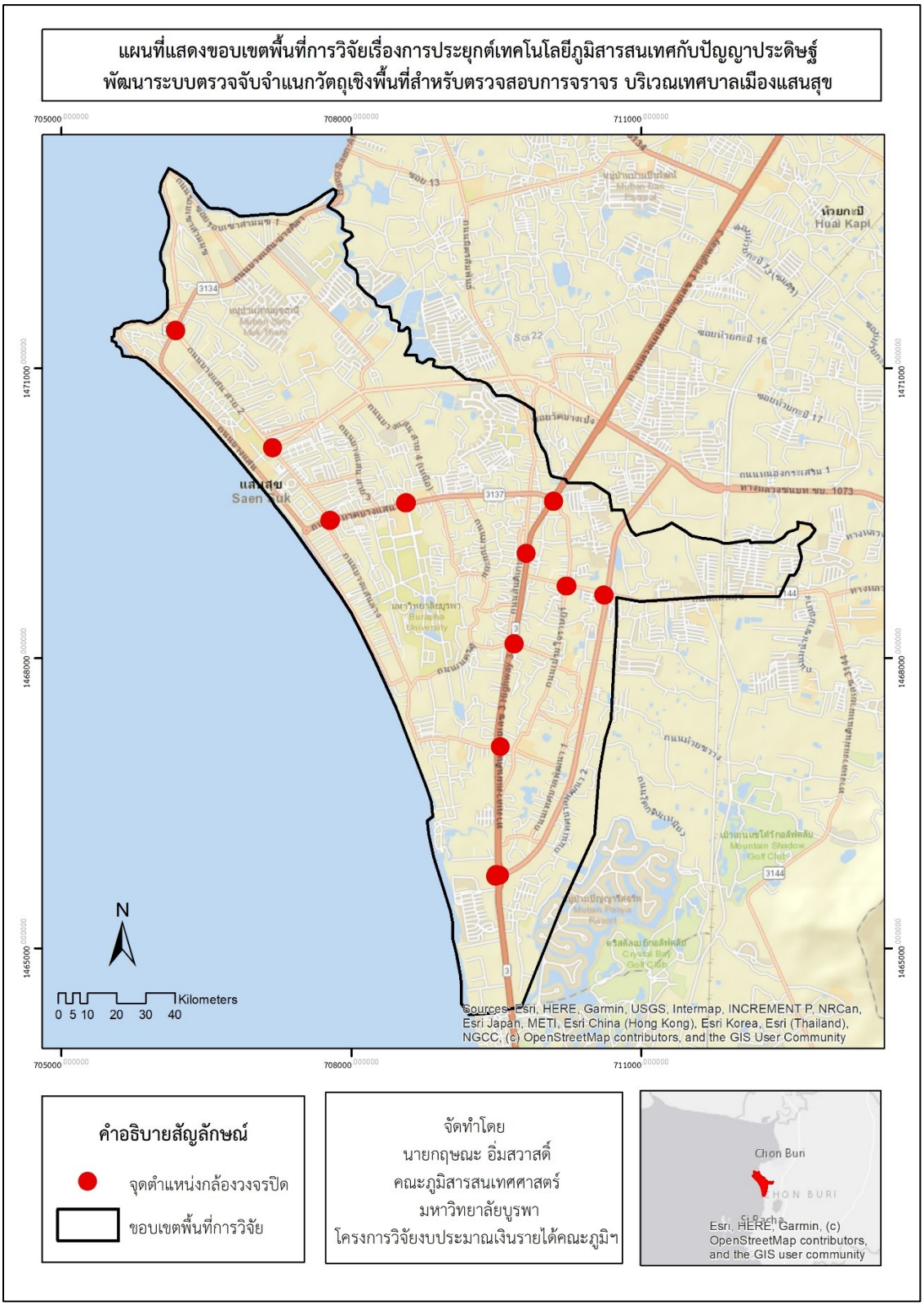
การนำเข้าข้อมูลภาพเคลื่อนไหวจากกล้องวงจรปิด (CCTV) ผ่านระบบ Web Service

วิเคราะห์จำแนกภาพด้วยแบบจำลอง (The You Only Look Once: YOLO) เวอร์ชัน 8 และแบ่งการติดตามจำแนกแยกประเภทยานพาหนะ

วิเคราะห์การเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชัน (DCNN) โดยให้มีความถูกต้องมากกว่า 60 เปอร์เซ็นต์ขึ้นไป

ขอบเขตด้านพื้นที่

พื้นที่ศึกษาในเขตเทศบาลเมืองแสนสุข ดังภาพที่ 1-1 มีอาณาเขต 20.268 ตารางกิโลเมตร ครอบคลุมพื้นที่ 3 ตำบล คือ ตำบลแสนสุขทั้งตำบล บางส่วนของตำบลเหมือง บางส่วนของตำบลห้วยกะปิ

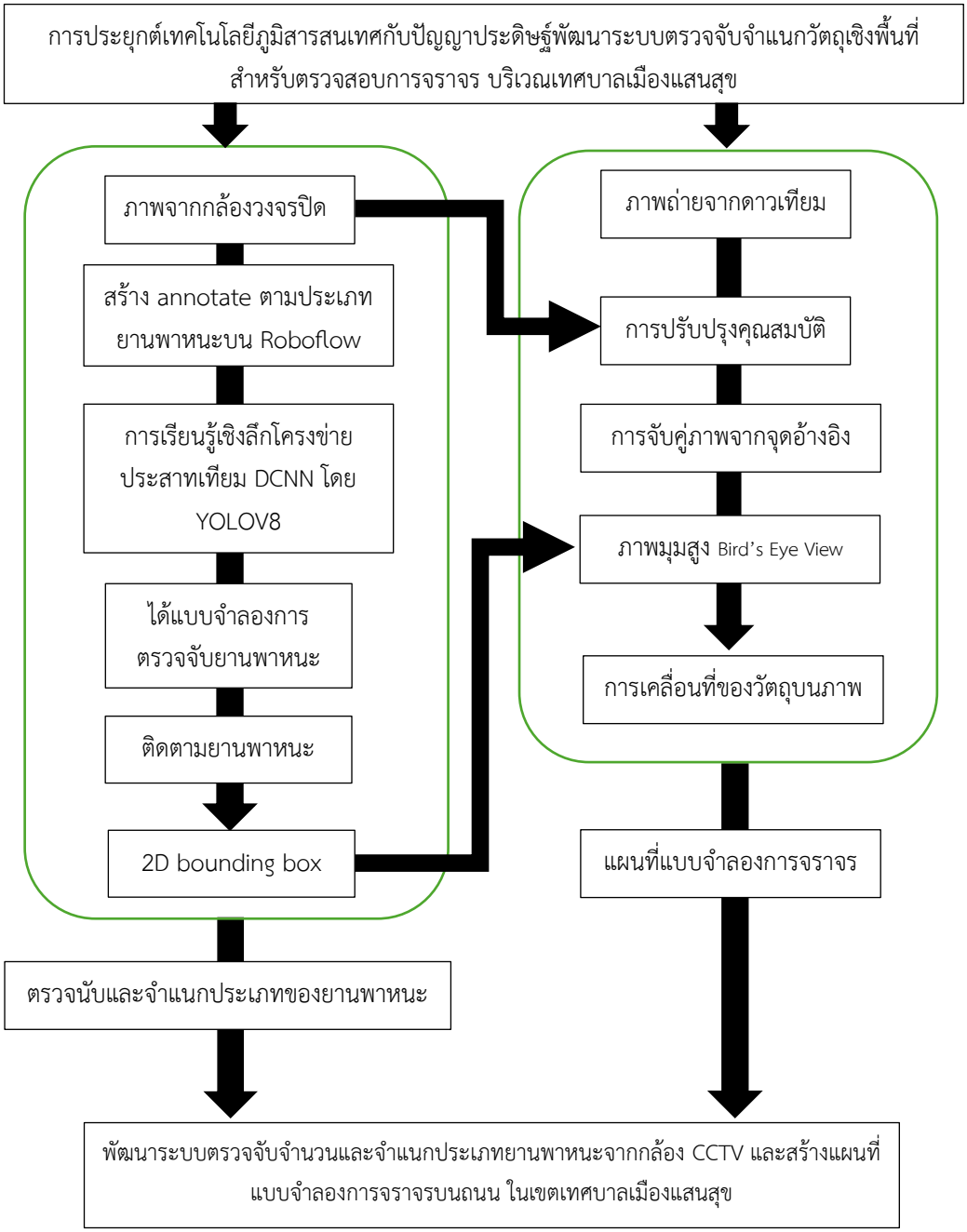



ภาพที่ 1-1 แผนที่แสดงขอบเขตพื้นที่การวิจัยบริเวณเทศบาลเมืองแสนสุข

**กรอบแนวความคิดของการวิจัย**

รวบรวมข้อมูลภาพจากกล้องวงจรปิด ที่มีการติดตั้งตามจุดต่างๆ ในเขตเทศบาลเมืองแสนสุข นำเข้าข้อมูลภาพใช้เทคนิคการเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) บน YOLOV8 เพื่อสร้างแบบจำลองการจำแนกและติดตามประเภทยานพาหนะในรูปแบบ 2D bounding box และกำหนดเส้นเพื่อสร้างเงื่อนไขการซ้อนทับระหว่างเส้นกับกรอบสี่เหลี่ยมที่ติดตามวัตถุและเขียนฟังก์ชันการนับจำนวนแต่ละประเภทของยานพาหนะ

ในส่วนของภาพถ่ายจากดาวทียมผ่านการปรับปรุงคุณสมบัติข้อมูลและนำไปจับคู่กับข้อมูลภาพจากกล้องวงจรปิด โดยใช้วิธีการ Homography ปรับมุมมองของกล้องวงจรปิดให้เป็นภาพมุมสูง Bird’s Eye View และทำการแปลงข้อมูลการติดตามวัตถุที่อยู่ในรูปแบบ 2D bounding box จากกรอบสี่เหลี่ยมให้เป็นจุดกึ่งกลางของสี่เหลี่ยมเพื่อใช้เป็นตัวแทนของการเคลื่อนที่ของยานพาหนะเพื่อนำไปแสดงผลการจำลองการจราจรบริเวณนั้น นำผลลัพธ์ที่ได้พัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV และสร้างแบบจำลองแผนที่การจราจรบนถนน ในเขตเทศบาลเมืองแสนสุข ดังภาพที่ 1-2



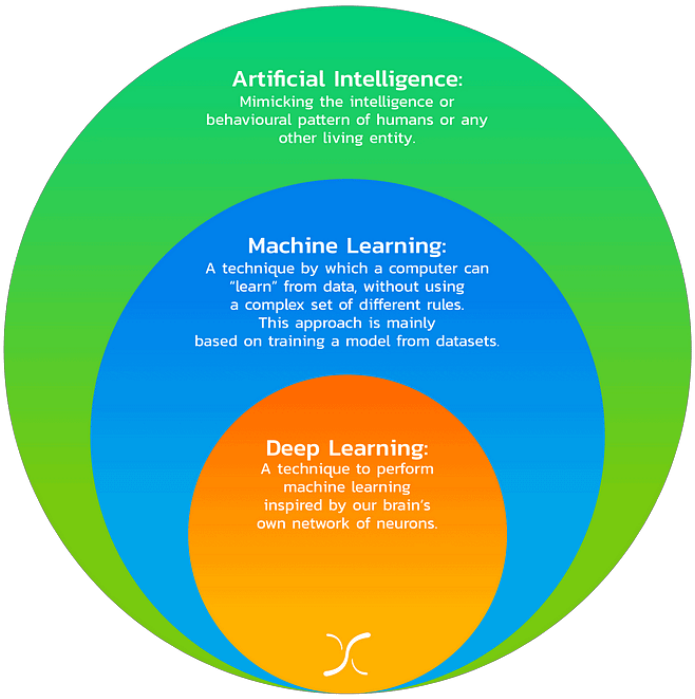

ภาพที่ 1-2 กรอบแนวความคิดของการวิจัย

# บทที่ 2

# เอกสารและงานวิจัยที่เกี่ยวข้อง

**การเรียนรู้เชิงลึก (Deep Learning)**

การเรียนรู้เชิงลึกเป็นส่วนย่อยหนึ่งของ Machine Learning (ML) และอยู่ภายใต้ปัญญาประดิษฐ์ Artificial Intelligence (AI) ซึ่งการรวมความฉลาดของมนุษย์สู่เครื่องจักร (Machine) คือชุดของคำสั่ง กระบวนการเทคนิค หรืออัลกอริทึม ที่ทำให้ระบบคอมพิวเตอร์สามารถเลียนแบบ พัฒนาและแสดงพฤติกรรมของมนุษย์ได้และสามารถแก้ปัญหาหรือแก้อัลกอริทึมตามชุดของคำสั่งที่สร้างไว้ได้สำเร็จ การทำงานเช่นนั้นเรียกว่า “ปัญญาประดิษฐ์” ดังภาพที่ 2-1




ภาพที่ 2-1 ลำดับขั้นของ Deep Learning (คลาวด์ เอซ จำกัด, 2563)

การสอนให้ระบบคอมพิวเตอร์ทำการเรียนรู้ได้ด้วยตนเองโดยการใช้ “ข้อมูล” เรียกว่า Machine Learning (ML) คือ การสอนอัลกอริทึมให้เรียนรู้ทำความเข้าใจและตัดสินใจได้ด้วยตัวเองจาก 'ข้อมูล' ที่ป้อนให้ การเรียนรู้ของ Machine นั้นเป็นไปในสองรูปแบบคือ การเรียนรู้โดยมีผู้กำกับดูแล (Supervised) หรือการเรียนรู้โดยไม่มีผู้กำกับดูแล (Unsupervised)

การเรียนรู้โดยมีผู้กำกับดูแล (Supervised) นั้นเครื่องจะเรียนรู้และทำนายผลลัพธ์ได้จากการช่วยเหลือของนักวิทยาศาสตร์ข้อมูล (Data Scientist) ส่วนการเรียนรู้โดยไม่มีผู้กำกับดูแล

(Unsupervised) นั้นเครื่องจะเรียนรู้และทำนายผลได้จากการจำแนกและสร้างแพทเทิร์นของมันจากข้อมูลที่ได้รับเมื่อเครื่องสามารถทำนายผลลัพธ์จากชุดข้อมูลจำนวนมากได้มากเท่าไร ก็จะยิ่งแสดงความสามารถในการเรียนรู้เชิงลึก (Deep Learning) มากเท่านั้น

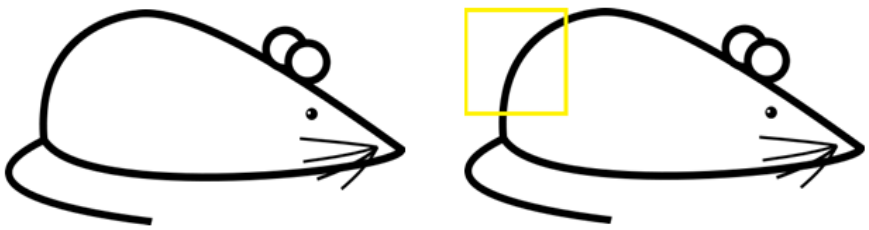
อัลกอริทึมแบบระบบเรียนรู้เชิงลึก (Deep learning) ต้องใช้ “โครงข่ายใยประสาทเสมือน” (Artificial Neural Networks (ANN)) ซึ่งก็เหมือนวิธีการทำงานของระบบประสาทในสมองมนุษย์ โครงข่ายเหล่านี้มี “เซลล์ประสาท” ที่เชื่อมต่อกันเป็น “ระบบประสาท” และสื่อสารกัน โดยใช้วิธีประมวลผลแบบขนาน (parallel processing) เพื่อทำให้มันสามารถเข้าใจและเรียนรู้จากข้อมูลจำนวนมากที่ได้รับอย่างต่อเนื่อง

สมองคนเรามักจะพยายามถอดรหัสข้อมูลที่ได้รับ อีกทั้งมักจะติดป้ายและการกำหนดสิ่งต่างๆ แบ่งแยกเป็นหมวดหมู่ เมื่อใดก็ตามที่เราได้รับข้อมูลใหม่สมองจะพยายามเปรียบเทียบกับสิ่งที่เราได้รู้ก่อนหน้า ก่อนที่จะทำความเข้าใจกับมัน เช่นเดียวกัน DL ก็สามารถถูกสอนให้ทำงานในลักษณะเดียวกันให้สำเร็จได้ (เทคซอร์ส, 2562)

## โครงข่ายประสาทเทียมแบบคอนโวลูชัน(Convolutional Neural Network: CNN)

Convolutional Neural Network (CNN) หรือ โครงข่ายประสาทแบบคอนโวลูชัน เป็นโครงข่ายประสาทเทียมหนึ่งในกลุ่ม bio-inspired โดยที่ CNN จะจำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อยๆ และนำกลุ่มของพื้นที่ย่อยๆ มาผสานกัน เพื่อจำแนกว่าสิ่งนั้นเป็นอะไร

การมองพื้นที่ย่อยของมนุษย์จะมีการแยกคุณลักษณะ (feature) ของพื้นที่ย่อย ๆ นั้น เช่น ลายเส้น และการตัดกันของสี ซึ่งการที่มนุษย์รู้ว่าพื้นที่ตรงนี้เป็นเส้นตรงหรือสีตัดกัน เพราะมนุษย์ดูทั้งจุดที่สนใจและบริเวณรอบ ๆ ประกอบกัน ถ้าเราเปรียบว่ากรอบสีเหลี่ยมสีเหลืองนั้น คือพื้นที่ที่มนุษย์กำลังให้ความสนใจอยู่ แต่เราสามารถรับรู้ได้ว่าสิ่งนี้คือหนู เพราะเรากวาดสายตามองรอบ ๆ ดังภาพที่ 2-2

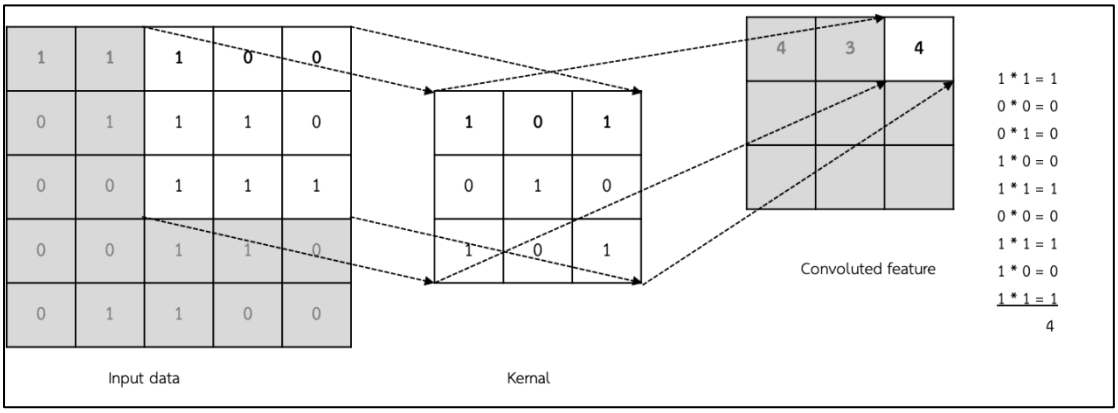


ภาพที่ 2-2 ภาพซ้ายมือรูปวาดหนูและทางขวามือในกรอบสี่เหลี่ยมพื้นที่จุดสนใจ
(Adit Deshpande, 2016)

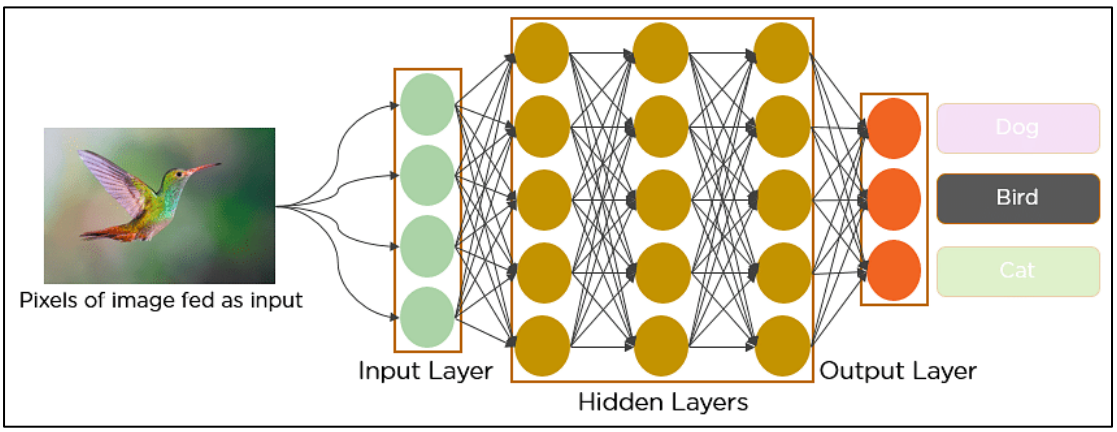
แนวคิดของ CNN นั้นค่อนข้างเป็นแนวคิดที่ดีมาก แต่สิ่งที่ซับซ้อนของมันคือระบบการคำนวณที่สอดคล้องกับ Concept ของมันเองและต้องมีคณิตศาสตร์มารองรับ โดยการคำนวณตาม

แนวคิดนี้ใช้หลักการเดียวกันกับ คอนโวลูชันเชิงพื้นที่ (Spatial Convolution) ในการทำงานด้าน Image Processing

การคำนวณนี้จะเริ่มจากการกำหนดค่าใน ตัวกรอง (filter) หรือ เคอร์เนล (kernel) ที่ช่วยดึงคุณลักษณะที่ใช้ในการรู้จำวัตถุออก โดยปกติตัวกรอง/เคอร์เนลอันหนึ่งจะดึงคุณลักษณะที่สนใจออกมาได้หนึ่งอย่าง เราจึงจำเป็นต้องตัวกรองหลายตัวกรองด้วย เพื่อหาคุณลักษณะทางพื้นที่หลายอย่างประกอบกันการประเมินผลของ Convolutional feature จะสามารถคำนวณได้ดังภาพที่ 2-3 และภาพที่ 2-4




ภาพที่ 2-3 การคำนวณ Convolutional feature (Put Dejudom, 2022)




ภาพที่ 2-4 อธิบายแผนผังการแสดงการทำงานของ CNN ในการระบุภาพนก (Put Dejudom, 2022)

## การเรียนรู้เชิงลึกของโครงข่ายประสาทเทียมแบบคอนโวลูชัน Deep convolutional neural networks (DCNN)

เป็นวิธีการเรียนรู้เชิงลึกซึ่งมีความแตกต่างจาก Convolutional Neural Network (CNN) โดยปกติตรงจำนวนชั้นที่ซ่อนอยู่ที่มีมากกว่า 5 ชั้น ซึ่งใช้การดึงคุณสมบัติเพิ่มเติมและเพิ่มความ

แม่นยำของการคาดคะเน โดยวิธี DCNN จะแบ่งออกเป็น 2 ส่วน ส่วนแรกจะเพิ่มจำนวนของชั้นข้อมูลที่ซ่อนอยู่ ส่วนที่สองเพิ่มจำนวนจุดเชื่อมต่อ (node) ในชั้นข้อมูลที่ซ่อนอยู่ วิธีการแบบ DCNN ใช้กันอย่างแพร่หลายในกลุ่มงานวิจัย Computer vision รวมถึงการแปลวัตถุ การตรวจจับและการจำแนกจากการเรียนรู้ภายใต้การกำกับดูแลที่ใช้ข้อมูลดิบเพื่อกำหนดคุณลักษณะการจัดหมวดหมู่ ซึ่งตรงกันข้ามกับเทคนิคการเรียนรู้ของเครื่อง (ML) อื่น ๆ ต้องการมีคุณสมบัติการป้อนข้อมูลล่วงหน้า โดยส่วนใหญ่ใช้ในการจำแนกแยกประเภทวัตถุจากภาพนิ่งหรือวิดีโอได้อย่างมีประสิทธิภาพ

จุดแข็งของ DCNN อยู่ในส่วนขั้นตอนการแบ่งชั้น DCNN ใช้โครงข่ายประสาทเทียมแบบสามมิติโดยจากประมวลผลภาพจากองค์ประกอบชั้นข้อมูลสีแดง สีเขียว และสีน้ำเงิน ในช่วงเวลาเดียวกัน ซึ่งช่วยลดจำนวนเซลล์ประสาทเทียมที่ต้องใช้ประมวลผลภาพได้อย่างมากเมื่อเทียบกับโครงข่ายประสาทเทียมแบบเดิม การทำงานของโครงข่ายประสาทเทียม Deep Convolutional neural networks จะรับภาพในส่วน Input และเข้าโครงข่ายเพื่อฝึกการเรียนรู้แยกประเภทโดยใช้คณิตศาสตร์แบบพิเศษที่เรียกว่า Convolution แทนการคูณแบบเมทริกซ์ โดยสถาปัตยกรรม

## โครงสร้างของโครงข่าย Deep Convolutional Neural Networks (DCNN)

โครงข่าย Deep Convolutional Neural Networks (DCNN) เป็นรูปแบบของโครงข่ายประสาทเทียมที่ใช้ในงานการประมวลผลภาพและการวิเคราะห์ข้อมูลเชิงพื้นที่อื่น ๆ โครงสร้างของ DCNN ประกอบด้วยหลายชั้น (layers) ที่มีความซับซ้อนและสามารถเรียนรู้คุณลักษณะเชิงลึกจากข้อมูลภาพดังนี้

### Convolutional Layer

Convolutional Layer เป็นชั้นแรกที่ใช้ใน DCNN ซึ่งทำหน้าที่สกัดคุณลักษณะพื้นฐานจากข้อมูลภาพ โดยใช้ตัวกรอง (filter หรือ kernel) ที่เคลื่อนที่ผ่านภาพ การดำเนินการคอนโวลูชันจะสร้างแผนที่ลักษณะ (feature map) ที่เก็บคุณลักษณะเชิงพื้นที่ของภาพ

Kernel/Filter: ตัวกรองขนาดเล็กที่ใช้ในการคอนโวลูชันกับภาพ

Stride: ระยะการเคลื่อนที่ของตัวกรองผ่านภาพ

Padding: การเพิ่มพิกเซลที่ขอบของภาพเพื่อรักษาขนาดของแผนที่ลักษณะ

### Activation Layer

Activation Layer เป็นชั้นที่ใช้ฟังก์ชันเปิดใช้งาน (activation function) เพื่อเพิ่มความไม่เป็นเชิงเส้นให้กับโมเดล ทำให้สามารถเรียนรู้คุณลักษณะที่ซับซ้อนได้มากขึ้น ฟังก์ชันเปิดใช้งานที่นิยมใช้ได้แก่

ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$

Sigmoid: $f(x) = \frac{1}{1+e^{-x}}$

Tanh: $f(x) = \tanh(x)$

### Pooling Layer

Pooling Layer ใช้เพื่อลดขนาดเชิงพื้นที่ของแผนที่ลักษณะ ลดจำนวนพารามิเตอร์และการคำนวณ ทำให้โมเดลมีความทนทานต่อการแปรผันของตำแหน่งและขนาดของคุณลักษณะ

Max Pooling: เลือกค่ามากที่สุดจากกลุ่มของพิกเซล

Average Pooling: คำนวณค่าเฉลี่ยของกลุ่มของพิกเซล

**Fully Connected Layer**

Fully Connected Layer เป็นชั้นที่เชื่อมต่อทุกนิวรอนในชั้นปัจจุบันกับทุกนิวรอนในชั้นถัดไป มักใช้ในขั้นตอนสุดท้ายของ DCNN เพื่อนำคุณลักษณะที่สกัดได้มาทำการจำแนกประเภท (classification)

**Dropout Layer**

Dropout Layer เป็นเทคนิคการเรียนรู้เชิงลึกที่ใช้เพื่อลดการปรับตัวเกิน (overfitting) โดยการปิดใช้งานนิวรอนบางส่วนในระหว่างการฝึก (training) แบบสุ่ม

**Softmax Layer**

Softmax Layer มักเป็นชั้นสุดท้ายในโครงข่าย DCNN ที่ใช้สำหรับการจำแนกประเภทหลายกลุ่ม (multi-class classification) โดยแปลงค่าล็อกิต (logits) ให้เป็นค่าความน่าจะเป็นที่รวมกันได้ 1

**โครงสร้างตัวอย่างของ DCNN**

ตัวอย่างโครงสร้างของ DCNN ที่ใช้ในงานจำแนกภาพ ดังภาพที่ 2-5

Input Layer: รับข้อมูลภาพขนาด 32×32×3 (เช่นภาพสี)

Convolutional Layer 1: ใช้ตัวกรอง 5×5 จำนวน 32 ตัว

Activation Layer (ReLU)

Pooling Layer (Max Pooling)

Convolutional Layer 2: ใช้ตัวกรอง 5×5 จำนวน 64 ตัว

Activation Layer (ReLU)

Pooling Layer (Max Pooling)

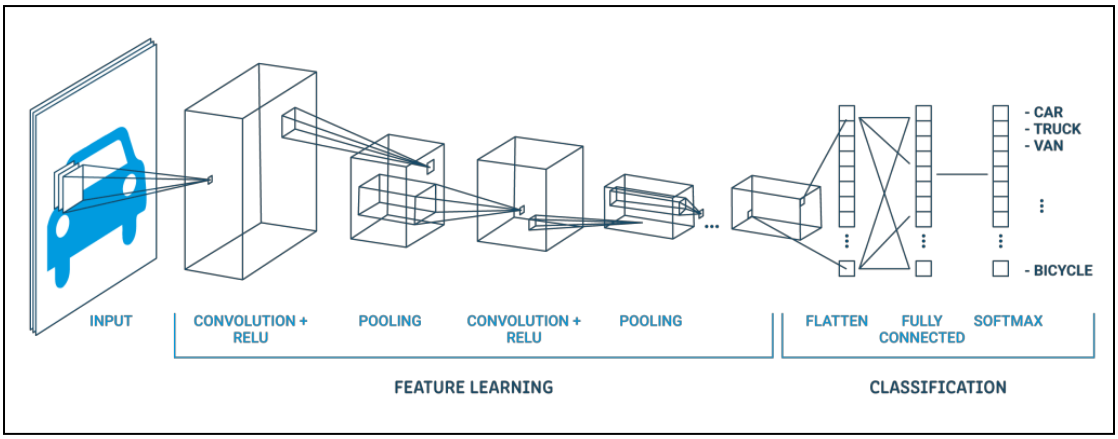
Fully Connected Layer: มี 1024 นิวรอน

Dropout Layer

Fully Connected Layer: มี 10 นิวรอน (สำหรับการจำแนก 10 กลุ่ม)

Softmax Layer

โครงข่ายนี้สามารถนำไปใช้ในการจำแนกภาพที่มีหลายกลุ่ม เช่น การจำแนกภาพสัตว์ต่างๆ การประยุกต์ใช้โครงข่าย DCNN ในการจำแนกวัตถุในพื้นที่สามารถช่วยเพิ่มความแม่นยำในการวิเคราะห์และตรวจจับวัตถุในระบบจราจรได้อย่างมีประสิทธิภาพ




ภาพที่ 2-5 สถาปัตยกรรมโครงสร้างของโครงข่าย Deep Convolutional neural networks (run.ai, 2023)

การประยุกต์ตัวกรอง Convolution กับรูปภาพพื่อตรวจจับคุณลักษณะของภาพได้ดังนี้

Convolution คือ การนำชุดค่าน้ำหนักชุดหนึ่งมาคูณกับข้อมูลภาพที่นำเข้าจากโครงข่ายประสาทเทียม

Kernels หรือตัวกรอง เป็นส่วนในกระบวนการคูณกับค่าน้ำหนักอาร์เรย์ 2 มิติ หรือตัวกรองที่ใช้โครงสร้างแบบ 3 มิติ ที่ส่งผ่านภาพหลายครั้งเพื่อให้ครอบคลุมทั้งภาพ โดยกรองจากขวาไปซ้ายและบนลงล่าง

Dot หรือ Scalar product เป็นกระบวนการทางคณิตศาสตร์ที่ดำเนินการระหว่างการทำ Convolution แต่ละตัวกรองจะคูณน้ำหนักด้วยค่าข้อมูลนำเข้าที่แตกต่างกัน

## การทำงาน Roboflow

เป็นแพลตฟอร์มที่ช่วยในการจัดการข้อมูลสำหรับการฝึกสอนข้อมูลโมเดลการเรียนรู้ของเครื่อง โดยเฉพาะในงานที่เกี่ยวข้องกับการตรวจจับวัตถุ (object detection), การแบ่งประเภทภาพ (image classification), และการแบ่งส่วนภาพ (image segmentation) ในส่วนของการจัดการข้อมูลการติดป้าย (label) และข้อมูลการฝึกสอนข้อมูล (train data) มีความสำคัญอย่างยิ่งในการพัฒนาโมเดลที่มีประสิทธิภาพและแม่นยำ ดังนี้

**1. การติดป้ายข้อมูล (Labeling Data)**

1.1 การสร้างชุดข้อมูล

Roboflow ช่วยให้ผู้ใช้สามารถอัปโหลดรูปภาพที่ต้องการใช้ในการฝึกอบรมโมเดลได้ง่าย ๆ ผ่านอินเตอร์เฟซที่ใช้งานง่าย ผู้ใช้สามารถอัปโหลดภาพที่มีอยู่ในระบบ หรือใช้ API ในการดึงภาพจากแหล่งข้อมูลอื่น ๆ ได้

1.2 การติดป้ายภาพ (Image Annotation)

Roboflow มีเครื่องมือสำหรับการติดป้ายภาพที่ใช้งานง่าย ทำให้ผู้ใช้สามารถกำหนดตำแหน่งและประเภทของวัตถุที่ต้องการตรวจจับได้อย่างแม่นยำ ไม่ว่าจะเป็นการวาดกรอบสี่เหลี่ยม (bounding box) สำหรับการตรวจจับวัตถุ หรือการวาดโพลิกอน (polygon) สำหรับการแบ่งส่วนภาพ

1.3 การจัดการป้ายข้อมูล

Roboflow ช่วยในการจัดการและจัดระเบียบป้ายข้อมูล โดยสามารถแก้ไข เพิ่ม หรือลบป้ายข้อมูลได้อย่างสะดวก นอกจากนี้ยังสามารถสร้างป้ายข้อมูลหลายประเภทได้ เช่น การตรวจจับวัตถุหลายชนิดในภาพเดียวกัน

**2. การฝึกสอนข้อมูล (Training Data)**

2.1 การเตรียมชุดข้อมูลฝึกสอนข้อมูล

หลังจากที่มีการติดป้ายภาพเสร็จเรียบร้อย Roboflow ช่วยในการแบ่งชุดข้อมูลออกเป็นส่วนต่าง ๆ เช่น ชุดข้อมูลฝึกสอนข้อมูล (training set), ชุดข้อมูลทดสอบ (testing set), และชุดข้อมูลตรวจสอบ (validation set) ซึ่งเป็นขั้นตอนสำคัญในการฝึกสอนโมเดล

2.2 การปรับแต่งข้อมูล (Data Augmentation):

Roboflow มีฟังก์ชันการปรับแต่งข้อมูล (data augmentation) ที่ช่วยในการเพิ่มความหลากหลายของข้อมูลฝึกอบรม เช่น การหมุนภาพ (rotation), การพลิกภาพ (flipping), การปรับแสง (brightness), และการซูม (zooming) การปรับแต่งข้อมูลช่วยในการเพิ่มขนาดของชุดข้อมูลฝึกอบรมและทำให้โมเดลมีความทนทานต่อการเปลี่ยนแปลงของข้อมูลมากขึ้น

2.3 การส่งออกข้อมูล

Roboflow ช่วยในการส่งออกชุดข้อมูลฝึกอบรมในรูปแบบต่าง ๆ ที่ใช้กับเฟรมเวิร์กการเรียนรู้ของเครื่องที่หลากหลาย เช่น TensorFlow, PyTorch, Darknet, และอื่น ๆ การส่งออกข้อมูลในรูปแบบที่ต้องการทำให้การฝึกอบรมโมเดลสามารถเริ่มต้นได้อย่างรวดเร็วและสะดวก

**3. การใช้งาน Roboflow ในการพัฒนาโมเดล แบ่งออกได้ดังนี้**

3.1 การอัปโหลดและติดป้ายข้อมูล

3.1.1 อัปโหลดภาพที่ต้องการใช้ในการฝึกสอนข้อมูล

3.1.2 ใช้เครื่องมือติดป้ายข้อมูลเพื่อกำหนดตำแหน่งและประเภทของวัตถุในภาพ

3.2 การปรับแต่งข้อมูลฝึกอบรม

3.2.1 ใช้ฟังก์ชันการปรับแต่งข้อมูลเพื่อเพิ่มความหลากหลายของภาพ

3.2.2 แบ่งชุดข้อมูลออกเป็นชุดฝึกสอนข้อมูล ชุดทดสอบ และชุดตรวจสอบ

การส่งออกข้อมูล

3.2.3 เลือกรูปแบบการส่งออกข้อมูลที่ตรงกับเฟรมเวิร์กการฝึกสอนข้อมูลที่ใช้งาน

3.2.4 นำเข้าข้อมูลไปยังเฟรมเวิร์กการเรียนรู้ของเครื่องที่ต้องการ

Roboflow ช่วยลดความซับซ้อนและเพิ่มประสิทธิภาพในกระบวนการติดป้ายข้อมูลและการเตรียมข้อมูลฝึกสอนข้อมูล ทำให้ผู้พัฒนาโมเดลสามารถมุ่งเน้นไปที่การพัฒนาและปรับปรุงประสิทธิภาพของโมเดลได้อย่างเต็มที่

## Google Colab (Google Colaboratory)

เป็นเครื่องมือที่พัฒนาโดย Google ซึ่งให้บริการโฮสต์ Jupyter Notebook ฟรี บนคลาวด์ มันถูกออกแบบมาเพื่อให้ผู้ใช้งานสามารถเขียนและรันคำสั่ง Python บนเว็บเบราว์เซอร์โดยไม่ต้องตั้งค่าหรือกำหนดค่าภายในเครื่อง การใช้ Google Colab มีข้อดีหลายประการโดยเฉพาะในงานด้านการวิจัย การเรียนรู้ของเครื่อง (Machine Learning) และการวิเคราะห์ข้อมูล (Data Analysis)

**1. คุณสมบัติหลักของ Google Colab**

1.1 การรันคำสั่ง Python

Google Colab สนับสนุนการเขียนและรันคำสั่ง Python พร้อมทั้งสนับสนุนฟังก์ชันและไลบรารีที่ใช้ในงานการเรียนรู้ของเครื่อง เช่น TensorFlow, Keras, PyTorch และ OpenCV

1.2 การใช้ GPU และ TPU

Google Colab มีการสนับสนุนการใช้ GPU (Graphics Processing Unit) และ TPU (Tensor Processing Unit) ฟรี ทำให้การรันโมเดลการเรียนรู้เชิงลึก (Deep Learning) เป็นไปได้อย่างรวดเร็ว

1.3 การแบ่งปันและการทำงานร่วมกัน

ผู้ใช้สามารถแชร์โน้ตบุ๊กกับคนอื่น ๆ ได้ง่าย ๆ เหมือนการแชร์เอกสารใน Google Drive ทำให้สามารถทำงานร่วมกันได้อย่างสะดวก

1.4 การเชื่อมต่อกับ Google Drive

ผู้ใช้สามารถเชื่อมต่อและเข้าถึงไฟล์ใน Google Drive ของตนเองจากโน้ตบุ๊ก Colab ได้โดยตรง ทำให้การจัดการไฟล์และการเข้าถึงข้อมูลสะดวกมากยิ่งขึ้น

1.5 การติดตั้งไลบรารีเพิ่มเติม

ผู้ใช้สามารถติดตั้งไลบรารี Python เพิ่มเติมที่ต้องการผ่านการใช้คำสั่ง !pip ในเซลล์คำสั่งได้

**2. วิธีการใช้งาน Google Colab**

2.1 การเริ่มต้นใช้งาน

2.1.1 เข้าสู่เว็บไซต์ Google Colab

2.1.2 ลงชื่อเข้าใช้ด้วยบัญชี Google ของคุณ

2.1.3 สามารถสร้างโน้ตบุ๊กใหม่ เปิดโน้ตบุ๊กที่มีอยู่แล้วจาก Google Drive หรือ GitHub ได้

2.2 การสร้างและรันคำสั่งในโน้ตบุ๊ก:

2.2.1 ในแต่ละเซลล์สามารถเขียนคำสั่ง Python และรันคำสั่งโดยการกดปุ่ม "Run" หรือใช้คีย์ลัด Shift + Enter

2.2.2 นอกจากนี้ยังสามารถเขียนข้อความบรรยายในรูปแบบ Markdown ได้โดยเปลี่ยนประเภทของเซลล์เป็น "Text"

**3. การใช้งาน GPU/TPU**

3.1 ไปที่เมนู "Runtime" -> "Change runtime type"

3.2 ในส่วนของ "Hardware accelerator" ให้เลือก "GPU" หรือ "TPU" ตามต้องการ

3.3 ใช้ GPU/TPU จะช่วยให้การประมวลผลงานที่มีความซับซ้อนและใช้เวลานานเสร็จสิ้นได้เร็วขึ้น

**4. การเชื่อมต่อกับ Google Drive**

4.1 รันคำสั่งในเซลล์เพื่อเชื่อมต่อกับ Google Drive ดังคำสั่งตัวอย่าง

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

4.2 คลิกลิงก์เพื่ออนุญาตการเข้าถึง Google Drive ของคุณ จากนั้นให้คัดลอกคำสั่งที่ได้มาวางในช่องที่กำหนด

**5. การติดตั้งไลบรารีเพิ่มเติม**

5.1 สามารถติดตั้งไลบรารี Python เพิ่มเติมได้โดยใช้คำสั่ง !pip หรือ !apt ในเซลล์ ดังคำสั่งตัวอย่าง

```
!pip install numpy
!apt-get install -y htop
```

**6. ตัวอย่างการใช้งาน Google Colab**

6.1 การสร้างกราฟด้วย Matplotlib ดังคำสั่งตัวอย่าง

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.show()
```

6.2 การใช้ TensorFlow ใน Google Colab ดังคำสั่งตัวอย่าง

```
import tensorflow as tf
# สร้างโมเดลง่ายๆ ด้วย Keras
model = tf.keras.models.Sequential([
tf.keras.layers.Dense(128, activation='relu', input_shape=(784,)),
tf.keras.layers.Dense(10, activation='softmax')])
# สรุปโมเดล
model.summary()
```

Google Colab เป็นเครื่องมือที่มีประสิทธิภาพและใช้งานง่ายสำหรับนักวิจัย นักพัฒนา และนักเรียนที่ต้องการทำงานกับข้อมูลและการเรียนรู้ของเครื่องโดยไม่ต้องกังวลเกี่ยวกับการตั้งค่าระบบและฮาร์ดแวร์

## OpenCV (Open Source Computer Vision)

เป็นไลบรารีโอเพนซอร์สที่ใช้สำหรับการประมวลผลภาพและการเรียนรู้ของเครื่อง มีความนิยมในวงกว้างทั้งในวงการวิจัยและอุตสาหกรรมเนื่องจากความสามารถและความยืดหยุ่นในการใช้งาน OpenCV ถูกพัฒนาในภาษา C++ แต่ก็มีการสนับสนุนการใช้งานในหลายภาษาเช่น Python, Java, และ MATLAB

**1. คุณสมบัติและความสามารถของ OpenCV**

1.1 การประมวลผลภาพ (Image Processing)

การอ่านและเขียนภาพในหลายรูปแบบ เช่น JPEG, PNG, TIFF, BMP การปรับแต่งภาพ เช่น การครอบตัด (cropping), การหมุน (rotation), การปรับแสงและสี (brightness and contrast adjustment), การย่อและขยายภาพ (resizing) การใช้ฟิลเตอร์ (filtering) เช่น Gaussian blur, median blur, bilateral filter

1.2 การวิเคราะห์และตรวจจับวัตถุ (Object Detection and Recognition)

1.2.1 การตรวจจับใบหน้า (face detection) ด้วย Haar Cascades และ DNN (Deep Neural Networks) การตรวจจับและติดตามวัตถุ (object tracking) เช่น KLT tracker, Meanshift, Camshift การตรวจจับขอบ (edge detection) ด้วย Canny edge detector

1.2.2 การวิเคราะห์รูปร่างและการจับคู่ลักษณะ (shape analysis and feature matching)

1.3 การประมวลผลวิดีโอ (Video Processing)

1.3.1 การอ่านและเขียนวิดีโอจากไฟล์หรือกล้อง

1.3.2 การประมวลผลเฟรมต่อเฟรมในวิดีโอ เช่น การปรับปรุงคุณภาพวิดีโอ การตรวจจับการเคลื่อนไหว

1.4 การใช้ Machine Learning

1.4.1 การฝึกอบรมและการทำนายโมเดลการเรียนรู้ของเครื่อง (machine learning models)

1.4.2 การใช้งานโมเดลการเรียนรู้เชิงลึก (deep learning models) ผ่านการใช้งาน DNN module ที่รองรับเฟรมเวิร์กเช่น TensorFlow, Caffe, PyTorch

1.5 การทำงานร่วมกับฮาร์ดแวร์

1.5.1 การใช้งานกล้องและการประมวลผลภาพเรียลไทม์ (real-time image processing)

1.5.2 การใช้ GPU acceleration เพื่อเพิ่มประสิทธิภาพในการประมวลผล Homography คือการเปลี่ยนแปลงเชิงเส้นที่แสดงถึงการเปลี่ยนแปลงของภาพระหว่างมุมมองต่าง ๆ ของระนาบเดียวกัน เช่น การเปลี่ยนมุมมองของกล้อง การหมุน การเคลื่อนที่ การซูมเข้าและซูมออก ซึ่งกระบวนการนี้มักใช้ในการประยุกต์ต่าง ๆ ในด้านคอมพิวเตอร์วิชั่น เช่น การจับภาพร่วม (image stitching), การสังเคราะห์มุมมองใหม่ (view synthesis), และการทำเสมือนจริง (augmented reality) เป็นต้น

## กระบวนการทำงานของ Homography

การหาจุดที่ตรงกัน (Feature Matching) เริ่มต้นด้วยการหาจุดที่ตรงกันระหว่างภาพสองภาพ โดยใช้เทคนิคการตรวจจับจุดเด่น (feature detection) เช่น SIFT, SURF, หรือ ORB เพื่อค้นหาจุดเด่นในแต่ละภาพ จากนั้นใช้เทคนิคการจับคู่จุดเด่น (feature matching) เพื่อหาคู่จุดที่ตรงกันระหว่างภาพสองภาพ

การคำนวณ Homography Matrix จากคู่จุดที่ตรงกันที่ได้ จะนำมาใช้คำนวณ Homography matrix $H$ ซึ่งเป็นเมตริกซ์ขนาด 3×3 ที่เปลี่ยนแปลงพิกัดของจุดในภาพแรกไปยังพิกัดของจุดที่ตรงกันในภาพที่สอง โดยมีสมการที่ 2.1 ดังนี้

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ w \end{bmatrix} \qquad (2\text{-}1)$$

โดย $(x,y,w)$ เป็นพิกัดของจุดในภาพแรก และ $(x',y',w')$ เป็นพิกัดของจุดที่ตรงกันในภาพที่สอง หลังการคำนวณจะทำการ normalize ค่า $w'$ ให้เป็น 1 เพื่อได้ค่าพิกัด $(x',y')$

**1. การประมาณค่า Homography Matrix**

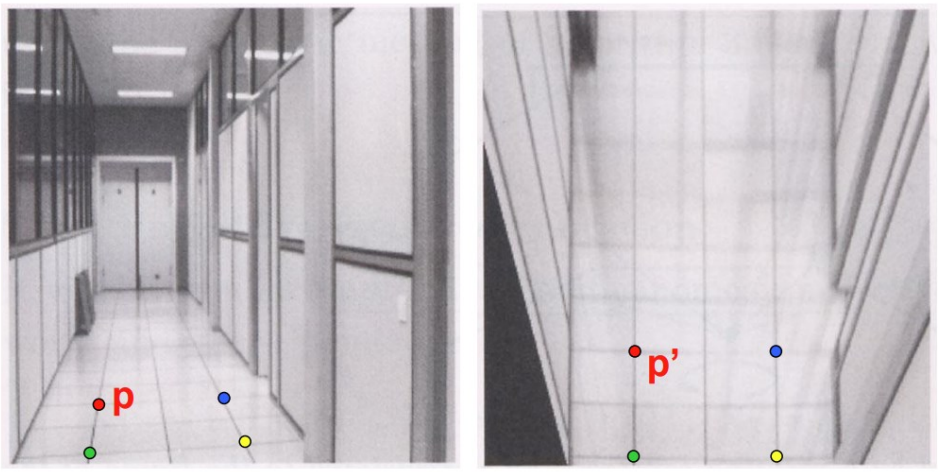
การประมาณค่าของ Homography matrix มักใช้วิธีการที่เรียกว่า Direct Linear Transformation (DLT) ซึ่งต้องการคู่จุดที่ตรงกันอย่างน้อย 4 คู่เพื่อคำนวณ $H$ จากสมการเชิงเส้น

การกรองคู่จุดที่ผิดพลาด เนื่องจากคู่จุดที่ตรงกันบางคู่ที่ได้จากการจับคู่อาจมีข้อผิดพลาด ดังนั้น จะต้องใช้วิธีการกรองเช่น Random Sample Consensus (RANSAC) เพื่อกำจัดคู่จุดที่ผิดพลาดและรักษาคู่จุดที่ถูกต้องไว้ ดังภาพที่ 2-6

ภาพที่ 2-6 การแมตซ์ของภาพทั้ง 2 จากแบบจำลอง RANSAC

**2. การใช้ Homography**

หลังจากที่ได้ Homography matrix ที่ถูกต้องแล้ว จะสามารถใช้เมตริกซ์นี้ในการเปลี่ยนแปลงพิกัดของจุดในภาพแรกไปยังภาพที่สอง หรือเพื่อทำการปรับแก้ (warp) ภาพตามที่ต้องการ ดังภาพที่ 2-7



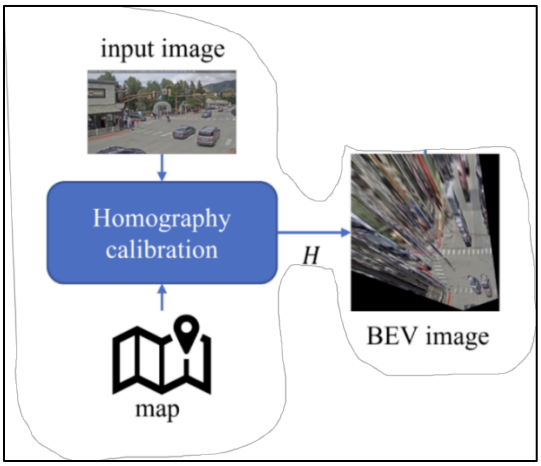

ภาพที่ 2-7 การแก้ไขภาพ (มุมมองสี่เหลี่ยมจัตุรัส) Image rectification (square views) (Gerhard Roth, 2011)

## การสอบเทียบค่าจากภาพของ กล้อง CCTV และภาพถ่ายจากดาวเทียมด้วยวิธีโฮโมกราฟี (Calibration of homography)

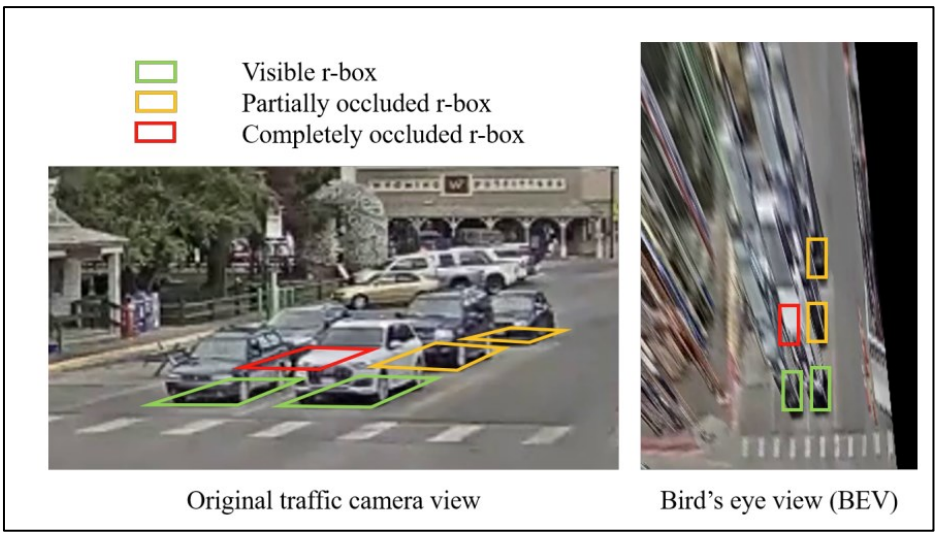
การแปลงข้อมูลพื้นระนาบของภาพจากกล้อง CCTV ตามภาพถ่ายจากดาวเทียม เพื่อต้องการปรับมุมมองให้เป็นภาพมุมสูงหรือ Bird's Eye View ต้องเข้าใจตำแหน่งบนพื้นระนาบจากกล้อง CCTV ที่ตรวจจับยานพาหนะในมุมระนาบแนวนอนและภาพถ่ายจากดาวเทียมที่เป็นแนวดิ่งที่มีค่าพิกัดเพื่อหาจุดในภาพที่เป็นจุดเหมือนกันใช้สำหรับการอ้างอิงของภาพทั้ง 2 ภาพ โดยเทคนิควิธีการที่นำมาใช้คือ วิธีแบบโฮโมกราฟี (Homography) ที่กล่าวมาข้างต้น เข้ากระบวนการสอบเปรียบตำแหน่งของจุดภาพให้มีความใกล้เคียงกันมาที่สุด โดยใช้สมการที่ 2-2 การสอบเทียบ

$$H_{ori}^{bev} = H_{world}^{bev} \; H_{bev}^{world} \qquad (2\text{-}2)$$

โดย $\mathrm{sp}_a = H_b^a p_b$ แสดงถึงพิกัดของ $H_b^a$ บนแผนที่ในเฟรม $b$ เพื่อจุดประสานความเชื่อมต่อในเฟรม $a$ จนไปถึงค่าสเกลเฟคเตอร์ (scale factor) $s$, และ $p = [x, y, 1]^T$ คือพิกัดต่อเนื่องของ $a$ ชี้ไปยังพื้นราบ $bev$ คือ ระนาบของภาพมุมสูง (Bird's Eye View) $world$ คือ ค่าพื้นระนาบของพื้นผิวโลก $ori$ คือ ภาพจากกล้อง CCTV $H_{world}^{bev}$ ผู้ใช้สามารถกำหนดได้อย่างอิสระที่มีการเปลี่ยนแปลงที่คล้ายคลึงกันโดยคงมุมไว้ระหว่างระนาบบนถนนจริงและมุมมองจากมุมสูงของภาพถ่ายดาวเทียม จำเป็นที่ต้องมีการสอบเทียบค่า $H_{ori}^{world}$ หมายถึง ภาพจากกล้อง CCTV $(ori)$ และ ข้อมูลภาพถ่ายจากดาวเทียมที่มีระบบอ้างอิงตามพิกัดตำแหน่งโลก $(world)$ (Minghan Zhu & et al., 2022) ดังภาพที่ 2-8 และภาพที่ 2-9




ภาพที่ 2-8 ผังแสดงการสอบเทียบค่าและผลลัพธ์ของภาพจาก CCTV กับ ภาพถ่ายจากดาวเทียมที่มีพิกัดอ้างอิง ด้วยวิธีการ Homography ดัดแปลงมาจาก (Minghan Zhu & et al., 2022)

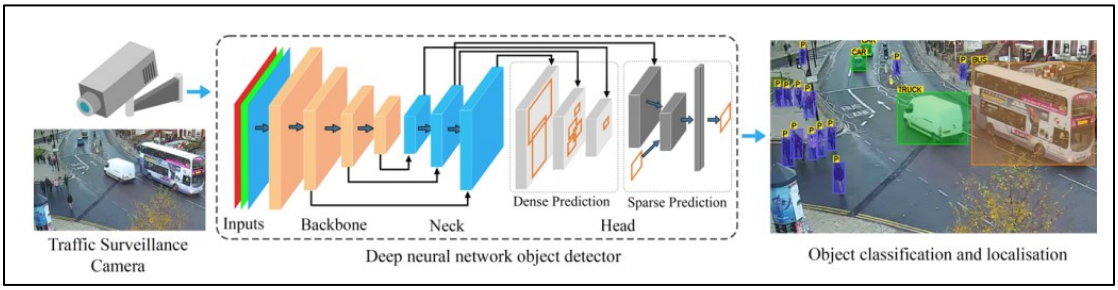



ภาพที่ 2-9 ผลลัพธ์หลังจากสอบเทียบของภาพจาก CCTV กับ ภาพถ่ายจากดาวเทียมที่มีพิกัดอ้างอิง ได้ภาพมุมสูง (Bird's Eye View) ดัดแปลงมาจาก (Minghan Zhu & et al., 2022)

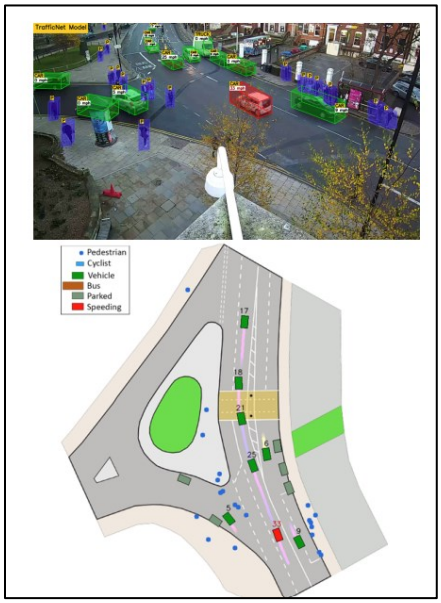
## การใช้เทคโนโลยีกล้องวงจรปิด (Smart video surveillance systems)

ระบบกล้องวงจรปิดอัจฉริยะกำลังกลายเป็นเทคโนโลยีทั่วไปสำหรับการตรวจสอบการจราจรและความแออัดการจัดการควบคู่ไปกับการปรับปรุงเทคโนโลยีให้สามารถจัดการความซับซ้อนของระบบการจราจรสำหรับการเฝ้าระวังการจราจรแบบอัตโนมัติที่เพิ่มขึ้น เนื่องจากปัจจัยหลายประการ เช่น การพัฒนาเมือง การผสมผสานระหว่างยานยนต์แบบมีคนขับและยานยนต์ไร้คนขับ การเติบโตของประชากร จำนวนคนเดินทางบนถนนที่เพิ่มขึ้นและผู้ใช้รถใช้ถนน ทำให้จำนวนกล้องวงจรปิดถูกติดตั้งอย่างมาก จากข้อมูลกล้อง CCTV มากกว่า 20 ล้านตัวในประเทศสหรัฐอเมริกาและสหราชอาณาจักร โดยเฉพาะถนนสายหลักของเมือง สถานที่แออัด ถนน ทางแยก และทางหลวง แสดงให้เห็นถึงความสำคัญของกล้องวงจรปิดสำหรับเจ้าหน้าที่ การเฝ้าระวังที่ต้องเชื่อมต่อถึงกัน อีกทั้งกล้องสามารถเป็นแพลตฟอร์มพิเศษสำหรับการศึกษาเพิ่มเติมเกี่ยวกับการจัดการจราจรและการวางผังเมือง อย่างไรก็ตาม ในลักษณะดังกล่าวสภาพแวดล้อมถนนที่หนาแน่นและซับซ้อน การตรวจสอบสภาพถนนแบบเดิม ๆ เป็นเรื่องที่น่าเบื่อและใช้เวลานาน และเป็นวิธีการที่แม่นยำน้อยกว่าการใช้เทคโนโลยี AI ที่สามารถติดตาม ตรวจสอบ และเฝ้าระวังดังภาพที่ 2-10 แทนมนุษย์อย่างอัตโนมัติ ได้รับการค้นคว้ามาหลายปีจึงค่อยมาทดแทนมนุษย์ด้วยคอมพิวเตอร์ที่สามารถวิเคราะห์การจราจรสดและมีประสิทธิภาพเพื่อรักษาความปลอดภัยในการขนส่งและความยั่งยืน

การรับรู้ภาพด้วยคอมพิวเตอร์ (Computer Vision) จึงเข้ามาช่วยเหลือมนุษย์มากขึ้น ในการวิเคราะห์วัตถุต่าง ๆ ที่ปรากฏในภาพวิดีโอหรือภาพนิ่ง และจำแนกแยกประเภทยานพาหนะได้อย่างแม่นยำ ตามกระบวนการฝึกฝนข้อมูลที่มีอยู่จำนวนมาก สามารถวิเคราะห์ถึงความเร็ว ความหนาแน่น ของจำนวนยานพาหนะและแสดงเป็นภาพมุมสูงได้อย่างมีประสิทธิภาพในด้านมุมมองที่ครอบคลุม อีกทั้งยังเป็นการสร้างข้อมูลเพื่อนำไปสร้างแบบจำลองเสมือนจากวัตถุทางกายภาพ หรือที่เรียกว่า Digital Twin (Mahdi Rezaei, Mohsen Azarmi & Farzam Mohammad Pour Mir, 2021) ดังภาพที่ 2-11




ภาพที่ 2-10 แสดงโครงสร้างการทำงานของกล้องวงจรปิด การใช้แบบจำลองทางปัญญาประดิษฐ์ และการตรวจจับ จำแนก ยานพาหนะ (Mahdi Rezaei, Mohsen Azarmi & Farzam Mohammad Pour Mir, 2021)




ภาพที่ 2-11 ภาพบน แสดงการตรวจจับยานพาหนะแบบ 3 มิติ และประมาณความเร็วของยานพาหนะ ภาพล่าง การแสดงแบบจำลองเสมือนจากวัตถุทางกายภาพ หรือ Digital Twin (Mahdi Rezaei, Mohsen Azarmi & Farzam Mohammad Pour Mir, 2021)

## งานวิจัยที่เกี่ยวข้อง

Minghan Zhu & et al. (2022) ได้ศึกษาพัฒนาวิธีการตรวจจับยานพาหนะ 3 มิติ โดยใช้กล้องจราจรที่ไม่ได้ปรับเทียบค่า ผ่านทางโฮโมกราฟี งานวิจัยนี้เป็นการนำภาพจากกล้องจราจร (Traffic CCTV) มาแปลงข้อมูลภาพให้เป็นเปลี่ยนมุมมองเป็นภาพมุมสูง (Bird's Eye View) โดยใช้เทคนิคโฮโมกราฟี (Homography) ร่วมกับแผนที่ที่มีระบบอ้างอิงพิกัดบนพื้นโลก และทำการแปลงข้อมูลการตรวจจับยานพาหนะแบบ 2 มิติ จากกล้อง CCTV ให้เป็นแบบ 3 มิติ เพื่อนำข้อมูลส่วนด้านล่างของยานพาหนะและการใช้ค่าถดถอยแบบใหม่ที่เรียกว่า tailed r-box มาปรับปัญหาการบิดเบี้ยวให้เป็นขนาดตามประเภทยานพาหนะให้มีความใกล้เคียงมากที่สุด เมื่อแสดงผลร่วมกับข้อมูลภาพมุมสูงที่ผ่านกระบวนการสอบเทียบโฮโมกราฟี

Joseph Birkner (2023) ได้ศึกษาวิธีการตรวจจับวัตถุโดยใช้แผนที่ High-definition (HD) โดยทดสอบความแตกต่างของโมเดลจากชุดข้อมูลสี่แยก Providentia ใกล้กับเมือง Garching Hochbrück ของประเทศเยอรมัน โดยงานวิจัยนี้จะเน้นระบบเซนเซอร์ตรวจจับภาพสี RGB ที่ติดตั้งตามจุดต่างๆ ของถนน และนำมาวิเคราะห์แบบ 3 มิติ ที่จะนำไปใช้เป็นแบบโมเดลจำลองเสมือนจากวัตถุทางกายภาพ และนำแผนที่รายละเอียดสูง Map HD มาทดสอบในงานวิจัยเพื่อตรวจสอบถึงความถูกต้อง แม่นยำโดยรายละเอียดภาพของระบบเซนเซอร์ตรวจจับภาพสี RGB อยู่ที่ 1920 x 1200 pixels สามารถถ่ายรายละเอียดต่อเฟรมภาพได้ 300 -1200 ภาพต่อวินาที ในข้อมูลช่วงอากาศแจ่มใสในช่วงเวลากลางวันและใช้ข้อมูลขนาดเฟรมภาพ 619 ภาพต่อวินาที ในช่วงฝนตก และเป็นช่วงเวลากลางคืน และใช้กระบวนการแปลงข้อมูลภาพ 2 มิติเป็น การแบ่งส่วนภาพ Segmentation ด้วยเทคนิค Mask-RCNN เพื่อเข้าสู่ การสร้างเส้นคอนทัวร์ในส่วนด้านล่างของ 3 มิติ

Mahdi Rezaei, Mohsen Azarmi & Farzam Mohammad Pour Mir (2021) ได้ศึกษาวิจัยเกี่ยวกับ Traffic-Net: การตรวจสอบการจราจรแบบ 3 มิติโดยใช้กล้องวงจรปิดแบบตัวเดียว โดยใช้เทคโนโลยี computer vision ให้มีบทบาทสำคัญในระบบขนส่งอัจฉริยะ (ITS) และการเฝ้าระวังการจราจร ประกอบกับการเติบโตอย่างรวดเร็วยานพาหนะอัตโนมัติและเมืองที่มีผู้คนพลุกพล่าน ระบบจัดการจราจรอัตโนมัติและขั้นสูง (ATMS) โดยใช้กล้องวงจรปิดโครงสร้างพื้นฐานได้รับการพัฒนาโดยการนำ Deep Neural Networks ไปใช้ ในการวิจัยนี้ เราได้จัดเตรียมแพลตฟอร์มที่ใช้งานได้จริงสำหรับการตรวจสอบสภาพการจราจรแบบเรียลไทม์ ได้แก่ การตรวจจับยานพาหนะ/คนเดินถนนแบบ 3 มิติ การตรวจจับความเร็ว การประมาณวิถี การตรวจจับความแออัด ตลอดจนติดตามปฏิสัมพันธ์ของยานพาหนะและคนเดินถนน โดยใช้กล้องวงจรปิดตัวเดียว เราปรับ Deep Neural ของ YOLO V5 แบบกำหนดเองโมเดลเครือข่ายสำหรับการตรวจจับยานพาหนะ/คนเดินถนน และอัลกอริธึมการติดตาม SORT Tracking ที่ได้รับการปรับปรุง และเป็นครั้งแรกที่ใช้ภาพถ่ายจากดาวเทียมมาใช้วิธีการแมปเปอร์สเปคทีฟแบบผกผัน (SG-IPM) สำหรับการสอบเทียบอัตโนมัติของกล้องได้รับการพัฒนาเช่นกัน ซึ่งนำไปสู่วัตถุ 3 มิติ ที่แม่นยำการตรวจจับและการมองเห็น

นอกจากนี้เรายังพัฒนาการสร้างแบบจำลองการรับส่งข้อมูลแบบลำดับชั้นตามข้อมูลวิดีโอแบบระยะสั้นและระยะยาวเพื่อทำความเข้าใจสภาพการจราจร ปัญหาเส้นทางคับแคบ และจุดเสี่ยงสำหรับผู้ใช้ถนนกลุ่มเปราะบาง การทดลองหลายครั้งในสถานการณ์จริงและการเปรียบเทียบกับความล้ำสมัยจะดำเนินการโดยใช้ชุดข้อมูลการตรวจสอบการรับส่งข้อมูลที่หลากหลาย รวมถึง MIO-TCD, UA-DETRAC และ GRAM-RTM รวบรวมจากทางหลวง ทางแยก และเขตเมืองภายใต้แสงและสภาพอากาศที่แตกต่างกัน

Priyanka Ankireddy, V. Siva Krishna Reddy, Dr.V. Lokeswara Reddy (2022) ศึกษาการตรวจจับและติดตามยานพาหนะ โดยใช้ YOLOv8 และ การเรียนรู้เชิงลึก (Deep Learning) เพื่อเพิ่มคุณภาพการประมวลผลภาพ การพัฒนาเทคโนโลยีภาพดิจิทัลและเพิ่มความสามารถในการประมวลผลการจราจรที่ใช้เทคโนโลยีที่ทันสมัยมากมาย เช่น การตรวจจับการชนของยานพาหนะอัตโนมัติ การเตือนการออกนอกเลน ตัวควบคุมสัญญาณไฟจราจร และการประมาณค่าความหนาแน่นของการจราจร เป็นต้น การวิจัยนี้มีวัตถุประสงค์เพื่อประเมินประสิทธิภาพของ YOLOv5 และ YOLOv8 เพื่อสร้างวิธีการแบบเรียลไทม์ สำหรับการตรวจสอบอัตโนมัติที่ใช้เทคนิค multi-threading เพื่อจัดการสตรีมวิดีโอหลายรายการบน GPU แบบตัวเดียว ในส่วนของการติดตามยานพาหนะ ค้นหาลำดับรูปภาพที่สัมพันธ์ตรงกับประเภทของวัตถุให้ดีที่สุดจากนั้นจึงสำรวจสภาพแวดล้อมโดยรอบเพื่อหาคุณลักษณะที่สัมพันธ์กับรายการเหล่านั้น ผลจากการประมวลผล YOLOv8 มีค่าแม่นยำโดยเฉลี่ยในการจำแนกวัตถุ Mean Average Precision (mAP) อยู่ที่ 98.4 เปอร์เซ็นต์ อัตราความเร็วของเฟรมภาพต่อการตรวจจับอยู่ที่ 39 เฟรมต่อวินาที ใช้หน่วยความจำ 15.9 MB สรุปได้ว่าการระบุยานพาหนะและความแม่นยำในการติดตามมีประสิทธิภาพสูง

# บทที่ 3

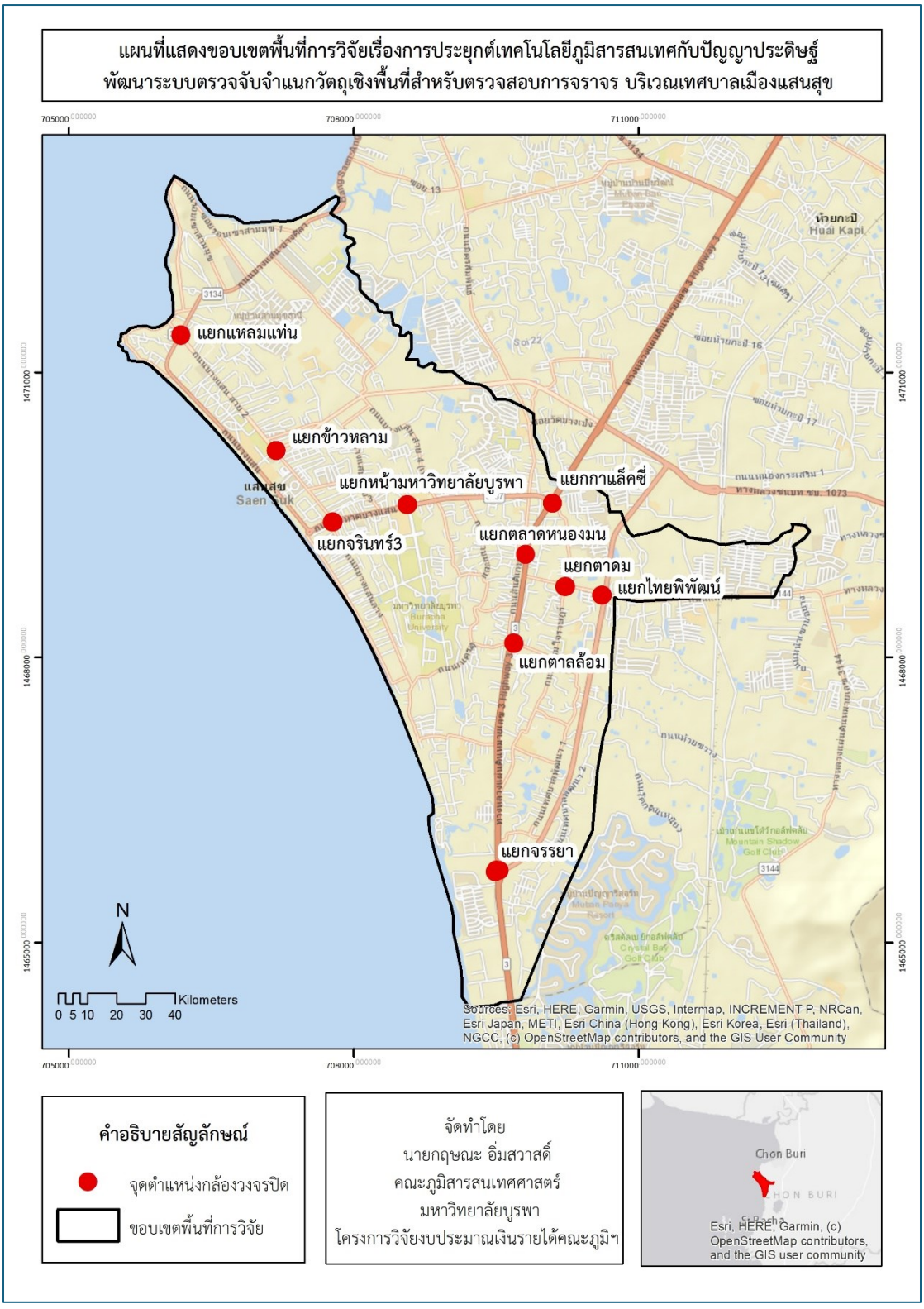
# วิธีดำเนินการวิจัย

## รูปแบบการศึกษา

การวิจัยนี้เป็นวิจัยที่ต้องการนำเทคโนโลยีปัญญาประดิษฐ์มาบูรณการกับระบบกล้องวงจรปิดของเทศบาลเมืองแสนสุข เพื่อพัฒนาให้สอดคล้องกับขับเคลื่อนด้วยเทคโนโลยีและนวัตกรรมของเมืองอัจฉริยะน่าอยู่ของเทศบาลเมืองแสนสุข โดยส่วนแรกเป็นการฝึกสอนข้อมูลยานพาหนะเพื่อสร้างเป็นชุดข้อมูล (Training Dataset) ส่วนที่สองการสร้างระบบตรวจจับ จำแนกประเภทยานพาหนะ นับจำนวนยานพาหนะ และส่วนที่สามสร้างแผนที่แบบจำลองการจราจร โดยงานวิจัยนี้จะเป็นการพัฒนาระบบต้นแบบเพื่อให้เริ่มกระบวนการเตรียมชุดข้อมูลยานพาหนะ รวมถึงเริ่มสร้างเป็นระบบตรวจสอบและจำแนกยานพาหนะแบบอัตโนมัติ และทดลองสร้างแบบจำลองแผนที่การจราจร

## พื้นที่ศึกษา

ขอบเขตพื้นที่ศึกษาบริเวณแยกในเขตเทศบาลเมืองแสนสุขที่มีกล้องวงจรปิดจำนวน 10 สถานที่ โดยแบ่งเป็นบริเวณแยกถนนลงหาดบางแสน 2 จุด คือ แยกหน้ามหาวิทยาลัยบูรพา และแยกจรินทร์ บริเวณถนนบางแสนสาย 2 จำนวน 2 จุด คือ แยกข้าวหลาม และแยกแหลมแท่น บริเวณถนนสุขุมวิท 4 จุด คือ แยกกาแล็คซี่ ตลาดหนองมนติดสุขุมวิท แยกตาลล้อม และแยกจรรยา และแยกถนนแสนสุข 2 จุด คือ แยกตาดม และแยกไทยพิพัฒน์ ดังภาพที่ 3-1 และการเข้าถึงข้อมูลกล้องวงจรปิดแบบ Realtime ทั้ง 10 จุด จากลิงก์ในตารางที่ 3-1




ภาพที่ 3-1 พื้นที่ขอบเขตการศึกษาบริเวณแยกในเขตพื้นที่เทศบาลเมืองแสนสุขที่สามารถเข้าถึงกล้องวงจรปิด

ตารางที่ 3-1 พื้นที่ศึกษาบริเวณแยกที่มีกล้องวงจรปิดในเขตเทศบาลเมืองแสนสุข

| ลำดับ | ชื่อสถานที่ | ลิงก์จากกล้องวงจรปิด CCTV |
|---|---|---|
| 1 | แยกกาแล็คซี่ | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=42&user=user&pass=pass |
| 2 | แยกหน้ามหาวิทยาลัยบูรพา | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=46&user=user&pass=pass |
| 3 | แยกจรินทร์ | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=49&user=user&pass=pass |
| 4 | แยกจรรยา | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=35&user=user&pass=pass |
| 5 | แยกตาดม | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=63&user=user&pass=pass |
| 6 | แยกไทยพิพัฒน์ | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=38&user=user&pass=pass |
| 7 | ตลาดหนองมนติดสุขุมวิท | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=64&user=user&pass=pass |
| 8 | แยกตาลล้อม | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=52&user=user&pass=pass |
| 9 | แยกข้าวหลาม | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=58&user=user&pass=pass |
| 10 | แยกแหลมแท่น | http://saensukcctv.citydata.in.th:5080/zm/cgi-bin/nph-zms?monitor=33&user=user&pass=pass |

## เครื่องมือและอุปกรณ์ที่ใช้ในการศึกษา

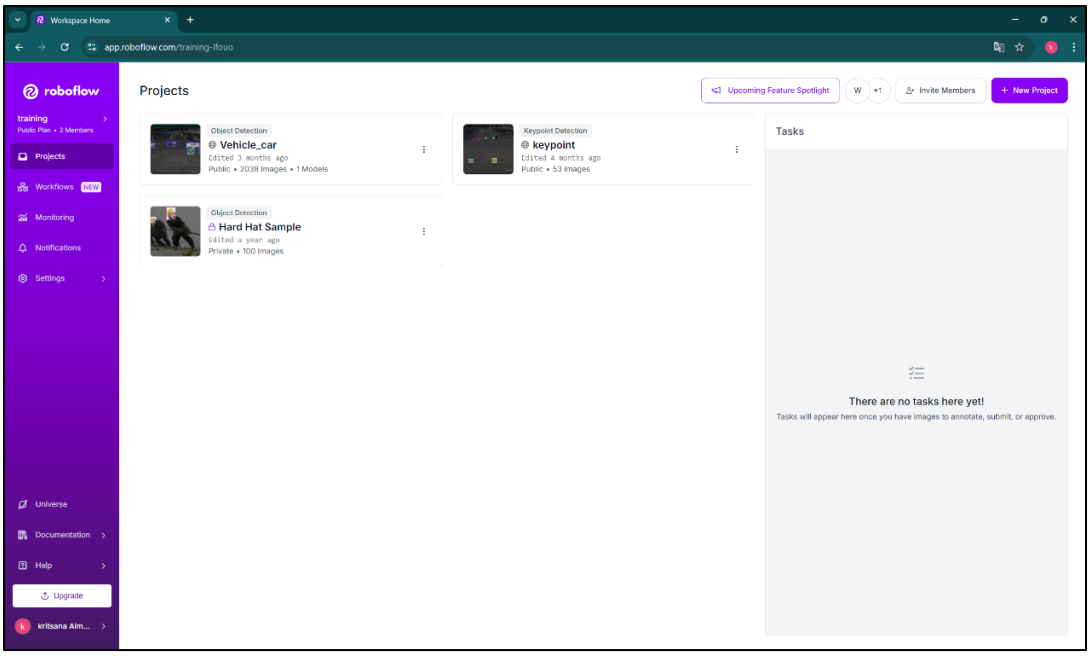
1. กล้องวงจรปิด CCTV ของเทศบาลเมืองแสนสุข
2. คอมพิวเตอร์ Laptop พร้อมด้วยการ์ดจอ NVIDIA GeForce RTX 4060 Laptop GPU
3. บริการเว็บของ Roboflow สำหรับการจำแนกข้อมูลประเภทยานพาหนะ
4. บริการเว็บของ Google Colab สำหรับการฝึกสอนข้อมูล (Training data)
5. โปรแกรม Visual Studio Code
6. OpenCV version 4.10.0.84
7. Python version 3.9.1
8. QGIS version 3.34.3
9. ภาพถ่ายจากดาวเทียมจากโปรแกรม Google Earth Pro

## วิธีการดำเนินงาน

พัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV ในเขตเทศบาลเมืองแสนสุข โดยใช้เทคนิคการเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) ส่วนวิธีการดำเนินการขั้นตอนนี้แบ่งออกเป็น 7 ขั้นตอน ดังนี้

1. ขั้นตอนการเตรียมข้อมูลภาพยานพาหนะแต่ละประเภทโดยเป็นภาพจากกล้องวงจรปิด CCTV ของเทศบาลเมืองแสนสุข

2. นำข้อมูลภาพจากกล้องวงจรปิดเข้าแพลตฟอร์มปัญญาประดิษฐ์ Roboflow ที่ให้บริการบนเว็บไซต์ https://app.roboflow.com/ โดยทำการสมัครสมาชิกและสร้างโปรเจค และนำเข้าไฟล์วิดีโอจากกล้องวงจรปิดเพื่อทำการแบ่งเฟรมวิดีโอให้เป็นภาพนิ่งแต่ละเฟรมและทำการอัพโหลดไฟล์เพื่อไปขั้นตอนการระบุประเภทให้กับวัตถุบนภาพ Annotate ดังภาพที่ 3-2




ภาพที่ 3-2 แสดงพื้นที่การทำงานโดยการสร้าง Project บน Roboflow

3. การระบุประเภทยานพาหนะในขั้นตอนนี้เรียกว่า Annotate บน Roboflow โดยทำการสร้างขอบเขตคลุมบนวัตถุบนภาพเพื่อระบุประเภทยานพาหนะจำนวนชุดข้อมูล 2,037 ชุด โดยในงานวิจัยครั้งนี้แยกประเภทพาหนะดังตารางที่ 3-2

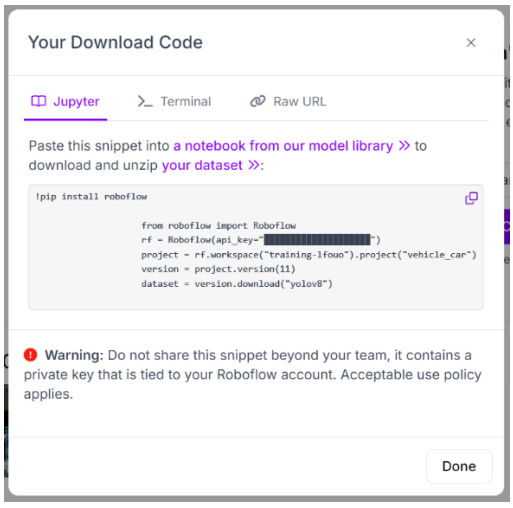
ตารางที่ 3-2 การกำหนด Annotate บน Roboflow ของยานพาหนะแต่ละประเภท

| Annotation บน Roboflow | ชื่อยานพาหนะ | ประเภทของยานพาหนะ |
|---|---|---|
| MPV | Multi Purpose Van | รถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 คน |
| Motorcycle | | รถจักรยานยนต์ |
| PPV | Pick-Up Passenger Vehicle | รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ |

ตารางที่ 3-2 (ต่อ)

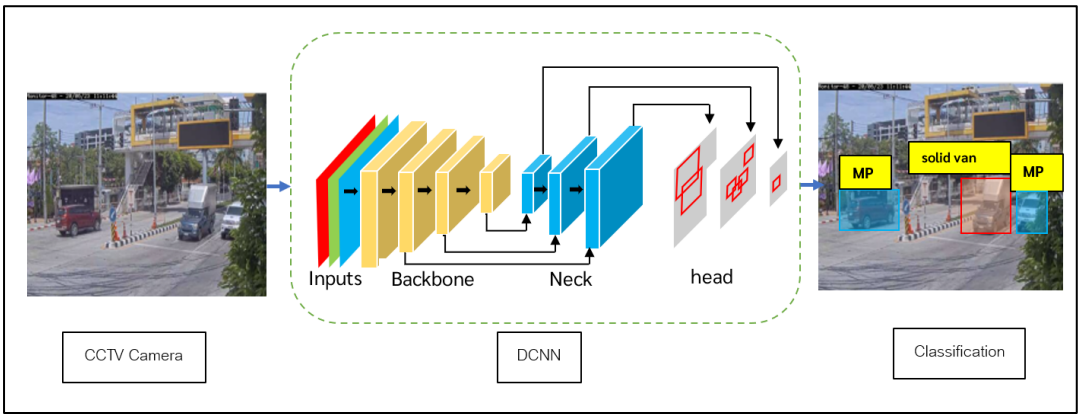
| Annotation บน Roboflow | ชื่อยานพาหนะ | ประเภทของยานพาหนะ |
|---|---|---|
| SUV | Sport Utility Vehicle | รถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 ที่นั่ง |
| Sidecar | Sidecar | รถมอเตอร์ไซต์พ่วงข้าง |
| Solid box pick-up | Solid box pick-up | รถกระบะประเภทขนส่งแบบตู้ทึบ |
| Songthaew | Songthaew | รถขนส่งสาธารณะแบบท้องถิ่น เรียกว่า สองแถว |
| Bus | Bus | รถประจำทาง |
| Pick-up | Pick-up | รถกระบะ |
| Sedan | Sedan | รถยนต์นั่งส่วนบุคคล หรือรถเก๋งที่มี 4 ประตู |
| Truck | Truck | รถบรรทุกขนของหรือสินค้า |
| Van | Van | รถตู้ |

4. การระบุประเภทยานพาหนะเรียบร้อยแล้ว ทำการส่งชุดข้อมูลไปฝึกฝนการเรียนรู้ (training dataset) โดยทำการส่งออกเป็นแบบ code ในรูปแบบ YOLOV8 ดังภาพที่ 3-3




ภาพที่ 3-3 การส่งออก code ผลลัพธ์ของการทำ Annotate บน Roboflow นำไปใช้บน Google Colab

5. การวิเคราะห์ข้อมูลฝึกฝนการเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) ซึ่งการจำแนกยานพาหนะใช้สถาปัตยกรรมของ YOLOV8 ที่แบ่งโครงสร้างประกอบด้วย backbone neck และ head ดังภาพที่ 3-4




ภาพที่ 3-4 โครงสร้างสถาปัตยกรรมโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN)

การวิเคราะห์บน Google Colab โดยใช้ภาษาไพธอนในการดำเนินการและสามารถใช้ทรัพยากรของ GPU Google Compute Engine เพื่อช่วยในการวิเคราะห์ข้อมูลรวดเร็วมากยิ่งขึ้น โดยกำหนดตัว runtime เป็น Python 3 ตัวฮาร์ดแวร์เป็น T4 GPU โดยกำหนดค่าพารามิเตอร์แบบฝึกสอนข้อมูลการวิเคราะห์การจำแนกแบบจำลองยานพาหนะบน YOLOV8 ตามชุดคำสั่ง !yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True ดังตารางที่ 3-3

ตารางที่ 3-3 แสดงการกำหนดค่าพารามิเตอร์ในการวิเคราะห์แบบฝึกสอนในระบบ YOLOV8 บน Google Colab

| พารามิเตอร์ | ค่าที่กำหนด | คำอธิบาย |
|---|---|---|
| **mode** | train | เลือกโหมดแบบฝึกสอนเป็น train |
| **model** | yolov8m.pt | เป็นโมเดลของ yolov8 กำหนดเป็น yolov8s.pt |
| **data** | {dataset.location}/data.yaml | กำหนดไปที่แหล่งเก็บไฟล์ data.yaml ได้มาจากการรันคำสั่ง roboflow บน google colab |
| **epochs=50** | 50 | การ train model โดยดึงข้อมูลเข้า Model โดยใช้ข้อมูลใน Dataset ที่มีอยู่จนครบทุกตัว เป็นจำนวน 50 ครั้ง |

ตารางที่ 3-3 (ต่อ)

| พารามิเตอร์ | ค่าที่กำหนด | คำอธิบาย |
|---|---|---|
| **imgsz** | 640*640 | การกำหนดขนาดของภาพเป็น 640 x 640 px |
| **plots** | true | แสดงผลลัพธ์ผ่านฟังก์ชั่น plots |

หลังจากวิเคราะห์แบบจำลองเสร็จจะมีการสร้างไฟล์ best.pt และ last.pt จากระบบ YOLOV8 ผลลัพธ์จากการจำแนกประเภทยานพาหนะแสดงผลดังภาพที่ 3-5


ภาพที่ 3-5 แสดงผลลัพธ์การเรียนรู้จากการฝึกฝนชุดข้อมูลแบบจำลอง YOLOV8S จำนวน 50 รอบ

6. ขั้นตอนการพัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV พัฒนาบนภาษาไพธอน โดยใช้โปรแกรม Visual Studio Code สำหรับการเขียนโปรแกรมพัฒนาระบบตรวจจับจำนวนยานพาหนะใช้ OpenCV เป็นไลบารีหลัก เช่น การตรวจจับจำแนกประเภทยานพาหนะจากแบบจำลอง YOLOV8 การวาดเส้นลงบนระบบตรวจจับเพื่อกำหนดระยะการนับโดยกำหนดเงื่อนไขซ้อนทับเมื่อมีวัตถุผ่านเส้นระบบจะเริ่มการนับจำนวนยานพาหนะแต่ละประเภท โดยไลบารีในขั้นตอนส่วนนี้มีดังตารางที่ 3-4

ตารางที่ 3-4 แสดงไลบารีหลักที่นำมาใช้ในการสร้างระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV

| ชื่อไลบารี | หน้าที่การทำงาน |
|---|---|
| **cv2** | Opencv version 2 ใช้สำหรับการอ่าน วิเคราะห์การตรวจจับวัตถุบนภาพ การต่อภาพของจุดที่เหมือนกัน Homography การสร้างการวาดเส้นบนภาพ และแสดงผลสามารถรองรับข้อมูลภาพและวิดีโอ |

ตารางที่ 3-4 (ต่อ)

| ชื่อไลบารี | หน้าที่การทำงาน |
|---|---|
| **streamlit** | library สำหรับการสร้าง Data Web Application อย่างเร็วและง่าย เพื่อใช้สำหรับการทำ Production หรือ Prototype ของเหล่า Data Expert |
| **ultralytics** | Ultralytics ใช้สำหรับการการอ่าน การคาดการณ์วัตถุการตรวจจับจากแบบจำลองรองรับไฟล์ .pt |
| **torch** | Pytorch ใช้สำหรับเรียกการทำงาน GPU ของคอมพิวเตอร์ร่วมประมวลผลการทำ Deep Learning |
| **numpy** | Numpy มีความสามารถในการจัดการกับอาเรย์หลายมิติและข้อมูลแบบเมทริกซ์ |

7. การสร้างแบบจำลองแผนที่การจราจรบนถนน (Digital replica of the road mapping) นำผลลัพธ์ของการตรวจจับยานพาหนะมาสร้างแผนที่เป็นภาพมุมสูง (BEV) Bird's Eye View วิธีแบบโฮโมกราฟีแมทริกซ์ Homography Matrix หาจากคู่จุดที่ตรงกัน และนำมาใช้คำนวณ Homography matrix $H$ ซึ่งเป็นเมตริกซ์ขนาด 3×3 ที่เปลี่ยนแปลงพิกัดของจุดในภาพแรกไปยังพิกัดของจุดที่ตรงกันในภาพที่สอง โดยมีสมการที่ 3-1 ดังนี้

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ w \end{bmatrix} \qquad (3\text{-}1)$$

โดย $(x,y,w)$ เป็นพิกัดของจุดในภาพแรก และ $(x',y',w')$ เป็นพิกัดของจุดที่ตรงกันในภาพที่สอง หลังการคำนวณจะทำการ normalize ค่า $w'$ ให้เป็น 1 เพื่อได้ค่าพิกัด $(x',y')$

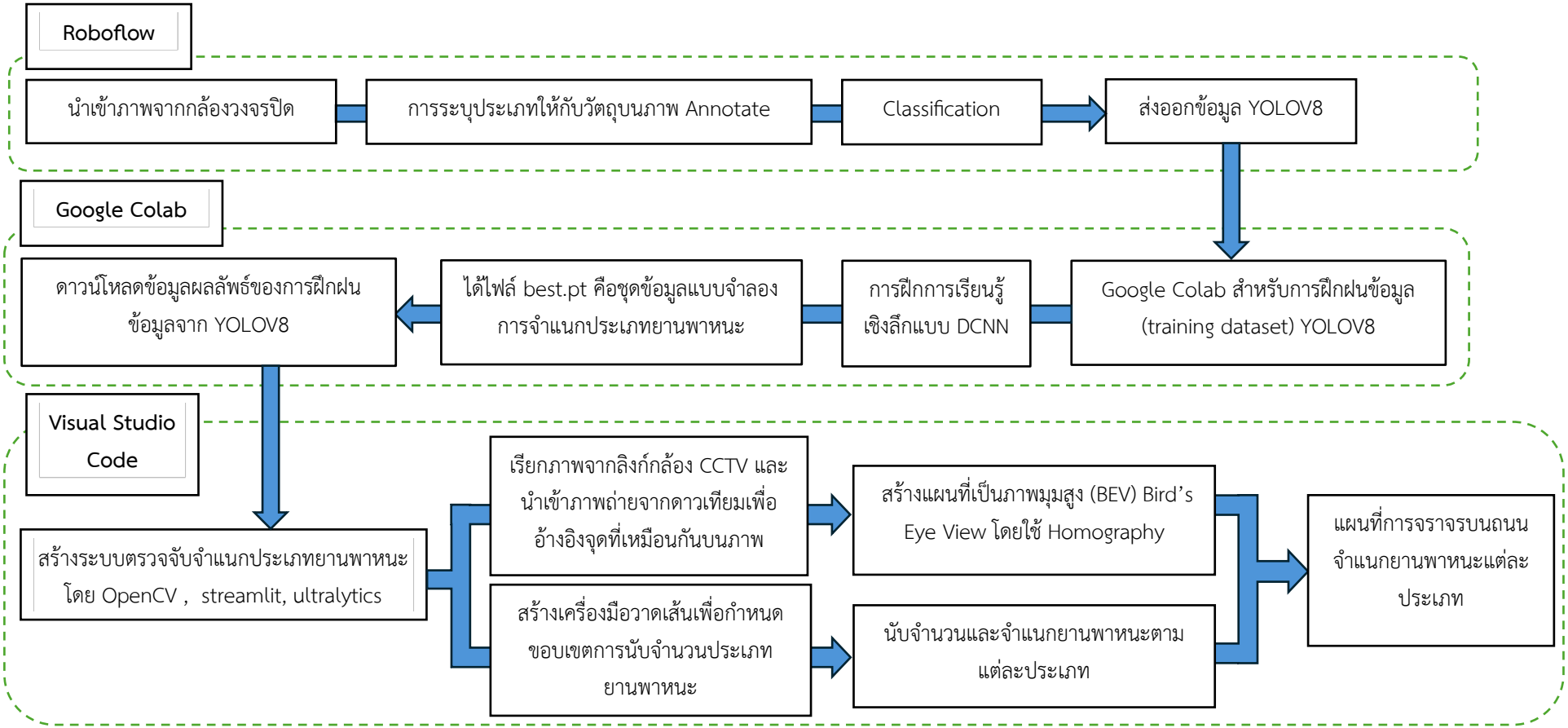
จากนั้นเข้ากระบวนการสอบเปรียบตำแหน่งของจุดภาพให้มีความใกล้เคียงกันมาที่สุด โดยใช้สมการที่ 3-2 การสอบเทียบ

$$H_{ori}^{bev} = H_{world}^{bev}\ H_{bev}^{world} \quad (3\text{-}2)$$

โดย $sp_a = H_b^a p_b$ แสดงถึงพิกัดของ $H_b^a$ บนแผนที่ในเฟรม $b$ เพื่อจุดประสานความเชื่อมต่อในเฟรม $a$ จนไปถึงค่าสเกลเฟคเตอร์ (scale factor) $s$, และ $p = [x, y, 1]^T$ คือพิกัดต่อเนื่องของ $a$ ชี้ไปยังพื้นราบ $bev$ คือ ระนาบของภาพมุมสูง (Bird's Eye View) $world$ คือ ค่าพื้นระนาบของพื้นผิวโลก $ori$ คือ ภาพจากกล้อง CCTV $H_{world}^{bev}$ ผู้ใช้สามารถกำหนดได้อย่างอิสระที่มีการเปลี่ยนแปลงที่คล้ายคลึงกันโดยคงมุมไว้ระหว่างระนาบบนถนนจริงและมุมมองจากมุมสูงของภาพถ่ายดาวเทียม จำเป็นที่ต้องมีการสอบเทียบค่า $H_{ori}^{world}$ หมายถึง ภาพจากกล้อง CCTV $(ori)$ และ ข้อมูลภาพถ่ายจากดาวเทียมที่มีระบบอ้างอิงตามพิกัดตำแหน่งโลก ($world$) (Minghan Zhu & et al., 2022)

โดยใช้ไลบารีของ OpenCV โดยหาจุดที่เหมือนกันของทั้ง 2 ภาพ คือ ภาพวิดีโอระบบตรวจจับยานพาหนะจากกล้องวงจรปิดและภาพถ่ายจากดาวเทียมบริเวณเดียวกัน เพื่อทำการแปลงภาพวิดีโอให้เป็นแนวดิ่งหรือภาพมุมสูง จากนั้นแปลงกรอบสี่เหลี่ยมที่ตรวจจับยานพาหนะให้เป็นจุดเพื่อแสดงเป็นแผนที่จำลองการจราจรของยานพาหนะบนถนนได้ จากขั้นตอนที่กล่าวมาข้างต้นแสดงขั้นตอนแบบผังขั้นตอนการวิจัยดังภาพที่ 3-6

**ผังขั้นตอนการวิจัย**




ภาพที่ 3-6 ผังขั้นตอนการวิจัย

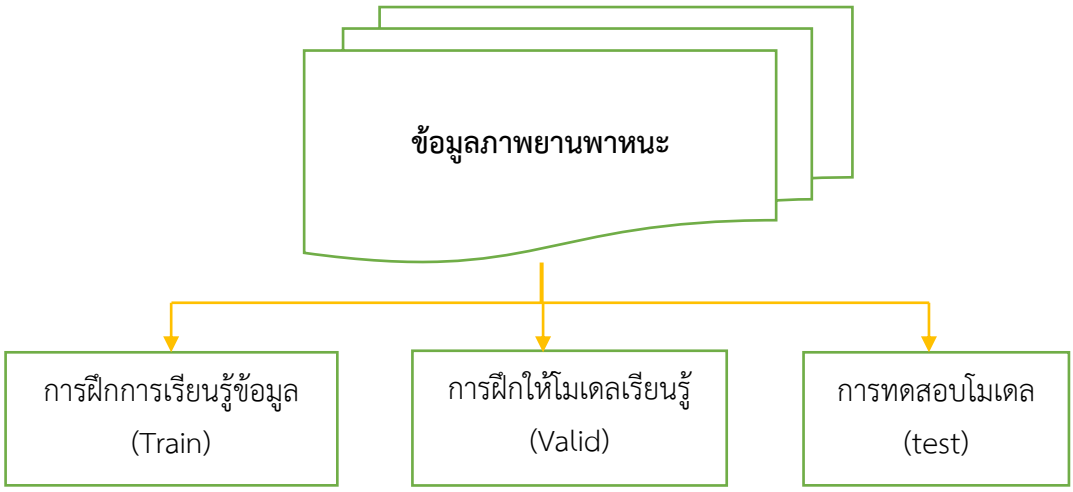
# บทที่ 4

# ผลการศึกษา

ผลการศึกษาแบ่งออกเป็น 4 ส่วน ได้แก่ ส่วนการเตรียมข้อมูลฝึกการเรียนรู้ ส่วนการเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) จาก YOLOV8 ส่วนการตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้องวงจรปิด CCTV และส่วนสร้างแบบจำลองการจราจรบนถนน บริเวณเทศบาลเมืองแสนสุข

## การเตรียมข้อมูลฝึกการเรียนรู้

การฝึกฝนข้อมูล training เป็นส่วนที่สำคัญในการฝึกการเรียนรู้ของระบบการจำแนกวัตถุ โดยนำข้อมูลภาพยานพาหนะทั้ง 12 ประเภท ได้แก่ รถยนต์นั่งส่วนบุคคล 4 ที่นั่ง (Sedan), รถอเนกประสงค์ ที่สามารถนั่งได้ประมาณ 5 - 7 คน (MPV), รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ (PPV), รถอเนกประสงค์ประเภทครอบครัว (SUV), รถสองแถว (songthaew), รถตู้ (Van), รถประจำทาง (Bus), รถมอเตอร์ไซต์ (Motorcycle), รถมอเตอร์ไซต์พ่วงข้าง (Sidecar), รถกระบะ (Pick-up), รถกระบะตู้ทึบ (Solid box pickup truck), รถบรรทุก (Truck) อัพโหลดไฟล์ภาพยานพาหนะรวมทั้งหมด 2,037 ภาพ และเข้ากระบวนการใส่ป้ายกำกับวัตถุในภาพเพื่อเตรียมข้อมูลไปสร้างเป็นแบบฝึกสอนให้กับแบบจำลอง โดยระบบ Roboflow จะแบ่งชุดการทำงานข้อมูล (Dataset Split) ข้อมูลจำนวนการฝึกการเรียนรู้ 3,011 ข้อมูล เพื่อวิเคราะห์ตามแบบฝึกสอนและสร้างแบบจำลองการจำแนกประเภทยานพาหนะ ดังภาพที่ 4-1




ภาพที่ 4-1 ขั้นตอนการแบ่งข้อมูลวิเคราะห์แบบฝึกสอน (Train validation test split data) ดัดแปลงมาจาก (Surapong Kanoktipsatharporn, 2019)

โดยแบ่งชุดการทำงานข้อมูลภาพใน Roboflow เพื่อเตรียมไปสร้างเป็นแบบจำลองการจำแนกยานพาหนะใน Google Colab ดังตารางที่ 4-1

ตารางที่ 4-1 การแบ่งชุดการทำงานข้อมูลภาพใน Roboflow เพื่อเตรียมไปสร้างเป็นแบบจำลองการจำแนกยานพาหนะใน Google Colab

| ประเภทยานพาหนะ | TRAIN SET | | VALID SET | | TEST SET | | รวม (ภาพ) |
|---|---|---|---|---|---|---|---|
| | จำนวนภาพ | ร้อยละ | จำนวนภาพ | ร้อยละ | จำนวนภาพ | ร้อยละ | |
| **รถยนต์นั่งส่วนบุคคล 4 ที่นั่ง** | 583 | 71.53 | 161 | 19.75 | 71 | 8.71 | 815 |
| **รถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 คน** | 86 | 68.25 | 23 | 18.25 | 17 | 13.49 | 126 |

ตารางที่ 4-1 (ต่อ)

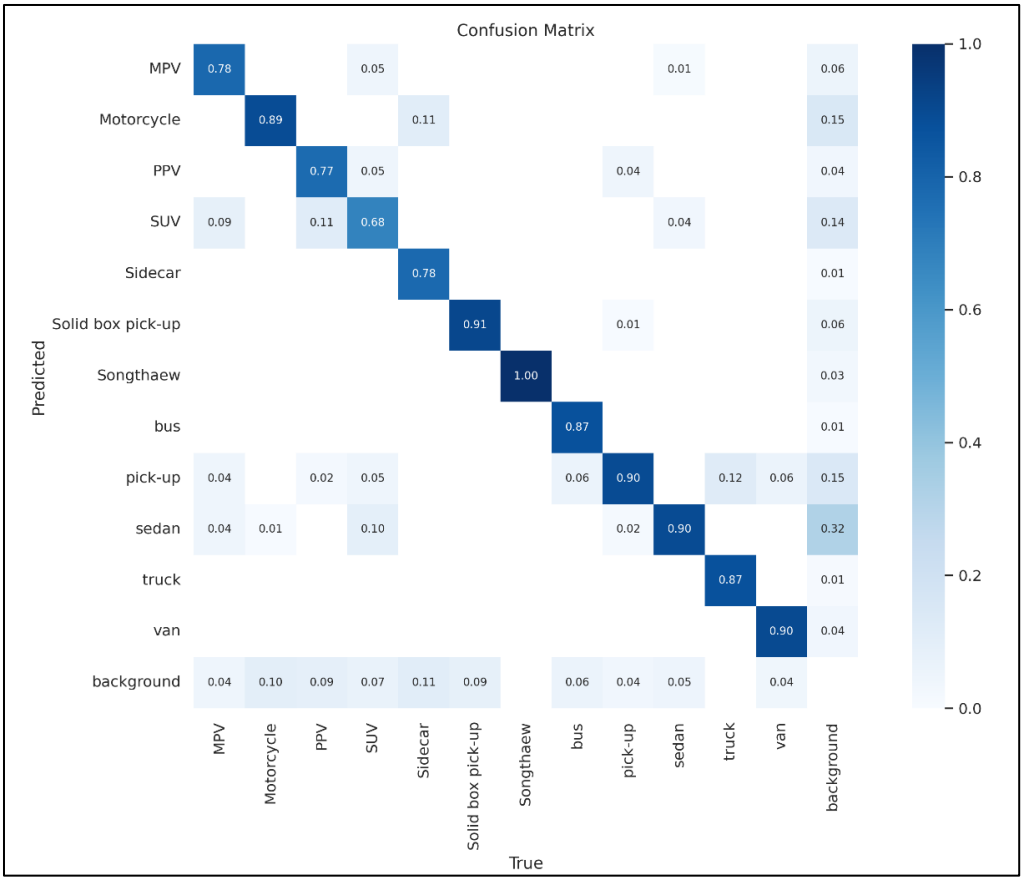
| ประเภทยานพาหนะ | TRAIN SET | | VALID SET | | TEST SET | | รวม (ภาพ) |
|---|---|---|---|---|---|---|---|
| | จำนวนภาพ | ร้อยละ | จำนวนภาพ | ร้อยละ | จำนวนภาพ | ร้อยละ | |
| **รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ** | 165 | 73.01 | 41 | 18.14 | 20 | 8.85 | 226 |
| **รถอเนกประสงค์ประเภทครอบครัว** | 20 | 25.64 | 38 | 48.72 | 20 | 25.64 | 78 |
| **รถสองแถว** | 18 | 31.58 | 21 | 36.84 | 18 | 31.58 | 57 |
| **รถตู้** | 132 | 68.04 | 41 | 21.13 | 21 | 10.82 | 194 |
| **รถประจำทาง** | 86 | 76.79 | 15 | 13.39 | 11 | 9.82 | 112 |
| **รถมอเตอร์ไซต์** | 363 | 65.76 | 103 | 18.66 | 56 | 10.14 | 522 |
| **รถมอเตอร์ไซต์พ่วงข้าง** | 38 | 70.37 | 9 | 16.67 | 7 | 12.96 | 54 |
| **รถกระบะ** | 449 | 72.07 | 119 | 19.10 | 55 | 8.83 | 623 |
| **รถกระบะตู้ทึบ** | 109 | 71.24 | 23 | 15.03 | 21 | 13.73 | 153 |
| **รถบรรทุก** | 32 | 62.75 | 15 | 29.41 | 4 | 7.84 | 51 |
| **รวม** | 2,081 | | 609 | | 321 | | 3,011 |

## การฝึกการเรียนรู้โครงข่ายประสาทเทียมเชิงลึกแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) บน YOLOV8

ผลลัพธ์จากการวิเคราะห์การจำแนกประเภทยานพาหนะเพื่อสร้างแบบจำลองโดยใช้ YOLOV8 โดยโครงสร้างแบบจำลองการจำแนกประเภทยานพาหนะเป็นกระบวนการเรียนรู้เชิงลึกโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) ที่อยู่ใน YOLOV8 โดยใช้คำสั่งในการฝึกฝนข้อมูลดังนี้

```
!yolo task=detect mode=train model=yolov8m.pt
data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True
```

โดยผลลัพธ์จากการจำแนกได้เปรียบเทียบสัดส่วนค่าความถูกต้องของการจำแนกภาพจากการคาดการณ์และภาพจริง ซึ่งและแสดงเป็นตารางภาพ Confusion Matrix เพื่อตรวจสอบความถูกต้องของการแยกประเภทยานพาหนะได้ถูกต้องอยู่ในช่วง 0.68 - 1 โดยค่าคาดการณ์ความถูกต้องแม่นยำของการจำแนกยานพาหนะที่มีค่าระหว่าง 0.68 - 0.78 คือ ประเภทรถอเนกประสงค์ประเภทครอบครัว (SUV), รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ (PPV) และ รถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 คน (MPV), ประเภทรถมอเตอร์ไซด์พ่วงข้าง (Sidecar) ส่วนค่าระหว่าง 0.87- 0.90 คือ รถประจำทาง (Bus), รถบรรทุก (truck), รถกระบะ (pick-up), รถยนต์นั่งส่วนบุคคล 4 ที่นั่ง (sedan), รถมอเตอร์ไซด์ (Motorcycle), รถตู้ (van) และค่า 0.91 - 1 คือ รถกระบะขนส่งตู้ทึบ (Solidbox pick-up), รถสองแถว (Songthaew) ดังภาพที่ 4-2



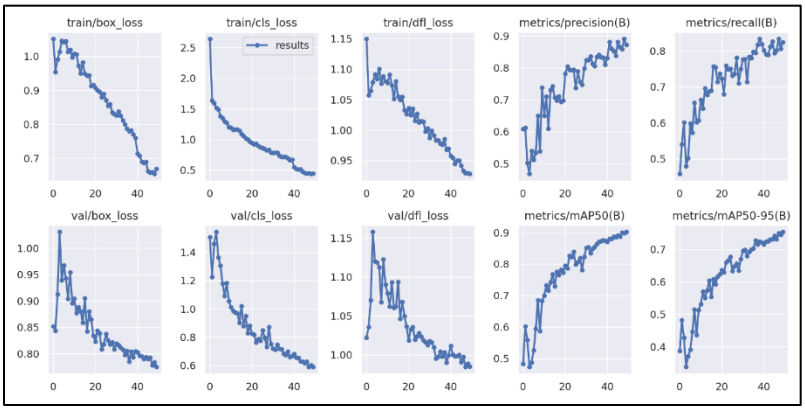

ภาพที่ 4-2 ค่าความถูกต้องจากการจำแนกยานพาหนะระหว่างภาพจริงเปรียบเทียบกับการจำแนกจากการคาดการณ์แสดงในรูปแบบ Confusion Matrix

ค่าเฉลี่ยความแม่นยำ (Mean Average Precision หรือ mAP) เป็นตัวชี้วัดที่สำคัญในการตรวจจับวัตถุ ซึ่งประเมินประสิทธิภาพของโมเดลโดยพิจารณาทั้งความแม่นยำ (precision) และการเรียกคืน (recall) ในหลายคลาสของวัตถุ โดยเฉพาะอย่างยิ่ง mAP50 จะเน้นที่ค่าขีดแบ่ง IoU (Intersection over Union) ที่ 0.5 ซึ่งวัดว่าโมเดลสามารถระบุวัตถุที่มีการซ้อนทับกันได้ดีคะแนน mAP50 ที่สูงขึ้น บ่งชี้ถึงประสิทธิภาพโดยรวมที่ดีกว่าโดยค่าหลังการทดสอบอยู่ประมาณ 0.87 – 0.90 และ ค่า mAP50 และ mAP50-95 เปรียบเทียบกัน ช่วยในการประเมินประสิทธิภาพของโมเดลในคลาสหลายประเภทและหลายสภาวะ โดยใช้ข้อมูลความแม่นยำในการตรวจจับวัตถุในการพิจารณาการแลกเปลี่ยนระหว่างความแม่นยำ (precision) และความครบถ้วน (recall) โดยโมเดลที่มีคะแนน mAP50 และ mAP50-95 สูงกว่าจะมีความน่าเชื่อถือมาก จากภาพที่ 4.3 แสดงค่า mAP50 สูงสุดประมาณ 0.87 – 0.90 และค่า mAP50-95 สูงสุดประมาณ 0.70- 0.79 เป็นค่าที่มีความน่าเชื่อถืออยู่ในระดับดี และเหมาะสมสำหรับการใช้งานที่ต้องการความแม่นยำของการจำแนกยานพาหนะได้

ในส่วนการฝึกฝนโมเดลการตรวจจับวัตถุฟังก์ชันการสูญเสีย (loss functions) มีบทบาทสำคัญอย่างยิ่ง ฟังก์ชันการสูญเสียทำหน้าที่วัดความแตกต่างระหว่างกรอบสี่เหลี่ยมที่ทำนาย (predicted bounding boxes) กับข้อมูลที่ถูกกำกับไว้ (ground truth annotations) ซึ่งเป็นการวัดว่าโมเดลกำลังเรียนรู้ได้ดีเพียงใดระหว่างการฝึกฝน องค์ประกอบของฟังก์ชันการสูญเสียที่พบบ่อยในการตรวจจับวัตถุดังนี้

ค่า box_loss แสดงค่าความผิดพลาดในการตีกรอบยานพาหนะที่ทำนายได้ เท่ากับ 0.65423 (ค่าเข้าใกล้ 0 ยิ่งเป็นค่าที่มีความแม่นยำ)

ค่า cls_loss s แสดงค่าความผิดพลาดในการทำนายยานพาหนะ เท่ากับ 0.43908 (ค่าเข้าใกล้ 0 ยิ่งเป็นค่าที่มีความแม่นยำ) ดังภาพที่ 4-3




ภาพที่ 4-3 แสดงกราฟของ Mean Average Precision (mAP) และ loss หลังจากการฝึกการเรียนรู้ (training) ของ YOLOV8 บนแบบจำลอง YOLOV8M.pt เพื่อจำแนกประเภทยานพาหนะ

การตรวจสอบความแม่นยำและความถูกต้องของโมเดลผ่านกระบวนการวิเคราะห์บนแบบจำลอง YOLOV8M และได้แบบจำลอง best.pt และข้อมูล data.yaml เป็นแบบจำลองการตรวจจับประเภทยานพาหนะ โดยนำแบบจำลองที่ได้ทดสอบความแม่นยำและถูกต้อง โดยสามารถตรวจสอบจากคำสั่งภาษาไพธอนดังนี้

!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt

data={dataset.location}/data.yaml

ผลลัพธ์ที่ได้จากการทดสอบความแม่นยำ และความถูกต้องของแบบจำลอง YOLOv8 จากข้อมูล Valid set ดังตารางที่ 4-2

ตารางที่ 4-2 ตารางแสดงความแม่นยำ และความถูกต้องของแบบจำลอง YOLOv8 จากข้อมูล Valid set

| Class | Images | Instances | Box | | | |
|---|---|---|---|---|---|---|
| | | | P | R | mAP50 | mAP50-95 |
| MPV | 408 | 23 | 0.833 | 0.783 | 0.858 | 0.806 |
| Motorcycle | 408 | 196 | 0.943 | 0.764 | 0.923 | 0.616 |
| PPV | 408 | 44 | 0.841 | 0.773 | 0.855 | 0.759 |
| SUV | 408 | 41 | 0.622 | 0.683 | 0.701 | 0.609 |
| Sidecar | 408 | 9 | 0.971 | 0.889 | 0.888 | 0.753 |
| Solid box pick-up | 408 | 23 | 0.776 | 0.826 | 0.925 | 0.825 |
| Songthaew | 408 | 24 | 0.855 | 0.917 | 0.979 | 0.765 |
| bus | 408 | 16 | 0.9 | 0.812 | 0.94 | 0.834 |
| pick-up | 408 | 162 | 0.866 | 0.827 | 0.931 | 0.792 |
| sedan | 408 | 244 | 0.87 | 0.865 | 0.932 | 0.773 |
| truck | 408 | 16 | 0.933 | 0.866 | 0.91 | 0.701 |
| van | 408 | 51 | 0.965 | 0.824 | 0.961 | 0.808 |
| all | 408 | 849 | 0.865 | 0.819 | 0.9 | 0.753 |

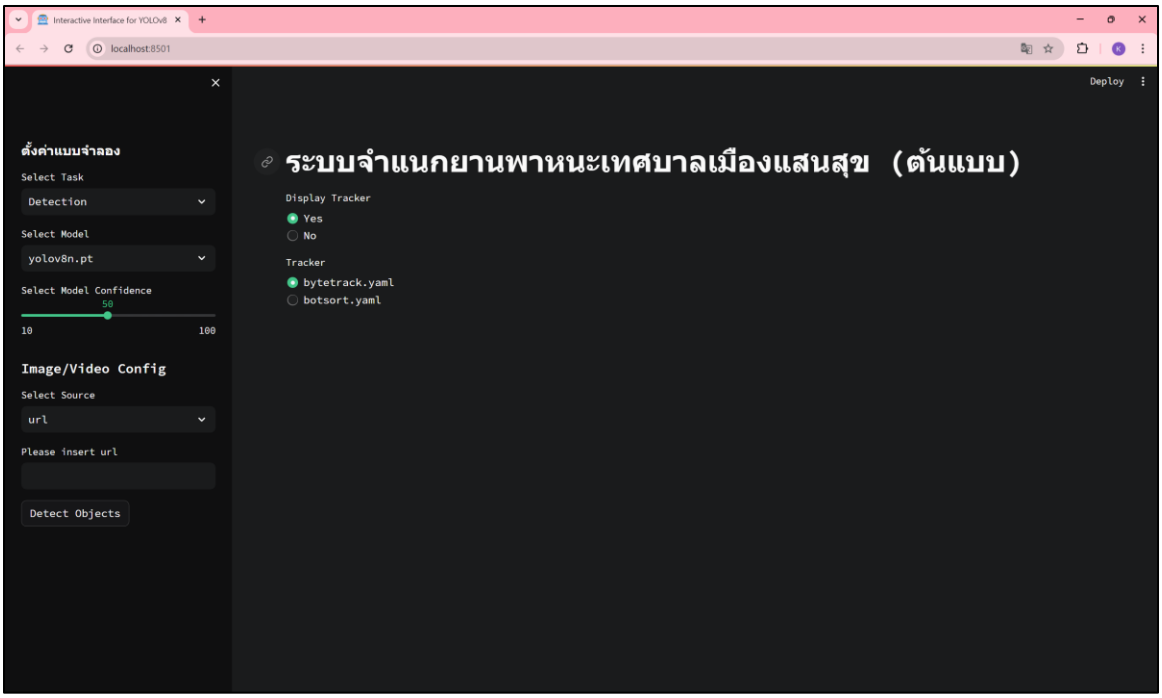
ผลจากการฝึกฝนข้อมูลเชิงลึกโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) ที่อยู่ในแบบจำลอง YOLOV8M จำแนกประเภทยานพาหนะโดยค่าความถูกต้องและแม่นยำรวมมากกว่า 0.8 ส่วนค่าความถูกต้องและแม่นยำที่น้อยกว่า 0.5 มีผลจากขนาดของยานพาหนะ ระยะตำแหน่งวัตถุบนภาพและความมืดของช่วงเวลาจากกล้องวงจรปิด ดังภาพที่ 4-4


ภาพที่ 4-4 ผลลัพธ์การคาดการณ์จากการฝึกฝนการเรียนรู้เชิงลึกการจำแนกประเภทยานพาหนะ

**ส่วนการตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้องวงจรปิด CCTV**

การสร้างพัฒนาต้นแบบของระบบตรวจจับและจำแนกประเภทยานพาหนะใช้ภาษาไพธอนในการเขียน โดยใช้ไลบารี OpenCV และ Streamlit ในการพัฒนาโปรแกรมตรวจจับจำนวนและจำแนกยานพาหนะโดยใช้แบบจำลองที่ได้จากการเรียนรู้เชิงลึกจาก YOLOV8 โดยการทำงานของระบบนี้จะแบ่งการทำงานออกเป็น 2 ส่วน คือ

1. ส่วนแรกการเรียกการใช้งานผ่าน Streamlit บน VScode โดยใช้คำสั่ง streamlit run app.py (ชื่อโปรแกรมชื่อ app.py) ระบบจะทำงานบน Web browser โดยจำลองเป็น http://localhost:8501/ และเลือกแบบจำลองแบบเพื่อการจำแนกยานพาหนะและวางลิงก์กล้องวงจรปิดในช่อง URL ดังภาพที่ 4-5

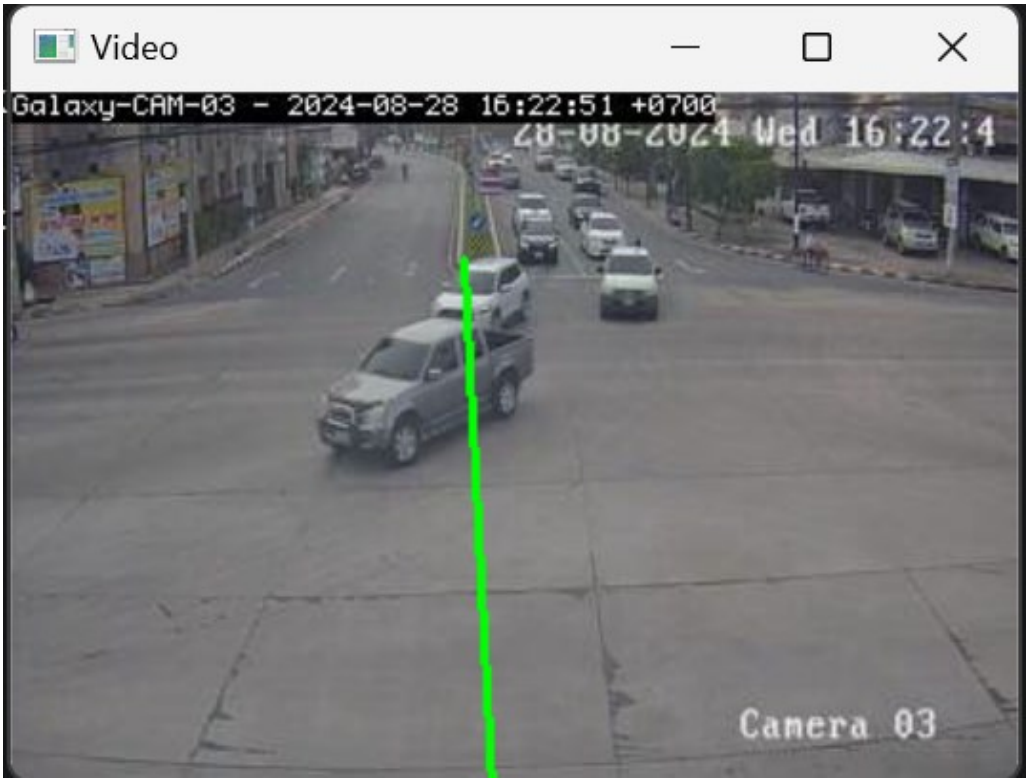



ภาพที่ 4-5 ภาพตัวอย่างเมื่อเข้าสู่การทำงานระบบการจำแนกยานพาหนะโดยฝั่งซ้ายเป็นการใส่พารามิเตอร์ เช่น การเลือกแบบจำลอง การกำหนดค่าความเชื่อมั่น และการใส่ลิงก์จากกล้องวงจรปิด

2. ส่วนการนับจำนวนประเภทยานพาหนะในการพัฒนาระบบการนับจำนวนและจำแนกนี้สามารถกำหนดลากเส้นลงบนหน้าต่างแสดง video โดยใช้เงื่อนไข for สำหรับดึงค่าพิกัดเส้นที่วาดขึ้นจาก draw.lines จะถูกจัดเก็บไว้ในตัวแปรอาร์เรย์ line โดยค่าพิกัด x จะถูกเก็บใน line[0] และค่า y จะถูกเก็บใน line [1] ตามคำสั่งนี้

```
line=[ ]
for line in draw.lines:
    cv2.line(img, line[0], line[1], (0, 255, 0), 2)
```

ผลลัพธ์ที่ได้จากการสร้างการวาดเส้นบนภาพสามารถกำหนดเส้นได้มากกว่าหนึ่งเส้น ดังภาพที่ 4-6




ภาพที่ 4-6 การวาดเส้นลงบนหน้าต่าง Video เพื่อกำหนดการนับจำนวนยานพาหนะ

การกำหนดจุดนับเมื่อมียานพาหนะผ่านเส้น โดยการกำหนดเงื่อนไข ถ้ามีวัตถุบนภาพเคลื่อนผ่านทับเส้นใช้คำสั่ง intersect เพื่อแยกประเภทยานพาหนะที่ทับเส้นที่วาดคำสั่งตัวอย่าง ดังภาพที่ 4-7

```
if intersect(data_deque[id][0], data_deque[id][1], line[0], line[1]):
    cv2.line(img, line[0],line[1], (255, 255, 255), 3)
```

การแสดงผลการนับจำแนกประเภทยานพาหนะเขียนตามคำสั่งตัวอย่าง

```
inText = 'Vehicle In'
outText = 'Vehicle Out'
if config.OBJECT_COUNTER != None:
    for _, (key, value) in enumerate(config.OBJECT_COUNTER.items()):
        inText += ' - ' + str(key) + ": " +str(value)
if config.OBJECT_COUNTER1 != None:
    for _, (key, value) in enumerate(config.OBJECT_COUNTER1.items()):
        outText += ' - ' + str(key) + ": " +str(value)
```


ภาพที่ 4-7 เส้นที่กำหนดจะปรากฏบนระบบการจำแนกแยกประเภทยานพาหนะและนับจำนวนแต่ละยานพาหนะ

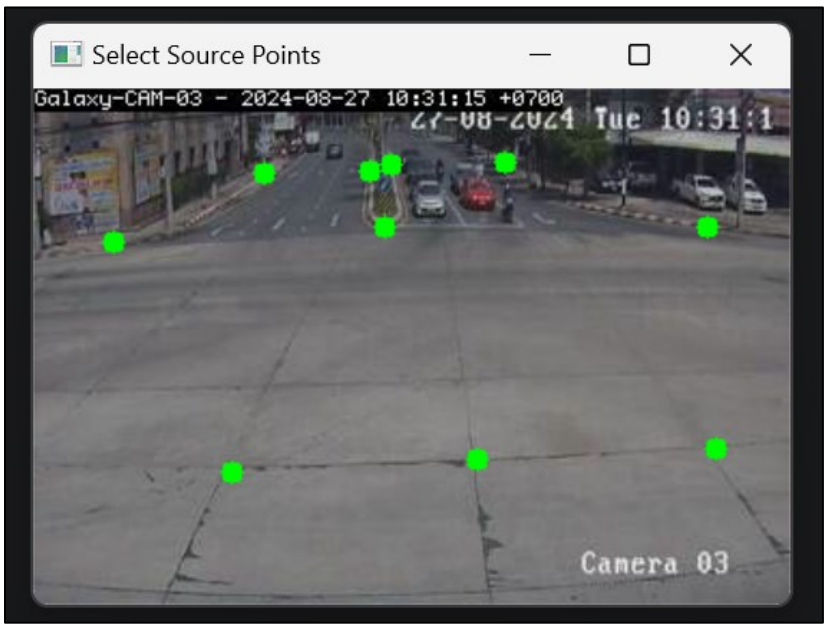
## ส่วนสร้างแบบจำลองการจราจรบนถนนจากกล้องวงจรปิด CCTV

การสร้างแบบจำลองการจราจรจากกล้องวงจรปิดร่วมกับภาพถ่ายจากดาวเทียมแบ่งออกเป็น 2 ส่วน ดังนี้

1. การวางจุดอ้างอิงพิกัด (Place a coordinate reference point) การกำหนดตำแหน่งจุดอ้างอิงพิกัดบนภาพถ่ายจากดาวเทียม จุดสีแดงที่ปรากฏ ดังภาพที่ 4-8

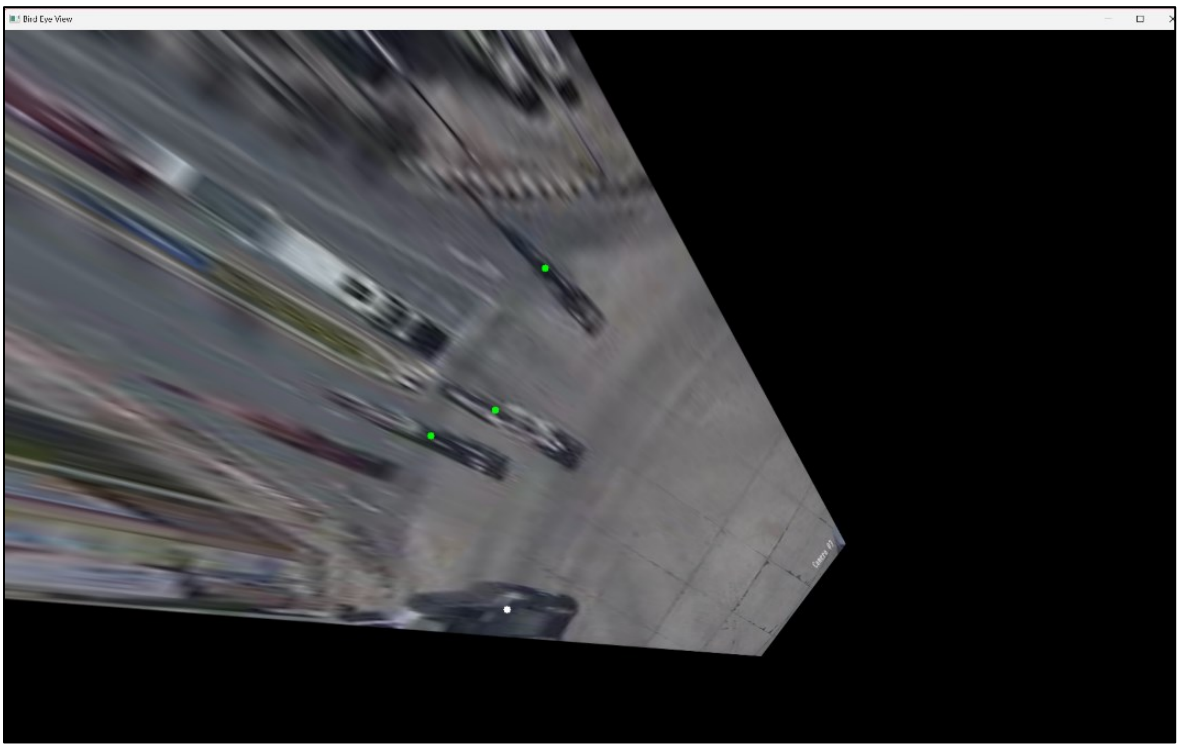
ภาพที่ 4-8 ภาพตัวอย่างการวางจุดอ้างอิงพิกัดบนภาพถ่ายจากดาวเทียม

เมื่อวางจุดอ้างอิงครบแล้วกดปุ่ม c (กำหนดไว้ในโปรแกรม) เพื่อวางจุดควบคุมบนกล้องวงจรปิด โดยวางให้ใกล้เคียงกับตำแหน่งกันบนภาพจากกล้องวงจรปิดตามภาพที่ 4-9



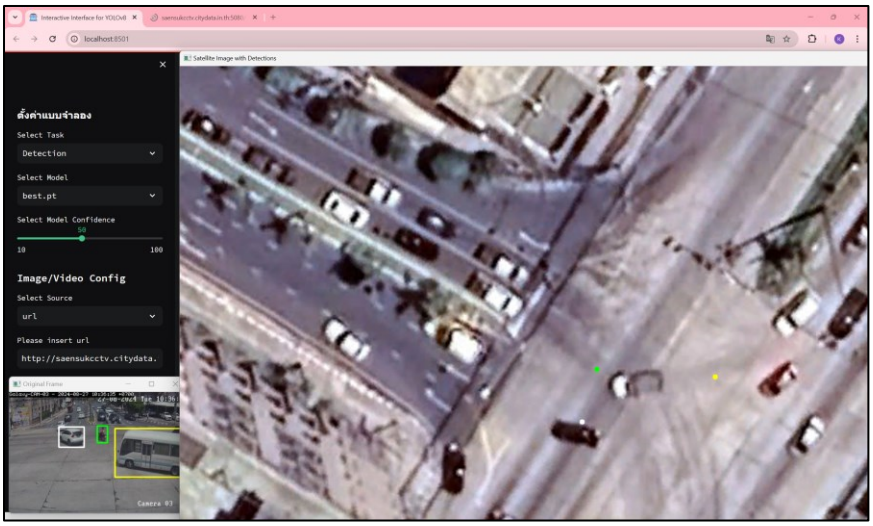

ภาพที่ 4-9 ภาพตัวอย่างการจุดอ้างอิงพิกัดบนกล้องวงจรปิดให้ใกล้เคียงกับตำแหน่งบนภาพถ่ายจากดาวเทียม

เมื่อวางจุดอ้างอิงบนกล้องวงจรปิดครบแล้ว กดปุ่ม c (กำหนดไว้ในโปรแกรม) ผลลัพธ์จากการวางจุดบังคับภาพทั้งสองเพื่อกำหนดหาจุดเหมือนบนภาพโดยใช้เทคนิค Homography หาจุดเหมือนและอ้างอิงเชิงตำแหน่งของภาพเพื่อทำการแปลงภาพ (Transform) จากกล้องวงจรปิดเป็นภาพมุมสูง Bird's eye view (BEV) ดังภาพที่ 4-10

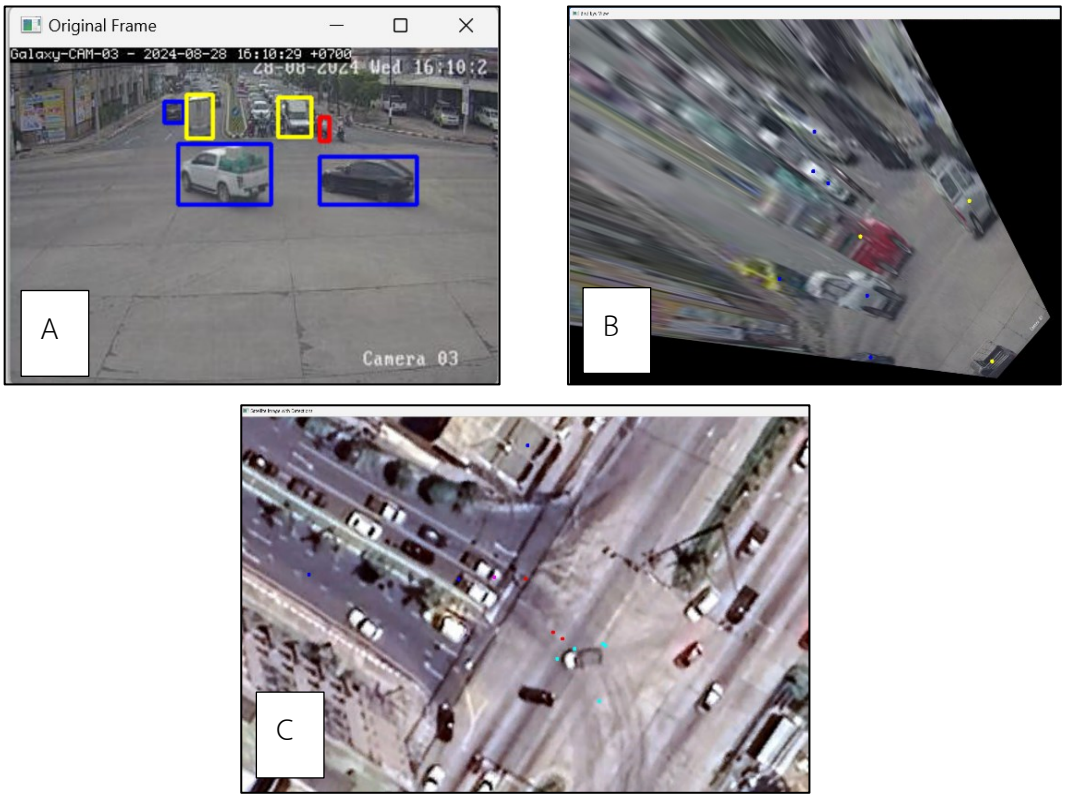



ภาพที่ 4-10 ผลลัพธ์จากการแปลงภาพจากกล้องวงจรปิดเป็นภาพมุมสูงใช้วิธี Homography

2. การจำลองการจราจรร่วมกับการจำแนกจากแบบจำลอง YOLOV8 ผลจากการแปลงข้อมูลภาพเคลื่อนไหวจากกล้องวงจรเป็นภาพมุมสูงผสมผสานกับการตรวจจับจำแนกยานพาหนะจากแบบจำลอง YOLOV8 จากกรอบสี่เหลี่ยมให้ข้อมูลจุดจุดของยานพาหนะที่เคลื่อนที่แสดงบนแผนที่จำลองการจราจร โดยการจำลองแผนที่การจราจรผู้วิจัยทดสอบทั้งหมด 10 สถานที่ตามการให้บริการให้ครอบคลุมเทศบาลเมืองแสนสุข โดยกำหนดจุดอ้างอิงพิกัด เพื่ออ้างอิงกับภาพถ่ายจากดาวเทียม และลักษณะการแสดงผลการทำงานบนแผนที่แบบจำลองการจราจรดังภาพที่ 4-11 และ 4-12




ภาพที่ 4-11 ผลลัพธ์ฟังก์ชั่นการแสดงแผนที่จำลองการจราจรโดยจุดที่แสดงเป็นยานพาหนะ




ภาพที่ 4-12 การตรวจจับยานพาหนะ (A) การแปลงข้อมูลภาพวงจรปิดเป็นภาพมุมสูง (B) และการจำลองจุดของยานพาหนะการจราจรแบบ Realtime บนภาพถ่ายจากดาวเทียม (C)

ผลการเปรียบเทียบการใช้ข้อมูลจากกล้องวงจรปิด (CCTV) กับ ภาพถ่ายจากดาวเทียมสร้างแบบจำลองแผนที่การจราจรบนถนนตามแยกทั้งหมด 10 สถานที่ โดยแบ่งการเปรียบเทียบในด้านความแม่นยำของตำแหน่งยานพาหนะและความถูกต้องของการจำแนกได้ดังนี้

1. บริเวณแยกหน้ามหาวิทยาลัยบูรพา

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนลงหาดบางแสนฝั่งขาเข้าชายหาดบางแสนและขาออกถนนสุขุมวิทและความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 75 เปอร์เซ็นต์ และเกิดความคลาดเคลื่อนของตำแหน่งของยานพาหนะที่ออกจากมหาวิทยาลัยบูรพาไปทางออกสุขุมวิท และเข้าสู่ชายหาดบางแสนที่แสดงตำแหน่งคลาดเคลื่อนเนื่องจากกล้องวงจรปิดติดตั้งฝั่งตรงข้ามมหาวิทยาลัยบูรพาประกอบกับมีมุมมองกว้าง (wide angle) ทำให้เกิดการบิดเบี้ยวของภาพและทำให้จำแนกและตรวจจับคลาดเคลื่อน

2. บริเวณแยกลงหาดบางแสน (แยกกาแล็คซี่)

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนสุขุมวิทฝั่งขาเข้าตัวเมืองชลบุรีและฝั่งขาออกไปตลาดหนองมน และความถูกต้องในการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 85 – 90 เปอร์เซ็นต์ เนื่องจากตำแหน่งของการติดตั้งและมุมกล้องวงจรปิดได้ระดับดี

3. บริเวณแยกลงหาดบางแสนสาย 2 (แยกจรินทร์)

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนลงหาดบางแสนฝั่งขาเข้าและขาออกชายหาดบางแสนและความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 75 - 80 เปอร์เซ็นต์ และเกิดความคลาดเคลื่อนของตำแหน่งและการจำแนกของยานพาหนะที่มาจากถนนลงหาดบางแสนสาย 2 เนื่องจากกล้องติดตั้งฝั่งตรงข้าม มีมุมมองของกล้องวงจรปิดดกว้าง ทำให้เกิดการบิดเบี้ยวของภาพและทำให้จำแนกและตรวจจับคลาดเคลื่อน

4. บริเวณแยกจรรยา (คลองน้ำเหม็น)

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนสุขุมวิทฝั่งขาเข้าไปเมืองชลบุรีและฝั่งขาออกมุ่งหน้าไปศรีราชาและความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 75 - 80 เปอร์เซ็นต์ บริเวณเกิดความคลาดเคลื่อนของตำแหน่งและการจำแนกของยานพาหนะที่เลี้ยวเข้าสู่ถนนเลี่ยงหนองมน เนื่องจากกล้องตั้งอยู่ฝั่งตรงข้ามกับถนนเลี่ยงหนองมนทำให้วัตถุที่เคลื่ออนตัวออกจากกล้องวงจรปิดเกิดความคลาดเคลื่อนได้

5. บริเวณแยกตาดม (ร้านเกาเหลาโต สาขา 2)

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนแสนสุขและถนนเปรมใจราษฎร์ ความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 78 -82 เปอร์เซ็นต์

6. บริเวณแยกไทยพิพัฒน์ (เลี่ยงหนองมน)

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนเลี่ยงหนองมน และความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 76 - 82 เปอร์เซ็นต์ ความคลาดเคลื่อนของตำแหน่งและการจำแนกของยานพาหนะเกิดบริเวณฝั่งมุ่งไปเทศบาลตำบลเหมือง เนื่องจากกล้องวงจรปิดอยู่ฝั่งตรงข้ามทำให้การตรวจจับและจำแนกไกลจากระยะทำให้คลาดเลื่อน

7. บริเวณตลาดหนองมน

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนสุขุมวิทฝั่งขาเข้าเมืองชลบุรีและความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 75 - 80 เปอร์เซ็นต์ ความคลาดเคลื่อนของตำแหน่งและการจำแนกของยานพาหนะเกิดบริเวณฝั่งเข้าออกไปศรีราชา เนื่องจากมีเกาะกลางถนนแบบรั้วกั้นทำให้บดบังและลดประสิทธิภาพในการตรวจจับและจำแนกยานพาหนะ

8. บริเวณแยกตาลล้อม

ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนสุขุมวิทฝั่งขาเข้ามุ่งหน้าไปเมืองชลบุรีและขาออกมุ่งหน้าไปศรีราชา ความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 75 - 78 เปอร์เซ็นต์ ความคลาดเคลื่อนของตำแหน่งและการจำแนกของยานพาหนะเกิดบริเวณฝั่งมุ่งหน้าถนนเนตรดี เนื่องจากกล้องอยู่ฝั่งตรงข้ามกับถนนและความกว้างของถนนสุขุมวิททำให้วัตถุบนภาพที่ออกมาจากถนนเนตรดีไม่ชัด

9. บริเวณแยกถนนข้าวหลาม

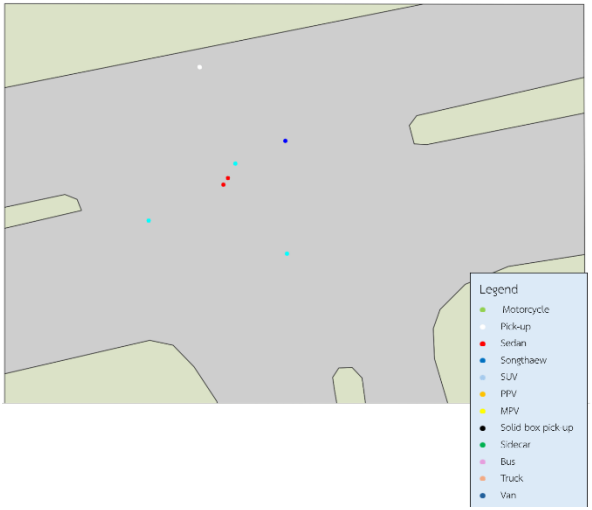
ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนข้าวหลามฝั่งเข้าชายหาดบางแสนความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 80 - 85 เปอร์เซ็นต์

10. บริเวณแยกแหลมแท่น

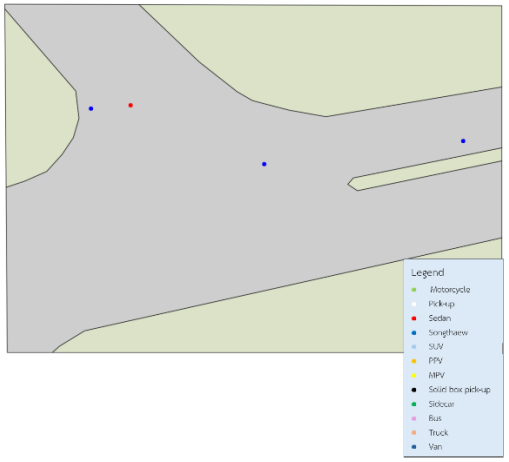
ความแม่นยำของตำแหน่งของยานพาหนะเคลื่อนที่ถูกต้องตามช่องทางบนถนนลงหาดบางแสนสาย 2 มุ่งหน้าไปถนนบางแสน - อ่างศิลา ความถูกต้องการจำแนกยานพาหนะมีความถูกต้องเฉลี่ย 80 – 85 เปอร์เซ็นต์

จากผลลัพธ์ระบบตรวจจับและจำแนกประเภทยานพาหนะเข้าสู่กระบวนการสร้างแผนที่จำลองการจราจรบนถนนตามแยกถนนทั้งหมด 10 สถานที่ ในเขตเทศบาลเมืองแสนสุข สามารถแสดงผลการแปลงภาพจากกล้องวงจรปิดเป็นภาพมุมสูงอ้างอิงตามภาพถ่ายจากดาวเทียมและแผนที่จำลองการจราจรของยานพาหนะ ดังตารางที่ 4-3

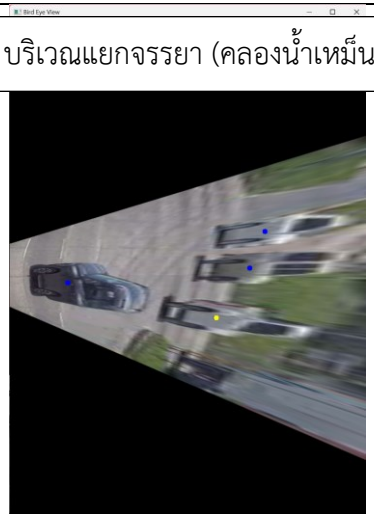
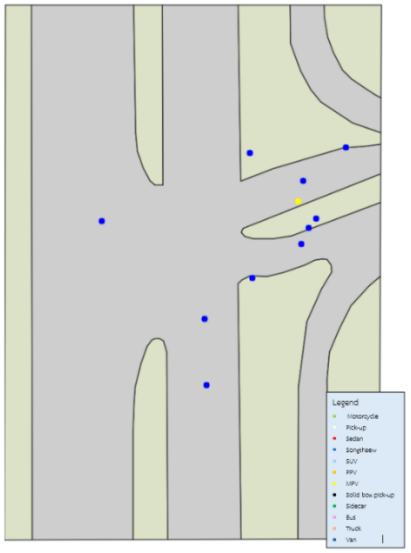
ตารางที่ 4-3 แสดงผลการแปลงภาพจากกล้องวงจรปิดเป็นภาพมุมสูงอ้างอิงตามภาพถ่ายจากดาวเทียมและแผนที่จำลองการจราจรของยานพาหนะบนถนน

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
|  |  |  |

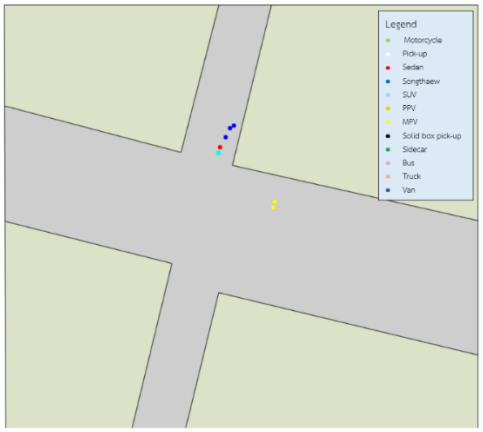
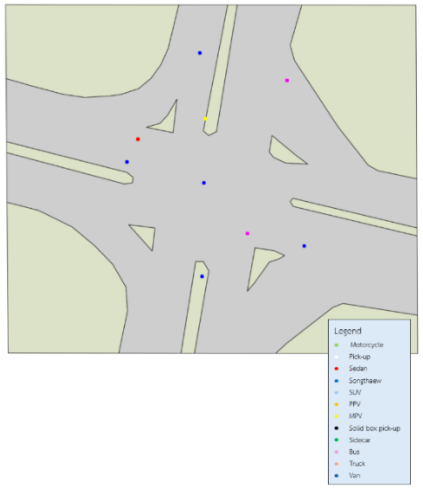
ตารางที่ 4-3 (ต่อ)

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
|  |  |  |
|  |  |  |

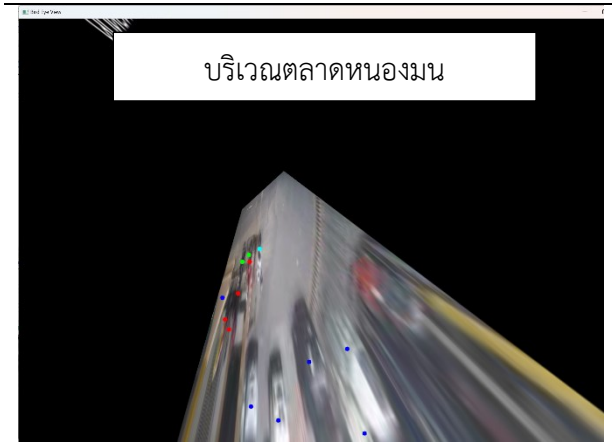
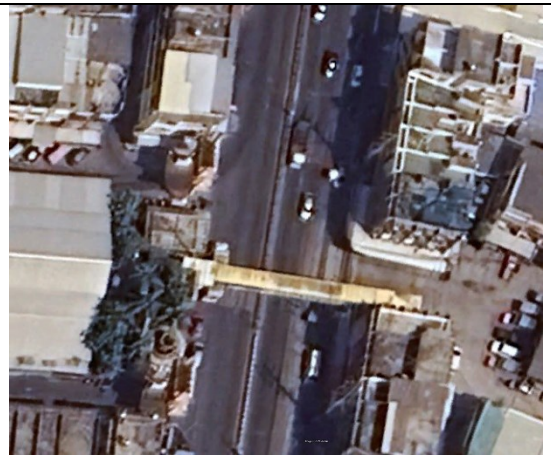
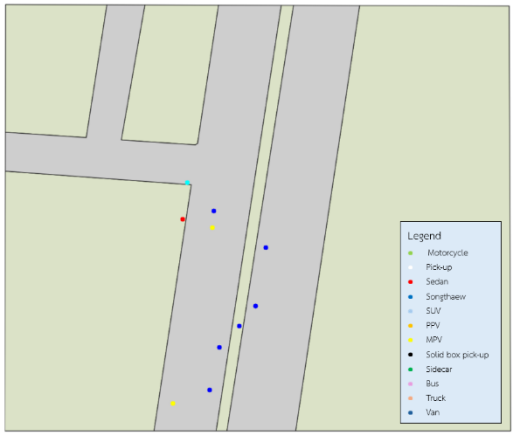
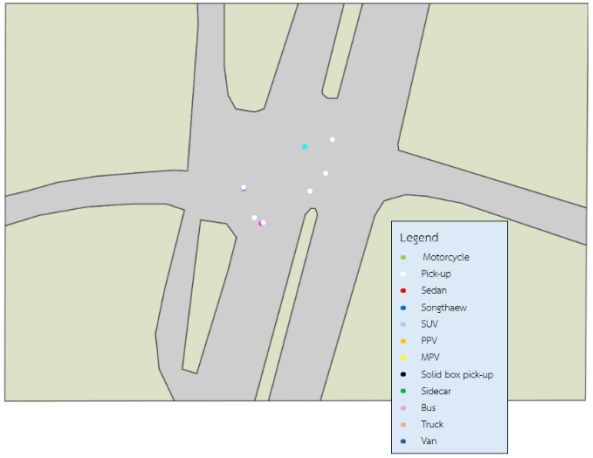
ตารางที่ 4-3 (ต่อ)

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
| บริเวณแยกจรรยา (คลองน้ำเหม็น)  |  |  |

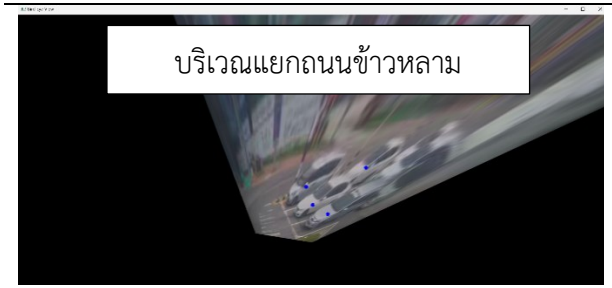
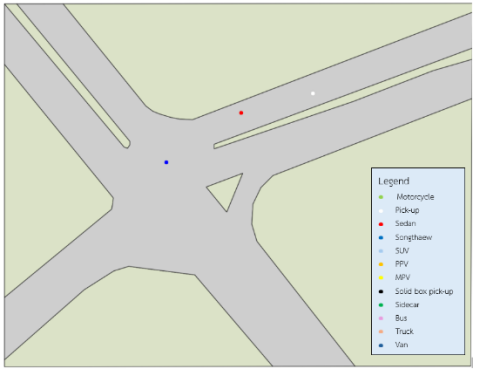
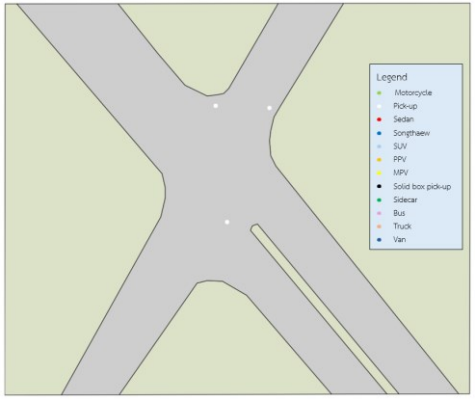
ตารางที่ 4-3 (ต่อ)

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
| บริเวณแยกตาดม (ร้านเกาเหลาโต สาขา 2)  |  |   |
| บริเวณแยกไทยพิพัฒน์ (เลี่ยงหนองมน)  |  |   |

ตารางที่ 4-3 (ต่อ)

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
|  |  |  |
|  |  |  |

ตารางที่ 4-3 (ต่อ)

| ภาพจากกล้องวงจรปรับเป็นภาพมุมสูง | ภาพถ่ายจากดาวเทียม | แผนที่จำลองการจราจรของยานพาหนะบนถนน |
|---|---|---|
|  |  |  |
|  |  |  |

# บทที่ 5
# สรุป อภิปรายผล และข้อเสนอแนะ

## สรุป

สรุปผลการพัฒนาระบบตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้อง CCTV ในเขตเทศบาลเมืองแสนสุข โดยใช้เทคนิคการเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) และสร้างแบบจำลองแผนที่การจราจรบนถนน (Digital replica of the road mapping)

1. ส่วนการเตรียมข้อมูลฝึกการเรียนรู้

การเตรียมข้อมูลภาพเพื่อการฝึกการเรียนรู้ใน Roboflow จำแนกประเภทยานพาหนะทั้งหมด 12 ประเภท จำนวน 2,037 ภาพ แบ่งชุดการทำงานเพื่อสร้างแบบจำลองการจำแนกยานพาหนะ แบ่งชุดการทำงานข้อมูล (Dataset Split) โดยชุดข้อมูลสำหรับการฝึกสอน (Train set) จำนวน 2,081 ชุด ข้อมูลฝึกให้แบบจำลองเรียนรู้ (Valid set) จำนวน 609 ชุด และชุดข้อมูล (Test set) จำนวน 321 ชุด รวมชุดข้อมูลทั้งหมด 3,011 ข้อมูล

2. ส่วนการเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) จาก YOLOV8 ใช้แบบจำลอง YOLOV8M ใช้การดึงข้อมูลเข้าไปฝึกสอนในชุดช้อมูแบบจำลองให้ครบ 12 ประเภทของยานพาหนะเป็นจำนวน 50 รอบ

สรุปผลจากการวิเคราะห์การเรียนรู้เชิงลึกของโครงข่ายประสาทแบบคอนโวลูชัน Deep Convolutional Neural Network (DCNN) บน YOLOV8 ประเภทยานพาหนะที่มีรูปแบบหรือลักษณะที่แตกต่างกันความถูกต้องของการจำแนกอยู่ในช่วง 0.91 - 1 คือ รถกระบะขนส่งตู้ทึบ (Solidbox pick-up), รถสองแถว (Songthaew) ในส่วนประเภทยานพาหนะที่มีลักษณะคล้ายกันทั้งรูปแบบและลักษณะการทดสอบค่าความถูกต้องในการจำแนกจะมีค่าประมาณ 0.87- 0.90 คือ รถประจำทาง (Bus), รถบรรทุก (truck), รถกระบะ (pick-up), รถยนต์นั่งส่วนบุคคล 4 ที่นั่ง (sedan), รถมอเตอร์ไซด์ (Motorcycle), รถตู้ (van) และประเภทยานพาหนะที่มีลักษณะคล้ายกันทั้งรูปแบบและลักษณะการทดสอบค่าความถูกต้องในการจำแนกจะมีค่าประมาณ 0.68 - 0.78 คือ ประเภทรถอเนกประสงค์ประเภทครอบครัว (SUV), รถอเนกประสงค์ที่มีพื้นฐานและดัดแปลงมาจากรถกระบะ (PPV) และ รถอเนกประสงค์ที่สามารถนั่งได้ประมาณ 5 - 7 คน (MPV), ประเภทรถมอเตอร์ไซด์พ่วงข้าง (Sidecar)

3. ส่วนการตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้องวงจรปิด CCTV

สรุปการตรวจจับและจำแนกประเภทยานพาหนะสามารถทำงานบน Web browser ผ่านการจำลองเครื่องแม่ข่ายแบบ localhost และกำหนดวาดเส้นบนภาพจากกล้องวงจรปิดร่วมกับแบบจำลองการจำแนกประเภทยานพาหนะเพื่อกำหนดจุดนับจำนวนยานพาหนะและแยกประเภทได้ โดยแสดงค่าการนับแยกตามประเภทของยานพาหนะ โดยปัจจัยที่ส่งผลการตรวจจับและการจำแนกคาดเคลื่อน คือ คุณภาพรายละเอียดของกล้องวงจรปิด ความสว่าง และความเร็วในการส่งสัญญาณข้อมูลภาพ

4. ส่วนสร้างแผนที่แบบจำลองการจราจรบนถนน

การสร้างแผนที่แบบจำลองการจราจรบนถนนโดยใช้ข้อมูลกล้องวงจรปิด ภาพถ่ายจากดาวเทียม ภาพแผนที่ถนนของแต่ละบริเวณ และแบบจำลองการตรวจจำแนกประเภทยานพาหนะจาก YOLOV8 สรุปว่าการทดสอบทั้งหมด 10 สถานที่ โดยแผนที่แบบจำลองการจราจรเป็นแผนที่แสดงการเคลื่อนไหวของยานพาหนะจากกล้องวงจรปิดตามแยกและใช้ปัญญาประดิษฐ์สำหรับฝึกสอนข้อมูลภาพให้สามารถจดจำ เรียนรู้ประเภทของยานพาหนะเพื่อสร้างแบบจำลองการตรวจจับ การจำแนกและประยุกต์บนข้อมูลภูมิสารสนเทศในรูปแบบแผนที่ถนนสำหรับการแสดงผลเป็นการเคลื่อนที่ของยานพาหนะบนแผนที่ในรูปแบบพลวัต (Map Dynamics) เพื่อเข้าสู่เทคโนโลยีโมเดลเสมือนจริงของวัตถุทางกายภาพหรือที่เรียกว่า Digital Twins ในอนาคต ในส่วนของการสร้างแผนที่แบบจำลองการจราจรบนถนนสามารถสรุปได้เป็นประเด็นสำคัญจากผลวิจัยคือ จุดการติดตั้งและมุมมองของกล้องวงจรมีผลกับการแสดงผลของแผนที่แบบจำลองการจราจร ซึ่งมุมกล้องบางบริเวณมีมุมมองแคบ (Narrow Field of View) เช่น แยกตาดม และแยกตลาดหนองมน มีผลกับการวางจุดอ้างอิง เนื่องจากหาจุดอ้างอิงที่เหมือนกันกับภาพถ่ายจากดาวเทียมค่อนข้างยาก และบริเวณมุมกล้องกว้าง (Wide Field of View) ส่งผลให้ภาพขอบภาพบิดเบี้ยวมากส่งผลให้ยานพาหนะที่ปรากฏจะบิดเบี้ยวส่งผลต่อการจำแนกที่คลาดเคลื่อนและความแม่นยำของตำแหน่งของยานพาหนะ เช่น บริเวณแยกลงหาดบางแสนสาย 2 (แยกจรินทร์) และแยกบริเวณหน้ามหาวิทยาลัยบูรพา สรุปผลแบบจำลองการจราจรจาก 10 สถานที่ ดังนี้ บริเวณแยกลงหาดบางแสน (แยกกาแล็คซี่) บริเวณหน้ามหาวิทยาลัยบูรพา บริเวณแยกตาลล้อม มีค่าความถูกต้องของการจำแนกและค่าความแม่นยำของตำแหน่งยานพาหนะมีค่าเฉลี่ย 85 – 90 เปอร์เซ็นต์ สูงกว่าบริเวณอื่นเพราะตำแหน่งของการติดตั้งและมุมกล้องวงจรปิดอยู่ในระดับดี ส่วนบริเวณแยกอื่นมีค่าความแม่นยำและถูกต้องของการจำแนกเฉลี่ยประมาณ 75 -82 เปอร์เซ็นต์ เพราะมุมกล้องมุมกว้างเกินและบางที่แคบเกินไปและระดับของการติดตั้งของกล้องวงจรปิด ส่งผลความคลาดเคลื่อนให้ค่าความแม่นยำของตำแหน่งของยานพาหนะที่แสดงผลและความถูกต้องของการจำแนกยานพาหนะบ้าง แต่อยู่ในเกณฑ์ที่ยอมรับได้

## อภิปรายผล

จากการศึกษาพบว่าการตรวจจับจำนวนและจำแนกประเภทยานพาหนะจากกล้องวงจรปิด การจราจรโดยเทคนิค Homography เป็นวิธีการจำลองการจราจรมุมสูงจากกล้องวงจรปิด ซึ่งสอดคล้องกับ Minghan Zhu, Songan Zhang, Yuanxin Zhong, Pingping Lu,Huei Peng & John Lenneman (2022) ได้ศึกษาวิธีการสกัดข้อมูลตำแหน่งและทิศทางของยานพาหนะโดยอาศัยข้อมูลจากกล้องวงจรปิดที่ยังไม่ได้ตรวจวัดค่าเชิงตำแหน่งร่วมกับข้อมูลอ้างอิงเชิงตำแหน่งบนแผนที่เพื่อแปลงข้อมูลกรอบสี่เหลี่ยมที่ตรวจจับวัตถุให้เป็นแนวระนาบเรียบตามพื้นผิวโดยใช้ภาพมุมสูง Bird's Eye View (BEV) ที่ได้จากกล้องวงจรปิด เพื่อจะนำข้อมูลสามมิติของรถยนต์ที่มาใส่แทนในกรอบสี่เหลี่ยม

นอกจากนี้การแสดงการจำลองการจราจรจากข้อมูลแบบจำลองจำแนกยานพาหนะร่วมกับภาพมุมสูงจากกล้องวรจรสอดคล้องกับวิจัยของ Torben Teepe Philipp Wolters Johannes Gilg Fabian Herzog & Gerhard Rigoll (2024) ในการใช้เทคนิค Homography สำหรับการแปลงข้อมูลให้เป็นแนวระนาบ ในส่วนของงานวิจัยนี้มีความต่างกันในส่วนข้อมูลกล้องวงจรที่ใช้เช่น รายละเอียดภาพและจำนวนกล้องวงจรปิดที่ใช้ในจุดนั้นๆ ซึ่งสามารถทำภาพซ้อนทับกัน (Overlap) ทำให้เห็นมุมทุกด้านของสี่แยกและส่งผลให้การคาดการณ์การจำลองการจราจรของยานพาหนะมีความแม่นยำเชิงตำแหน่งมากยิ่งขึ้น

การใช้เทคนิค Homography เพื่อใช้การจับคู่ภาพ (Feature Matching) สอดคล้องกับวิธีการเลือกการจับคู่ภาพของ Mahdi Rezaei, Mohsen Azarmi  & Farzam Mohammad Pour Mir (2021) วิจัยเรื่องการติดตตามการจราจรแบบสามมิติโดยใช้กล้องวงจรปิด ซึ่งในกระบวนการปรับแก้ภาพจากกล้องวงจรปิดใช้เทคนิคการจับคู่ภาพแบบ Random Sample Consensus (RANSAC) เป็นวิธีการแบบวนซ้ำเพื่อประมาณค่าพารามิเตอร์ของแบบจำลองทางคณิตศาสตร์จากชุดข้อมูลจุดบนภาพที่สังเกตได้ซึ่งต้องมีค่าผิดปกติ โดยค่าผิดปกติจะต้องไม่ส่งผลต่อค่าประมาณ ดังนั้น จึงสามารถตีความได้ว่าเป็นวิธีการตรวจจับค่าจับคู่ที่ผิดปกติและเลือกจุดคู่ภาพที่ดีที่สุด

การตรวจจับนับจำนวนและจำแนกประเภทยานพาหนะในงานวิจัยนี้ใช้แบบจำลอง YOLOV8 สอดคล้องกับงานวิจัยของPriyanka Ankireddy, V. Siva Krishna Reddy, Dr.V. Lokeswara Reddy (2022) ที่ทำการศึกษาการติดตามและจำแนกประเภทยานพาหนะโดยใช้ YOLOV8 พบว่าการจำแนกมีค่าความถูกต้องและแม่นยำสูงถึงแม้ว่าวัตถุบนภาพจะมีขนาดเล็กสามารถแยกประเภทยานพาหนะได้และมุมมองของการติดตั้งกล้องวงจรปิดส่งผลต่อความถูกต้องของการจำแนก YOLOV8 จำแนกยานพาหนะจากภาพวิดีโอแบบ Realtime และการตรวจจับวัตถุที่ซ้อนทับหรือมีสิ่งกีดขวางได้เป็นอย่างดี

ส่วนในการนับจำนวนของยานพาหนะบนภาพจากกล้องวงจรปิดมีวิธีการสอดคล้องกับงานวิจัยของ Ankit Kumar, Ritika Chandel & Monika Singh (2023) ที่ได้ศึกษาเทคนิคการประมวลผลภาพสำหรับการติดตามและนับจำนวนของยานพาหนะโดยใช้ ROI ใช้แบบจำลอง YOLOV8 สำหรับการตรวจจับจำแนกและใช้การสร้างพื้นที่ ROI ลงวิดีโอเพื่อกำหนดพื้นที่การเริ่มนับจำนวนยานพาหนะที่ผ่านพื้นที่นี้ และนับจำนวนตามประเภทของยานพาหนะที่จำแนกไว้บนแบบจำลอง YOLOV8 แสดงผลการนับบนจอภาพของวิดีโอสามารถนับได้แบบ Realtime และไม่มีปัญหาเรื่องอัตราความเร็วของวิดีโอ ที่เรียกว่า Frame rates และมีความยืดหยุ่นในการกำหนดพื้นที่การนับได้ตามมุมมองของกล้องวงจรปิดตามสถานการณ์

## ปัญหาและอุปสรรค

จากขั้นตอนการศึกษาที่ผ่านมาทั้งหมด พบปัญหาดังนี้

1. ปัญหามุมมองของกล้องวงจรบางจุดมีมุมมองแคบทำให้การตรวจจับไม่ทัน
2. ข้อจำกัดรายละเอียดภาพของกล้องวงจรปิดที่ทำให้การตรวจจับและจำแนกผิดพลาดได้
3. ปัญหาความไม่เสถียรของสัญญาณอินเตอร์เน็ตที่ทำให้การส่งสัญญาณภาพขัดข้องบางช่วงเวลา

## ข้อเสนอแนะ

1. หากต้องการใช้ระบบตรวจนับและจำแนกประเภทยานพาหนะจากกล้องวงจรปิดทุกตัวพร้อมกัน ต้องดำเนินการวางระบบโครงข่าย
2. ตัวกล้องวงจรปิดต้องมีระบบประมวลการตรวจนับ จำแนกในตัว สามารถส่งข้อมูลไปยังฐานข้อมูลเครื่องแม่ข่าย ซึ่งทำให้สามารถติดตามการจราจรในทุกช่วงเวลาพร้อมกันได้
3. การพัฒนาต่อยอดทางเทคโนโลยี ต้องมีงบประมาณเพียงพอสำหรับการวางระบบ
4. บุคลากรที่เกี่ยวข้องต้องมีองค์ความรู้ทางวิทยาการปัญญาประดิษฐ์ ข้อมูลเชิงพื้นที่ขนาดใหญ่ทางภูมิสารสนเทศ
5. ส่งเสริมการบูรณาการทางวิจัยให้มากขึ้น เพื่อเกิดนวัตกรรมเชิงข้อมูลและสามารถนำไปแก้ไขปัญหาการจราจรได้อย่างมีประสิทธิภาพ

# บรรณานุกรม


คลาวด์ เอซ จำกัด. (2563). *AI, Machine Learning (ML) คืออะไร?*. วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://cloud-ace.co.th/blogs/o0v9a6-ai-machine-learning-ml-ai-ml-google

เทคซอร์ส. (2562). ทำความรู้จัก *AI, Machine learning, Deep learning ฉบับเข้าใจง่าย*. วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://techsauce.co/tech-and-biz/ai-machine-learning-deep-learning-Differences

เทศบาลเมืองแสนสุข. (2562). *สถิติการท่องเที่ยวบางแสน ปี 2562*. วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://www.saensukcity.go.th/images/doc/stat-tourism-2562.pdf

ผู้จัดการออนไลน์. (2564). *ไอเดียเจ๋ง! สภ.แสนสุข ใช้โดรนตรวจสอบสภาพจราจรแก้ปัญหารถติดหาดบางแสน*. วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://mgronline.com/local/detail/9640000120553

พีทีที เอ็กซ์เพรสโซ่. (2563). *สร้าง AI ให้รู้สึก รู้จริง! ทำความรู้จักว่า Deep Learning คืออะไร.* วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://blog.pttexpresso.com/get-to-know-deep-learning/

สำนักงานส่งเสริมเศรษฐกิจดิจิทัล. (2564). *เทคโนโลยีที่สำคัญในยุคดิจิทัล: เทคโนโลยีปัญญาประดิษฐ์ Tech Series: Artificial Intelligence (AI).* วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://www.depa.or.th/th/article-view/tech-series-artificial-intelligence-ai

_____. (2565). *การคาดการณ์อนาคต เทคโนโลยีดิจิทัลประเทศไทย 2035*. วันที่ค้นข้อมูล 12 ธันวาคม พ.ศ. 2566, เข้าถึงได้จาก https://www.depa.or.th/storage/app/media/file/Second%20Deliverable%20RevVer%20TH%20V12%20140819%20FIN.pdf

Adit Deshpande. (2016*). A Beginner's Guide To Understanding Convolutional Neural Networks*. Retrieved from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Ankit Kumar, Ritika Chandel & Monika Singh. (2023). *Image Processing technique for Tracking andCounting of Vehicles using ROI*. Retrieved from Jan 12, 2024 https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10451264

esriThailand. (2023). *Future Impacts of GeoAI on Mapping*. Retrieved from https://www.esrith.com/blog/future-impacts-of-geoai-on-mapping/

Gerhard Roth. (2011). *Homography.* Retrieved from http://people.scs.carleton.ca /~roth/comp4900d-12/notes/homography.pdf

Joseph Birkner. (2023). *Monocular 3D Object Detection Using HD Maps*. Retrieved from https://www.ce.cit.tum.de/fileadmin/w00cgn/air/_my_direct_uploads /joseph.ausarbeitung.v1.1.0-1.pdf

Mahdi Rezaei, Mohsen Azarmi & Farzam Mohammad Pour Mir. (2021). *Traffic-Net: 3D traffic monitoringusing a single camera.* Retrieved from https://arxiv.org/pdf/2109.09165

Minghan Zhu, Songan Zhang, Yuanxin Zhong, Pingping Lu,Huei Peng & John Lenneman. (2022). *Monocular 3D Vehicle Detection Using Uncalibrated Traffic Cameras through Homography*. Retrieved from https://arxiv.org/pdf/2103.15293

Priyanka Ankireddy, V. Siva Krishna Reddy, Dr.V. Lokeswara Reddy. (2022). *Vehicle Detection and Tracking Using YOLOv8 and Deep Learning to Boost Image Processing Quality*. Retrieved from https://www.ijfans.org/uploads/paper /edff6aef6f768ed61a58c3f7a359e933.pdf

Put Dejudom. (2022). *Convolutional Neural Networks (CNN).* Retrieved from https://th.linkedin.com/pulse/%E0%B9%80%E0%B8%9A%E0%B8%AD%E0% B8%87%E0%B8%95%E0%B8%99%E0%B8%82%E0%B8%AD%E0%B8%87-convolutional-neural-networkscnn-put-dejudom

run.ai. (2023). *Deep Convolutional Neural Networks*. Retrieved from https://www.run.ai/guides/deep-learning-for-computer-vision/deep-convolutional-neural-networks

Surapong Kanoktipsatharporn. (2019). *Training Set Train / Test Split Training Set, Validation Set and Test Set in Machine Learning.* Retrieved from https://www.bualabs.com/archives/532/what-is-training-set-why-train- test-split-training-set-validation-set-test-set/

Torben Teepe Philipp Wolters Johannes Gilg Fabian Herzog & Gerhard Rigoll. (2024). *Lifting Multi-View Detection and Tracking to the Bird's Eye View*. Retrieved from https://arxiv.org/pdf/2403.12573

University of Florida. (2022). *GEO:AI.* Retrieved from https://geog.ufl.edu/geoai/

# ภาคผนวก

**ชุดคำสั่ง app.py**

```
from pathlib import Path
import streamlit as st

import config
from utils import load_model,url_video
from streamlit_drawable_canvas import st_canvas
#import draw
# setting page layout
st.set_page_config(
    page_title="Interactive Interface for YOLOv8",
    page_icon="🤖",
    layout="wide",
    initial_sidebar_state="expanded"
    )

# main page heading
st.title("ระบบจำแนกยานพาหนะเทศบาลเมืองแสนสุข (ต้นแบบ)")

# sidebar
st.sidebar.header("ตั้งค่าแบบจำลอง")

# model options
task_type = st.sidebar.selectbox(
    "Select Task",
    ["Detection"]
)
model_type = None
if task_type == "Detection":
    model_type = st.sidebar.selectbox(
        "Select Model",
        config.DETECTION_MODEL_LIST
```

```
    )
else:
    st.error("Currently only 'Detection' function is implemented")

confidence = float(st.sidebar.slider(
    "Select Model Confidence", 10, 100, 50)) / 100

model_path = ""
if model_type:
    model_path = Path(config.DETECTION_MODEL_DIR,
str(model_type))
else:
    st.error("Please Select Model in Sidebar")

# load pretrained DL model
try:
    model = load_model(model_path)
except Exception as e:
    st.error(f"Unable to load model. Please check the specified path:
{model_path}")

# image/video options
st.sidebar.header("Image/Video Config")
source_selectbox = st.sidebar.selectbox(
    "Select Source",
    config.SOURCES_LIST
)

#line = st.sidebar.number_input('Line
position',min_value=0.0,max_value=100.0,value=0.6, step=0.1)
source_img = None
if source_selectbox == config.SOURCES_LIST[0]: #  url
```

```
    url_video(confidence, model)
else:
    st.error("Currently only 'Image' and 'Video' source are
implemented")
```

**ชุดคำสั่ง config.py**

```
from pathlib import Path
import sys

# Get the absolute path of the current file
file_path = Path(__file__).resolve()

# Get the parent directory of the current file
root_path = file_path.parent

# Add the root path to the sys.path list if it is not already there
if root_path not in sys.path:
sys.path.append(str(root_path))
# Get the relative path of the root directory with respect to the
current working directory
ROOT = root_path.relative_to(Path.cwd())

# Source
SOURCES_LIST = ["url"]

# DL model config
DETECTION_MODEL_DIR = ROOT / 'weights' / 'detection'
YOLOv8n = DETECTION_MODEL_DIR / "yolov8n.pt"
YOLOv8s = DETECTION_MODEL_DIR / "yolov8s.pt"
YOLOv8m = DETECTION_MODEL_DIR / "yolov8m.pt"
YOLOv8l = DETECTION_MODEL_DIR / "yolov8l.pt"
YOLOv8x = DETECTION_MODEL_DIR / "yolov8x.pt"
Best = DETECTION_MODEL_DIR / "best.pt"
```

```
DETECTION_MODEL_LIST = [
"yolov8n.pt",
"yolov8s.pt",
"yolov8m.pt",
"yolov8l.pt",
"yolov8x.pt",
"best.pt"]

OBJECT_COUNTER = None
OBJECT_COUNTER1 = None
```

**ชุดคำสั่ง utils.py**

```
from ultralytics import YOLO
#from ultralytics.solutions import object_counter
import streamlit as st
import cv2
from PIL import Image
import tempfile
import config
from pytube import YouTube
from sort import *
import urllib.request as url
from tracker import *
import pandas as pd
import cvzone
from streamlit_drawable_canvas import st_canvas
import draw
import mysql.connector
from mysql.connector import Error
from datetime import datetime
import config
import numpy as np
```

```
import Homo
import connectdb
t=connectdb.vehicle_data_list

import streamlit as st
src_points = []
dst_points = []

# Load the satellite image to select destination points
satellite_img_path = 'galaxy_zoom.jpg'
satellite_img = cv2.imread(satellite_img_path)
satellite_img_copy = satellite_img.copy()
def clear_lines():
    draw.lines = []
    cv2.setMouseCallback("Video", draw.draw_line)

def _display_detected_frames(conf, model, st_count, st_frame,
image, is_display_tracking=None, tracker=None):
    """
    Display the detected objects on a video frame using the YOLOv8
model.
    :param conf (float): Confidence threshold for object detection.
    :param model (YOLOv8): An instance of the `YOLOv8` class
containing the YOLOv8 model.
    :param st_frame (Streamlit object): A Streamlit object to display
the detected video.
    :param image (numpy array): A numpy array representing the
video frame.
    :return: None
    """

    # Predict the objects in the image using YOLOv8 model
```

```
    res = model.predict(image, conf=conf)

    inText = 'Vehicle In'
    outText = 'Vehicle Out'
    if config.OBJECT_COUNTER != None:
        for _, (key, value) in
enumerate(config.OBJECT_COUNTER.items()):
            inText += ' - ' + str(key) + ": " +str(value)

    if config.OBJECT_COUNTER1 != None:
        for _, (key, value) in
enumerate(config.OBJECT_COUNTER1.items()):
            outText += ' - ' + str(key) + ": " +str(value)
    cv2.namedWindow("Video")
    cv2.setMouseCallback("Video", draw.draw_line)
    cv2.imshow("Video",image)
    """frame_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    st_frame.image(frame_rgb,
                   use_column_width=True)"""

    # Plot the detected objects on the video frame
    st_count.write(inText + '\n\n' + outText)
    res_plotted = res[0].plot()

    f = st_frame.image(res_plotted,
                   caption='Detected Video',
                   channels="BGR",
                   use_column_width=True
                   )
    @st.cache_resource
def load_model(model_path):
    """
```

```
    Loads a YOLO object detection model from the specified
model_path.
    Parameters:
        model_path (str): The path to the YOLO model file.

    Returns:
        A YOLO object detection model.
    """
    model = YOLO(model_path)
    return model

def url_video(conf, model):
    # global source_url
    source_url = st.sidebar.text_input("Please insert url")
    is_display_tracker, tracker = display_tracker_options()

    if st.sidebar.button('Detect Objects'):


        try:
            class_counts = {}
            #u = str(source_url)


            st_count = st.empty()
            st_frame = st.empty()
            vid_cap = cv2.VideoCapture(source_url,cv2.CAP_FFMPEG)
            vid_cap.set(cv2.CAP_PROP_BUFFERSIZE,0)
            #vid_cap.get(cv2.CAP_PROP_POS_MSEC) / 1000.0
            # Set up OpenCV to read from the URL
            #vid_cap = cv2.VideoCapture(source_url)
            #cv2.imshow("image",image)
            st.button("Clear Lines", on_click=clear_lines)
            #cv2.namedWindow("Video")
```

```
        #cv2.setMouseCallback("Video", draw_line)
        # Load the satellite image to select destination points
        # Load YOLOv8 model
        Homo.select_points
        Homo.select_dst_points
        Homo.get_bird_eye_view
        model

        # Set up the mouse callback to select destination points on
the satellite image
        cv2.namedWindow('Select Destination Points')
        cv2.setMouseCallback('Select Destination Points',
Homo.select_dst_points, satellite_img_copy)

        print("Select at least 4 points on the satellite image to
define the destination points.")
        while True:
            cv2.imshow('Select Destination Points',
satellite_img_copy)
            if len(Homo.dst_points) >= 4 and cv2.waitKey(1) & 0xFF
== ord('c'):  # Press 'c' to confirm selection
                break
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        cv2.destroyAllWindows()
        # Convert dst_points to a numpy array
        #dst_points = np.array(dst_points, dtype='float32')

        # Open the video and display the first frame to select
source points
        # source_url =
```

```
            # video_path = source_url
            video_path = source_url
            cap = cv2.VideoCapture(video_path)
            ret, image = cap.read()
            cap.release()

            if ret:
                cv2.namedWindow('Select Source Points')
                cv2.setMouseCallback('Select Source Points',
Homo.select_points, image)

                print("Select at least 4 points on the video frame to
define the source points.")
                while True:
                    cv2.imshow('Select Source Points', image)
                    if len(Homo.src_points) >= 4 and cv2.waitKey(1) & 0xFF
== ord('c'):  # Press 'c' to confirm selection
                        break
                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break

                cap.release()
                cv2.destroyAllWindows()

                # Define the size of the output video
                output_size = (satellite_img.shape[1],
satellite_img.shape[0])

                Homo.get_bird_eye_view(video_path, Homo.src_points,
Homo.dst_points, output_size, model, satellite_img_path)
            else:
                print("Failed to open video.")
```

```
            while vid_cap.isOpened():
                success, image = vid_cap.read()

                # Draw all the lines

                for line in draw.lines:
                    cv2.line(image, line[0], line[1], (0, 255, 0), 2)

                # Display the frame in Streamlit
                #print(printCoordinate)

                #print(draw.lines)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break

                #line = cv2.line(image,(170,250),(155,50),(0,0,255),2) #
galaxy
                #line = cv2.line(image,(40,70),(300,70),(0,0,255),2)
                #line = cv2.line(image,(300,150),(20,130),(0,0,255),2) #jarin
                # Draw all the lines

                if success:

                    _display_detected_frames(conf,
                                             model,
                                             st_frame,
                                             st_count,
                                             image
                    )

                else:
```

```
                    vid_cap.release()
                    cv2.destroyAllWindows()

                    break
            except Exception as e:
                st.sidebar.error("Error loading video:"+str(e))
    def display_tracker_options():
        display_tracker = st.radio("Display Tracker", ('Yes', 'No'))
        is_display_tracker = True if display_tracker == 'Yes' else False
        if is_display_tracker:
            tracker_type = st.radio("Tracker", ("bytetrack.yaml",
"botsort.yaml"))
            return is_display_tracker, tracker_type
        return is_display_tracker, None
```

**ชุดคำสั่ง predict.py**

```
# Ultralytics YOLO 🚀, GPL-3.0 license

import torch
import cv2
import torch
from numpy import random
from ultralytics.yolo.engine.predictor import BasePredictor
from ultralytics.yolo.engine.results import Results
from ultralytics.yolo.utils import DEFAULT_CFG, ROOT, ops
from ultralytics.yolo.utils.plotting import Annotator
import config
import cv2
from ultralytics.yolo.v8.detect.deep_sort_pytorch.utils.parser import
get_config
from ultralytics.yolo.v8.detect.deep_sort_pytorch.deep_sort import DeepSort
from collections import deque
import numpy as np
```

```
from os import getcwd
#import utils
import draw
palette = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)
data_deque = {}

deepsort = None
#count_up = 0
#count_down = 0
object_counter = {}

object_counter1 = {}

line=[(draw.lines)]
#line = [(170,250),(155,50)] #galaxy hori
#line = [(500, 100),(500, 1050)] #galaxy Verti
#line = [(90,50),(155,50)]
#line = [(300,150),(20,130)] #jarin
#line = lines[0][1]

cwd = getcwd()
cfg_deep = get_config()
cfg_deep.merge_from_file(cwd+"/ultralytics/yolo/v8/detect/deep_sort_pytorch/configs/deep_sort.yaml")

deepsort= DeepSort(cfg_deep.DEEPSORT.REID_CKPT,
                    max_dist=cfg_deep.DEEPSORT.MAX_DIST,
min_confidence=cfg_deep.DEEPSORT.MIN_CONFIDENCE,
                    nms_max_overlap=cfg_deep.DEEPSORT.NMS_MAX_OVERLAP,
max_iou_distance=cfg_deep.DEEPSORT.MAX_IOU_DISTANCE,
                    max_age=cfg_deep.DEEPSORT.MAX_AGE,
n_init=cfg_deep.DEEPSORT.N_INIT,
nn_budget=cfg_deep.DEEPSORT.NN_BUDGET,
                    use_cuda=True)
##########################################################
################################
```

```
def xyxy_to_xywh(xyxy):
    """ Calculates the relative bounding box from absolute pixel values. """
    x_c = (xyxy[..., 0] + xyxy[..., 2]) / 2  # x center
    y_c = (xyxy[..., 1] + xyxy[..., 3]) / 2  # y center
    w = xyxy[..., 2] - xyxy[..., 0]  # width
    h = xyxy[..., 3] - xyxy[..., 1]  # height
    return x_c, y_c, w, h

def xyxy_to_tlwh(bbox_xyxy):
    tlwh_bboxs = []
    for i, box in enumerate(bbox_xyxy):
        x1, y1, x2, y2 = [int(i) for i in box]
        top = x1
        left = y1
        w = int(x2 - x1)
        h = int(y2 - y1)
        tlwh_obj = [top, left, w, h]
        tlwh_bboxs.append(tlwh_obj)
    return tlwh_bboxs

def compute_color_for_labels(label):
    """
    Simple function that adds fixed color depending on the class
    """
    if label == 0: #person
        color = (85,45,255)
    elif label == 2: # Car
        color = (222,82,175)
    elif label == 3:  # Motobike
        color = (0, 204, 255)
    elif label == 5:  # Bus
        color = (0, 149, 255)
    else:
        color = [int((p * (label ** 2 - label + 1)) % 255) for p in palette]
    return tuple(color)
```

```
def draw_border(img, pt1, pt2, color, thickness, r, d):
    x1,y1 = pt1
    x2,y2 = pt2
    # Top left
    cv2.line(img, (x1 + r, y1), (x1 + r + d, y1), color, thickness)
    cv2.line(img, (x1, y1 + r), (x1, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x1 + r, y1 + r), (r, r), 180, 0, 90, color, thickness)
    # Top right
    cv2.line(img, (x2 - r, y1), (x2 - r - d, y1), color, thickness)
    cv2.line(img, (x2, y1 + r), (x2, y1 + r + d), color, thickness)
    cv2.ellipse(img, (x2 - r, y1 + r), (r, r), 270, 0, 90, color, thickness)
    # Bottom left
    cv2.line(img, (x1 + r, y2), (x1 + r + d, y2), color, thickness)
    cv2.line(img, (x1, y2 - r), (x1, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x1 + r, y2 - r), (r, r), 90, 0, 90, color, thickness)
    # Bottom right
    cv2.line(img, (x2 - r, y2), (x2 - r - d, y2), color, thickness)
    cv2.line(img, (x2, y2 - r), (x2, y2 - r - d), color, thickness)
    cv2.ellipse(img, (x2 - r, y2 - r), (r, r), 0, 0, 90, color, thickness)
    cv2.rectangle(img, (x1 + r, y1), (x2 - r, y2), color, -1, cv2.LINE_AA)
    cv2.rectangle(img, (x1, y1 + r), (x2, y2 - r - d), color, -1, cv2.LINE_AA)

    cv2.circle(img, (x1 +r, y1+r), 2, color, 12)
    cv2.circle(img, (x2 -r, y1+r), 2, color, 12)
    cv2.circle(img, (x1 +r, y2-r), 2, color, 12)
    cv2.circle(img, (x2 -r, y2-r), 2, color, 12)

    return img

def UI_box(x, img, color=None, label=None, line_thickness=None):
    # Plots one bounding box on image img
    tl = line_thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) +
1   # line/font thickness
    color = color or [random.randint(0, 255) for _ in range(3)]
    c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))
```

```
    cv2.rectangle(img, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)
    if label:
        tf = max(tl - 1, 1)  # font thickness
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]

        img = draw_border(img, (c1[0], c1[1] - t_size[1] -3), (c1[0] + t_size[0],
c1[1]+3), color, 1, 8, 2)

        cv2.putText(img, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255, 255],
thickness=tf, lineType=cv2.LINE_AA)

def intersect(A,B,C,D):
    return ccw(A,C,D) != ccw(B,C,D) and ccw(A,B,C) != ccw(A,B,D)

def ccw(A,B,C):
    return (C[1]-A[1]) * (B[0]-A[0]) > (B[1]-A[1]) * (C[0]-A[0])

def get_direction(point1, point2):
    direction_str = ""

    # calculate y axis direction
    if point1[1] > point2[1]:
        direction_str += "South"
    elif point1[1] < point2[1]:
        direction_str += "North"
    else:
        direction_str += ""

    # calculate x axis direction
    if point1[0] > point2[0]:
        direction_str += "East"
    elif point1[0] < point2[0]:
        direction_str += "West"
    else:
        direction_str += ""
```

```
    return direction_str
def draw_boxes(img, bbox, names, object_id, identities=None, offset=(0, 0)):
    line=[]
    for line in draw.lines:
        cv2.line(img, line[0], line[1], (0, 255, 0), 2)
        print(line[0],line[1])
    #cv2.line(img, line[0], line[1], (46,162,112), 3)

    #global count_up
    #global count_down
        height, width, _ = img.shape
        # remove tracked point from buffer if object is losts
        for key in list(data_deque):
            if key not in identities:
                data_deque.pop(key)
        for i, box in enumerate(bbox):
            x1, y1, x2, y2 = [int(i) for i in box]
            x1 += offset[0]
            x2 += offset[0]
            y1 += offset[1]
            y2 += offset[1]

            # code to find center of bottom edge
            #center = (int((x2+x1)/ 2), int((y2+y2)/2))
            center = (int((x1+x2)/ 2), int((y1+y2)/2))
            #cv2.circle(img,(int((x1+x1)/ 2),int((y2+y1)/2)),4,(0,0,255), 5)
            # get ID of object
            id = int(identities[i]) if identities is not None else 0

            # create new buffer for new object
            if id not in data_deque:
                data_deque[id] = deque(maxlen= 64)
            color = compute_color_for_labels(object_id[i])
            obj_name = names[object_id[i]]
            label = '{}{:d}'.format("", id) + ":"+ '%s' % (obj_name)
```

```
    # add center to buffer
    data_deque[id].appendleft(center)
    if len(data_deque[id]) >= 2:
        direction = get_direction(data_deque[id][0], data_deque[id][1])
        if intersect(data_deque[id][0], data_deque[id][1], line[0], line[1]):
            cv2.line(img, line[0],line[1], (255, 255, 255), 3)
            if "South" in direction:
                if obj_name not in object_counter:
                    object_counter[obj_name] = 1
                    #count_up += 1
                else:
                    object_counter[obj_name] += 1
                    #count_up += 1
            if "North" in direction:
                if obj_name not in object_counter1:
                    object_counter1[obj_name] = 1
                    #count_down += 1
                else:
                    object_counter1[obj_name] += 1
                    #count_down += 1
            if "East" in direction:
                if obj_name not in object_counter:
                    object_counter[obj_name] = 1
                    #count_up += 1
                else:
                    object_counter[obj_name] += 1
                    #count_up += 1
            if "West" in direction:
                if obj_name not in object_counter1:
                    object_counter1[obj_name] = 1
                    #count_down += 1
                else:
                    object_counter1[obj_name] += 1
                    #count_down +=1
UI_box(box, img, label=label, color=color, line_thickness=2)
# draw trail
```

```
        for i in range(1, len(data_deque[id])):
            # check if on buffer value is none
            if data_deque[id][i - 1] is None or data_deque[id][i] is None:
                continue
            # generate dynamic thickness of trails
            thickness = int(np.sqrt(64 / float(i + i)) * 1.5)
            # draw trails
            cv2.line(img, data_deque[id][i - 1], data_deque[id][i], color, thickness)

    #4. Display Count in top right corner
        for idx, (key, value) in enumerate(object_counter1.items()):
            cnt_str = str(key) + ":" +str(value)
            cv2.line(img, (width - 500,25), (width,25), [85,45,255], 40)
            cv2.putText(img, f'Number of Vehicles Entering', (width - 500, 35), 0, 1,
[225, 255, 255], thickness=2, lineType=cv2.LINE_AA)
            #cv2.putText(img, "Total Count:"+str(count_up),(width -
250,75),0,1,[225,0,0])
            cv2.line(img, (width - 150, 65 + (idx*40)), (width, 65 + (idx*40)), [85, 45,
255], 30)
            cv2.putText(img, cnt_str, (width - 150, 75 + (idx*40)), 0, 1, [255, 255,
255], thickness = 2, lineType = cv2.LINE_AA)

        for idx, (key, value) in enumerate(object_counter.items()):
            cnt_str1 = str(key) + ":" +str(value)
            cv2.line(img, (20,25), (500,25), [85,45,255], 40)
            cv2.putText(img, f'Numbers of Vehicles Leaving', (11, 35), 0, 1, [225,
255, 255], thickness=2, lineType=cv2.LINE_AA)
            cv2.line(img, (20,65+ (idx*40)), (127,65+ (idx*40)), [85,45,255], 30)
            cv2.putText(img, cnt_str1, (11, 75+ (idx*40)), 0, 1, [225, 255, 255],
thickness=2, lineType=cv2.LINE_AA)

        config.OBJECT_COUNTER = object_counter
        config.OBJECT_COUNTER1 = object_counter1

        return img
```

```
class DetectionPredictor(BasePredictor):

    def get_annotator(self, img):
        return Annotator(img, line_width=self.args.line_width,
example=str(self.model.names))

    def postprocess(self, preds, img, orig_img):
        preds = ops.non_max_suppression(preds,
                                        self.args.conf,
                                        self.args.iou,
                                        agnostic=self.args.agnostic_nms,
                                        max_det=self.args.max_det)

        results = []
        for i, pred in enumerate(preds):
            orig_img = orig_img[i] if isinstance(orig_img, list) else orig_img
            if not isinstance(orig_img, torch.Tensor):
                pred[:, :4] = ops.scale_boxes(img.shape[2:], pred[:, :4],
orig_img.shape)
            path = self.batch[0]
            img_path = path[i] if isinstance(path, list) else path
            results.append(Results(orig_img=orig_img, path=img_path,
names=self.model.names, boxes=pred))
        return results

    def write_results(self, idx, results, batch):
        """Write inference results to a file or directory."""
        p, im, im0 = batch
        log_string = ''
        all_outputs = []
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        self.seen += 1
        im0 = im0.copy()
        if self.source_type.webcam or self.source_type.from_img:  # batch_size
>= 1
```

```
            log_string += f'{idx}: '
            frame = self.dataset.count
        else:
            frame = getattr(self.dataset, 'frame', 0)
        self.data_path = p
        self.txt_path = str(self.save_dir / 'labels' / p.stem) + ('' if
self.dataset.mode == 'image' else f'_{frame}')
        log_string += '%gx%g ' % im.shape[2:]  # print string
        result = results[idx]
        log_string += result.verbose()

        if self.args.save or self.args.show:  # Add bbox to image
            plot_args = dict(line_width=self.args.line_width,
                             boxes=self.args.boxes,
                             conf=self.args.show_conf,
                             labels=self.args.show_labels)
            if not self.args.retina_masks:
                plot_args['im_gpu'] = im[idx]
            self.plotted_img = result.plot(**plot_args)
        self.annotator = self.get_annotator(im0)

        all_outputs.append(result)

        if len(result) == 0:
            return log_string
        for c in result.boxes.cls.unique():
            n = (result.boxes.cls == c).sum()  # detections per class
            log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "

        # Write
        if self.args.save_txt:
            result.save_txt(f'{self.txt_path}.txt', save_conf=self.args.save_conf)
        if self.args.save_crop:
            result.save_crop(save_dir=self.save_dir / 'crops',
file_name=self.data_path.stem)
```

```
        xywh_bboxs = []
        confs = []
        oids = []
        outputs = []
        for r in result:
            x_c, y_c, bbox_w, bbox_h = xyxy_to_xywh(r.boxes.xyxy)
            xywh_obj = [x_c, y_c, bbox_w, bbox_h]
            xywh_bboxs.append(xywh_obj)
            confs.append([r.boxes.conf.item()])
            oids.append(int(r.boxes.cls))
        xywhs = torch.Tensor(xywh_bboxs)
        confss = torch.Tensor(confs)

        outputs = deepsort.update(xywhs, confss, oids, im0)
        if len(outputs) > 0:
            bbox_xyxy = outputs[:, :4]
            identities = outputs[:, -2]
            object_id = outputs[:, -1]

            draw_boxes(im0, bbox_xyxy, self.model.names, object_id,identities)

        return log_string

def predict(cfg=DEFAULT_CFG, use_python=False):
    """Runs YOLO model inference on input image(s)."""
    model = cfg.model or 'yolov8n.pt'
    source = cfg.source if cfg.source is not None else ROOT / 'assets' if (ROOT /
'assets').exists() \
        else 'https://ultralytics.com/images/bus.jpg'

    args = dict(model=model, source=source)
    if use_python:
        from ultralytics import YOLO
        YOLO(model)(**args)
    else:
        predictor = DetectionPredictor(overrides=args)
```

```
        predictor.predict_cli()

if __name__ == "__main__":

    predict()
```

**ชุดคำสั่ง Homography.py**

```
import cv2
import numpy as np
from ultralytics import YOLO
#from utils import source_url
# Global variables to store points
src_points = []
dst_points = []

#import streamlit as st
def select_points(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        src_points.append([x, y])
        print(f"Source point selected: {x, y}")
        # Draw a small circle at the selected point
        cv2.circle(param, (x, y), 5, (0, 255, 0), -1)
        cv2.imshow('Select Source Points', param)

def select_dst_points(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        dst_points.append([x, y])
        print(f"Destination point selected: {x, y}")
        # Draw a small circle at the selected point
        cv2.circle(param, (x, y), 5, (0, 0, 255), -1)
        cv2.imshow('Select Destination Points', param)
# def generate_heatmap(satellite_img, transformed_points,
heatmap_intensity=1):
#     # Initialize an empty heatmap with the same size as the bird's-eye view
#     heatmap = np.zeros((satellite_img.shape[0], satellite_img.shape[1]),
dtype=np.float32)
```

```
def get_bird_eye_view(video_path, src_points, dst_points, output_size, model, satellite_img_path):
    # Open the video file

    cap = cv2.VideoCapture(video_path)

    # Compute the homography matrix using RANSAC for robustness
    H, status = cv2.findHomography(np.array(src_points, dtype='float32'), np.array(dst_points, dtype='float32'), cv2.RANSAC, 5.0)

    # Define a color mapping for different classes
    class_colors = {
        0: (0, 0, 255),     # Person - Red
        1: (0, 255, 0),     # Bicycle - Green
        2: (255, 0, 0),     # Car - Blue
        3: (255, 255, 0),   # Motorcycle - Cyan
        5: (255, 0, 255),   # Bus - Magenta
        7: (0, 255, 255)    # Truck - Yellow
        # Add more classes and colors as needed
    }
    transformed_points = []
    while True:
        ret, image = cap.read()
        if not ret:
            break

        # Apply the homography matrix to the frame
        bird_eye_view = cv2.warpPerspective(image, H, output_size)

        # Reload the original satellite image to clear previous detections
        satellite_img = cv2.imread(satellite_img_path)

        # Detect vehicles in the original frame
        results = model(image)
        detections = results[0].boxes.data.cpu().numpy()  # Extract detections
```

```
        for detection in detections:
            xmin, ymin, xmax, ymax, conf, cls = detection[:6]
            xmin, ymin, xmax, ymax = map(int, [xmin, ymin, xmax, ymax])
            cls = int(cls)

            # Draw bounding box in the original frame
            color = class_colors.get(cls, (255, 255, 255))  # Default to white if
class not found
            cv2.rectangle(image, (xmin, ymin), (xmax, ymax), color, 2)
            # Center of the bounding box
            center = np.array([[0.5 * (xmin + xmax)], [0.5 * (ymin + ymax)], [1]])
            # Apply homography to center point
            transformed_center = np.dot(H, center)
            transformed_center /= transformed_center[2]  # Normalize
            x,y = int(transformed_center[0]),int(transformed_center[1])
            # Draw transformed point in bird eye view
            cv2.circle(bird_eye_view, (x, y), 5, color, -1)

            # Draw transformed point on the satellite image
            cv2.circle(satellite_img, (x, y), 5, color, -1)
            transformed_points.append((x, y))
        # heatmap_overlay = generate_heatmap(satellite_img,
transformed_points)
         # Display the frames
        cv2.imshow('Original Frame', image)
        cv2.imshow('Bird Eye View', bird_eye_view)
        cv2.imshow('Satellite Image with Detections', satellite_img)
        # cv2.imshow('Bird Eye View with Heatmap', heatmap_overlay)
          # Generate the heatmap and overlay it on the bird-eye view

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release everything if job is finished
    cap.release()
    cv2.destroyAllWindows()
```

**ชุดคำสั่ง draw.py**

```
import cv2
drawing = False  # True if mouse is pressed
ix, iy = -1, -1
lines = []

# Mouse callback function

def draw_line(event, x, y, flags, param):
    global ix, iy, drawing, lines,lin

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix, iy = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:

            if lines:
                lines.pop()
            lines.append(((ix, iy), (x, y)))

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        lines.append(((ix, iy), (x, y)))
    #return draw_line ()
```