การพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็ก

Development of an Automatic Approach to Create a High-Performance Object Detection Model from Small Datasets

ธีรธร รักษาเมือง
สุปวีณ์ สัญจร
ชญาดา เอียดทองใส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
สำนักวิชาวิศวกรรมศาสตร์และเทคโนโลยี มหาวิทยาลัยวลัยลักษณ์

ปีการศึกษา 2567

การพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็ก

Development of an Automatic Approach to Create a High-Performance Object Detection Model from Small Datasets

ธีรธร รักษาเมือง
สุปวีณ์ สัญจร
ชญาดา เอียดทองใส

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
สำนักวิชาวิศวกรรมศาสตร์และเทคโนโลยี มหาวิทยาลัยวลัยลักษณ์

ปีการศึกษา 2567

ชื่อเรื่อง	การพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็ก
ชื่อผู้พัฒนา	ธีรธร รักษาเมือง สุปวีณ์ สัญจร ชญาดา เอียดทองใส
อาจารย์ที่ปรึกษา	รศ.ดร. วัฒนพงศ์ เกิดทองมี
ปริญญาและสาขาวิชา	หลักสูตรวิศวกรรมศาสตร์และเทคโนโลยี สาขาวิชาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
ปีการศึกษา	2567

บทคัดย่อ

รายงานวิจัยนี้มีวัตถุประสงค์เพื่อการพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็ก มีเป้าหมายเพื่อแก้ไขข้อจำกัดของการใช้ชุดข้อมูลขนาดใหญ่ โดยโครงการนี้มุ่งเน้นการออกแบบและพัฒนาเฟรมเวิร์คที่สามารถสร้างชุดข้อมูลจำลอง (Synthetic Dataset) โดยอัตโนมัติ การประยุกต์ใช้เทคนิคการเพิ่มข้อมูล (Data Augmentation) และการฝึกโมเดลโดยใช้ YOLOv8 ที่สามารถตรวจจับวัตถุได้อย่างแม่นยำและรวดเร็วในสถานการณ์ที่มีข้อมูลจำกัด นอกจากนี้ ยังได้พัฒนาเครื่องมือที่ช่วยให้กระบวนการสร้างข้อมูลและฝึกโมเดลเป็นไปอย่างอัตโนมัติ โดยวัดผลสำเร็จจากค่าความแม่นยำเฉลี่ย (mAP), Precision, และ Recall ผลลัพธ์ที่ได้ช่วยลดต้นทุนและเพิ่มความสะดวกในการพัฒนาโมเดล ซึ่งเป็นประโยชน์ต่อการนำไปใช้ในงานที่ต้องการความแม่นยำสูงและมีข้อจำกัดด้านทรัพยากร

คำสำคัญ: การตรวจจับวัตถุ, YOLOv8, ชุดข้อมูลขนาดเล็ก, ชุดข้อมูลจำลอง, การเพิ่มข้อมูล

Title	Development of an Automatic Approach to Create a High-Performance Object Detection Model from Small Datasets
Student's Name	Teeratorn Raksamuang Supavee Sonjohn Chayada Iadthongsai
Advisor	Assoc. Prof. Dr. Watthanapong Kerdthongmee
Degree and Program	Bachelor of Engineering Program in Computer Engineering and Artificial Intelligence
Academic Year	2024

Abstract

This research aims to develop an automated approach for creating high-performance object detection models from small datasets. The objective is to address the limitations of using large datasets. The project focuses on designing and developing a framework capable of automatically generating synthetic datasets, applying data augmentation techniques, and training models using YOLOv8 to achieve accurate and efficient object detection in scenarios with limited data. Additionally, tools were developed to automate data generation and model training processes. The success of the model was evaluated using metrics such as mean Average Precision (mAP), Precision, and Recall. The results demonstrate the model's ability to perform effectively in terms of accuracy and speed, reducing development costs and improving convenience. This work is beneficial for applications requiring high precision while operating under resource constraints.

Keywords: Object detection, YOLOv8, small datasets, synthetic datasets, data augmentation

กิตติกรรมประกาศ

รายงานวิจัยฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาและความเอาใจใส่อย่างยิ่งจาก รศ.ดร. วัฒนพงศ์ เกิดทองมี อาจารย์ที่ปรึกษา ผู้ซึ่งให้คำแนะนำปรึกษาตลอดจนปรับปรุงแก้ไขข้อบกพร่องต่างๆ รวมถึงการตรวจสอบและปรับปรุงแก้ไขในทุกขั้นตอน ด้วยความมุ่งมั่นและใส่ใจในทุกขั้นตอนของการดำเนินงาน คณะผู้พัฒนาขอแสดงความขอบคุณอย่างสูงไว้ ณ โอกาสนี้

คณะผู้พัฒนายังขอแสดงความขอบคุณไปยังทีมพัฒนาแบบจำลอง YOLOv8 และ Google Inc. ที่ได้จัดเตรียมแหล่งข้อมูลและเครื่องมือที่มีประโยชน์อย่างยิ่งต่อการศึกษาวิจัยครั้งนี้ ทั้งในด้านการค้นคว้า การพัฒนา และการประยุกต์ใช้งาน ทำให้การดำเนินงานบรรลุเป้าหมายได้อย่างมีประสิทธิภาพ

สุดท้ายนี้ คณะผู้พัฒนาหวังเป็นอย่างยิ่งว่ารายงานวิจัยฉบับนี้จะเป็นประโยชน์แก่ผู้ที่สนใจในหัวข้อดังกล่าว และหากมีข้อบกพร่องใด ๆ คณะผู้พัฒนาขอน้อมรับคำแนะนำเพื่อปรับปรุงและพัฒนางานในอนาคตให้ดียิ่งขึ้นต่อไป

คณะผู้พัฒนา
ธีรธร รักษาเมือง
สุปวีณ์ สัญจร
ชญาดา เอียดทองใส

สารบัญ

เรื่อง **หน้า**

สารบัญตาราง

ตาราง **หน้า**

สารบัญภาพ

ภาพ **หน้า**

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ปัจจุบัน โมเดลตรวจจับวัตถุที่มีประสิทธิภาพสูงส่วนใหญ่มักพัฒนามาจากชุดข้อมูลขนาดใหญ่ ซึ่งมีความหลากหลายของข้อมูลสูง ชุดข้อมูลเหล่านี้ช่วยให้โมเดลสามารถเรียนรู้รูปแบบและความแตกต่างของวัตถุได้อย่างมีประสิทธิภาพ อย่างไรก็ตาม การจัดเตรียมชุดข้อมูลขนาดใหญ่นั้นมีต้นทุนสูง ทั้งในด้านเวลา แรงงาน และทรัพยากรในการเก็บรวบรวมและจัดการข้อมูล ในทางตรงกันข้าม การพัฒนาโมเดลจากชุดข้อมูลขนาดเล็กนั้นมีข้อจำกัดหลายประการ โดยเฉพาะในด้านความแม่นยำ และความหลากหลายของข้อมูลที่ไม่ดีเท่าชุดข้อมูลขนาดใหญ่ ซึ่งประสิทธิภาพของโมเดลที่พัฒนาจากชุดข้อมูลขนาดเล็กอาจลดลงเมื่อเผชิญกับสภาพแวดล้อมที่ซับซ้อน ความท้าทายนี้จึงเป็นจุดเริ่มต้นของโครงการนี้ ที่มุ่งเน้น การพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุที่มีประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็ก

ส่วนสาเหตุที่เรามุ่งเน้นไปในการตรวจจับขวดพลาสติกบนน้ำนั้น เนื่องจากปัญหาขยะลอยน้ำ โดยเฉพาะขวดพลาสติกที่ถูกทิ้งลงแหล่งน้ำธรรมชาติ เป็นหนึ่งในประเด็นสำคัญที่ส่งผลกระทบอย่างร้ายแรงต่อสิ่งแวดล้อมและระบบนิเวศ ขวดพลาสติกที่สะสมในแม่น้ำ ลำคลอง หรือทะเล ไม่เพียงแต่ทำให้แหล่งน้ำสกปรก แต่ยังเป็นอันตรายต่อสิ่งมีชีวิต และยังส่งผลกระทบต่อคุณภาพน้ำที่ใช้ในชีวิตประจำวันของมนุษย์

แม้ว่าจะมีความพยายามในการเก็บรวบรวมขยะจากแหล่งน้ำ แต่การทำงานด้วยแรงงานมนุษย์ล้วน ๆ มีข้อจำกัดดังนั้นการตรวจจับขยะลอยน้ำโดยใช้ AI เป็นทางเลือกที่ช่วยเพิ่มประสิทธิภาพในการจัดการขยะเหล่านี้ได้ อย่างไรก็ตาม การพัฒนาโมเดล AI ที่สามารถตรวจจับขยะลอยน้ำได้อย่างแม่นยำ ยังคงเผชิญกับปัญหาหลักสองประการ ได้แก่ การขาดแคลนชุดข้อมูลที่เหมาะสมสำหรับการฝึกโมเดล และความยากลำบากในการตรวจจับขยะในสภาพแวดล้อมที่หลากหลาย เช่น เงาสะท้อนบนผิวน้ำ หรือสีที่กลมกลืนระหว่างขวดพลาสติกและพื้นน้ำ

1.2 วัตถุประสงค์

1) พัฒนาแนวทางประยุกต์ที่เหมาะสมในการขยายจำนวนของชุดข้อมูลภาพ
2) สร้างโมเดล AI ที่สามารถตรวจจับขวดพลาสติกบนพื้นน้ำได้ด้วยชุดข้อมูลจำลองและมีความแม่นยำไม่ต่ำกว่า 80% เมื่อเทียบกับโมเดลที่สร้างจากชุดข้อมูลจริง
3) พัฒนาเฟรมเวิร์ค AI เพื่อการสร้างชุดข้อมูลจำลองแบบอัตโนมัติ

1.3 ขอบเขตของงาน

1.3.1 การเก็บข้อมูลและจัดการข้อมูล

- เก็บข้อมูลภาพขวดพลาสติกที่ถ่ายในบริบทที่ขวดพลาสติกลอยอยู่บนน้ำในแฟล่งน้ำบริเวณมหาวิทยาลัยวลัยลักษณ์

- ใช้กระบวนการ Data Augmentation เช่น การหมุน การปรับขนาด และการย้ายตำแหน่ง เพื่อเพิ่มความหลากหลายของข้อมูล
- จัดเก็บข้อมูลที่แยกเป็น Training Set และ Test Set สำหรับการพัฒนาและทดสอบโมเดล
- ตรวจสอบและจัดเตรียมข้อมูล เช่น การสร้าง Mask ของพื้นที่น้ำ เพื่อให้เหมาะสมกับการสร้างชุดข้อมูลจำลอง (Synthetic Dataset)

1.3.2. การพัฒนาและฝึกอบรมโมเดล

- พัฒนาโมเดลตรวจจับวัตถุโดยใช้เทคนิค Deep Learning เช่น YOLO (You Only Look Once)
- สร้างชุดข้อมูลจำลอง (Synthetic Dataset) ที่มี Annotation พร้อมใช้งาน เพื่อฝึกโมเดลตรวจจับขวดพลาสติกบนพื้นที่น้ำ
- ใช้กระบวนการ Fine-tuning เพื่อปรับปรุงโมเดลให้เหมาะสมกับการตรวจจับวัตถุในชุดข้อมูลขนาดเล็ก
- ทดสอบและวิเคราะห์โมเดลด้วยเมตริก เช่น mAP (Mean Average Precision), Precision, และ Recall

1.3.3. การพัฒนาระบบอัตโนมัติ

- พัฒนากระบวนการอัตโนมัติสำหรับการสร้างชุดข้อมูลจำลอง ที่สามารถวางฟีเจอร์ขวดพลาสติกบนพื้นที่น้ำในภาพพื้นหลัง
- เฟรมเวิร์คประกอบด้วยฟังก์ชัน rename, extract, และ generate ที่รวมไว้ใน main.py เพื่ออำนวยความสะดวกในการสร้าง dataset ด้วยคำสั่งเดียว

1.3.4. การประเมินผลและปรับปรุงระบบ

- ประเมินความแม่นยำของโมเดลด้วยเมตริก เช่น mAP, Precision, Recall บนชุดข้อมูล Test Set
- เก็บข้อเสนอแนะและข้อคิดเห็นจากผู้ใช้งาน เพื่อนำมาปรับปรุงโมเดลและกระบวนการอัตโนมัติ
- อัปเดตและปรับปรุงโมเดลเพิ่มเติมเมื่อมีข้อมูลใหม่ เช่น การขยายขอบเขตของวัตถุที่ตรวจจับ หรือบริบทของภาพพื้นหลัง

1.3.5. ข้อจำกัดของโครงการ

- การพัฒนาเน้นการทำงานกับ ชุดข้อมูลขนาดเล็ก และ Synthetic Dataset อาจมีข้อจำกัดด้านความแม่นยำเมื่อเทียบกับชุดข้อมูลขนาดใหญ่

- ระบบต้องพึ่งพาคุณภาพของ Mask น้ำ และกระบวนการ Augmentation ซึ่งอาจส่งผลต่อความสมจริงของชุดข้อมูล
- ระบบต้องสามารถปรับตัวรองรับการอัปเดตในอนาคต เพื่อให้เพิ่มฟีเจอร์หรือปรับปรุงโมเดลได้ง่าย

1.4 ประโยชน์ที่คาดว่าจะได้รับ

การพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุจากชุดข้อมูลขนาดเล็กนี้จะช่วยลดเวลาและความซับซ้อนในการสร้างโมเดล ทำให้ผู้พัฒนาสามารถมุ่งเน้นไปที่การปรับปรุงประสิทธิภาพของโมเดลได้อย่างเต็มที่ โดยไม่ต้องเสียเวลาทำซ้ำขั้นตอนเดิม

ระบบเฟรมเวิร์คที่พัฒนาขึ้นสามารถใช้งานได้อย่างยืดหยุ่น โดยผู้ใช้งานสามารถเพิ่มฟีเจอร์ เปลี่ยนพื้นหลัง หรือปรับจำนวนภาพที่ต้องการสร้างได้โดยไม่ต้องแก้โค้ดหลัก ทำให้เหมาะสำหรับการนำไปต่อยอดกับวัตถุประเภทอื่นในอนาคต

นอกจากนี้ โมเดลที่พัฒนาขึ้นจากชุดข้อมูลขนาดเล็กจะมีความสามารถในการใช้งานได้จริง แม้ว่าจะมีข้อมูลจำกัด เหมาะสำหรับองค์กรหรือบุคคลที่มีทรัพยากรน้อย โดยสามารถนำโมเดลไปประยุกต์ใช้ในงานที่ต้องการการตรวจจับวัตถุในเวลาที่รวดเร็วและมีต้นทุนต่ำ

การใช้ชุดข้อมูลขนาดเล็กยังทำให้การเก็บรวบรวมและการเตรียมข้อมูลมีความง่ายและรวดเร็วมากขึ้น ทำให้สามารถสร้างและปรับปรุงโมเดลได้อย่างต่อเนื่องและมีประสิทธิภาพ

ดังนั้น การพัฒนานี้จึงมีความสำคัญต่อการส่งเสริมการเติบโตในด้านเทคโนโลยีการตรวจจับวัตถุ และเปิดโอกาสใหม่ในการสร้างผลิตภัณฑ์ที่มีประสิทธิภาพสูงยิ่งขึ้น

1.5 ผลที่คาดว่าจะได้รับเมื่อเสร็จสิ้นโครงการ

เมื่อโครงการการพัฒนาแนวทางอัตโนมัติในการสร้างโมเดลตรวจจับวัตถุประสิทธิภาพสูงจากชุดข้อมูลขนาดเล็กเสร็จสิ้นลง คาดว่าจะได้รับผลลัพธ์ที่มีความสำคัญหลายประการ ดังนี้:

1. แนวทางประยุกต์ที่เหมาะสมในการขยายจำนวนของชุดข้อมูลภาพ
2. โมเดล AI ที่สามารถตรวจจับขวดพลาสติกบนพื้นน้ำได้ด้วยชุดข้อมูลจำลองและมีความแม่นยำไม่ต่ำกว่า 80% เมื่อเทียบกับโมเดลที่สร้างจากชุดข้อมูลจริง
3. เฟรมเวิร์ค AI เพื่อการสร้างชุดข้อมูลจำลองแบบอัตโนมัติ

ผลที่คาดว่าจะได้รับเหล่านี้จะทำให้โครงการมีผลกระทบเชิงบวกต่อการพัฒนาเทคโนโลยีและการดำเนินงานในภาคธุรกิจต่าง ๆ โดยเฉพาะในด้านการตรวจจับวัตถุที่ต้องการความแม่นยำและรวดเร็ว

1.6 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

ลำดับที่	แผนการดำเนินงาน	ตุลาคม				พฤศจิกายน				ธันวาคม				มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	เขียนข้อเสนอโครงงาน																																
2	สอบหัวข้อเสนอโครงงาน					^	^																										
3	วิเคราะห์และออกแบบระบบ																																
4	สร้าง/train model																																
4.1	data preprocessing																																
4.2	train และ ปรับจูน hyperparameter																																
4.3	ทดสอบประสิทธิภาพของ model																																
4.4	ปรับปรุงประสิทธิภาพของ model																																
5	สอบกลางภาคโครงงาน										^1																						
6	เตรียมความพร้อมก่อนทดสอบประสิทธิภาพของโครงงาน																																
7	สอบความก้าวหน้าโครงงาน															^2	^2																
8	ทดสอบประสิทธิภาพของโครงงาน																																
9	ประเมินและสรุปผลการวิจัย																																

หมายเหตุ หมายถึง ระยะเวลาแผนการดำเนินงาน

^ หมายถึง สอบข้อเสนอโครงงาน

^1 หมายถึง สอบกลางภาคโครงงาน

^2 หมายถึง สอบความก้าวหน้าโครงงาน

1.7 อุปกรณ์ที่ใช้ในการพัฒนางาน

การพัฒนาระบบ ผู้พัฒนาได้เลือกใช้เครื่องมือที่หลากหลาย เพื่อการใช้งานที่หลากหลาย ครบถ้วนตามความต้องการซึ่งมีทั้งซอฟต์แวร์และฮาร์ดแวร์ดังนี้

1.7.1 ด้านซอฟต์แวร์ (Software)

ระบบปฏิบัติการ

- Windows 11

ซอฟต์แวร์สำหรับการพัฒนาระบบ

- **Jupyter Notebook:** ใช้พัฒนาและทดลองโมเดล Machine Learning รวมถึงจัดการข้อมูล (Data Preprocessing)
- **VS Code:** ใช้เขียนและแก้ไขโค้ด โดยเฉพาะโค้ด Python สำหรับการสร้างและจัดการ Synthetic Dataset
- **Google Colab:** ใช้สำหรับพัฒนาและทดสอบโมเดลโดยใช้ทรัพยากรประมวลผลบนคลาวด์ (GPU/TPU)
- **Roboflow:** แพลตฟอร์มที่ใช้จัดการชุดข้อมูล เช่น การทำ Augmentation, การจัดการ Annotation และการเชื่อมต่อกับโมเดล Machine Learning

ซอฟต์แวร์สำหรับจัดการเอกสาร

- **Microsoft Word:** ใช้จัดทำเอกสารรายงาน ผลการศึกษา และบันทึกข้อมูลสำคัญของโครงการ เพื่อให้ได้เอกสารที่มีมาตรฐานและสามารถแก้ไขได้สะดวก
- **Microsoft Excel:** ใช้จัดการและวิเคราะห์ข้อมูลเบื้องต้น รวมถึงการเก็บผลการทดสอบโมเดลต่าง ๆ ในรูปแบบตาราง เพื่อความสะดวกในการเปรียบเทียบและสรุปผล

ไลบรารีและเฟรมเวิร์กสำหรับการพัฒนา

- **Python 3.10:** ภาษาหลักที่ใช้ในการพัฒนาเฟรมเวิร์คทั้งหมด
- **OpenCV (cv2):** สำหรับตรวจจับพื้นที่น้ำด้วย HSV และวางฟีเจอร์ลงบนภาพพื้นหลัง รวมถึงจัดการการรวมภาพและประมวลผลต่าง ๆ
- **NumPy:** ใช้ประมวลผลข้อมูลเชิงตัวเลข และจัดการอาร์เรย์ของภาพ
- **Pillow (PIL):** ใช้สำหรับการ post-process ฟีเจอร์ เช่น การเบลอและเพิ่มความคมชัด
- **rembg:** ไลบรารีสำหรับลบพื้นหลังจากภาพ เพื่อแยกขวดพลาสติกให้มีพื้นหลังโปร่งใสโดยอัตโนมัติ
- **TensorFlow / Keras:** ใช้สำหรับฝึกและปรับแต่งโมเดล Deep Learning เช่น YOLOv8
- **Matplotlib / Seaborn:** สำหรับแสดงผลข้อมูล เช่น กราฟเปรียบเทียบค่าความแม่นยำของโมเดล

1.7.2 ด้านฮาร์ดแวร์ (Hardware)

การพัฒนาระบบ ผู้พัฒนาได้ใช้เครื่องคอมพิวเตอร์สำหรับการพัฒนา ซึ่งคอมพิวเตอร์ผู้พัฒนาสามารถระบุคุณสมบัติทางฮาร์ดแวร์ได้ดังนี้

- **MSI Bravo 17 A4DDK-077TH**
  - CPU: AMD Ryzen 7 4800H (2.90 GHz up to 4.20 GHz)
  - GPU: AMD Radeon RX 5500M
  - RAM: 8 GB DDR4 3200 MHz
  - OS: Windows 10 Home (64 Bit)
- **Acer Aspire 7 A715-42G-R4BX**
  - CPU: AMD Ryzen 5 5500U (2.10 GHz up to 4.00 GHz)
  - GPU: NVIDIA GeForce GTX 1650 4 GB GDDR6
  - RAM: 24 GB DDR4 3200 MHz
  - OS: Windows 11 Home (64 Bit)
- **HP Victus 16-d1213TX**

- CPU: Intel Core i5-12500H (2.50 GHz up to 4.50 GHz)
- GPU: Nvidia GeForce RTX 3050 4GB GDDR6
- RAM: 8 GB DDR5 4800 MHz
- OS: Windows 11 Home 64bit

บทที่ 2

ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การตรวจจับวัตถุ (Object Detection)

การตรวจจับวัตถุ (Object Detection) เป็นเทคนิคในวิทยาการคอมพิวเตอร์ที่ผสานการจำแนกภาพ (Image Classification) และการระบุตำแหน่งวัตถุ (Object Localization) เข้าด้วยกัน เพื่อระบุและหาตำแหน่งของวัตถุภายในภาพหรือวิดีโอ เทคนิคนี้ช่วยให้ระบบสามารถตรวจจับวัตถุหลายชนิดในภาพเดียวกัน กำหนดป้ายกำกับให้กับวัตถุเหล่านั้น และให้ข้อมูลตำแหน่งที่แน่นอนของวัตถุในภาพ

การตรวจจับวัตถุมีบทบาทสำคัญในงานด้านวิทยาการคอมพิวเตอร์หลายด้าน เช่น การใส่คำอธิบายภาพ (Image Annotation) การนับจำนวนยานพาหนะ การรู้จำกิจกรรม (Activity Recognition) การตรวจจับใบหน้า (Face Detection) และการรู้จำใบหน้า (Face Recognition) วิธีการตรวจจับวัตถุสามารถแบ่งออกเป็นสองประเภทหลัก ได้แก่ วิธีที่ใช้เครือข่ายประสาทเทียม (Neural Network-Based) และวิธีที่ไม่ใช้เครือข่ายประสาทเทียม (Non-Neural Approaches)

สำหรับวิธีที่ไม่ใช้เครือข่ายประสาทเทียม จำเป็นต้องกำหนดคุณลักษณะของวัตถุล่วงหน้า เช่น การใช้คุณลักษณะ Haar ในกรอบการทำงานของ Viola-Jones หรือการใช้คุณลักษณะ Histogram of Oriented Gradients (HOG) ในทางกลับกัน วิธีที่ใช้เครือข่ายประสาทเทียมสามารถทำการตรวจจับวัตถุแบบ end-to-end โดยไม่ต้องกำหนดคุณลักษณะเฉพาะ และมักอิงกับเครือข่ายประสาทคอนโวลูชัน (Convolutional Neural Networks - CNN)

โมเดลการตรวจจับวัตถุที่ใช้เครือข่ายประสาทเทียมที่ได้รับความนิยม ได้แก่ R-CNN, Fast R-CNN, Faster R-CNN, YOLO (You Only Look Once), และ SSD (Single Shot MultiBox Detector) และการตรวจจับวัตถุยังมีการประยุกต์ใช้ในงานต่าง ๆ เช่น การติดตามวัตถุในวิดีโอ การวิเคราะห์พฤติกรรม และการนำไปใช้ในระบบยานยนต์อัตโนมัติ

การทำงานทั่วไปของการตรวจจับวัตถุมีดังนี้:

- ภาพอินพุต: กระบวนการตรวจจับวัตถุเริ่มต้นด้วยการวิเคราะห์ภาพหรือวิดีโอ
- การประมวลผลล่วงหน้า: ภาพจะได้รับการประมวลผลล่วงหน้าเพื่อให้แน่ใจว่ารูปแบบเหมาะสมกับแบบจำลองที่ใช้

- การสกัดคุณลักษณะ: แบบจำลอง CNN ถูกใช้เป็นตัวสกัดคุณลักษณะ โดยแบบจำลองจะรับผิดชอบในการแยกภาพออกเป็นส่วนๆ และดึงคุณลักษณะออกจากแต่ละส่วนเพื่อตรวจจับรูปแบบของวัตถุต่างๆ
- การจำแนกประเภท: แต่ละส่วนของภาพจะถูกจำแนกประเภทตามคุณลักษณะที่สกัดออกมา งานการจำแนกประเภทจะดำเนินการโดยใช้ SVM หรือเครือข่ายประสาทเทียมอื่นๆ ที่คำนวณความน่าจะเป็นของแต่ละหมวดหมู่ที่ปรากฏในบริเวณนั้น
- การระบุตำแหน่ง: พร้อมกันกับกระบวนการจำแนกประเภท แบบจำลองจะกำหนดกรอบขอบเขตสำหรับวัตถุที่ตรวจพบแต่ละชิ้น ซึ่งเกี่ยวข้องกับการคำนวณพิกัดสำหรับกรอบที่ล้อมรอบวัตถุแต่ละชิ้น จึงสามารถระบุตำแหน่งภายในภาพได้อย่างแม่นยำ
- การระงับค่าที่ไม่ใช่ค่าสูงสุด: เมื่อแบบจำลองระบุกรอบขอบเขตหลายกรอบสำหรับวัตถุเดียวกัน การระงับค่าที่ไม่ใช่ค่าสูงสุดจะถูกใช้เพื่อจัดการกับการทับซ้อนเหล่านี้ เทคนิคนี้จะเก็บเฉพาะกรอบขอบเขตที่มีคะแนนความเชื่อมั่นสูงสุดเท่านั้น และจะลบกรอบขอบเขตอื่นๆ ที่ทับซ้อนกันออกไป
- ผลลัพธ์: กระบวนการสิ้นสุดลงด้วยภาพต้นฉบับที่ถูกทำเครื่องหมายด้วยกรอบขอบเขตและป้ายกำกับที่แสดงวัตถุที่ตรวจพบและหมวดหมู่ที่เกี่ยวข้อง

วิธีการเรียนรู้เชิงลึกสำหรับการตรวจจับวัตถุ มีวิธีการตรวจจับวัตถุอยู่ 2 ประเภทหลัก:

Two-stage Object Detection: จะแบ่งการทำงานเป็นสองขั้นตอน: ขั้นตอนแรกจะเสนอภูมิภาคที่เป็นตัวเลือก จากนั้นจึงจัดหมวดหมู่ภูมิภาคนั้น เครื่องตรวจจับสองขั้นตอนบางประเภทได้แก่ R-CNN, Fast R-CNN และ Fastพ R-CNN

Single-Stage Detectors for Object Detection จะคาดการณ์กล่องขอบเขตและความน่าจะเป็นของคลาสสำหรับทุกพื้นที่ของภาพได้อย่างแม่นยำ เน้นที่การรวมงานการระบุตำแหน่งวัตถุและการจำแนกประเภทเข้าในเครือข่ายประสาทเทียมแบบผ่านครั้งเดียว

โครงการนี้เลือกใช้ YOLOv8 ซึ่งเป็นโมเดลตรวจจับวัตถุแบบ Single-Stage Detector ที่มีความเร็วและแม่นยำสูง สามารถนำไปฝึกด้วยชุดข้อมูลที่มีจำนวนจำกัดได้ โดยเฉพาะเมื่อชุดข้อมูลนั้นถูกสร้างขึ้นอย่างเหมาะสมจากการทำ Synthetic Dataset

2.1.2 การเพิ่มข้อมูล (Data Augmentation)

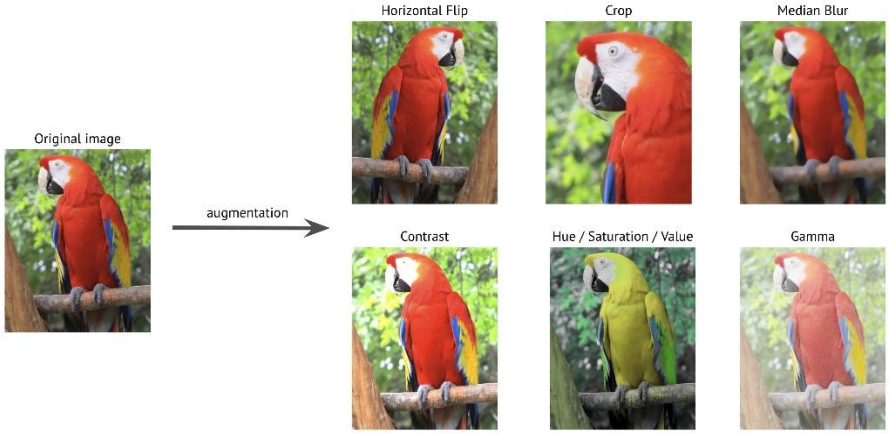
การทำ Augmented Images เพื่อที่จะปรับปรุงประสิทธิภาพของโมเดล ซึ่งเป็นขั้นตอนที่การ augmentations จะถูกนำไปใช้กับรูปภาพที่มีอยู่ในชุดข้อมูล กระบวนการนี้สามารถช่วยปรับปรุงความแม่นยำของโมเดล เมื่อโมเดลชุดข้อมูลรูปภาพใหม่ที่ไม่เคยเห็น โดยการ augmentations ภาพมีดังต่อไปนี้: พลิกกลับ หมุน 90 องศา หมุนแบบสุ่ม ครอบตัดแบบสุ่ม การเฉือนแบบสุ่ม การเบลอ การเปิดรับแสง สัญญาณรบกวนแบบสุ่ม การตัดออก และโมเสก

รูปแบบการปรับแต่งข้อมูล (Augmentation Options)

- การพลิกภาพ (Flip):การพลิกภาพคือการสะท้อนภาพไปตามแกนเฉพาะเพื่อสร้างความหลากหลายในชุดข้อมูลและจำลองสถานการณ์ในโลกจริงที่วัตถุอาจสะท้อนกลับได้ มีสองประเภทหลัก ได้แก่ การพลิกแนวนอน ซึ่งจะพลิกภาพในทิศทางซ้าย/ขวา และการพลิกแนวตั้ง ซึ่งจะพลิกภาพในทิศทางขึ้น/ลง
- การหมุนแบบสุ่ม (Random Rotation): หมุนภาพรอบจุดศูนย์กลางด้วยมุมที่สุ่มเลือก โดยอาจหมุนตามหรือทวนเข็มนาฬิกา วิธีนี้ช่วยให้โมเดลมีความยืดหยุ่นต่อการเปลี่ยนแปลงทิศทางของวัตถุในภาพ
- การครอบแบบสุ่ม (Random Crop): ตัดหรือเลือกส่วนหนึ่งของภาพในขณะที่ละทิ้งส่วนที่เหลือ การครอบแบบสุ่มช่วยสร้างความหลากหลายในชุดข้อมูลโดยเน้นไปที่พื้นที่ต่างๆ ของภาพ และช่วยให้โมเดลเรียนรู้คุณลักษณะที่สำคัญได้ดียิ่งขึ้น
- การบิดแบบสุ่ม (Random Shear): บิดภาพในแนวนอนหรือแนวตั้งแบบสุ่ม เพื่อเพิ่มความหลากหลายในมุมมองของวัตถุ
- การปรับแสง (Exposure): เปลี่ยนค่าความสว่างหรือความมืดของภาพโดยการปรับค่าการรับแสง (gamma exposure)
- การเบลอ (Blur): เพิ่มเอฟเฟกต์เบลอให้กับภาพ เช่น การเบลอแบบ Gaussian เพื่อสร้างความสมจริงหรือจำลองการถ่ายภาพที่มีความไม่ชัดเจน
- การเพิ่มสัญญาณรบกวนแบบสุ่ม (Random Noise): เพิ่มสัญญาณรบกวนในภาพ เช่น สัญญาณรบกวนแบบเกลือและพริกไทย (salt and pepper noise) เพื่อจำลองภาพที่มีคุณภาพต่ำหรือมีความซับซ้อน
- การปรับแต่งกรอบคำอธิบายภาพ (Bounding Box Augmentation): ปรับเปลี่ยนกรอบคำอธิบายภาพ (bounding box) เพื่อให้เน้นเฉพาะเนื้อหาภายในกรอบข้อมูล วิธีนี้ช่วยสร้างชุดข้อมูลที่เหมาะสมสำหรับการเรียนรู้เชิงลึก

การใช้เทคนิคทั้งหมดนี้ช่วยเพิ่มความหลากหลายในชุดข้อมูล ทำให้โมเดลสามารถรับมือกับความแตกต่างของข้อมูลในโลกจริงได้ดียิ่งขึ้น

สำหรับโครงการนี้ การทำ Data Augmentation ดำเนินการผ่านแพลตฟอร์ม Roboflow ซึ่งสามารถกำหนดค่าการหมุน ปรับแสง และเพิ่ม Noise ได้ตามต้องการ เพื่อเสริมความหลากหลายให้กับชุดข้อมูลที่ได้จากเฟรมเวิร์ค



ภาพที่ 1 การทำ Augmented Images

https://albumentations.ai/docs/images/introduction/image_augmentation/augmentation.jpg

2.1.3 Mean Average Precision (mAP)

ความแม่นยำเฉลี่ย (Mean Average Precision หรือ mAP) คือตัวชี้วัดประสิทธิภาพที่ใช้ในการประเมินโมเดลการเรียนรู้ของเครื่อง โดยเป็นตัวชี้วัดที่ได้รับความนิยมมากที่สุดซึ่งใช้โดยความท้าทายด้านเกณฑ์มาตรฐานต่างๆ เช่น PASCAL VOC, COCO, ImageNET challenge, Google Open Image Challenge เป็นต้น ความแม่นยำเฉลี่ยเฉลี่ยมีความหมายต่างกันไปในแต่ละแพลตฟอร์ม ดังนั้น จึงมักจะทำให้สับสนได้

$$mAP = \frac{1}{N}\sum_{i}^{N} AP_i$$

โดยที่ mAP จะใช้เกณฑ์ย่อยดังต่อไปนี้

Confusion Matrix เป็นเครื่องมือที่ใช้ในการประเมินประสิทธิภาพของ Machine Learning โดยเฉพาะในงานจำแนกประเภท (Classification) เป็นวิธีการเปรียบเทียบระหว่างค่าที่ทำนาย (Predicted values) และค่าจริง (True values) โดยตารางที่แสดงจำนวนของการคาดการณ์ที่ถูกต้องและผิดพลาดในแต่ละคลาส (Class) ตารางนี้จะมีขนาด n×n โดยที่ n คือจำนวนคลาสทั้งหมดในข้อมูล และมี 4 ส่วนหลัก ๆ คือ:

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

- False Negative (FN) = สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้น คือ จริง

Intersection Over Union (IoU) คือหน่วยวัดที่ใช้ประเมินความแม่นยำการตรวจจับวัตถุโดยทั่วไป IoU จะใช้ร่วมกับเกณฑ์เพื่อพิจารณาว่าการตรวจจับนั้นถูกต้องหรือไม่ แบบจำลองการตรวจจับวัตถุโดยเปรียบเทียบกล่องขอบเขตที่คาดการณ์ไว้กับกล่องขอบเขตพื้นฐาน หน่วยวัดนี้จะวัดว่ากล่องทั้งสองทับซ้อนกันมากเพียงใดเป็นเศษส่วนของพื้นที่รวมของกล่องทั้งสอง ดังสมการที่ (2)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \qquad (2)$$

โดยที่:
Area of Overlap: พื้นที่ที่กรอบขอบเขตที่คาดการณ์ไว้และกรอบขอบเขตของข้อมูลพื้นฐานทับซ้อนกัน
Area of Union: พื้นที่ทั้งหมดที่ครอบคลุมโดยทั้งกรอบขอบเขตที่คาดการณ์ไว้และกรอบขอบเขตของข้อมูลพื้นฐาน ไม่รวมพื้นที่ทับซ้อน โดยจะสามารถแสดงเป็นสูตรทางคณิตศาสตร์ ดังสมการที่ (3)

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \qquad (3)$$

โดยที่
A คือกรอบขอบเขตพื้นฐาน
B คือกรอบขอบเขตที่คาดการณ์ไว้
$|A \cap B|$ คือพื้นที่จุดตัดของทั้งสอง
$|A \cup B|$ คือพื้นที่การรวมตัวของทั้งสอง

Recall หรือ Sensitivity บอกถึง ความครอบคลุม ว่ากฎหรือตัวจำแนกสามารถตรวจพบกรณีบวกจริงได้มากน้อยเพียงใด มันมุ่งเน้นไปที่การลดจำนวน False Negative เพื่อไม่ให้พลาดกรณีสำคัญที่ควรจะถูกระบุ

ในด้าน Information Retrieval (IR) นั้น Recall มักไม่ได้รับความสำคัญมากนัก เนื่องจากสมมติฐานที่ว่าเอกสารที่เกี่ยวข้องมีอยู่จำนวนมาก และไม่สำคัญว่าเราจะค้นพบชุดย่อยใด นอกจากนี้เรายังไม่สามารถทราบเกี่ยวกับความเกี่ยวข้องของเอกสารที่ไม่ได้ถูกดึงมาได้ ในด้าน Machine Learning และ Computational Linguistics Recall มักถูกละเลยหรือนำมารวมเฉลี่ยไปกับปัจจัยอื่น ๆ โดยเน้นไปที่ความมั่นใจในกฎหรือตัวจำแนกเป็นหลักดังสมการที่ (4)

$$\text{Recall} = TP / RP = A / (A + C) \qquad (4)$$

โดยที่
TP (True Positive): จำนวนกรณีบวกจริงที่ถูกทำนายถูกต้อง
RP (Real Positive): จำนวนกรณีทั้งหมดที่เป็นบวกจริง
A: กรณีที่เป็นบวกจริงและทำนายถูกต้อง
C: กรณีที่เป็นบวกจริงแต่ทำนายผิด (False Negative)

Precision หรือที่บางครั้งเรียกว่า Confidence (ในบริบทของ Data Mining) หมายถึง สัดส่วนของกรณีที่ทำนายว่าเป็นบวก (Predicted Positive) ที่เป็นบวกจริง (Real Positive)

ความสำคัญของ Precision ในด้าน Machine Learning, Data Mining และ Information Retrieval, Precision เป็นตัววัดที่สำคัญ เพราะแสดงให้เห็นว่าคำทำนายบวกที่ทำไปนั้นมีความถูกต้องแม่นยำเพียงใด ในการระบุกรณีบวก โดยเป็นสัดส่วนระหว่างความสำเร็จ (True Positive) กับความพยายามทั้งหมด (Predicted Positive) อย่างไรก็ตาม ใน ROC Analysis, Precision มักไม่ได้รับการพิจารณาโดยตรง ดังสมการที่ (5)

$$\text{Precision} = TP / PP = A / (A + B) \qquad (5)$$

โดยที่
TP (True Positive): จำนวนกรณีบวกจริงที่ถูกทำนายถูกต้อง
PP (Predicted Positive): จำนวนกรณีทั้งหมดที่ถูกทำนายว่าเป็นบวก
A: กรณีที่เป็นบวกจริงและทำนายถูกต้อง
B: กรณีที่ถูกทำนายว่าเป็นบวก แต่ไม่ใช่บวกจริง (False Positive)

2.1.4 การประมวลผลภาพ (Image Processing)

การประมวลผลภาพประกอบด้วยขั้นตอนหลักๆ ดังต่อไปนี้:

1. การนำเข้าภาพผ่านเครื่องมือรับภาพ
2. การวิเคราะห์และปรับแต่งภาพ
3. เอาต์พุตที่สามารถเปลี่ยนผลลัพธ์ได้ ภาพหรือรายงานที่อิงจากการวิเคราะห์ภาพนั้น

โดยที่ภาพจะถูกกำหนดให้เป็นฟังก์ชันสองมิติ F(x,y) โดยที่ x และ y เป็นพิกัดเชิงพื้นที่ และแอมพลิจูดของ F ที่พิกัดคู่ใดๆ (x,y) เรียกว่าความเข้มของภาพนั้น ณ จุดนั้น เมื่อค่า x, y และแอมพลิจูดของ F มีจำกัด เราจะเรียกว่าภาพดิจิทัล กล่าวอีกนัยหนึ่ง ภาพสามารถกำหนดได้โดยอาร์เรย์สองมิติที่จัดเรียงเป็นแถวและคอลัมน์โดยเฉพาะ

ภาพดิจิทัลประกอบด้วยองค์ประกอบจำนวนจำกัด โดยแต่ละองค์ประกอบจะมีค่าเฉพาะที่ตำแหน่งเฉพาะ องค์ประกอบเหล่านี้เรียกว่าองค์ประกอบภาพ องค์ประกอบภาพ และพิกเซล พิกเซลถูกใช้กันอย่างแพร่หลายที่สุดเพื่อระบุองค์ประกอบของภาพดิจิทัล

ประเภทของภาพ:

- ภาพไบนารี นั้นประกอบด้วยองค์ประกอบพิกเซลเพียงสององค์ประกอบ คือ 0 และ 1 โดย 0 หมายถึงสีดำ และ 1 หมายถึงสีขาว ภาพนี้เรียกอีกอย่างว่าภาพขาวดำ
- ภาพขาวดำ เป็นภาพที่ประกอบด้วยสีขาวดำเท่านั้นเรียกว่าภาพขาวดำ
- รูปแบบสี 8 บิต เป็นรูปแบบภาพที่มีชื่อเสียงที่สุด มีเฉดสีที่แตกต่างกัน 256 เฉด และเรียกอีกอย่างว่าภาพระดับสีเทา ในรูปแบบนี้ 0 หมายถึงสีดำ 255 หมายถึงสีขาว และ 127 หมายถึงสีเทา
- รูปแบบสี 16 บิต เป็นรูปแบบภาพสี มีสีที่แตกต่างกัน 65,536 สี เรียกอีกอย่างว่ารูปแบบสีสูง ในรูปแบบนี้ การกระจายสีจะไม่เหมือนกับภาพระดับสีเทา และรูปแบบ 16 บิตยังแบ่งออกเป็นรูปแบบเพิ่มเติมอีกสามรูปแบบ ได้แก่ สีแดง สีเขียว และสีน้ำเงิน ซึ่งเป็นรูปแบบ RGB

ขั้นตอนการประมวลผลภาพ:

1. การรับภาพ (Image Acquisition):

ขั้นตอนนี้เกี่ยวข้องกับการได้มาซึ่งภาพและการปรับให้เหมาะสมกับการประมวลผล โดยงานหลักประกอบด้วย:

- การปรับขนาดภาพให้เหมาะสมกับการใช้งาน
- การแปลงสี เช่น การแปลงภาพจาก RGB เป็นสีเทา หรือในทางกลับกัน

2. การปรับปรุงภาพ (Image Enhancement):

เป็นกระบวนการที่ง่ายและน่าสนใจที่สุดในงานประมวลผลภาพ ใช้เพื่อเพิ่มคุณภาพภาพหรือดึงรายละเอียดที่ซ่อนอยู่ในภาพ ซึ่งมักขึ้นอยู่กับความต้องการของผู้ใช้งาน

3. การฟื้นฟูภาพ (Image Restoration):

มุ่งเน้นการปรับปรุงภาพที่เสียหายหรือด้อยคุณภาพ โดยใช้แบบจำลองทางคณิตศาสตร์หรือความน่าจะเป็นเพื่อฟื้นฟูภาพให้ใกล้เคียงกับต้นฉบับมากที่สุด

4. การประมวลผลภาพสี (Color Image Processing):

เกี่ยวข้องกับการจัดการภาพสีและการปรับปรุงภาพที่มีสีเต็มรูปแบบหรือสีเทียม การใช้แบบจำลองสีช่วยในการปรับแต่งสีในภาพดิจิทัลได้อย่างมีประสิทธิภาพ

5. เวฟเล็ตและการประมวลผลความละเอียดหลายระดับ (Wavelets and Multiresolution Processing):

เป็นพื้นฐานในการแสดงภาพในหลายระดับความละเอียด ช่วยให้สามารถประมวลผลภาพในหลายสเกลได้อย่างละเอียด

6. การบีบอัดภาพ (Image Compression):

กระบวนการลดขนาดหรือความละเอียดของภาพ เพื่อประหยัดพื้นที่จัดเก็บและเพิ่มความเร็วในการส่งข้อมูล

7. การประมวลผลทางสัณฐานวิทยา (Morphological Processing):

ใช้สำหรับการแยกและวิเคราะห์ส่วนประกอบในภาพ เช่น รูปร่าง โครงสร้าง หรือเส้นขอบ เพื่อช่วยอธิบายคุณลักษณะต่างๆ

8. การแบ่งส่วนภาพ (Image Segmentation):

เป็นกระบวนการแยกภาพออกเป็นส่วนต่างๆ หรือวัตถุที่สนใจ การแบ่งส่วนอัตโนมัติถือเป็นงานที่ซับซ้อนที่สุดในงานประมวลผลภาพ

9. การแสดงและคำอธิบาย (Representation and Description):

กระบวนการที่ต่อเนื่องจากการแบ่งส่วน ใช้เพื่อแปลงข้อมูลดิบที่ได้จากการแบ่งส่วนให้เป็นข้อมูลที่พร้อมใช้งาน โดยการเลือกวิธีการแสดงผลมีความสำคัญต่อการแปลความหมายของข้อมูล

10. การตรวจจับและการรับรู้วัตถุ (Object Detection and Recognition):

มุ่งเน้นการกำหนดป้ายกำกับให้กับวัตถุในภาพตามลักษณะหรือตัวอธิบายของวัตถุนั้น ช่วยให้ระบบสามารถระบุและจดจำวัตถุได้อย่างแม่นยำ

ข้อดีของการประมวลผลภาพดิจิทัล:

- คุณภาพของภาพที่ดีขึ้น: อัลกอริธึมการประมวลผลภาพดิจิทัลสามารถปรับปรุงคุณภาพของภาพของภาพ ทำให้ภาพชัดเจนขึ้น คมชัดขึ้น และให้ข้อมูลมากขึ้น
- งานที่ใช้ภาพอัตโนมัติ: การประมวลผลภาพดิจิทัลสามารถทำให้การทำงานที่ใช้ภาพหลายอย่างเป็นอัตโนมัติ เช่น การจดจำวัตถุ การตรวจจับรูปแบบ และการวัด
- ประสิทธิภาพที่เพิ่มขึ้น: อัลกอริธึมการประมวลผลภาพดิจิทัลสามารถประมวลผลภาพได้เร็วกว่ามนุษย์มาก ทำให้สามารถวิเคราะห์ข้อมูลจำนวนมากได้ในเวลาอันสั้น
- ความแม่นยำที่เพิ่มขึ้น: อัลกอริธึมการประมวลผลภาพดิจิทัลสามารถให้ผลลัพธ์ที่แม่นยำกว่ามนุษย์ โดยเฉพาะสำหรับงานที่ต้องมีการวัดที่แม่นยำหรือการวิเคราะห์เชิงปริมาณ

ข้อเสียของการประมวลผลภาพดิจิทัล:

- ต้นทุนการคำนวณสูง: อัลกอริธึมการประมวลผลภาพดิจิทัลบางตัวต้องใช้การคำนวณจำนวนมากและต้องใช้ทรัพยากรการคำนวณจำนวนมาก
- ความสามารถในการตีความจำกัด: อัลกอริธึมการประมวลผลภาพดิจิทัลบางตัวอาจให้ผลลัพธ์ที่มนุษย์ตีความได้ยาก โดยเฉพาะอย่างยิ่งสำหรับอัลกอริธึมที่ซับซ้อนหรือล้ำสมัย
การพึ่งพาคุณภาพของอินพุต: คุณภาพของเอาต์พุตของอัลกอริธึมการประมวลผลภาพ

- ดิจิทัลนั้นขึ้นอยู่กับคุณภาพของภาพอินพุตเป็นอย่างมาก ภาพอินพุตที่มีคุณภาพต่ำอาจส่งผลให้เอาต์พุตมีคุณภาพต่ำ
- ข้อจำกัดของอัลกอริธึม: อัลกอริธึมการประมวลผลภาพดิจิทัลมีข้อจำกัด เช่น ความยากลำบากในการจดจำวัตถุในฉากที่รกหรือมีแสงไม่เพียงพอ หรือไม่สามารถจดจำวัตถุที่มีการบิดเบือนหรือการบดบังอย่างมีนัยสำคัญ
- การพึ่งพาข้อมูลการฝึกอบรมที่ดี: ประสิทธิภาพของอัลกอริธึมการประมวลผลภาพดิจิทัลจำนวนมากขึ้นอยู่กับคุณภาพของข้อมูลการฝึกอบรมที่ใช้ในการพัฒนาอัลกอริธึม ข้อมูลการฝึกอบรมที่มีคุณภาพต่ำอาจส่งผลให้ประสิทธิภาพของอัลกอริธึมลดลง

2.2 เทคโนโลยีที่นำมาใช้ในการพัฒนาระบบ

2.2.1 RoboFlow

RoboFlow เป็นเครื่องมือที่ออกแบบมาสำหรับนักพัฒนา เพื่อช่วยปรับปรุงกระบวนการสร้างและพัฒนาโมเดลคอมพิวเตอร์วิชัน (Computer Vision) ซึ่งกำลังได้รับความนิยมในหลากหลายอุตสาหกรรม ด้วยฟีเจอร์ที่ครอบคลุม RoboFlow ช่วยลดความซับซ้อนและเร่งกระบวนการพัฒนาโมเดลตั้งแต่ต้นจนจบ

จุดเด่นของ RoboFlow

- การจัดการชุดข้อมูลภาพ
- รองรับการอัปโหลด ใส่คำอธิบายประกอบ (Annotation) และประมวลผลชุดข้อมูลภาพล่วงหน้า
- ช่วยให้นักพัฒนาสามารถมุ่งเน้นไปที่การฝึกและปรับแต่งโมเดลโดยไม่เสียเวลาไปกับการเตรียมข้อมูล

คุณสมบัติหลักของ RoboFlow

- การเพิ่มข้อมูล (Data Augmentation): RoboFlow นำเสนอเทคนิคการเพิ่มข้อมูลที่หลากหลาย เช่น การหมุน (Rotation), การพลิก (Flip), และการปรับขนาด (Resizing) เพื่อสร้างความหลากหลายในชุดข้อมูลการฝึกอบรม และช่วยเพิ่มประสิทธิภาพของโมเดล
- การฝึกอบรมโมเดล (Model Training):
- สามารถใช้โมเดลที่ผ่านการฝึกอบรมล่วงหน้า (Pre-trained Models)
- ฝึกอบรมโมเดลเฉพาะของตนเองบนแพลตฟอร์ม RoboFlow
- ปรับแต่งไฮเปอร์พารามิเตอร์และสถาปัตยกรรมโมเดลเพื่อเพิ่มความแม่นยำ

การนำไปใช้งาน (Deployment): RoboFlow รองรับการปรับใช้โมเดลที่ฝึกอบรมแล้วในแพลตฟอร์มต่างๆ เช่น:

- บริการคลาวด์ (Cloud Services): สำหรับแอปพลิเคชันออนไลน์
- อุปกรณ์เอดจ์ (Edge Devices): เช่น กล้อง IoT หรือระบบฝังตัว

ประโยชน์ของการใช้ RoboFlow

- ประหยัดเวลา: ลดเวลาในกระบวนการเตรียมข้อมูลและการตั้งค่า
- ความยืดหยุ่นสูง: รองรับการปรับแต่งโมเดลตามความต้องการ
- เพิ่มประสิทธิภาพของโมเดล: ด้วยเครื่องมือสำหรับการเพิ่มข้อมูลและการทดลองไฮเปอร์พารามิเตอร์

ระบบเฟรมเวิร์คที่พัฒนาขึ้น ได้เตรียมชุดข้อมูลให้อยู่ในโครงสร้างที่ตรงกับมาตรฐานของ Roboflow โดยอัตโนมัติ เพื่อให้สามารถนำเข้าไปใช้งานได้ทันทีโดยไม่ต้องปรับโครงสร้างเพิ่มเติม

2.2.2 opencv-python

OpenCV (Open-Source Computer Vision Library) เป็นไลบรารีโอเพ่นซอร์สที่ได้รับความนิยมอย่างมากในด้านการประมวลผลภาพและการมองเห็นด้วยคอมพิวเตอร์ (Computer Vision) โดย OpenCV-Python เป็นตัวเชื่อม (binding) ที่ช่วยให้นักพัฒนาสามารถใช้ฟังก์ชันของ OpenCV ได้ง่ายผ่านภาษา Python โดยในปัจจุบัน Opencv มีบทบาทสำคัญในการทำงานแบบเรียลไทม์ ซึ่งมีความสำคัญมากในระบบปัจจุบัน โดยการใช้ Opencv จะทำให้สามารถประมวลผลรูปภาพและวิดีโอเพื่อระบุวัตถุ ใบหน้า หรือแม้แต่ลายมือของมนุษย์ได้

เมื่อรวมเข้ากับไลบรารีต่างๆ เช่น NumPy แล้ว Python จะสามารถประมวลผลโครงสร้างอาร์เรย์ของ Opencv เพื่อวิเคราะห์ได้ เพื่อระบุรูปแบบภาพและคุณลักษณะต่างๆ ของภาพ เราใช้เวกเตอร์สเปซและดำเนินการทางคณิตศาสตร์กับคุณลักษณะเหล่านี้

การใช้งาน OpenCV

คุณสมบัติที่สำคัญของ OpenCV-Python

การประมวลผลภาพพื้นฐาน:

- อ่าน/เขียนรูปภาพและวิดีโอ (cv2.imread, cv2.imwrite, cv2.VideoCapture)
- การแปลงสี เช่น จาก RGB เป็น Grayscale หรือ HSV (cv2.cvtColor)
- การครอป (Crop), ย่อขยาย (Resize), และการหมุนภาพ (Rotation)

การตรวจจับและการรู้จำวัตถุ:

- การตรวจจับขอบ (Edge Detection) เช่น Canny Edge Detector
- การตรวจจับเส้น (Hough Line Detection)
- การตรวจจับใบหน้า (Face Detection) ด้วย Haar Cascade หรือ DNN models

ฟิลเตอร์และการประมวลผลเชิงพื้นที่:

- การปรับความคมชัด (Sharpening) และเบลอ (Blurring)
- การทำ Morphological Transformations เช่น Erosion, Dilation

การทำงานกับวิดีโอและการติดตามวัตถุ:

- การอ่านและเขียนวิดีโอ

- การติดตามวัตถุด้วยวิธี Optical Flow หรือ Tracking Algorithms

การรู้จำรูปแบบ (Pattern Recognition):

- การจดจำใบหน้า, ป้ายทะเบียนรถ
- การจับคู่รูปภาพ (Template Matching)

การประยุกต์ใช้ Deep Learning:

- รองรับการใช้งานโมเดลที่เทรนจาก TensorFlow, PyTorch, หรือ Caffe ผ่านโมดูล cv2.dnn

ในโครงการนี้ OpenCV ถูกใช้อย่างครอบคลุมทั้งในการตรวจจับพื้นที่น้ำ (Water Mask Generation), การวางฟีเจอร์บนพื้นหลัง, การรวมภาพ และการหมุนภาพฟีเจอร์ด้วย Rotation Matrix เพื่อสร้างภาพจำลองให้มีความสมจริงมากขึ้น

2.2.3 TensorFlow

TensorFlow เป็นกรอบงานสำหรับกำหนดและเรียกใช้การคำนวณที่เกี่ยวข้องกับเทนเซอร์ตามชื่อ เทนเซอร์เป็นการสรุปเวกเตอร์และเมทริกซ์ไปยังมิติที่สูงกว่า โดยภายใน TensorFlow แสดงเทนเซอร์เป็นอาร์เรย์ n มิติของประเภทข้อมูลพื้นฐาน แต่ละองค์ประกอบใน Tensor มีประเภทข้อมูลเดียวกัน และประเภทข้อมูลจะเป็นที่ทราบเสมอ รูปร่าง (นั่นคือ จำนวนมิติที่มีและขนาดของแต่ละมิติ) อาจเป็นที่รู้จักเพียงบางส่วนเท่านั้น การดำเนินการส่วนใหญ่สร้างเทนเซอร์ที่มีรูปร่างที่ทราบทั้งหมดหากรูปร่างของอินพุตเป็นที่รู้จักอย่างสมบูรณ์เช่นกัน แต่ในบางกรณี จะสามารถค้นหารูปร่างของเทนเซอร์ได้เฉพาะในเวลาที่ดำเนินการกราฟเท่านั้น

คุณลักษณะของ TensorFlow

- การแยกความแตกต่างโดยอัตโนมัติ (AutoDifferentiation) TensorFlow มีฟีเจอร์การแยกความแตกต่างโดยอัตโนมัติ ซึ่งช่วยคำนวณเวกเตอร์การไล่ระดับ (gradient) ของโมเดลได้อย่างมีประสิทธิภาพ โดยเฉพาะในอัลกอริทึมอย่างแบ็กโพรพาเกชัน (Backpropagation) ที่ใช้ในการฝึกโมเดล เฟรมเวิร์กจะติดตามลำดับการดำเนินการของเทนเซอร์อินพุตและคำนวณการไล่ระดับตามพารามิเตอร์ที่เหมาะสม
- การดำเนินการอย่างกระตือรือร้น (Eager Execution) TensorFlow รองรับโหมดการดำเนินการอย่างกระตือรือร้น ซึ่งทำให้การดำเนินการสามารถประเมินผลได้ทันที โดยไม่ต้องสร้างกราฟการคำนวณล่วงหน้า การทำงานในโหมดนี้ช่วยให้โค้ดอ่านง่ายขึ้นและแก้ไขข้อผิดพลาดได้ง่ายขึ้น เนื่องจากสามารถตรวจสอบผลลัพธ์ทีละขั้นตอนผ่านดีบักเกอร์ได้
- การกระจายการคำนวณ (Distribution) TensorFlow สนับสนุนการคำนวณแบบกระจายบนหลายอุปกรณ์ เช่น CPU, GPU หรือ TPU ผ่าน API และกลยุทธ์การกระจายที่หลากหลาย ซึ่งช่วยเพิ่มประสิทธิภาพในการฝึกอบรมและประเมินโมเดล โดยเฉพาะสำหรับโมเดลขนาดใหญ่

- ฟังก์ชันการสูญเสีย (Loss Functions) TensorFlow มีฟังก์ชันการสูญเสียหลากหลาย เช่น ข้อผิดพลาดกำลังสองเฉลี่ย (MSE) และเอนโทรปีแบบไบนารีครอส (BCE) เพื่อใช้ประเมินผลลัพธ์ของโมเดลและช่วยในการปรับปรุงโมเดล
- เมตริก (Metrics) สำหรับการประเมินประสิทธิภาพของโมเดล TensorFlow มีเมตริกที่ใช้กันทั่วไป เช่น ความแม่นยำ (Accuracy), การเรียกคืน (Recall), ความแม่นยำ (Precision) และ Intersection-over-Union (IoU)
- TF.nn TensorFlow.nn เป็นโมดูลสำหรับดำเนินการเครือข่ายประสาทเทียม เช่น การคอนโวลูชัน (Convolution), ฟังก์ชันการเปิดใช้งาน (Activation Functions) เช่น ReLU, Softmax, Sigmoid และการดำเนินการอื่น ๆ เช่น Max-Pooling และ Bias-Addition
- ตัวเพิ่มประสิทธิภาพ (Optimizers) TensorFlow มีตัวเพิ่มประสิทธิภาพที่หลากหลาย เช่น Adam, Adagrad, และ Stochastic Gradient Descent (SGD) ซึ่งช่วยปรับค่าพารามิเตอร์ของโมเดลให้เหมาะสมเพื่อปรับปรุงการเรียนรู้และประสิทธิภาพของโมเดล

ขั้นตอนการใช้งาน TensorFlow

1. นำเข้าหรือสร้างข้อมูล ข้อมูลเป็นปัจจัยสำคัญสำหรับการเรียนรู้ของเครื่อง TensorFlow มีชุดข้อมูลที่พร้อมใช้งาน เช่น MNIST และ CIFAR-10 ซึ่งช่วยให้เริ่มต้นได้ง่ายขึ้น
2. แปลงและปรับมาตรฐานข้อมูล ก่อนใช้งาน ข้อมูลต้องถูกแปลงให้อยู่ในรูปแบบและประเภทที่อัลกอริทึม TensorFlow คาดหวัง และปรับมาตรฐานข้อมูล (Normalization) เพื่อให้เหมาะสมกับการฝึกโมเดล
3. ตั้งค่าพารามิเตอร์ของอัลกอริทึม กำหนดค่าคงที่ เช่น จำนวนการวนซ้ำ (Epochs), อัตราการเรียนรู้ (Learning Rate) และพารามิเตอร์อื่น ๆ ที่จำเป็น
4. กำหนดค่าตัวแปรและตัวแทน (Placeholders) TensorFlow ต้องการให้กำหนดค่าตัวแปร (Variables) และตัวแทน (Placeholders) เพื่อระบุว่าส่วนใดของข้อมูลสามารถปรับเปลี่ยนได้ระหว่างการฝึก
5. กำหนดโครงสร้างของโมเดล สร้างกราฟการคำนวณโดยกำหนดลำดับการดำเนินการและการเปลี่ยนแปลงของข้อมูลในโมเดล
6. ประกาศฟังก์ชันการสูญเสีย กำหนดฟังก์ชันการสูญเสียเพื่อตรวจสอบความคลาดเคลื่อนระหว่างค่าที่คาดการณ์และค่าจริง
7. เริ่มต้นและฝึกโมเดล เริ่มต้นกราฟการคำนวณ และป้อนข้อมูลเข้าสู่โมเดลเพื่อปรับค่าพารามิเตอร์และลดค่าฟังก์ชันการสูญเสีย
8. ประเมินโมเดล (Optional) ตรวจสอบประสิทธิภาพของโมเดลโดยใช้ข้อมูลใหม่และเมตริกที่กำหนดไว้
9. ทำนายผลลัพธ์ใหม่ (Optional) ใช้โมเดลที่ฝึกแล้วเพื่อทำนายผลลัพธ์สำหรับข้อมูลที่ไม่เคยเห็นมาก่อน

โครงการนี้ใช้ TensorFlow ผ่าน Roboflow ในขั้นตอนการฝึกโมเดล YOLOv8 โดย Roboflow รองรับการส่งออกโมเดลในรูปแบบที่สามารถนำไปใช้งานต่อกับ TensorFlow ได้

2.3 งานวิจัยหรือระบบงานใกล้เคียง

ในปัจจุบันมีงานวิจัยและซอฟต์แวร์ที่มุ่งเน้นการสร้างและฝึกอบรมโมเดลตรวจจับวัตถุที่มีประสิทธิภาพสูง แม้ว่างานส่วนใหญ่จะเน้นการใช้ชุดข้อมูลขนาดใหญ่ แต่งานวิจัยบางชิ้นยังมุ่งเน้นไปที่การพัฒนาโมเดลสำหรับชุดข้อมูลขนาดเล็ก เพื่อลดความจำเป็นในการใช้ทรัพยากรที่สูงในขั้นตอนการฝึกอบรม โดยระบบงานใกล้เคียงที่มีแนวคิดใกล้เคียงกับโครงการนี้ ได้แก่ YOLO (You Only Look Once), การพัฒนาการตรวจจับรอยเปื้อนบนเสื้อผ้า (Development of Stain Detection on Clothes), และ พฤติกรรมการกินถุงพลาสติกของเต่าทะเลและผลกระทบต่อสุขภาพ: กรณีศึกษามลภาวะพลาสติกในทะเลไทย (The Feeding Behavior of Sea Turtles on Plastic Bags and Its Health Impacts: A Case Study of Marine Plastic Pollution in Thailand) ซึ่งเป็นตัวอย่างที่มีความสามารถสูงในการตรวจจับวัตถุ โดยมีรายละเอียดในข้อ 2.1, 2.2, และ 2.3 และการเปรียบเทียบระบบต่างๆ ไว้ในข้อ 2.4

2.3.1 YOLO (You Only Look Once)

YOLO เป็นโมเดลที่สามารถตรวจจับวัตถุได้แบบเรียลไทม์โดยใช้การเรียนรู้เชิงลึก (Deep Learning) การตรวจจับภาพด้วย YOLO ทำได้ในขั้นตอนเดียว โดยแบ่งภาพเป็นกริดแล้วคำนวณกรอบและคลาสของวัตถุ YOLO มีข้อได้เปรียบด้านความเร็วสูง แต่ยังต้องการชุดข้อมูลขนาดใหญ่ในการฝึกอบรมให้ได้ผลลัพธ์ที่แม่นยำ

(https://pjreddie.com/darknet/yolo/)

จุดเด่น

- สามารถตรวจจับวัตถุได้อย่างรวดเร็วและแม่นยำ
- สามารถตรวจจับวัตถุหลายชนิดในภาพเดียวได้
- เหมาะกับการใช้งานแบบเรียลไทม์

จุดด้อย

- ต้องใช้ชุดข้อมูลขนาดใหญ่ในการฝึกอบรม
- อาจมีปัญหาในการตรวจจับวัตถุที่มีขนาดเล็กหรืออยู่ใกล้กันมาก

2.3.2 การพัฒนาการตรวจจับรอยเปื้อนบนเสื้อผ้า (Development of Stain Detection on Clothes)

การเสริมข้อมูล (Data Augmentation) เป็นเทคนิคสำคัญที่ใช้เพิ่มประสิทธิภาพของโมเดลตรวจจับรอยเปื้อนบนเสื้อผ้า โดยการปรับเปลี่ยนข้อมูลภาพที่มีอยู่ เช่น การหมุนภาพ ปรับขนาด เปลี่ยนสี หรือเพิ่มสัญญาณรบกวน เพื่อเพิ่มจำนวนและความหลากหลายของชุดข้อมูล
(มหาวิทยาลัยวลัยลักษณ์)

จุดเด่น

- เพิ่มปริมาณข้อมูล: ช่วยเพิ่มจำนวนภาพรอยเปื้อนบนเสื้อผ้า ทำให้โมเดลเรียนรู้ได้ดีขึ้น
- เพิ่มความหลากหลาย: สร้างความหลากหลายของรอยเปื้อน เช่น ขนาด รูปร่าง และตำแหน่ง เพื่อให้โมเดลมีความยืดหยุ่นในการตรวจจับ
- ลดการเกิด Overfitting: ช่วยให้โมเดลไม่จดจำเฉพาะรูปแบบของข้อมูลที่มีอยู่ แต่สามารถทั่วไปกับข้อมูลใหม่ได้

จุดด้อย

- ความสมจริงของข้อมูล: การเสริมข้อมูลอาจไม่สามารถสร้างรอยเปื้อนที่มีความสมจริงเหมือนในชีวิตจริงได้ทั้งหมด
- ความซับซ้อนในการสร้างข้อมูล: การสร้างรอยเปื้อนที่เหมือนจริงต้องใช้ความรู้และเทคนิคที่ซับซ้อน
- ความเสี่ยงในการสร้างข้อมูลที่ไม่เหมาะสม: หากการเสริมข้อมูลไม่ถูกต้อง อาจทำให้โมเดลเรียนรู้ข้อมูลที่ไม่ถูกต้องและลดประสิทธิภาพ

2.3.3 พฤติกรรมการกินถุงพลาสติกของเต่าทะเลและผลกระทบต่อสุขภาพ: กรณีศึกษามลภาวะพลาสติกในทะเลไทย (The Feeding Behavior of Sea Turtles on Plastic Bags and Its Health Impacts: A Case Study of Marine Plastic Pollution in Thailand)

การพัฒนาชุดข้อมูลเทียม (Synthesize Dataset) สำหรับศึกษาพฤติกรรมของเต่าทะเลในการกินถุงพลาสติก มีบทบาทสำคัญในการสร้างข้อมูลที่หลากหลายและเหมือนจริง โดยเฉพาะเมื่อการเก็บข้อมูลภาคสนามมีข้อจำกัด ชุดข้อมูลนี้สามารถใช้ฝึกโมเดลการเรียนรู้เชิงลึก (Deep Learning) เพื่อตรวจจับหรือทำนายพฤติกรรมของเต่ารวมถึงการคาดการณ์ผลกระทบจากพลาสติกในระบบนิเวศ (https://accstr.ufl.edu/wpcontent/uploads/sites/98/Pham_et_al_MarPolBull_2017.pdf)

จุดเด่น

- ลดข้อจำกัดของข้อมูลภาคสนาม
- เพิ่มความหลากหลายของข้อมูล
- สนับสนุนการพัฒนาระบบอัตโนมัติ

- เสริมการทดลองในหลากหลายบริบท
- ลดผลกระทบต่อสัตว์จริง

จุดด้อย

- ขาดความสมจริงในข้อมูล
- การสร้างข้อมูลที่มีคุณภาพต้องใช้ทรัพยากรสูง
- ความเสี่ยงของการแปลผลที่ไม่ถูกต้อง
- ข้อจำกัดของการประยุกต์ใช้

2.4 **เปรียบเทียบระบบงานใกล้เคียงกับระบบที่พัฒนา**

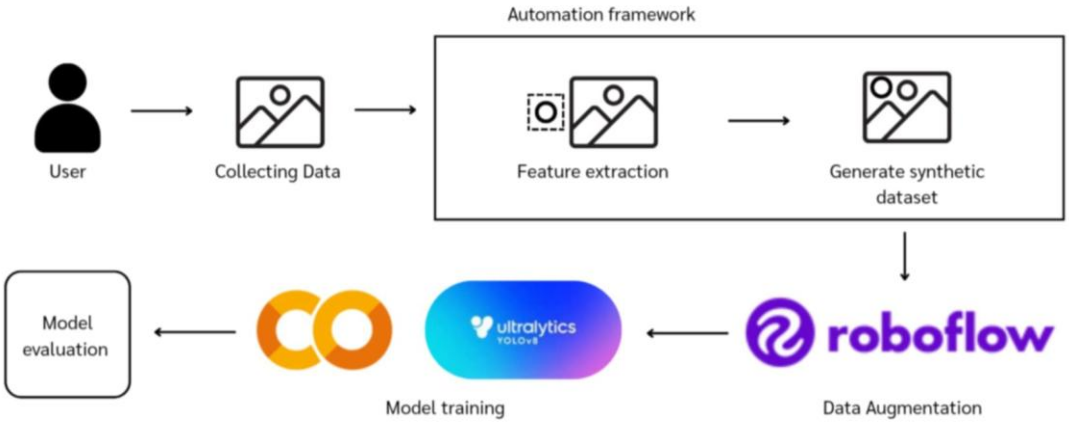
- ระบบงานใกล้เคียงยังไม่สามารถใช้ประโยชน์จากชุดข้อมูลขนาดเล็กได้อย่างมีประสิทธิภาพ เนื่องจากระบบส่วนใหญ่ต้องการชุดข้อมูลขนาดใหญ่ในการฝึกอบรม
- ระบบงานใกล้เคียงไม่สามารถสร้างขั้นตอนการฝึกอบรมที่เป็นอัตโนมัติเต็มรูปแบบ ซึ่งโครงการนี้พยายามลดขั้นตอนที่ซับซ้อนและเพิ่มความสะดวกในการสร้างโมเดล
- ระบบงานใกล้เคียงไม่มีเครื่องมือที่เน้นเฉพาะการทำงานกับชุดข้อมูลขนาดเล็ก ซึ่งโครงการนี้จะช่วยพัฒนาโมเดลที่สามารถฝึกอบรมได้บนชุดข้อมูลที่จำกัด

บทที่ 3

การออกแบบระบบ

3.1 สถาปัตยกรรมของระบบ (System architecture)

ในการศึกษาครั้งนี้ ผู้พัฒนาได้ออกแบบขอบเขตขั้นตอนการดำเนินงานไว้ดังต่อไปนี้ โดยแสดงตามดังรูปแผนผังสถาปัตยกรรมระบบ



ภาพที่ 2 แผนผังสถาปัตยกรรมระบบ

ซึ่งสามารถแบ่งและอธิบายรายละเอียดขั้นตอนการทำงานได้ ดังนี้

1) **การเก็บรวบรวมข้อมูล**

ทำการเก็บรวบรวมข้อมูลจากการถ่ายภาพแหล่งน้ำที่มีขยะในบริเวณรอบมหาวิทยาลัยวลัยลักษณ์ เช่น ขวดพลาสติก และภาพพื้นหลังที่ไม่มีขวดพลาสติก รวมถึงการนำข้อมูลภาพจากแหล่งข้อมูลออนไลน์ที่เกี่ยวข้อง เช่น Roboflow เพื่อใช้เป็นชุดข้อมูลเริ่มต้นสำหรับการพัฒนาโมเดลตรวจจับ

2) **การสร้างและจัดเตรียมชุดข้อมูล (Synthetic Dataset)**

นำข้อมูลภาพพื้นหลังและฟีเจอร์ (ขวดพลาสติก) มาสร้างชุดข้อมูลภาพจำลอง (Synthetic Dataset) พร้อม Annotation โดยใช้เทคนิคการวางฟีเจอร์บนภาพพื้นหลังแบบสุ่ม และปรับปรุงคุณภาพข้อมูลด้วยการสร้าง Mask น้ำ เพื่อกำหนดพื้นที่น้ำในภาพให้เหมาะสม

3) **การจัดการข้อมูลและอัปโหลดเข้า Roboflow**

นำชุดข้อมูลที่สร้างขึ้นไปอัปโหลดเข้าสู่แพลตฟอร์ม Roboflow เพื่อทำการจัดการและปรับปรุง Annotation และใช้ฟีเจอร์ Data Augmentation เพื่อเพิ่มความหลากหลายของชุดข้อมูล ชุดข้อมูลจะถูกส่งออกในฟอร์แมตที่เหมาะสมสำหรับการฝึกโมเดล YOLO

4) **การฝึกฝนแบบจำลอง YOLO**

นำชุดข้อมูลจาก Roboflow ไปฝึกโมเดลตรวจจับขวดพลาสติกโดยใช้ Ultralytics YOLOv8 บนแพลตฟอร์ม Google Colab เพื่อใช้ทรัพยากรประมวลผลจากคลาวด์ ในขั้นตอนนี้จะทำการประเมินและปรับปรุงโมเดลด้วยเมตริก เช่น Accuracy, Precision และ Recall เพื่อหาโมเดลที่มีประสิทธิภาพสูงสุด

5) **การพัฒนาเฟรมเวิร์คสำหรับเตรียมชุดข้อมูลฝึกโมเดลแบบอัตโนมัติ**

เมื่อได้โมเดลที่มีคุณภาพสูงสุด ระบบจะถูกพัฒนาเป็นเฟรมเวิร์ค AI โดยใช้ Visual Studio Code เพื่อออกแบบโครงสร้างการทำงานและการใช้งาน เฟรมเวิร์คดังกล่าวจะสามารถสร้างชุดข้อมูลจำลองสำหรับการเทรนโมเดลคุณภาพสูงให้โดยอัตโนมัติตามค่าที่ตั้งไว้

ระบบที่พัฒนาขึ้นมีโครงสร้างเป็นเฟรมเวิร์คแบบอัตโนมัติ ประกอบด้วย 3 โมดูลหลัก ได้แก่:

1. โมดูลเปลี่ยนชื่อไฟล์ (Rename) – สำหรับจัดระเบียบไฟล์ภาพให้เรียงตามลำดับเพื่อความสะดวกในการใช้งานในขั้นตอนต่อไป

2. โมดูลแยกฟีเจอร์ (Extract) – สำหรับลบพื้นหลังภาพขวดพลาสติกและทำ post-process ให้พร้อมนำไปใช้

3. โมดูลสร้างข้อมูลจำลอง (Generate) – สำหรับสุ่มภาพพื้นหลัง + ฟีเจอร์ แล้วสร้าง synthetic image พร้อม annotation แบบ YOLO

ทั้ง 3 โมดูลถูกรวมอยู่ใน `main.py` ซึ่งสามารถรันงานทั้งหมดแบบอัตโนมัติจากคำสั่งเดียว ช่วยลดเวลาและความซับซ้อนในการสร้างชุดข้อมูลจำลอง และทำให้กระบวนการสามารถทำซ้ำได้อย่างมีประสิทธิภาพ

3.2 การเก็บรวบรวมข้อมูล

การเก็บรวบรวมข้อมูลในโครงการนี้มุ่งเน้นการเตรียมข้อมูลที่เหมาะสมสำหรับการพัฒนาโมเดลตรวจจับขวดพลาสติกบนพื้นน้ำ โดยมีการวางแผนขั้นตอนการดำเนินงานอย่างเป็นระบบ เพื่อให้ได้ข้อมูลที่มีคุณภาพสูงเพียงพอและครอบคลุมความหลากหลายของสถานการณ์จริงที่อาจเกิดขึ้นในแหล่งน้ำธรรมชาติ

3.2.1 การถ่ายภาพฟีเจอร์ (ขวดพลาสติก)

กระบวนการถ่ายภาพฟีเจอร์เริ่มจากการรวบรวมขวดพลาสติกหลากหลายประเภท เช่น ขวดใส ขวดที่มีฉลากสีต่าง ๆ และขวดที่มีรูปทรงหลากหลาย เพื่อให้ข้อมูลที่ได้ครอบคลุมสถานการณ์ที่พบได้ในแหล่งน้ำจริง ภาพถ่ายจะถูกถ่ายบนพื้นน้ำเพื่อช่วยให้การแยกฟีเจอร์ออกจากพื้นหลังในขั้นตอนถัดไปทำได้ง่ายและสมจริง นอกจากนี้ ขวดพลาสติกแต่ละใบยังถูกถ่ายในมุมมองที่หลากหลาย ได้แก่ ด้านหน้า ด้านซ้าย ด้านขวา และด้านหลัง เพื่อเพิ่มความหลากหลายของชุดข้อมูลที่ใช้ในการฝึกโมเดล

ภาพที่ 3 ตัวอย่างภาพถ่ายขวดพลาสติกบนพื้นน้ำ

3.2.2 การจัดการพื้นหลังของฟีเจอร์ (Feature Extraction)

หลังจากถ่ายภาพขวดพลาสติก ระบบจะทำการลบพื้นหลังออกโดยใช้สคริปต์ Python ที่พัฒนาขึ้นโดยใช้ไลบรารี OpenCV กระบวนการนี้ประกอบด้วยการใช้ค่าช่วงสี (HSV Range) เพื่อแยกพื้นหลังออกจากฟีเจอร์ และบันทึกฟีเจอร์ที่ได้ในรูปแบบไฟล์ PNG พร้อมพื้นหลังโปร่งใส สำหรับปรับปรุงคุณภาพของภาพจะมีการใช้ตัวกรอง Gaussian Blur เพื่อลบขอบที่ไม่สมบูรณ์ และ Sharpen Filter เพื่อเพิ่มความคมชัดของฟีเจอร์ก่อนบันทึก โดยเราจะทำการเลือกภาพจาก Dataset จริงที่ถ่ายมาบางส่วน เพื่อเป็นต้นแบบสำหรับฟีเจอร์ขวดพลาสติก ซึ่งเราจะเลือกภาพที่เห็นขวดชัดและเลือกหลายขวดที่ไม่ซ้ำกันเพื่อความหลากหลายของฟีเจอร์

ภาพที่ 4 ตัวอย่างภาพฟีเจอร์ในรูปแบบไฟล์ PNG ที่มีพื้นหลังโปร่งใส

3.2.3 การเก็บรวบรวมภาพพื้นหลัง

ภาพพื้นหลังสำหรับโครงการนี้ถูกถ่ายจากแหล่งน้ำจริง เช่น แม่น้ำ ลำคลอง และบ่อน้ำ ในบริเวณมหาวิทยาลัยวลัยลักษณ์ โดยเลือกภาพที่ไม่มีขวดพลาสติกหรือวัตถุอื่นในเฟรม เพื่อใช้เป็นพื้นฐานในการสร้างชุดข้อมูลจำลอง

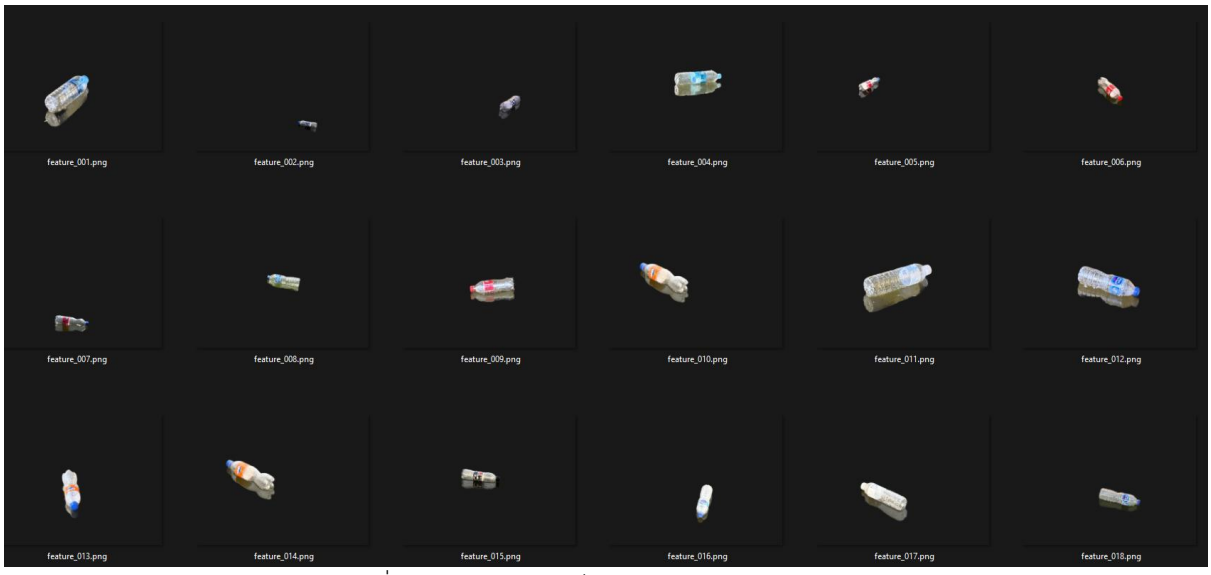
ภาพที่ 5 ตัวอย่างภาพพื้นหลังจากแหล่งน้ำบริเวณมหาวิทยาลัยวลัยลักษณ์

3.2.4 การจัดเก็บข้อมูล

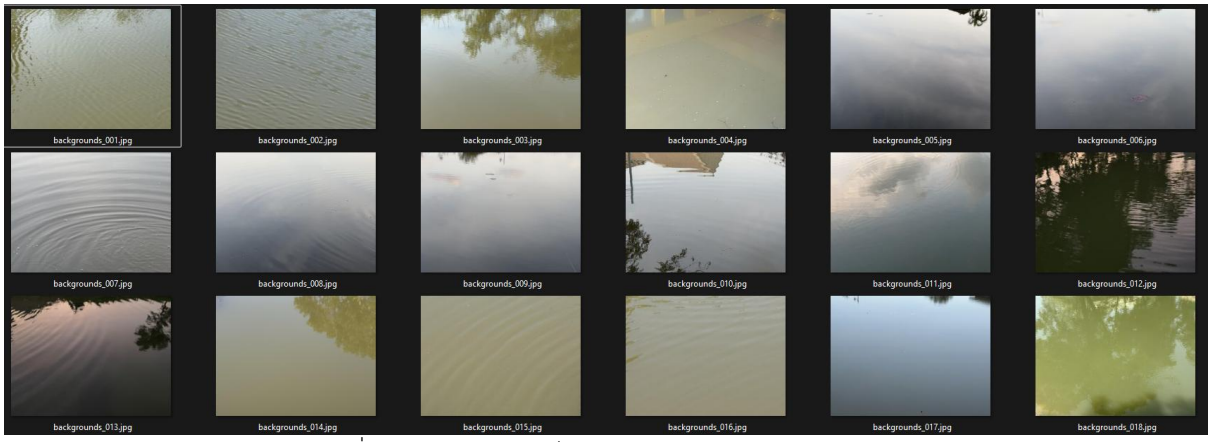
ภาพฟีเจอร์และพื้นหลังที่ได้จะถูกจัดเก็บอย่างเป็นระบบในโฟลเดอร์เฉพาะ ได้แก่:

- โฟลเดอร์ features/: สำหรับเก็บภาพขวดพลาสติกพร้อมพื้นหลังโปร่งใส
- โฟลเดอร์ backgrounds/: สำหรับเก็บภาพพื้นหลังจากแหล่งน้ำ

หลังจากถ่ายภาพเสร็จแล้ว ระบบเฟรมเวิร์คจะทำการเปลี่ยนชื่อไฟล์ให้อยู่ในรูปแบบที่เป็นมาตรฐาน เช่น `raw_image_001.jpg` และ `backgrounds_001.jpg` โดยอัตโนมัติ เพื่อเตรียมความพร้อมในการประมวลผลขั้นต่อไป



ภาพที่ 6 ตัวอย่างการจัดเก็บข้อมูลในโฟลเดอร์ features/



ภาพที่ 7 ตัวอย่างการจัดเก็บข้อมูลในโฟลเดอร์ backgrounds/

3.2.5 การตรวจสอบคุณภาพข้อมูล

ข้อมูลทั้งหมดจะถูกตรวจสอบคุณภาพก่อนนำไปใช้ เพื่อให้มั่นใจว่า ไม่มีสิ่งรบกวน เช่น เงาสะท้อน หรือการแยกพื้นหลังที่ผิดพลาด ภาพฟีเจอร์มีความคมชัดและรายละเอียดเพียงพอสำหรับกระบวนการสร้างชุดข้อมูลจำลอง ข้อมูลที่ได้จากกระบวนการนี้จะถูกนำไปใช้ในขั้นตอนการสร้างชุดข้อมูลภาพจำลอง (Synthetic Dataset) เพื่อสนับสนุนการพัฒนาโมเดลตรวจจับขวดพลาสติกในขั้นตอนถัดไป

ระบบจะตรวจสอบคุณภาพภาพขณะประมวลผล หากพบว่าภาพไม่มีพื้นที่น้ำที่เหมาะสมหรือไม่สามารถวางฟีเจอร์ได้ จะข้ามการสร้างภาพนั้นและบันทึกภาพที่ล้มเหลวไว้เพื่อการวิเคราะห์ภายหลัง

3.3 การสร้างชุดข้อมูลภาพจำลอง (Synthetic Dataset)

การสร้างชุดข้อมูลภาพจำลองในโครงการนี้ใช้กระบวนการเชิงโปรแกรมที่ออกแบบมาเพื่อเพิ่มปริมาณและความหลากหลายของข้อมูล ชุดข้อมูลที่สร้างขึ้นจะช่วยสนับสนุนการพัฒนาโมเดลตรวจจับขวดพลาสติกบนพื้นที่น้ำ โดยการวางฟีเจอร์ (ขวดพลาสติก) บนภาพพื้นหลังด้วยวิธีการสุ่มและการสร้าง Annotation อัตโนมัติ ซึ่งสามารถอธิบายรายละเอียดได้ดังนี้:

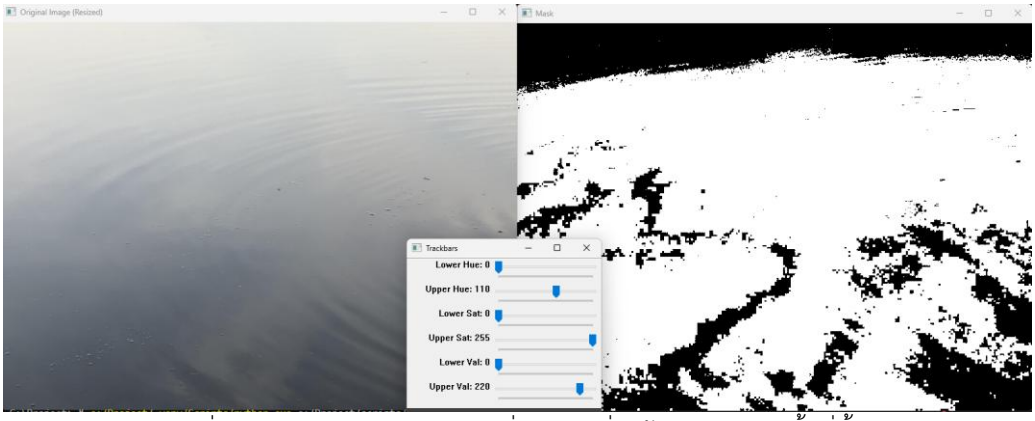
3.3.1 การนำฟีเจอร์และภาพพื้นหลังมาใช้งาน

ฟีเจอร์ที่แยกพื้นหลังแล้ว (ขวดพลาสติก) จะถูกนำมาจากโฟลเดอร์ features/ ในขณะที่ภาพพื้นหลังจะนำมาจากโฟลเดอร์ backgrounds/ สำหรับแต่ละรอบของการสร้างภาพจำลอง ฟีเจอร์และภาพพื้นหลังจะถูกสุ่มเลือกเพื่อสร้างความหลากหลายของชุดข้อมูล ช่วยให้ข้อมูลมีความครอบคลุมในลักษณะที่ใกล้เคียงกับสถานการณ์จริงในแหล่งน้ำ

ระบบจะสุ่มเลือกฟีเจอร์และพื้นหลังในแต่ละรอบโดยอัตโนมัติ และใช้โค้ด Python ในการวางฟีเจอร์ลงบนตำแหน่งที่มีน้ำอย่างเหมาะสม โดยอิงจากการตรวจจับ Mask น้ำที่ได้จาก OpenCV

3.3.2 การตรวจจับพื้นที่น้ำ (Water Mask Generation)

พื้นที่น้ำในภาพพื้นหลังถูกตรวจจับโดยใช้ฟังก์ชันที่พัฒนาด้วย OpenCV การตรวจจับเริ่มจากการแปลงภาพพื้นหลังเป็นโหมดสี HSV และใช้ค่าช่วงสี (HSV Range) ที่กำหนด เพื่อสร้าง Mask น้ำ ตัวกรอง Morphological เช่น Closing และ Opening ถูกใช้เพื่อลด Noise และปรับปรุงความแม่นยำของ Mask ซึ่ง Mask ที่ได้จะถูกใช้เพื่อกำหนดพื้นที่น้ำที่เหมาะสมสำหรับการวางฟีเจอร์



ภาพที่ 8 ตัวอย่างการค่าช่วงสี HSV ที่กำหนดเพื่อสร้าง Mask ของพื้นที่น้ำ

จากภาพที่ 8 ซึ่งพื้นที่สีขาวในรูปด้านขวา (Mask) คือพื้นที่ที่เราสามารถวางฟีเจอร์เข้าไปได้ สาเหตุที่ต้องสร้างเป็น mask น้ำเนื่องจากในภาพแบกกราวด์อาจมีสิ่งของอื่นที่ไม่ใช่พื้นน้ำปะปนมา จึงต้องทำการกรองเอาเฉพาะส่วนนี้เป็นน้ำอย่างชัดเจน เพื่อป้องกันไม่ให้เกิดการวางฟีเจอร์ทับบนสิ่งของอื่น เช่น เรือ หรือ พืชต่างๆบนน้ำ

3.3.3 การวางฟีเจอร์บนพื้นที่น้ำ

การวางฟีเจอร์ใช้การสุ่มขนาด (Scale), ตำแหน่ง (x, y), และมุมหมุน (Angle) ด้วยฟังก์ชันของ NumPy และ OpenCV จากนั้นจะตรวจสอบให้ฟีเจอร์ไม่ล้นกรอบภาพ และอย่างน้อย 50% อยู่บนพื้นที่น้ำที่ตรวจจับได้

3.3.4 การสร้างภาพจำลอง

การรวมฟีเจอร์เข้ากับภาพพื้นหลัง ซึ่งจะหมุนฟีเจอร์แบบสุ่ม (0° ถึง 360°) ด้วย Rotation Matrix จากนั้นใช้เทคนิค Alpha Blending เพื่อผสมฟีเจอร์กับภาพพื้นหลังให้ดูสมจริง

ภาพที่ 9 ตัวอย่างการวางฟีเจอร์บนพื้นที่น้ำโดยสุ่มขนาด พิกัด และมุมหมุน

ชุดข้อมูลจำลอง หรือ Synthetic Dataset ที่ได้ก็จะมีความสมจริงประมาณหนึ่ง แต่ขาดความสมจริงเรื่องแสงเราไป ซึ่งไม่ส่งผลกระทบต่อการเทรนโมเดล เนื่องจากจะมีการทำ Data Augmentation เพิ่มเติมในภายหลัง จะมีการหมุนภาพอยู่ในนั้นอยู่แล้วทำให้ไม่เกิดปัญหาต่อโมเดลที่ได้

3.3.5 การสร้าง Annotation

ระบบจะคำนวณ Bounding Box จริงจาก alpha channel ของฟีเจอร์ แล้วแปลงเป็นค่า Normalize (x_center, y_center, width, height) ตามฟอร์แมต YOLO โดยอัตโนมัติ และบันทึกเป็น `.txt` ให้พร้อมใช้งานทันทีใน Roboflow

Annotation ที่สร้างขึ้นจะถูกจัดเก็บในโฟลเดอร์ annotations/ โดยมีชื่อไฟล์ตรงกับชื่อภาพ เช่น synthetic_image_1.txt

3.3.6 การจัดเก็บข้อมูล

ภาพที่สร้างสำเร็จจะถูกจัดเก็บในโฟลเดอร์ synthetic_dataset/ โดยตั้งชื่อไฟล์ในรูปแบบ synthetic_image_1.jpg หากการวางฟีเจอร์ไม่สำเร็จ เช่น ไม่มีพื้นที่น้ำเพียงพอ ระบบจะบันทึกภาพพื้นหลังและ Mask น้ำในโฟลเดอร์สำหรับการตรวจสอบและการปรับปรุงพารามิเตอร์ในอนาคต

ภาพและ annotation จะถูกจัดเก็บพร้อมกันในโฟลเดอร์ที่กำหนด โดยใช้การตั้งชื่ออัตโนมัติ เช่น `synthetic_image_001.jpg` และ `synthetic_image_001.txt` เพื่อให้ชื่อไฟล์ตรงกันโดยไม่ต้องตั้งเอง

ภาพที่ 10 ตัวอย่างการจัดเก็บข้อมูลในโฟลเดอร์ synthetic_dataset/

3.3.7 การตรวจสอบและประเมินผล

ระบบจะบันทึกข้อมูลภาพที่ล้มเหลวในการวางฟีเจอร์ เพื่อใช้ในการวิเคราะห์และปรับปรุงพารามิเตอร์ เช่น การปรับช่วงค่า HSV หรือการขยายขนาด Mask น้ำ หลังจากระบบสร้างภาพครบตามจำนวนที่กำหนด ระบบจะหยุดทำงาน

ชุดข้อมูลภาพจำลองที่ได้จากขั้นตอนนี้จะมีความหลากหลายและสมจริง พร้อมทั้ง Annotation ในรูปแบบที่เหมาะสมสำหรับการฝึกโมเดล YOLO ในขั้นตอนถัดไป

การทำงานทั้งหมดถูกควบคุมด้วยไฟล์ `main.py` ที่เป็นตัวกลางเรียกใช้ทุกขั้นตอนจากต้นจนจบภายในคำสั่งเดียว

3.4 การอัปโหลดชุดข้อมูลเข้า Roboflow (สำหรับ YOLOv8)

การอัปโหลดชุดข้อมูลเข้าแพลตฟอร์ม Roboflow สำหรับโมเดล YOLOv8 เป็นขั้นตอนสำคัญที่ช่วยปรับปรุงคุณภาพข้อมูลและทำให้ข้อมูลพร้อมสำหรับการฝึกโมเดล Roboflow ช่วยจัดการ Annotation, เพิ่มความหลากหลายของข้อมูลด้วย Data Augmentation และแบ่งข้อมูลออกเป็น Training, Validation และ Test Sets พร้อมทั้งส่งออกในฟอร์แมตที่รองรับ YOLOv8 ได้อย่างมีประสิทธิภาพ

3.4.1 การจัดเตรียมชุดข้อมูล

ชุดข้อมูลที่สร้างขึ้นจากขั้นตอนก่อนหน้าประกอบด้วยไฟล์ภาพและไฟล์ Annotation ที่จับคู่กันอย่างถูกต้อง ชุดข้อมูลภาพถูกจัดเก็บในโฟลเดอร์ synthetic_dataset/ เช่น synthetic_image_001.jpg และไฟล์ Annotation ในโฟลเดอร์ annotations/ เช่น synthetic_image_001.txt ชื่อไฟล์ของภาพและ Annotation ต้องตรงกันเพื่อให้ Roboflow สามารถประมวลผลข้อมูลได้อย่างถูกต้อง

เฟรมเวิร์คที่พัฒนาขึ้นสามารถจัดโครงสร้างของชุดข้อมูลให้ตรงตามมาตรฐาน Roboflow ได้โดยอัตโนมัติ ทำให้สามารถนำเข้าไปใช้งานได้ทันทีโดยไม่ต้องแก้ไขเพิ่มเติม

3.4.2 การสร้างโปรเจกต์ใน Roboflow

เข้าสู่ระบบ Roboflow โดยลงชื่อเข้าใช้บนเว็บไซต์ Roboflow และสร้างโปรเจกต์ใหม่ผ่านหน้าจอ “Create New Project” ตั้งชื่อโปรเจกต์ เช่น YOLOv8 Plastic Bottle Detection และเลือกฟอร์แมตเป็น YOLOv8 PyTorch TXT เพื่อรองรับการใช้งานโมเดล YOLOv8 ในขั้นตอนการฝึก

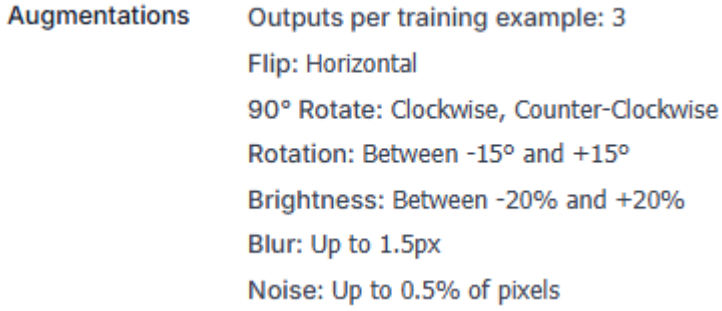
3.4.3 การอัปโหลดชุดข้อมูล

หลังจากสร้างโปรเจกต์เรียบร้อยแล้ว ไฟล์ภาพและไฟล์ Annotation จากโฟลเดอร์ที่เตรียมไว้จะถูกอัปโหลดเข้าสู่ Roboflow โดยเลือกไฟล์ภาพ เช่น synthetic_image_001.jpg และไฟล์ Annotation เช่น synthetic_image_001.txt พร้อมกัน Roboflow จะตรวจสอบความถูกต้องของ

ข้อมูลและแสดงผลภาพพร้อม Annotation ให้ผู้ใช้งานตรวจสอบ นอกจากนี้ ยังมีการระบุ Class สำหรับการตรวจจับ เช่น ขวดพลาสติก เพื่อกำหนดเป้าหมายของโมเดล

3.4.4 การปรับแต่งชุดข้อมูล

Roboflow มีฟีเจอร์สำหรับเพิ่มความหลากหลายของข้อมูล (Data Augmentation) เช่น การหมุนภาพ (Rotation), การปรับความสว่าง (Brightness Adjustment) และการเพิ่ม Noise หรือ Blur เพื่อให้ชุดข้อมูลมีความครอบคลุมและเหมาะสมกับการฝึกโมเดล หลังจากการปรับแต่งชุดข้อมูล ระบบจะแบ่งข้อมูลเป็น Training Set (80%), Validation Set (10%) และ Test Set (10%) โดยอัตโนมัติ เพื่อเตรียมความพร้อมสำหรับการฝึกโมเดลและการประเมินผล



ภาพที่ 11 ค่าต่างๆที่ใช้ในฟีเจอร์ Data Augmentation จาก Roboflow

3.4.5 การส่งออกชุดข้อมูล

เมื่อปรับแต่งชุดข้อมูลเสร็จสิ้นแล้ว Roboflow จะให้ผู้ใช้งานเลือกส่งออกชุดข้อมูลในฟอร์แมต YOLOv8 PyTorch TXT ไฟล์ที่ได้จะอยู่ในรูปแบบ ZIP ซึ่งประกอบด้วยภาพและ Annotation ที่แยกโฟลเดอร์สำหรับ train, valid และ test ไว้เรียบร้อยแล้ว พร้อมใช้งานสำหรับการฝึกโมเดล YOLOv8 ในขั้นตอนถัดไป ผู้ใช้งานสามารถดาวน์โหลดไฟล์ ZIP และนำไปใช้ในระบบฝึกโมเดลบนแพลตฟอร์ม เช่น Google Colab หรือ Local Environment ได้อย่างสะดวก

ชุดข้อมูลที่ผ่านกระบวนการใน Roboflow นี้ ช่วยให้ข้อมูลพร้อมใช้งานและมีความหลากหลาย รองรับการฝึกโมเดล YOLOv8 ได้อย่างมีประสิทธิภาพและแม่นยำในขั้นตอนต่อไป

3.5 การฝึกฝนแบบจำลอง YOLOv8

การฝึกฝนแบบจำลอง YOLOv8 เป็นขั้นตอนสำคัญในการพัฒนาโมเดลที่สามารถตรวจจับขวดพลาสติกบนพื้นที่น้ำได้อย่างมีประสิทธิภาพ โดยใช้ชุดข้อมูลที่ปรับแต่งและจัดการผ่าน Roboflow และดำเนินการฝึกโมเดลบนแพลตฟอร์ม Google Colab ซึ่งมีทรัพยากร GPU สำหรับประมวลผลที่เหมาะสม ช่วยเพิ่มความเร็วและประสิทธิภาพในการฝึกโมเดล

ขั้นตอนการฝึกประกอบด้วยรายละเอียดดังนี้:

3.5.1 การเตรียมสภาพแวดล้อม

กระบวนการฝึกโมเดลเริ่มต้นจากการอัปโหลดชุดข้อมูลจาก Roboflow ไปยังโฟลเดอร์ของโปรเจกต์บน Google Drive จากนั้นดำเนินการฝึกโมเดลผ่าน Google Colab โดยได้มีการติดตั้งไลบรารี Ultralytics YOLOv8 เพื่อใช้เป็นโครงสร้างพื้นฐานในการพัฒนาและฝึกสอนโมเดลสำหรับการตรวจจับวัตถุ

3.5.2 การโหลดชุดข้อมูล

ชุดข้อมูลที่ได้รับจาก Roboflow จะถูกจัดเก็บในรูปแบบไฟล์ YAML เพื่อระบุโครงสร้างของชุดข้อมูลที่ใช้ในการฝึก โดยกำหนดเส้นทางของชุดข้อมูลสำหรับการฝึก (Training Set) และการตรวจสอบความถูกต้อง (Validation Set) พร้อมทั้งระบุจำนวนคลาสของวัตถุที่ต้องการตรวจจับ ซึ่งในกรณีนี้มีเพียง 1 คลาส คือ ขวดพลาสติก

3.5.3 กระบวนการฝึกโมเดล

โมเดล YOLOv8 ทั้ง 5 เวอร์ชัน ได้แก่ yolov8n, yolov8s, yolov8m, yolov8l และ yolov8x ถูกนำมาใช้ในการฝึกเพื่อนำผลลัพธ์มาเปรียบเทียบประสิทธิภาพของแต่ละเวอร์ชัน โดยหลังจากกระบวนการฝึกฝนเสร็จสิ้น ระบบจะให้ผลลัพธ์ที่สำคัญ ได้แก่ ค่า mAP (Mean Average Precision), Precision และ Recall เพื่อใช้ในการประเมินประสิทธิภาพของโมเดลแต่ละเวอร์ชัน

3.5.4 การประเมินผลลัพธ์ของโมเดล

หลังการฝึก โมเดลจะถูกประเมินด้วยชุดข้อมูล Validation เพื่อวิเคราะห์ประสิทธิภาพ โดยใช้ตัวชี้วัดหลัก ได้แก่:

- mAP (Mean Average Precision): วัดความสามารถของโมเดลในการตรวจจับวัตถุ
- Precision: วัดความถูกต้องของการตรวจจับ
- Recall: วัดความครอบคลุมของการตรวจจับ

3.5.5 การปรับปรุงโมเดล (Fine-tuning)

หากผลลัพธ์ของโมเดลยังไม่เป็นที่น่าพอใจ การปรับปรุงโมเดลสามารถทำได้โดยการเพิ่มจำนวน Epoch เพื่อให้โมเดลเรียนรู้มากขึ้น ปรับพารามิเตอร์ เช่น Batch Size หรือใช้ Data Augmentation เพิ่มเติมใน Roboflow เพื่อเพิ่มความหลากหลายของข้อมูล

3.5.6 การบันทึกและส่งออกโมเดล

เมื่อการฝึกเสร็จสิ้น โมเดลที่ผ่านการปรับแต่งจะถูกบันทึกและส่งออกในฟอร์แมตที่เหมาะสม เช่น ONNX หรือ PyTorch เพื่อใช้งานในเฟรมเวิร์คที่พัฒนาขึ้นในขั้นตอนถัดไป โมเดลนี้จะถูกนำไปใช้ในการตรวจจับขวดพลาสติกบนพื้นน้ำในระบบที่ออกแบบไว้

3.6 การพัฒนาเฟรมเวิร์คสำหรับเตรียมชุดข้อมูลฝึกโมเดลแบบอัตโนมัติ

การพัฒนาเฟรมเวิร์คในโครงการนี้มีวัตถุประสงค์เพื่ออำนวยความสะดวกและลดภาระในกระบวนการเตรียมชุดข้อมูลฝึกโมเดลตรวจจับวัตถุ โดยเฉพาะในกรณีที่มีจำนวนข้อมูลจริงจำกัด เฟรมเวิร์คได้รับการออกแบบให้สามารถดำเนินงานได้โดยอัตโนมัติ ตั้งแต่การแยกวัตถุ (ฟีเจอร์) จากภาพจริง การสร้างชุดข้อมูลภาพจำลอง (Synthetic Dataset) ไปจนถึงการจัดเก็บชุดข้อมูลในรูปแบบที่พร้อมนำไปใช้งานใน Roboflow เพื่อการทำ Data Augmentation ต่อไป โดยผู้ใช้งานเพียงแค่เตรียมข้อมูลเบื้องต้นและกำหนดค่าพื้นฐาน จากนั้นระบบจะดำเนินการในทุกขั้นตอนให้โดยอัตโนมัติ

3.6.1 การแยกวัตถุจากภาพจริงโดยอัตโนมัติ

เฟรมเวิร์คจะรับภาพขวดพลาสติกจริงจากโฟลเดอร์ raw_images/ ซึ่งภาพเหล่านี้อาจถูกถ่ายในฉากหลังจริง โดยไม่จำเป็นต้องเป็นฉากเขียวหรือฉากสตูดิโอ จากนั้นระบบจะใช้ไลบรารี rembg เพื่อทำการลบพื้นหลังของภาพโดยอัตโนมัติ ด้วยการเปิดใช้งานโหมด alpha_matting และตั้งค่า alpha_matting_foreground_threshold ไว้ที่ 240 เพื่อให้สามารถแยกวัตถุออกจากพื้นหลังได้อย่างแม่นยำ

หลังจากได้ภาพที่มีพื้นหลังโปร่งใส ระบบจะดำเนินการ Post-processing เพื่อเพิ่มคุณภาพของฟีเจอร์ โดยใช้ Gaussian Blur เพื่อลดขอบภาพที่ไม่เรียบ และการ Sharpen ภาพเพื่อเพิ่มความคมชัด จากนั้นจึงบันทึกภาพในโฟลเดอร์ features/ โดยใช้ชื่อไฟล์ตามลำดับ เช่น feature_001.png จนถึงจำนวนที่ต้องการ

3.6.2 การสร้างชุดข้อมูลภาพจำลอง

หลังจากมีฟีเจอร์พร้อมใช้งาน เฟรมเวิร์คจะทำการสร้างชุดข้อมูลภาพจำลองโดยอัตโนมัติ โดยนำภาพพื้นหลังจากโฟลเดอร์ backgrounds/ ซึ่งเป็นภาพแหล่งน้ำต่าง ๆ มาจับคู่กับฟีเจอร์ในแต่ละรอบอย่างสุ่ม พร้อมทั้งตรวจสอบว่าในแต่ละภาพได้เลือกพื้นหลังและฟีเจอร์ที่แตกต่างกันเพื่อเพิ่มความหลากหลายของข้อมูล

ระบบจะเริ่มต้นด้วยการแปลงภาพพื้นหลังเป็นสีในโหมด HSV และใช้ค่าช่วงสี HSV ที่กำหนดไว้ (LOWER_BOUND = [0, 0, 0] และ UPPER_BOUND = [110, 255, 220]) เพื่อสร้าง Mask ที่ระบุตำแหน่งของพื้นที่น้ำในภาพ หลังจากนั้นระบบจะสุ่มขนาด สุ่มตำแหน่ง และมุมหมุนของฟีเจอร์

โดยตรวจสอบว่าฟีเจอร์ที่วางลงไปอย่างน้อย 50% ของพื้นที่ต้องอยู่ในตำแหน่งที่ตรงกับ Mask น้ำ เพื่อให้ภาพที่ได้ดูสมจริงและสอดคล้องกับวัตถุประสงค์ของโมเดล

เมื่อวัตถุถูกวางลงบนภาพพื้นหลังแล้ว ระบบจะทำการรวมภาพเข้าด้วยกัน และบันทึกเป็นไฟล์ .jpg ในโฟลเดอร์ synthetic_dataset/ พร้อมกับสร้างไฟล์ .txt สำหรับ Annotation โดยใช้ Bounding Box ที่คำนวณจากตำแหน่งจริงของพิกเซลใน alpha channel และแปลงให้เป็นค่าที่ Normalize แล้วตามฟอร์แมตของ YOLOv8

3.6.3 การเตรียมข้อมูลสำหรับนำเข้าสู่ Roboflow

เมื่อชุดข้อมูลจำลองและ Annotation ถูกสร้างเสร็จ ระบบจะจัดเรียงโครงสร้างของโฟลเดอร์ให้อยู่ในรูปแบบที่สอดคล้องกับ Roboflow โดยแบ่งเป็น images/ และ labels/ ภายในโฟลเดอร์ synthetic_dataset/ ซึ่งชื่อของภาพและไฟล์ Annotation จะต้องตรงกันทุกคู่ เช่น synthetic_image_001.jpg และ synthetic_image_001.txt เพื่อความถูกต้องในการอัปโหลด

Roboflow จะถูกนำมาใช้ในขั้นตอนถัดไปเพื่อทำ Data Augmentation ซึ่งช่วยให้ได้ชุดข้อมูลที่มีความหลากหลายมากขึ้น โดยผู้ใช้งานสามารถกำหนดการ Augment เช่น การหมุนภาพ การเพิ่ม Noise หรือปรับแสงได้ตามความเหมาะสม ทั้งนี้แม้กระบวนการ Augment จะดำเนินการภายนอกเฟรมเวิร์ค แต่เฟรมเวิร์คที่พัฒนาขึ้นสามารถสร้างข้อมูลที่พร้อมสำหรับ Roboflow ได้โดยไม่ต้องจัดการเพิ่มเติมใด ๆ

3.6.4 ความสามารถและความยืดหยุ่นของเฟรมเวิร์ค

เฟรมเวิร์คที่พัฒนาขึ้นในโครงการนี้ประกอบด้วย 3 โมดูลหลัก ได้แก่:

1. create_name.py – ใช้เปลี่ยนชื่อภาพพื้นหลังและภาพวัตถุดิบให้อยู่ในรูปแบบที่เป็นมาตรฐาน
2. extract_features.py – สำหรับลบพื้นหลังของภาพขวดพลาสติก และทำ post-processing ด้วยการเบลอและ sharpen
3. generate_synthetic_dataset.py – สำหรับนำภาพพื้นหลังและฟีเจอร์มาวางรวมกัน พร้อมสร้างไฟล์ annotation แบบ YOLO

ทั้งสามโมดูลถูกเรียกใช้งานผ่าน `main.py` ซึ่งรวมกระบวนการทั้งหมดไว้ในคำสั่งเดียว เพื่อให้ผู้ใช้งานสามารถสร้างชุดข้อมูลจำลองแบบสมบูรณ์ได้โดยไม่ต้องเขียนโค้ดเพิ่มเติม

เฟรมเวิร์คนี้มีความสามารถในการปรับแต่งได้สูง ทั้งในด้านการเพิ่มฟีเจอร์ การเปลี่ยนค่าพารามิเตอร์สำหรับ Mask สี หรือแม้แต่การเพิ่มเติมชนิดของฟีเจอร์ใหม่ ๆ ในอนาคต โดยไม่จำเป็นต้องเปลี่ยนโครงสร้างระบบหลัก นอกจากนี้ยังสามารถกำหนดจำนวนภาพที่ต้องการสร้างได้อย่างยืดหยุ่น และระบบมีการจัดการกรณีเกิดข้อผิดพลาด เช่น การวางฟีเจอร์ในพื้นที่ที่ไม่เหมาะสม หรือไม่พบพื้นที่น้ำในภาพ

ด้วยกระบวนการที่ครอบคลุมทั้งหมด เฟรมเวิร์คนี้จึงถือเป็นเครื่องมือที่มีประสิทธิภาพในการเตรียมข้อมูลจากชุดข้อมูลจริงขนาดเล็กให้กลายเป็นชุดข้อมูลจำลองที่พร้อมใช้งานในระบบตรวจจับวัตถุแบบอัตโนมัติ และสามารถนำไปประยุกต์ใช้ได้ในหลายสถานการณ์

บทที่ 4

การพัฒนาและทดสอบระบบ

บทนี้นำเสนอรายละเอียดกระบวนการพัฒนาเฟรมเวิร์คสำหรับสร้างชุดข้อมูลจำลอง (Synthetic Dataset) แบบอัตโนมัติ รวมถึงการสร้างชุดข้อมูลจริง การฝึกโมเดล และการทดสอบประสิทธิภาพของระบบ โดยมีการใช้ภาพจริง และทดลองกับทั้งชุดข้อมูลจริง (Real Dataset) และชุดข้อมูลจำลอง (Synthetic Dataset) อย่างเป็นระบบ

4.1 การพัฒนาเฟรมเวิร์คสำหรับสร้างชุดข้อมูลจำลอง

การสร้างชุดข้อมูลจำลอง (Synthetic Dataset) เป็นกระบวนการสำคัญที่ช่วยให้สามารถขยายปริมาณข้อมูลจากชุดข้อมูลจริงที่มีจำกัดได้อย่างมีประสิทธิภาพ โดยชุดข้อมูลจำลองจะถูกสร้างจากการนำภาพฟีเจอร์ของวัตถุ (เช่น ขวดน้ำ) ที่ถูกแยกออกมาจากภาพจริง มาวางลงบนภาพพื้นหลังที่ไม่มีวัตถุ เพื่อจำลองสถานการณ์ต่าง ๆ อย่างสมจริง และสร้างคำอธิบายตำแหน่งวัตถุสำหรับใช้ในการฝึกโมเดลตรวจจับวัตถุได้

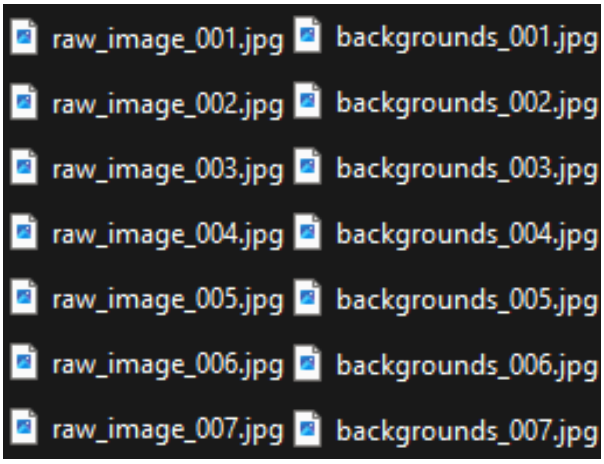
ในการดำเนินงาน โครงการได้พัฒนาเฟรมเวิร์คขึ้นด้วยภาษา Python เวอร์ชัน 3.10 ซึ่งมีไลบรารีที่เหมาะสมสำหรับการประมวลผลภาพ ได้แก่:

- **OpenCV** สำหรับจัดการภาพ การหมุน การย่อขยาย และตรวจจับพื้นที่น้ำในภาพ
- **NumPy** สำหรับการประมวลผลเชิงตัวเลข เช่น การจัดการเมทริกซ์และพิกัดตำแหน่ง
- **rembg** สำหรับลบพื้นหลังของภาพวัตถุ
- **Pillow** สำหรับการ post-processing ภาพ เช่น การเบลอหรือทำให้คมชัด
- **os** สำหรับจัดการไฟล์และโฟลเดอร์ภายในระบบ

เฟรมเวิร์คแบ่งออกเป็น 3 โมดูลหลัก โดยมีไฟล์ main.py เป็นตัวกลางในการเรียกใช้งานแต่ละโมดูลแบบอัตโนมัติและต่อเนื่อง

4.1.1 โมดูล rename.py

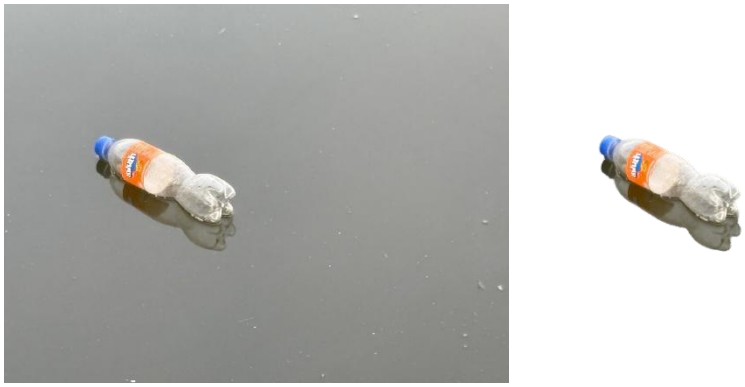
หน้าที่ของโมดูลนี้คือการจัดระเบียบชื่อไฟล์ภาพให้เป็นรูปแบบมาตรฐาน เช่น raw_image_001.jpg, background_001.jpg เพื่อให้สามารถเรียกใช้งานได้สะดวกในกระบวนการต่อไป ลดความผิดพลาดจากการเรียงลำดับผิด หรือชื่อซ้ำกันในระบบไฟล์



ภาพที่ 12 การจัดระเบียบชื่อไฟล์ด้วยโมดูล rename.py

4.1.2 โมดูล extract_features.py

โมดูลนี้ใช้สำหรับแยกภาพของวัตถุออกจากพื้นหลัง โดยไม่จำเป็นต้องใช้ฉากเขียว แต่ใช้ภาพจริงที่ถ่ายจากกล้อง แล้วใช้ไลบรารี rembg เพื่อทำการลบพื้นหลังโดยอิงตามโครงสร้างของภาพ จากนั้นใช้ Pillow เพื่อ post-process เช่น การทำให้ภาพขวดมีความคมชัดมากขึ้น การปรับขอบภาพให้กลมกลืน และการหมุนหรือย่อขนาดให้เหมาะสมกับการวางบนพื้นหลังในขั้นตอนถัดไป



ภาพที่ 13 การแยกภาพของวัตถุออกจากพื้นหลังด้วย extract_features.py

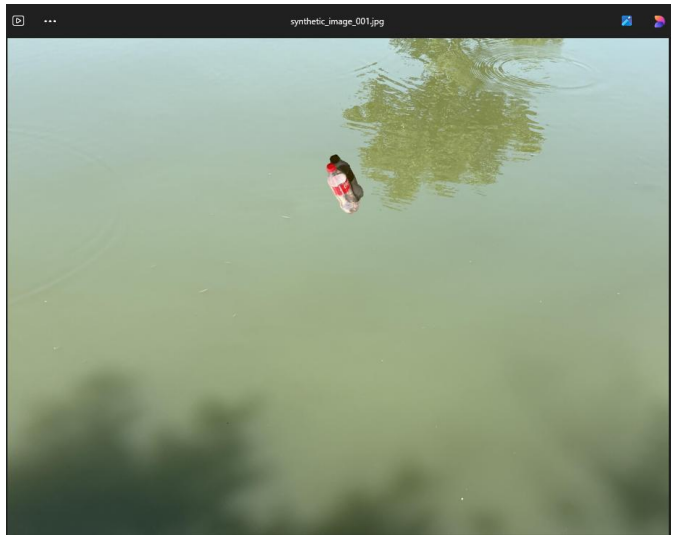
4.1.3 โมดูล generate_synthetic_dataset.py

โมดูลนี้เป็นหัวใจของระบบ โดยมีหน้าที่หลักในการสร้างภาพจำลองแบบสุ่ม โดยระบบจะเลือกพื้นหลังจากภาพที่ถ่ายไว้ และเลือกฟีเจอร์ (ภาพขวดที่ถูกลบพื้นหลังแล้ว) มาหนึ่งภาพ จากนั้นระบบจะตรวจหาพื้นที่ในภาพพื้นหลังที่มีลักษณะเป็น “น้ำ” ด้วยการแปลงภาพเป็น HSV และใช้ค่าช่วงสีในการสร้าง Mask ของบริเวณที่มีโอกาสเป็นน้ำมากที่สุด

เมื่อได้พื้นที่น้ำแล้ว ระบบจะสุ่มตำแหน่งที่เหมาะสมจากบริเวณนั้น เพื่อวางขวดน้ำให้ดูสมจริง โดยมีเงื่อนไขว่าขวดต้องอยู่ภายในขอบเขตของภาพและวางบนพื้นที่น้ำจริงมากกว่า 50% จากนั้นขวดจะถูกหมุนและย่อขนาดในระดับที่สุ่มได้ และทำการวางลงบนภาพพื้นหลังแบบไม่ทับซ้อนกัน

หลังจากรวมภาพเรียบร้อยแล้ว ระบบจะสร้างไฟล์ .jpg ของภาพใหม่ และสร้างไฟล์ .txt ซึ่งเป็น annotation แบบ YOLO โดยคำนวณพิกัด Bounding Box ของขวดตามตำแหน่งจริงที่วาง พร้อมทั้ง normalize ค่าให้เหมาะกับรูปแบบ YOLO ที่ต้องการค่าในช่วง 0 ถึง 1

หากพบว่าภาพพื้นหลังไม่มีพื้นที่น้ำเลย หรือไม่สามารถหาตำแหน่งที่เหมาะสมในการวางฟีเจอร์ ระบบจะข้ามภาพนั้นทันที และแจ้งเตือนผู้ใช้ใน log เพื่อให้สามารถตรวจสอบได้ภายหลัง



ภาพที่ 14 การสร้าง Synthetic Dataset และ Annotation ด้วยโมดูล generate_synthetic_dataset.py

ลำดับการทำงานของระบบ

เมื่อผู้ใช้เรียกใช้งานไฟล์ main.py ระบบจะดำเนินการตามลำดับดังนี้:

1. อ่านภาพจากโฟลเดอร์ raw_images และ backgrounds แล้วเปลี่ยนชื่อให้เป็นมาตรฐาน
2. ทำการลบพื้นหลังภาพขวดน้ำ และเก็บไว้ในโฟลเดอร์ features
3. เลือกภาพพื้นหลังและฟีเจอร์แบบสุ่ม แล้วประมวลผลเพื่อตรวจจับพื้นที่น้ำ
4. วางฟีเจอร์ลงบนพื้นหลัง และบันทึกภาพใหม่พร้อม annotation

ระบบนี้สามารถสร้างภาพได้จำนวนมากในเวลาอันสั้น เช่น สร้างภาพ 200 ภาพ ได้ภายในเวลาไม่ถึง 5 นาที ซึ่งลดเวลาการเตรียมชุดข้อมูลจากเดิมที่ใช้แรงงานคนได้หลายชั่วโมง และสามารถนำไปใช้งานกับชุดข้อมูลอื่นได้ง่าย เพียงเปลี่ยนภาพพื้นหลังและฟีเจอร์เท่านั้น ไม่ต้องแก้ไขโค้ดภายในระบบเลย

C: > Project > scripts > main.py > ...

```python
from create_name_functional import rename_image_files
from extract_features_functional import extract_features
from generate_synthetic_functional import generate_synthetic_dataset

#  กำหนดโฟลเดอร์หลัก
BACKGROUND_FOLDER = r"C:\\Project\\backgrounds"
RAW_IMAGES_FOLDER = r"C:\\Project\\raw_images"
FEATURE_OUTPUT_FOLDER = r"C:\\Project\\features"
SYNTHETIC_OUTPUT_FOLDER = r"C:\\Project\\synthetic_dataset"
ANNOTATION_OUTPUT_FOLDER = r"C:\\Project\\annotations"

#  1. เปลี่ยนชื่อไฟล์ให้เป็นฟอร์แมต xxx_###.ext
rename_image_files(BACKGROUND_FOLDER, prefix="backgrounds")
rename_image_files(RAW_IMAGES_FOLDER, prefix="raw_image")

#  2. ลบพื้นหลังและสร้างฟีเจอร์จาก raw images
extract_features(
    input_folder=RAW_IMAGES_FOLDER,
    output_folder=FEATURE_OUTPUT_FOLDER
)

#  3. สร้าง synthetic dataset พร้อม annotation
generate_synthetic_dataset(
    backgrounds_path=BACKGROUND_FOLDER,
    features_path=FEATURE_OUTPUT_FOLDER,
    output_path=SYNTHETIC_OUTPUT_FOLDER,
    annotations_path=ANNOTATION_OUTPUT_FOLDER,
    num_images=200  # ปรับจำนวนตามต้องการ
)

print("\n เสร็จสมบูรณ์ ")

```

ภาพที่ 15 โค้ด main.py สำหรับการสร้าง Synthetic Dataset จำนวน 200 ภาพ

4.2 การสร้างชุดข้อมูลและการฝึกโมเดลตรวจจับวัตถุ

เพื่อวัตถุประสงค์ในการเปรียบเทียบประสิทธิภาพของการฝึกโมเดลระหว่างชุดข้อมูลที่มาจากภาพจริงและชุดข้อมูลที่สร้างขึ้นแบบจำลอง โครงการนี้จึงได้ดำเนินการสร้างชุดข้อมูล 2 ประเภท ได้แก่:

- **Real Dataset**: ภาพถ่ายจริงของขวดน้ำในสถานการณ์จริง เช่น บริเวณที่มีน้ำขัง สระน้ำ หรือถังน้ำ เป็นต้น โดยไม่มีการตกแต่งหรือปรับฉากถ่าย

- **Synthetic Dataset**: ภาพที่สร้างขึ้นโดยการนำฟีเจอร์ของขวดน้ำที่แยกจากภาพจริงมาวางลงบนพื้นหลังของภาพน้ำเปล่าที่ไม่มีวัตถุใด ๆ ผ่านกระบวนการของเฟรมเวิร์คที่พัฒนาในบทก่อนหน้า

ภาพที่ 16 การเปรียบเทียบระหว่าง Real Dataset (ซ้าย) และ Synthetic Dataset (ขวา)

4.2.1 การจัดเตรียมชุดข้อมูล

ชุดข้อมูลจริงถ่ายจากกล้องมือถือในบริบทที่แตกต่างกัน จำนวน 200 ภาพ และชุดข้อมูลจำลองสร้างขึ้นให้มีจำนวนเท่ากันคือ 200 ภาพเช่นกัน โดยภาพทั้งหมดจะถูกนำไปผ่านกระบวนการ **Data Augmentation** บนแพลตฟอร์ม Roboflow ซึ่งใช้เทคนิคดังต่อไปนี้:

- Flip แบบ Horizontal เพื่อจำลองภาพที่กลับด้าน
- หมุนภาพ 90 องศา ทั้งตามเข็มและทวนเข็มนาฬิกา (90° Rotate: Clockwise & Counter-Clockwise)
- หมุนแบบอิสระในช่วง -15 ถึง +15 องศา (Rotation)
- ปรับระดับความสว่างระหว่าง -20% ถึง +20% (Brightness)
- เพิ่มการเบลอภาพสูงสุดที่ 1.5 พิกเซล (Blur)

- เติม noise แบบสุ่มไม่เกิน 0.5% ของจำนวนพิกเซลในภาพ (Noise)

ระบบจะสร้างภาพใหม่ 3 ภาพจากภาพต้นฉบับ 1 ภาพ ทำให้เมื่อเริ่มจาก 200 ภาพ จะได้ภาพรวมทั้งหมด 600 ภาพในแต่ละชุด (real และ synthetic) ซึ่งเพิ่มความหลากหลายของภาพได้อย่างมีนัยสำคัญและเป็นประโยชน์ต่อการฝึกโมเดล

4.2.2 การแบ่งชุดข้อมูล

หลังจากได้ภาพทั้งหมด 600 ภาพจากขั้นตอนการทำ Augmentation แล้ว จึงทำการแบ่งชุดข้อมูลออกเป็น 3 กลุ่ม ได้แก่

- **Training Set** จำนวน 420 ภาพ (คิดเป็น 70%)
- **Validation Set** จำนวน 120 ภาพ (คิดเป็น 20%)
- **Test Set** จำนวน 60 ภาพ (คิดเป็น 10%)

การแบ่งชุดข้อมูลเป็น 3 ชุดนี้มีวัตถุประสงค์เพื่อใช้ฝึก ตรวจสอบ และประเมินผลโมเดลตามลำดับ โดยชุด Validation ใช้ตรวจสอบความแม่นยำระหว่างการฝึก และ Test Set ใช้ทดสอบโมเดลในภาพที่ไม่เคยเห็นมาก่อน ซึ่งช่วยประเมินประสิทธิภาพได้อย่างเป็นกลาง

4.2.3 การฝึกโมเดลด้วย Google Colab

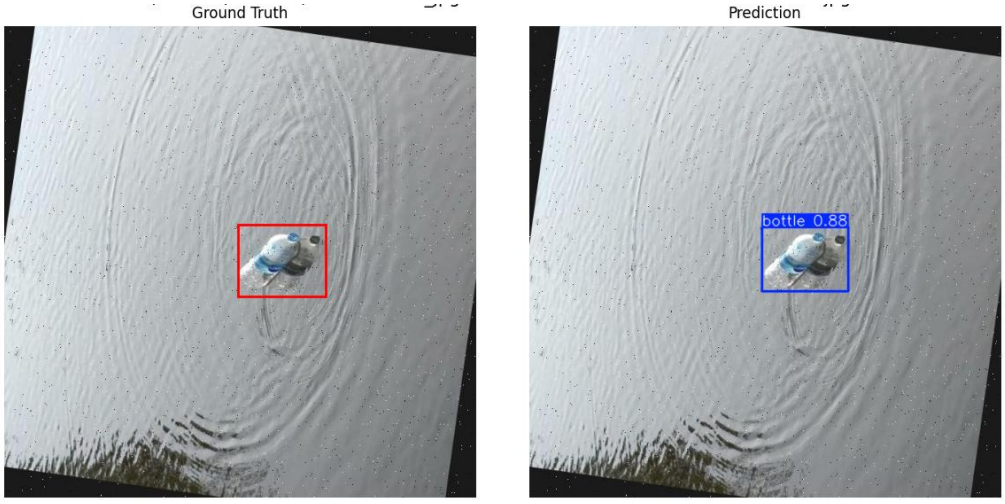
การฝึกโมเดลทั้งหมดดำเนินการผ่าน Google Colab ซึ่งเป็นแพลตฟอร์มที่สามารถใช้งาน GPU ได้ฟรี โดยเชื่อมต่อกับ Google Drive เพื่อเข้าถึงชุดข้อมูลที่ถูกบีบอัดไว้ในรูปแบบ .zip จาก Roboflow จากนั้นทำการแตกไฟล์ออกมาเก็บไว้แยกกันระหว่างชุดข้อมูลจริง (realdata) และข้อมูลจำลอง (syntheticdata) หลังจากเตรียมโครงสร้างไฟล์แล้ว ได้ติดตั้งไลบรารีที่จำเป็น เช่น ultralytics สำหรับเรียกใช้งาน YOLOv8 และ onnx สำหรับการ export โมเดลออกไปใช้งานต่อ

ในการฝึกโมเดลได้เลือก YOLOv8 ทั้งหมด 5 เวอร์ชัน ได้แก่ yolov8n, yolov8s, yolov8m, yolov8l และ yolov8x โดยกำหนด hyperparameters หลักดังนี้:

- จำนวนรอบการฝึก (Epochs): 32
- ขนาดชุดข้อมูลย่อยต่อรอบ (Batch size): 16
- ขนาดภาพ (imgsz): 640

ข้อมูล training และ validation ถูกอ้างอิงผ่านไฟล์ data.yaml ที่สร้างขึ้นเฉพาะในแต่ละรอบการฝึก โดยกำหนดชื่อ class ว่า 'bottle' และใช้ชุดข้อมูลฝึก/ตรวจสอบจากโฟลเดอร์ real หรือ synthetic ตามลำดับ หลังการฝึก โมเดลแต่ละเวอร์ชันจะถูกประเมินโดยใช้ validation set และแสดงค่า mAP@0.5, mAP@0.5:0.95, Precision และ Recall จากนั้น export โมเดลในรูปแบบ .onnx เพื่อใช้งานต่อได้ในระบบอื่น

สุดท้าย ได้ประเมินผลจาก test set โดยใช้ฟังก์ชัน visualize_comparisons_test() ที่แสดงภาพจริงพร้อม bounding box จาก ground truth เทียบกับคำทำนายของโมเดล ซึ่งช่วยให้สามารถประเมินความแม่นยำทั้งเชิงตัวเลขและภาพได้อย่างครบถ้วนหัวข้อถัดไปจะเป็นการประเมินผลอย่างละเอียดเปรียบเทียบระหว่างชุดข้อมูลจริงและชุดข้อมูลจำลอง ทั้งในเชิงปริมาณและเชิงคุณภาพ



ภาพที่ 17 การเปรียบเทียบระหว่างภาพจริงพร้อม bounding box จาก ground truth เทียบกับคำทำนายของโมเดล

4.3 การประเมินผลระบบและการทดสอบการทำงาน

หลังจากทำการฝึกโมเดลด้วยชุดข้อมูลจริงและชุดข้อมูลจำลองแล้ว ได้มีการประเมินผลลัพธ์ของโมเดลในสองด้าน ได้แก่ การประเมินเชิงปริมาณ (Quantitative Evaluation) และการประเมินเชิงคุณภาพ (Qualitative Evaluation) เพื่อให้เห็นถึงประสิทธิภาพของระบบทั้งในเชิงตัวเลขและในทางปฏิบัติจริง

4.3.1 การประเมินเชิงปริมาณ

การประเมินนี้อาศัยค่าเมตริกที่ใช้ทั่วไปในงานตรวจจับวัตถุ ได้แก่:

- **mAP@0.5**: Mean Average Precision ที่ IoU = 0.5
- **mAP@0.5:0.95**: ค่าเฉลี่ยความแม่นยำที่คำนวณจาก IoU หลายค่า ตั้งแต่ 0.5 ถึง 0.95
- **Precision**: ความแม่นยำของโมเดลว่าทำนายถูกมากน้อยแค่ไหน
- **Recall**: ความสามารถของโมเดลในการตรวจจับวัตถุทั้งหมดที่ควรพบ

ค่าที่ได้จากการประเมินบนชุดข้อมูล test set มีดังนี้:

ตารางที่ 4. 1 ผลการประเมินจาก Real Dataset

Model	mAP50	mAP50-95	Precision	Recall	epochs	batch
yolov8n	0.995	0.760	0.991	0.997	32	16
yolov8s	0.995	0.754	0.991	0.991	32	16
yolov8m	0.993	0.728	0.983	0.987	32	16
yolov8l	0.992	0.703	0.974	0.975	32	16
yolov8x	0.971	0.661	0.938	0.925	32	16

ตารางที่ 4. 2 ผลการประเมินจาก Synthetic Dataset

Model	mAP50	mAP50-95	Precision	Recall	epochs	batch
yolov8n	0.984	0.727	0.968	0.933	32	16
yolov8s	0.973	0.699	0.915	0.925	32	16
yolov8m	0.949	0.710	0.911	0.933	32	16
yolov8l	0.951	0.672	0.877	0.951	32	16
yolov8x	0.945	0.582	0.907	0.898	32	16

จากข้อมูลพบว่าโมเดลทุกตัวที่ฝึกจาก Real Dataset ให้ค่า mAP และ Precision สูงกว่าชุด Synthetic ในทุกเวอร์ชัน โดยเฉพาะ YOLOv8n และ YOLOv8s ที่ให้ค่า mAP@0.5 สูงถึง 0.995 ทั้งคู่ ซึ่งแสดงถึงความสามารถในการเรียนรู้ลักษณะวัตถุจากข้อมูลจริงได้ดีมาก

ขณะที่ชุด Synthetic Dataset ก็สามารถให้ผลลัพธ์ที่ใกล้เคียง โดยเฉพาะในกรณีของ YOLOv8n ที่มี mAP@0.5 ถึง 0.984 และ YOLOv8l ที่ให้ Recall สูงถึง 0.951 ซึ่งหมายความว่าแม้

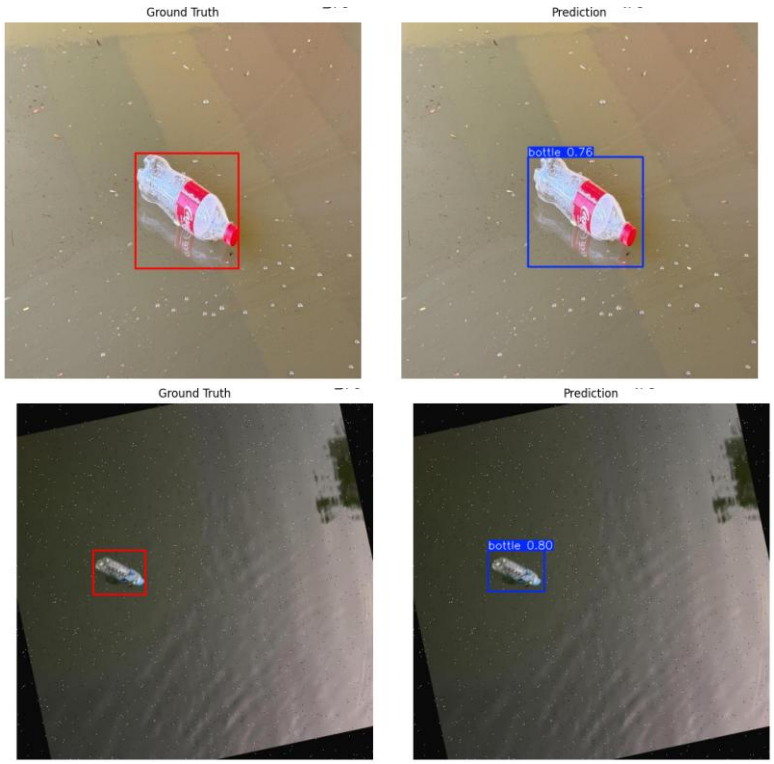
จะสร้างภาพจำลองขึ้นมาโดยไม่ได้อิงจากสถานการณ์จริงโดยตรง แต่หากออกแบบอย่างเหมาะสมและครอบคลุมลักษณะของวัตถุและฉากแวดล้อมได้ดี ก็สามารถฝึกโมเดลให้มีประสิทธิภาพที่ใกล้เคียงกับข้อมูลจริงได้ในระดับหนึ่ง โดยเฉพาะในกรณีที่มีข้อจำกัดด้านทรัพยากร

ผลที่ได้ชี้ให้เห็นว่าการใช้ชุดข้อมูลจำลองสามารถทดแทนการใช้ข้อมูลจริงได้ในระดับหนึ่ง โดยเฉพาะเมื่อต้องการลดเวลาและต้นทุนในการจัดเก็บภาพจริงจำนวนมาก อย่างไรก็ตาม ข้อมูลจริงยังคงให้ความแม่นยำสูงกว่าในเชิงสถิติ ซึ่งยืนยันว่าการใช้ภาพจากสถานการณ์จริงยังคงมีความสำคัญในงานที่ต้องการความแม่นยำสูง

4.3.2 การประเมินเชิงคุณภาพ

ได้มีการแสดงภาพจาก test set พร้อมแสดงผลการตรวจจับของโมเดลในรูปแบบ Bounding Box โดยเปรียบเทียบระหว่างกรอบจาก ground truth กับกรอบที่โมเดลทำนายได้ ซึ่งภาพส่วนใหญ่โมเดลสามารถตรวจจับขวดน้ำได้อย่างแม่นยำ ตรงตำแหน่ง และมีขนาดของกรอบใกล้เคียงกับความจริง โดยไม่พบปัญหา false positive หรือการตรวจจับซ้ำซ้อนในภาพ

ภาพเปรียบเทียบถูกนำเสนอผ่านฟังก์ชัน visualize_comparisons_test() ที่แสดง Bounding Box จากทั้ง ground truth และ prediction บนภาพเดียวกัน ซึ่งช่วยให้ผู้ใช้สามารถประเมินความถูกต้องของโมเดลได้อย่างชัดเจนและเข้าใจง่าย

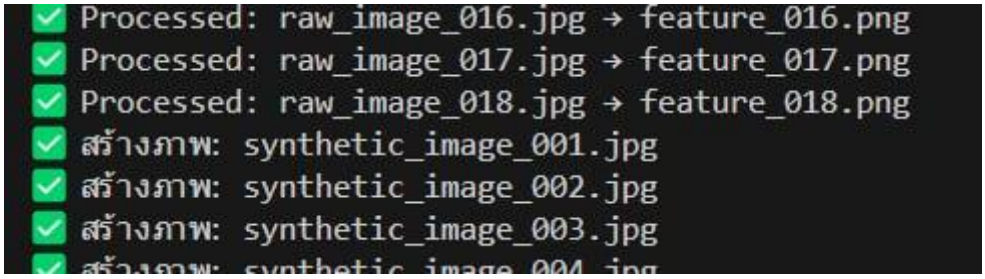


ภาพที่ 18 ภาพการตรวจจับขวดพลาสติกของโมเดล

4.3.3 การทดสอบเฟรมเวิร์ค

นอกจากการประเมินผลของโมเดลแล้ว ยังได้มีการทดสอบเฟรมเวิร์คที่ใช้ในการสร้างชุดข้อมูลจำลอง โดยทดลองรัน main.py ตั้งแต่ต้นจนจบ พบว่าระบบสามารถทำงานได้ครบถ้วนในทุกขั้นตอน ตั้งแต่การเปลี่ยนชื่อไฟล์ การแยกฟีเจอร์ การวางฟีเจอร์ลงบนพื้นหลัง และการสร้างไฟล์ annotation ในรูปแบบ YOLO อย่างถูกต้อง

ระบบยังสามารถจัดการกรณีที่ไม่พบพื้นที่น้ำในภาพพื้นหลังได้อย่างเหมาะสม โดยมีการข้ามภาพนั้นและบันทึก log แจ้งเตือน ทำให้สามารถใช้งานได้ต่อเนื่องโดยไม่เกิดข้อผิดพลาดร้ายแรง ซึ่งสะท้อนถึงความเสถียรของเฟรมเวิร์คที่พัฒนาได้เป็นอย่างดี



ภาพที่ 19 การทำงานของเฟรมเวิร์คตั้งแต่การแยกฟีเจอร์และสร้าง Synthetic Dataset

บรรณนานุกรม

ปรียาพร เหล่าติวานนท์, และสุภัสสร ไชยวรรณ. (2567). การพัฒนาการตรวจจับรอยเปื้อนบนเสื้อผ้า (Development of Stain Detection on Clothes). มหาวิทยาลัยวลัยลักษณ์.

Amazon Web Services (AWS). (n.d.). What is synthetic data? *Amazon Web Services (AWS).* Retrieved from https://aws.amazon.com/th/what-is/synthetic-data/

Ball, R., & Lynn, T. (2024). Blue Eco Line Reduces River Pollution with Roboflow and Intel Sapphire Rapids. *Roboflow Blog.* Retrieved from https://blog.roboflow.com/autonomous-river-cleaning-computer-vision/

Bee, P. (2018). What is mAP Evaluation? *Medium*. Retrieved from https://medium.com/boobeejung/what-is-map-evaluation-8d7a4adc67da

Big Data Institute. (2023). การตรวจจับวัตถุด้วยรูปภาพ (Image Detection). BDI. Retrieved from https://bdi.or.th/big-data-101/image-detection/

DEF Thailand. (2024). TensorFlow คืออะไร. *DEF Thailand.* Retrieved from https://def.co.th/th/blog/article/364

Expert Programming Tutor. (n.d.). Object detection and tracking with OpenCV - Using optical flow for motion detection. *Expert Programming Tutor.* Retrieved from https://expert-programming-tutor.com/tutorial/article/KE003613_Object_Detection_and_Tracking_with_OpenCV_-_Using_Optical_Flow_for_Motion_Detection.php

Expert Programming Tutor. (n.d.). Object detection with OpenCV - Object Detection with YOLO (You Only Look Once). *Expert Programming Tutor.* Retrieved from https://expert-programming-tutor.com/tutorial/article/KE003608_Object_detection_with_OpenCV_-Object_Detection_with_YOLO__You_Only_Look_Once.php

Expert Programming Tutor. (n.d.). OpenCV-Python: Library for image processing, what it works, and an example in Python. *Expert Programming Tutor.* Retrieved from https://expert-programming-tutor.com/tutorial/article/KE000730_OpenCV-Python_Library_for_Image_Processing_is_what_it_works_And_give_an_example_in_Python_How_to_do_with_Code.php

Fastwork. (n.d.). รู้ให้ลึก! Image processing เทคโนโลยีการประมวลผลภาพ คืออะไร. *Fastwork*. Retrieved from https://blog.fastwork.co/what-is-image-processing/

GFStealer-666. (2023). การใช้ YOLO อัลกอริทึ่มในการตรวจจับวัตถุ (Object detection). *Dev.to.* Retrieved from https://dev.to/gfstealer666/kaaraich-yolo-alkrithuemainkaartrwcchcchabwatthu-object-detection-3lef

Juras, P. (2020). Practical introduction to object detection: Training and deployment. *Edge AI and Vision.* Retrieved from https://www.edge-ai-vision.com/wp-content/uploads/2020/12/Juras_EJConsulting_2020_Embedded_Vision_Summit_Slides_Final.pdf

Kanoktipsatharporn, S. (2020). Object Detection คืออะไร บทความสอน AI ตรวจจับวัตถุ TensorFlow.js หลักการทำ Object Detection การตรวจจับวัตถุในรูปภาพ จากโมเดลสำเร็จรูป COCO-SSD – tfjs ep.8. *Bualabs*. Retrieved from https://www.bualabs.com/archives/3453/

Le, B. (2023). Get Started with Training a YOLOv8 Object Detection Model. *Datature Blog.* Retrieved from https://www.datature.io/blog/get-started-with-training-a-yolov8-object-detection-model

Mindphp. (n.d.). Roboflow ตัวช่วยสำคัญในการสร้างและปรับแต่ง Dataset. *Mindphp*. Retrieved from https://www.mindphp.com/บทเรียนออนไลน์/python-tensorflow/10210-roboflow-ตัวช่วยสำคัญในการสร้างและปรับแต่ง-dataset.html

Pham, C. K., Bessa, F., Otero, V., Frias, J. P. G. L., & Sobral, P. (2017). The Feeding Behavior of Sea Turtles on Plastic Bags and Its Health Impacts: A Case Study of Marine Plastic Pollution in Thailand. Retrieved from https://accstr.ufl.edu/wp-content/uploads/sites/98/Pham_et_al_MarPolBull_2017.pdf

Pinyorattanakit, N. (2023). สร้าง AI ทำนายรูปภาพด้วย Roboflow. *Medium*. Retrieved from https://nattakit-nice2580.medium.com/สร้าง-ai-ทำนายรูปภาพด้วย-roboflow-6bb96a632f4d

Ponnipa, O. (2021). Data Augmenting for object detection in YOLO. *Super AI Engineer.* Retrieved from https://medium.com/super-ai-engineer/augmenting-a-dataset-for-object-detection-in-yolo-234783155f41

Ponnipa, O. (2021). การประมวลผลภาพเบื้องต้น. *Super AI Engineer.* Retrieved from https://medium.com/super-ai-engineer/การประมวลผลภาพเบื้องต้น-fae1f73d8933

Redmon, J. (n.d.). YOLO (You Only Look Once): Real-Time Object Detection Algorithm. Retrieved from https://pjreddie.com/darknet/yolo/

Shah, D. (2022). Mean Average Precision (mAP) Explained: Everything You Need to Know. *V7 Labs.* Retrieved from https://www.v7labs.com/blog/mean-average-precision

Shaip. (2022). ข้อมูลสังเคราะห์และบทบาทในโลกของ AI – ประโยชน์ กรณีใช้งาน ประเภท & ความท้าทาย. *Shaip*. Retrieved from https://th.shaip.com/blog/synthetic-data-and-ai/

Skalski, P. (2023). How to Train YOLOv8 Object Detection on a Custom Dataset. *Roboflow Blog*. Retrieved from https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/

T., Chanisara. (2021). การประมวลผลภาพด้วย OpenCV & Python. *Medium.* Retrieved from https://medium.com/@chanisara.tan/การประมวลผลภาพด้วย-opencv-python-2e86f4ed41c7

TESR Shop. (2021). ตรวจหาวัตถุจากภาพ โดยวิธีการ HSV filter. *TESR Shop*. Retrieved from https://www.tesrshop.com/blog/ตรวจหาวัตถุจากภาพ-โดยวิธีการ-hsv-filter/9

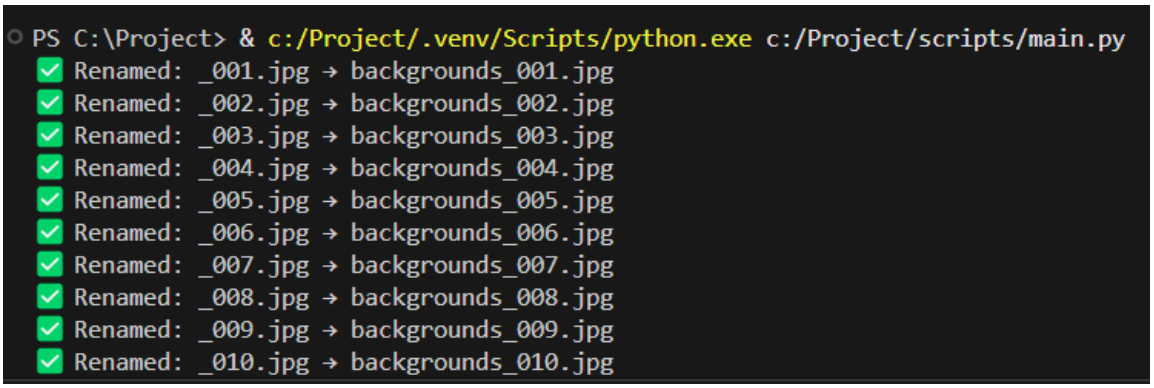
ภาคผนวก

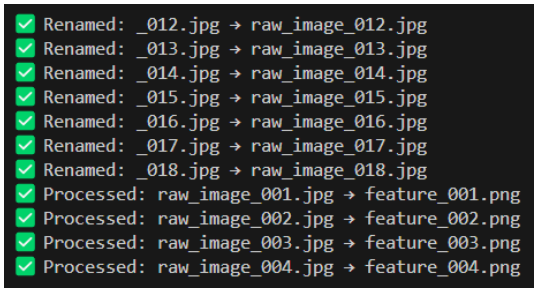
ภาคผนวก ก

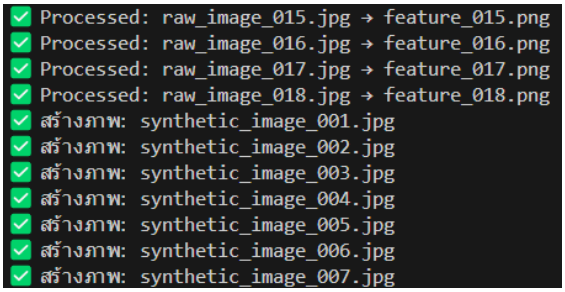
ภาพหน้าจอยืนยันการทำงานของระบบ (System Execution Screenshots)

ในภาคผนวกนี้แสดงภาพหน้าจอที่ใช้เป็นหลักฐานประกอบการพัฒนาระบบจริง ตั้งแต่กระบวนการรันเฟรมเวิร์คสร้างชุดข้อมูลจำลอง (Synthetic Dataset), การใช้ Roboflow เพื่อจัดการข้อมูล, การฝึกโมเดลผ่าน Google Colab และการทดสอบผลลัพธ์จากโมเดลที่ฝึกสำเร็จแล้ว

1. ภาพหน้าจอขณะรัน main.py





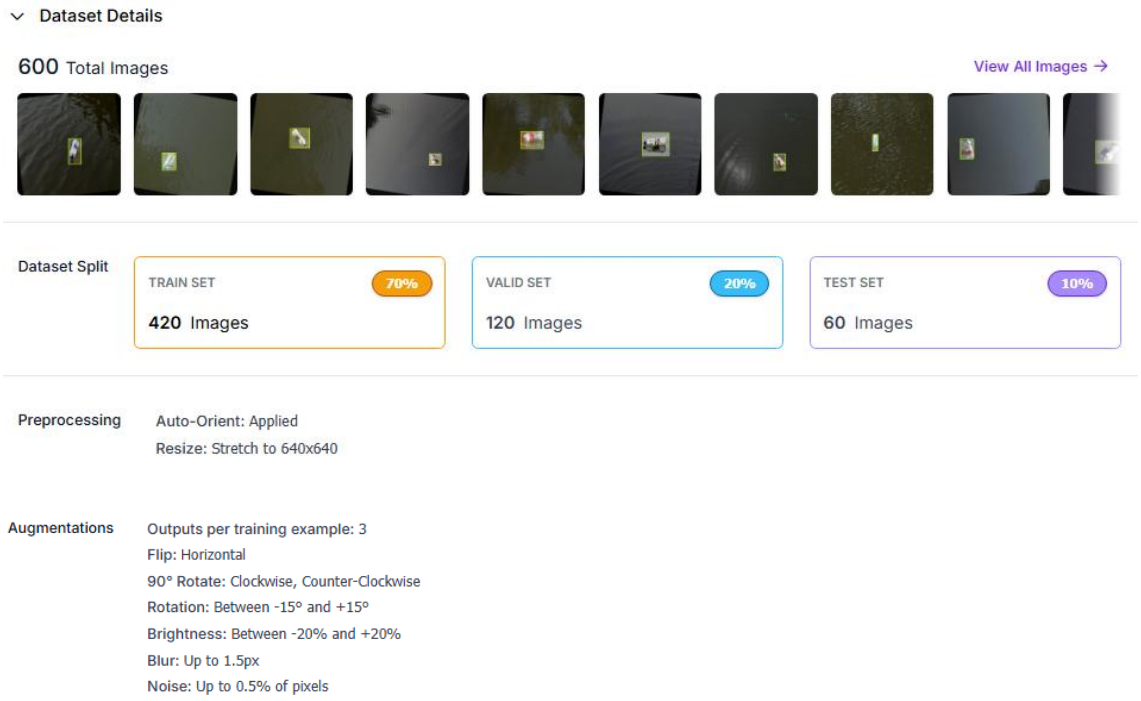


ภาพนี้แสดงหน้าจอการรันสคริปต์ main.py ที่เป็นศูนย์กลางของระบบเฟรมเวิร์คแบบอัตโนมัติ โดยเรียกใช้ฟังก์ชันจากไฟล์ rename.py, extract_features.py และ generate_synthetic_dataset.py ตามลำดับ แสดงการทำงานแบบต่อเนื่องของระบบเฟรมเวิร์ค ที่สามารถจัดการตั้งแต่เปลี่ยนชื่อภาพ ลบพื้นหลัง และสร้างภาพจำลองพร้อม annotation ได้ในคำสั่งเดียว

2. ภาพ input/output ขณะสร้างภาพ synthetic

ภาพด้านซ้ายคือ input พื้นหลังและฟีเจอร์ (ภาพขวดน้ำที่ผ่านการแยกพื้นหลังแล้ว) ส่วนภาพด้านขวาคือผลลัพธ์จากการรวมฟีเจอร์ลงพื้นหลังน้ำแบบสุ่ม แสดงกระบวนการรวมฟีเจอร์กับพื้นหลัง เพื่อสร้างข้อมูลจำลองที่หลากหลาย โดยกำหนดพิกัดการวางตามตำแหน่งพื้นที่น้ำในภาพ

3. ภาพหน้าจอจาก Roboflow ขณะ export dataset (Train / Val / Test)



ภาพนี้แสดงขั้นตอนการนำภาพเข้าสู่ Roboflow เพื่อทำ Data Augmentation และแบ่งชุดข้อมูลออกเป็น Train, Validation และ Test พร้อม export ในรูปแบบ YOLOv8 และแสดงการตั้งค่าและแบ่งชุดข้อมูลใน Roboflow ซึ่งช่วยให้ได้ dataset ที่สมดุลและสะดวกต่อการฝึกโมเดล

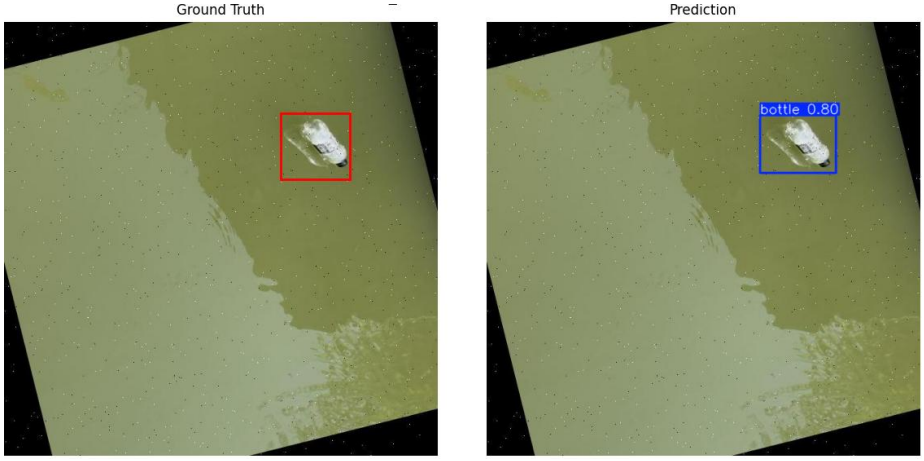
4. ภาพหน้าจอ Google Colab ช่วงฝึก YOLOv8 (แสดง mAP)

```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
28/32     3.04G     0.7764     0.4559      1.018          4        640: 100%|██████████| 27/27 [00:07<00:00,  3.79it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  2.93it/s]
             all        120        120      0.986      0.983      0.995      0.744

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
29/32     3.06G     0.7874     0.4668       1.02          4        640: 100%|██████████| 27/27 [00:05<00:00,  4.69it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  3.20it/s]
             all        120        120          1      0.991      0.995      0.753

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
30/32     3.08G      0.776     0.4534      1.012          4        640: 100%|██████████| 27/27 [00:07<00:00,  3.66it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  3.40it/s]
             all        120        120      0.992      0.998      0.995       0.76

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
31/32     3.08G     0.7639     0.4496      1.013          4        640: 100%|██████████| 27/27 [00:05<00:00,  4.77it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  3.43it/s]
             all        120        120          1      0.998      0.995      0.754

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
32/32     3.11G     0.7314     0.4386     0.9832          4        640: 100%|██████████| 27/27 [00:07<00:00,  3.65it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 4/4 [00:01<00:00,  3.58it/s]
             all        120        120      0.998          1      0.995      0.755
```

เป็นภาพตอนที่ฝึกโมเดล YOLOv8 บน Google Colab โดย mAP@0.5 และ mAP@0.5:0.95 ในรอบต่าง ๆ ภาพนี้แสดงผลการฝึก YOLOv8s ที่รอบที่ 32 โดยมีค่า mAP@0.5 ถึง 0.995 ซึ่งบ่งบอกถึงประสิทธิภาพในการเรียนรู้จากข้อมูลจริงได้อย่างแม่นยำ

5. ภาพทำนาย Bounding Box จาก test set (Prediction vs Ground Truth)

ภาพนี้แสดงผลการทำนายของโมเดลบนชุดข้อมูลทดสอบ โดยมีกรอบสีน้ำเงินเป็น bounding box ที่โมเดลตรวจจับได้ พร้อมค่าความมั่นใจ (confidence score)



ภาพนี้แสดงการทดสอบโมเดลที่ฝึกแล้ว พบว่าขวดน้ำในภาพถูกตรวจจับอย่างถูกต้องใกล้เคียงกับตำแหน่งจริง แสดงถึงความแม่นยำของระบบ

ภาคผนวก ข

โค้ดที่พัฒนาขึ้นพร้อมคำอธิบาย

ภาคผนวกนี้แสดงตัวอย่างโค้ดที่นักศึกษาได้พัฒนาขึ้นเพื่อใช้ในระบบอัตโนมัติสำหรับการสร้างชุดข้อมูลจำลอง (Synthetic Dataset) โดยแบ่งเป็น 4 โมดูลสำคัญ ได้แก่ rename.py, extract_features.py, generate_synthetic_dataset.py, และ main.py ซึ่งทั้งหมดถูกออกแบบให้สามารถทำงานร่วมกันอย่างเป็นระบบภายใต้เฟรมเวิร์คอัตโนมัติ โดยมีคำอธิบายประกอบแต่ละส่วนเพื่อแสดงแนวคิดและเหตุผลในการออกแบบ

1. rename_image_files() จากไฟล์: create_name_functional.py

```python
def rename_image_files(folder_path, prefix="backgrounds", extensions=None):
    """
    เปลี่ยนชื่อไฟล์ภาพในโฟลเดอร์ให้เป็นรูปแบบ prefix_001.jpg, prefix_002.jpg, ...
    """
    if extensions is None:
        extensions = ['.jpg', '.jpeg', '.png', '.heic']  # นามสกุลไฟล์ภาพที่รองรับ

    count = 1
    for filename in sorted(os.listdir(folder_path)):
        name, ext = os.path.splitext(filename)
        if ext.lower() in extensions:
            # สร้างชื่อใหม่แบบมีเลข 3 หลัก เช่น backgrounds_001.jpg
            new_name = f"{prefix}_{count:03d}{ext.lower()}"
            src_path = os.path.join(folder_path, filename)
            dst_path = os.path.join(folder_path, new_name)

            # เปลี่ยนชื่อถ้าไม่ซ้ำ
            if not os.path.exists(dst_path):
                os.rename(src_path, dst_path)
                print(f"✅ Renamed: {filename} → {new_name}")
                count += 1
            else:
                print(f"⚠️ Skipped (duplicate name): {new_name}")
```

ฟังก์ชันนี้ใช้สำหรับ เปลี่ยนชื่อไฟล์ภาพในโฟลเดอร์ ให้อยู่ในรูปแบบเรียงลำดับ เช่น backgrounds_001.jpg, raw_image_002.jpg เป็นต้น เพื่อให้สามารถนำไปใช้งานในขั้นตอนถัดไปได้สะดวก

- ใช้ os.listdir() ดึงชื่อไฟล์ทั้งหมด แล้ว sorted() เพื่อเรียงตามลำดับชื่อ
- ตรวจสอบว่านามสกุลไฟล์ตรงกับประเภทที่รองรับ (.jpg, .png, .heic)
- เปลี่ยนชื่อโดยใช้ format prefix_XXX.ext โดย XXX เป็นเลข 3 หลัก (:03d)
- ใช้ os.rename() เปลี่ยนชื่อและ os.path.exists() ป้องกันชื่อซ้ำ

2. extract_features() จากไฟล์: extract_features_functional.py

```python
def extract_features(input_folder, output_folder, prefix="feature"):
    """
    ลบพื้นหลังจาก raw images และบันทึกภาพ feature ที่ post-processed แล้ว
    โดยตั้งชื่อเป็น feature_001.png, feature_002.png, ...
    """
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    count = 1
    for filename in sorted(os.listdir(input_folder)):
        if filename.lower().endswith(".jpg"):
            input_path = os.path.join(input_folder, filename)
            output_filename = f"{prefix}_{count:03d}.png"
            output_path = os.path.join(output_folder, output_filename)
            temp_output_path = os.path.join(output_folder, f"temp_{output_filename}")

            try:
                # 🔍 ลบพื้นหลังจากภาพต้นฉบับ
                with open(input_path, "rb") as input_file:
                    input_data = input_file.read()
                    output_data = remove(
                        input_data,
                        alpha_matting=True,
                        alpha_matting_foreground_threshold=240,
                    )

                # 💾 บันทึกภาพชั่วคราว
                with open(temp_output_path, "wb") as temp_output_file:
                    temp_output_file.write(output_data)

                # ✨ Post-processing และบันทึกผลลัพธ์สุดท้าย
                processed_image = post_process_image(temp_output_path)
                processed_image.save(output_path)

                os.remove(temp_output_path)

                print(f"✅ Processed: {filename} → {output_filename}")
                count += 1

            except Exception as e:
                print(f"❌ Error processing {filename}: {e}")
```

ฟังก์ชันนี้ทำหน้าที่ แยกฟีเจอร์ของวัตถุออกจากพื้นหลัง ของ raw image และ บันทึกเป็นภาพโปร่งใส (PNG)

การทำงานหลัก:

- ใช้ rembg.remove() เพื่อลบพื้นหลังแบบอัตโนมัติ
- ใช้ ImageFilter.GaussianBlur และ ImageFilter.SHARPEN เพื่อ ปรับความคมของภาพ หลังลบพื้นหลังแล้ว

- บันทึกภาพที่แยกแล้วออกมาเป็น .png พร้อมตั้งชื่อใหม่

3. generate_synthetic_functional.py

3.1 detect_water_area()

หน้าที่: ตรวจจับพื้นที่น้ำในภาพพื้นหลัง ด้วยการแปลงภาพเป็น HSV แล้วกรองตามช่วงสี (Hue, Saturation, Value)

```python
def detect_water_area(image):
    """ตรวจจับพื้นที่น้ำในภาพด้วย HSV และ Morphological filter"""
    LOWER_BOUND = np.array([0, 0, 0])
    UPPER_BOUND = np.array([110, 255, 220])

    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv_image, LOWER_BOUND, UPPER_BOUND)

    kernel = np.ones((5, 5), np.uint8)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel)
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

    return mask
```

- ใช้ cv2.inRange เพื่อสร้าง Mask เฉพาะจุดที่อยู่ในช่วงค่าที่นิยามว่าเป็นน้ำ
- จากนั้นใช้ Morphological Filters (Close + Open) เพื่อลบจุด noise และเติมจุดว่างเล็ก ๆ ให้เนียน

3.2 place_feature_on_water()

หน้าที่: วางฟีเจอร์ (ขวดน้ำแบบมี alpha) ลงในบริเวณที่ตรวจจับว่าเป็นน้ำ

```python
def place_feature_on_water(background, feature, water_mask):
    """วางฟีเจอร์ (ภาพโปร่งใส) ลงบนพื้นที่น้ำของภาพพื้นหลัง"""
    bg_height, bg_width, _ = background.shape
    fg_height, fg_width, _ = feature.shape

    scale = np.random.uniform(0.4, 0.8)
    new_fg_width = int(fg_width * scale)
    new_fg_height = int(fg_height * scale)
    feature_resized = cv2.resize(feature, (new_fg_width, new_fg_height), interpolation=cv2.INTER_AREA)

    water_coords = np.column_stack(np.where(water_mask > 0))
    if len(water_coords) == 0:
        print("❌ ไม่มีพื้นที่น้ำในภาพพื้นหลัง")
        return None, None, None

    attempts = 0
    while attempts < 100:
        y, x = random.choice(water_coords)
        if (x + new_fg_width <= bg_width and y + new_fg_height <= bg_height):
            region = water_mask[y:y + new_fg_height, x:x + new_fg_width]
            if np.count_nonzero(region) / (new_fg_width * new_fg_height) >= 0.5:
                break
        attempts += 1

    if attempts >= 10000:
        print("❌ ไม่พบตำแหน่งวางที่เหมาะสม")
        return None, None, None

    angle = np.random.uniform(0, 360)
    M = cv2.getRotationMatrix2D((new_fg_width // 2, new_fg_height // 2), angle, 1)
    rotated_feature = cv2.warpAffine(feature_resized, M, (new_fg_width, new_fg_height), borderMode=cv2.BORDER_CONSTANT, borderValue=(0, 0, 0, 0))

    return rotated_feature, x, y
```

- ย่อขนาดฟีเจอร์แบบสุ่ม ด้วย scale = np.random.uniform(0.4, 0.8)
- สุ่มพิกัดวาง ใน Mask ของพื้นที่น้ำ (water_mask) โดย:
    - ต้องแน่ใจว่าพิกัดที่เลือกอยู่ภายในภาพ
    - พื้นที่นั้นมีน้ำครอบคลุม ≥ 50%
    - จำกัดการพยายามไม่เกิน 100 ครั้ง
- หมุนภาพแบบสุ่ม เพื่อความสมจริง โดยใช้ cv2.getRotationMatrix2D และ warpAffine

3.3 overlay_feature()

หน้าที่: วางภาพฟีเจอร์ที่มี alpha channel ลงบนภาพพื้นหลังจริง

```
def overlay_feature(background, feature, x, y):
    """วางภาพฟีเจอร์ลงบนพื้นหลัง โดยดูจาก alpha channel"""
    fg_height, fg_width, _ = feature.shape
    overlay = background.copy()
    for i in range(fg_height):
        for j in range(fg_width):
            if feature[i, j, 3] > 0:
                overlay[y + i, x + j] = feature[i, j, :3]
    return overlay
```

- ใช้ลูปซ้อน for i, j เช็คค่าความโปร่งใสของพิกเซล (feature[i, j, 3])
- ถ้า pixel นั้นไม่โปร่งใส (alpha > 0) ⟶ วางลงบน background

3.4 get_true_bbox_from_alpha()

หน้าที่: คำนวณ Bounding Box จากขอบเขตของพิกเซลที่มี alpha > 0

```
def get_true_bbox_from_alpha(feature_image):
    """คำนวณ Bounding Box จาก alpha channel ของฟีเจอร์"""
    alpha = feature_image[:, :, 3]
    coords = cv2.findNonZero(alpha)
    if coords is None:
        return 0, 0, feature_image.shape[1], feature_image.shape[0]
    x, y, w, h = cv2.boundingRect(coords)
    return x, y, w, h
```

- ใช้ cv2.findNonZero(alpha) หาตำแหน่งทั้งหมดที่มีพิกเซล
- ใช้ cv2.boundingRect เพื่อหาขอบเขตสี่เหลี่ยม

3.5 save_annotation()

หน้าที่: เขียนไฟล์ .txt สำหรับ annotation แบบ YOLO

```python
def save_annotation(annotation_path, x, y, width, height, angle, bg_width, bg_height):
    """บันทึก annotation แบบ YOLO format"""
    x_center = (x + width / 2) / bg_width
    y_center = (y + height / 2) / bg_height
    norm_width = width / bg_width
    norm_height = height / bg_height

    with open(annotation_path, 'w') as f:
        f.write(f"0 {x_center} {y_center} {norm_width} {norm_height}\n")
```

- แปลงตำแหน่งให้อยู่ในรูปแบบ normalized
- บันทึกค่า class_id x_center y_center width height ลงในไฟล์ .txt

3.6 generate_synthetic_dataset()

หน้าที่: เป็นฟังก์ชันหลักที่รันทุกกระบวนการจากโฟลเดอร์ภาพพื้นหลัง + ฟีเจอร์ แล้วสร้างภาพจำลองและ annotation ทั้งชุด

```python
def generate_synthetic_dataset(backgrounds_path, features_path, output_path, annotations_path, num_images):
    """
    สร้าง Synthetic Dataset ด้วยการวางฟีเจอร์บนภาพพื้นหลังที่มีน้ำ
    และสร้างไฟล์ Annotation ประกอบ
    """
    backgrounds = [os.path.join(backgrounds_path, f) for f in sorted(os.listdir(backgrounds_path)) if f.endswith('.jpg')]
    features = [os.path.join(features_path, f) for f in sorted(os.listdir(features_path)) if f.endswith('.png')]

    os.makedirs(output_path, exist_ok=True)
    os.makedirs(annotations_path, exist_ok=True)

    for i in range(1, num_images + 1):
        try:
            bg_path = random.choice(backgrounds)
            feature_path = random.choice(features)

            background = cv2.imread(bg_path)
            feature = cv2.imread(feature_path, cv2.IMREAD_UNCHANGED)

            if background is None or feature is None:
                print(f"❌ ไม่สามารถโหลดภาพ: {bg_path} หรือ {feature_path}")
                continue

            water_mask = detect_water_area(background)
            placed_feature, x, y = place_feature_on_water(background, feature, water_mask)

            if placed_feature is None:
                continue

            synthetic_image = overlay_feature(background, placed_feature, x, y)
            image_name = f"synthetic_image_{i:03d}.jpg"
            annotation_name = f"synthetic_image_{i:03d}.txt"

            cv2.imwrite(os.path.join(output_path, image_name), synthetic_image)

            rel_x, rel_y, rel_w, rel_h = get_true_bbox_from_alpha(placed_feature)
            save_annotation(
                os.path.join(annotations_path, annotation_name),
                x + rel_x, y + rel_y, rel_w, rel_h, 0,
                background.shape[1], background.shape[0]
            )

            print(f"✅ สร้างภาพ: {image_name}")

        except Exception as e:
            print(f"❌ Error at image {i}: {e}")
```

- ใช้ detect_water_area, place_feature_on_water, overlay_feature, get_true_bbox_from_alpha, save_annotation
- วนสร้างภาพจำนวน num_images (200 ภาพ)
- ตรวจสอบความผิดพลาดทุกขั้นตอน (เช่น โหลดภาพไม่ได้

4. main.py จากไฟล์: main.py

```python
from create_name_functional import rename_image_files
from extract_features_functional import extract_features
from generate_synthetic_functional import generate_synthetic_dataset

#  กำหนดโฟลเดอร์หลัก
BACKGROUND_FOLDER = r"C:\\Project\\backgrounds"
RAW_IMAGES_FOLDER = r"C:\\Project\\raw_images"
FEATURE_OUTPUT_FOLDER = r"C:\\Project\\features"
SYNTHETIC_OUTPUT_FOLDER = r"C:\\Project\\synthetic_dataset"
ANNOTATION_OUTPUT_FOLDER = r"C:\\Project\\annotations"

#  1. เปลี่ยนชื่อไฟล์ให้เป็นฟอร์แมต xxx_###.ext
rename_image_files(BACKGROUND_FOLDER, prefix="backgrounds")
rename_image_files(RAW_IMAGES_FOLDER, prefix="raw_image")

#  2. ลบพื้นหลังและสร้างฟีเจอร์จาก raw images
extract_features(
    input_folder=RAW_IMAGES_FOLDER,
    output_folder=FEATURE_OUTPUT_FOLDER
)

#  3. สร้าง synthetic dataset พร้อม annotation
generate_synthetic_dataset(
    backgrounds_path=BACKGROUND_FOLDER,
    features_path=FEATURE_OUTPUT_FOLDER,
    output_path=SYNTHETIC_OUTPUT_FOLDER,
    annotations_path=ANNOTATION_OUTPUT_FOLDER,
    num_images=200  # ปรับจำนวนตามต้องการ
)

print("\n เสร็จสมบูรณ์ ")
```

โค้ดนี้เรียกใช้ฟังก์ชันทั้งหมดที่กล่าวมา เพื่อทำงานแบบ อัตโนมัติจบในครั้งเดียว:

1. เปลี่ยนชื่อภาพให้เป็นรูปแบบ prefix_xxx
2. ลบพื้นหลังและสร้างฟีเจอร์
3. สร้างภาพจำลองและ annotati