ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู
Line Chatbot for Snake Classification

โดย

นายชลภพ	พอกพูล	รหัสนิสิต 6430200132
นางสาวธนพร	เกิดผล	รหัสนิสิต 6430200281

อาจารย์ที่ปรึกษา
ผู้ช่วยศาสตราจารย์ ดร.ชโลธร ชูทอง

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา
01418499 โครงงานวิทยาการคอมพิวเตอร์
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ ศรีราชา มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา
ประจำภาคปลาย ปีการศึกษา 2567

ชื่อโครงงาน ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู
(Snake species classification on line chatbot)
ผู้จัดทำ นายชลภพ พอกพูล รหัสนักศึกษา 6430200132
นางสาวธนพร เกิดผล รหัสนักศึกษา 6430200281
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. ชโลธร ชูทอง
หลักสูตร ปริญญาวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์

บทคัดย่อ

โครงงานศึกษานี้มีวัตถุประสงค์ในการพัฒนาระบบจำแนกสายพันธุ์งูโดยใช้เทคโนโลยีปัญญาประดิษฐ์และการประมวลผลภาพ เพื่อแก้ปัญหาการไม่สามารถระบุชนิดของงูที่กัดได้อย่างแม่นยำ ซึ่งเป็นปัจจัยสำคัญที่ส่งผลต่อประสิทธิภาพการรักษา ในประเทศไทยพบว่าปัญหาการถูกงูกัดยังคงเป็นปัญหาสาธารณสุขที่สำคัญ เนื่องจากผู้ที่ถูกงูกัดไม่สามารถระบุชนิดของงูได้ถึงร้อยละ 69.24 ซึ่งอาจทำให้เกิดข้อผิดพลาดในการรักษาและการใช้เซรุ่มที่ไม่ตรงกับชนิดของพิษจากงูแต่ละสายพันธุ์

โครงงานนี้ได้นำอัลกอริทึมที่ออกแบบมาเพื่อการตรวจจับวัตถุในภาพ ได้แก่ YOLOv5 และ YOLOv11 สำหรับการตรวจจับงูจากภาพ และใช้เทคนิค Deep Learning ที่หลากหลายมาใช้ ได้แก่ YOLOv5, YOLOv8, YOLOv11, MobileNetV2 และ EfficientNet ในการจำแนกสายพันธุ์ของงูจากภาพ โดยระบบนี้สามารถแยกแยะสายพันธุ์งูได้ทั้งหมด 10 สายพันธุ์ ทั้งงูมีพิษและงูไม่มีพิษ ซึ่งจะช่วยลดความเสี่ยงในการรักษาและเพิ่มประสิทธิภาพในการปฐมพยาบาล การรักษา รวมถึงช่วยในการตัดสินใจในกรณีฉุกเฉิน ระบบนี้ได้เชื่อมต่อกับ LINE Chatbot ซึ่งเป็นแพลตฟอร์มที่มีผู้ใช้งานจำนวนมากในประเทศไทย โดยผู้ใช้งานสามารถส่งภาพของงูเข้ามาวิเคราะห์และได้รับข้อมูลเบื้องต้นเกี่ยวกับสายพันธุ์งู รวมถึงคำแนะนำในการปฐมพยาบาลและข้อมูลการติดต่อหน่วยงานที่เกี่ยวข้องในกรณีฉุกเฉิน

ผลการทดสอบโมเดลพบว่ามีค่าความถูกต้องในการตรวจจับงู 0.806 และ ค่าความถูกต้องในการจำแนกสายพันธุ์งู 0.769 แม้ว่ายังมีข้อจำกัดในการใช้งานในบางกรณี แต่ระบบนี้ยังแสดงให้เห็นถึงศักยภาพในการพัฒนาและปรับปรุงในอนาคต เพื่อเพิ่มความแม่นยำและประสิทธิภาพในการใช้งานจริง โดยโครงงานนี้ยังช่วยสร้างความรู้และความตระหนักเกี่ยวกับการจัดการกรณีฉุกเฉินเมื่อถูกงูกัดอย่างมีประสิทธิภาพ.

Thesis Title Line Chatbot for Snake Classification
Author Mr.Chonlapop Pokpool Student ID 6430200132
Miss.Thanaporn Keadphol Student ID 6430200281
Thess Advisor Dr. Chalothon Chootong
Degree Bachelor of Science (Computer Science)

ABSTRACT

This study aims to develop a snake species classification system using artificial intelligence and image processing technology to address the issue of inaccurate snake identification after a bite, which is a crucial factor affecting treatment effectiveness. In Thailand, snakebite remains a significant public health problem, as 69.24% of snakebite victims are unable to identify the snake species, potentially leading to errors in treatment and the administration of inappropriate antivenom.

This project utilizes object detection algorithms, including YOLOv5 and YOLOv11, for detecting snakes in images. Various deep learning techniques, such as YOLOv5, YOLOv8, YOLOv11, MobileNetV2, and EfficientNet, are employed for snake species classification. The system is capable of distinguishing 10 snake species, both venomous and non-venomous, which helps reduce treatment risks and improve the efficiency of first aid, medical care, and decision-making in emergency situations. The system is integrated with a LINE Chatbot, a widely used platform in Thailand, allowing users to send snake images for analysis and receive preliminary information about the species, first aid guidelines, and emergency contact information.

The model evaluation results show an accuracy of 0.806 for snake detection and 0.769 for snake species classification. Although there are some limitations in certain cases, the system demonstrates potential for further development and improvement to enhance accuracy and real-world usability. Additionally, this project contributes to increasing knowledge and awareness of effective emergency management in snakebite incidents.

กิตติกรรมประกาศ

โครงงานวิทยาการคอมพิวเตอร์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับการสนับสนุนและความช่วยเหลือจากหลายฝ่าย ข้าพเจ้าขอขอบพระคุณอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. ชโลธร ชูทอง อาจารย์ที่ปรึกษาโครงงาน ที่ได้ให้คำแนะนำที่มีคุณค่าอุทิศเวลา และให้การสนับสนุนตลอดระยะเวลาการดำเนินโครงงาน นอกจากนี้ขอขอบพระคุณคณะกรรมการสอบโครงงาน ที่ได้ให้ข้อเสนอแนะและคำชี้แนะอันเป็นประโยชน์ต่อการพัฒนาโครงงานให้มีประสิทธิภาพมากยิ่งขึ้น

ขอขอบพระคุณ มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา ที่ได้มอบโอกาสและทรัพยากรทางวิชาการ รวมถึงสิ่งอำนวยความสะดวกที่จำเป็นต่อการดำเนินโครงงาน ตลอดจนการสนับสนุนด้านทุนวิจัยที่ทำให้สามารถดำเนินการศึกษาและพัฒนาโครงงานนี้ได้อย่างมีประสิทธิภาพ

นอกจากนี้ ขอขอบคุณ เพื่อนร่วมโครงการและเพื่อนร่วมชั้นเรียน ที่ได้ร่วมแลกเปลี่ยนความคิดเห็น ให้คำแนะนำ และให้การสนับสนุนทั้งทางด้านวิชาการและกำลังใจในการดำเนินโครงงานนี้ ซึ่งเป็นแรงผลักดันสำคัญที่ช่วยให้โครงงานสำเร็จลุล่วง

ท้ายที่สุด ข้าพเจ้าขอขอบพระคุณ ครอบครัว ที่คอยเป็นกำลังใจและสนับสนุนข้าพเจ้าในทุกช่วงเวลา ไม่ว่าจะเป็นด้านจิตใจหรือทรัพยากรอื่น ๆ ซึ่งมีส่วนสำคัญอย่างยิ่งต่อความสำเร็จของโครงงานนี้

ข้าพเจ้าหวังว่าโครงงานนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษาและสามารถนำไปต่อยอดเพื่อพัฒนาระบบการแบ่งส่วนภาพทางการแพทย์ในอนาคต

คณะผู้จัดทำ

นายชลภพ พอกพูล รหัสนักศึกษา 6430200132

นางสาวธนพร เกิดผล รหัสนักศึกษา 6430200281

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

ภาพ **หน้า**

สารบัญรูป (ต่อ)

ภาพ **หน้า**

สารบัญรูป (ต่อ)

ภาพ **หน้า**

สารบัญรูป (ต่อ)

ภาพ **หน้า**

สารบัญรูป (ต่อ)

ภาพ **หน้า**

สารบัญรูป (ต่อ)

ภาพ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

จากข้อมูลกองระบาดวิทยา สรุปรายงานการเฝ้าระวังโรคจากการประกอบอาชีพและสิ่งแวดล้อมโรคไม่ติดต่อ และโรคจากการป้องกันการบาดเจ็บ ประจำปี 2562 ได้สรุปข้อมูลจากรายงานองค์การอนามัยโลก ประมาณการว่าในแต่ละปีประชากรทั่วโลกถูกงูกัด 5.4 ล้านคน โดยเป็นงูพิษกัดร้อยละ 50 หรือประมาณ 2.7 ล้านคน และ เสียชีวิตเฉลี่ยปีละ 81,000 – 138,000 คน ซึ่งมีผู้พิการจำนวนไม่น้อยจากการถูกตัดอวัยวะในระหว่างการรักษา ในประเทศไทยพบว่า ปัญหาการถูกงูกัดก็มีความรุนแรงไม่ต่างกัน จากรายงานกรณีผู้ถูกงูกัดมีจำนวนรวมรวม 4,991 คน ซึ่งผู้ที่ถูกงูกัดไม่สามารถระบุชนิดได้ร้อยละ 69.24 หรือประมาณ 3,456 คน และผู้ที่ถูกงูกัดสามารถแยกชนิดได้ร้อยละ 30.76 หรือประมาณ 1,535 คน การไม่สามารถระบุชนิดของงูที่กัดได้นั้นเป็นเรื่องที่น่าเป็นห่วง เนื่องจากการรักษาจะมีความแตกต่างกันตามชนิดของงูพิษ ซึ่งส่งผลต่อการวินิจฉัยในการรักษาและการป้องกันความเสียหายที่อาจเกิดขึ้น (กรมควบคุมโรค, 2562)

สภาพภูมิประเทศในประเทศไทยเป็นป่าดงดิบมีความชื้นสูง โดยเฉพาะที่ที่ยังคงมีป่าและสภาพแวดล้อมที่อุดมสมบูรณ์ ซึ่งเอื้อต่อดำรงชีวิตอยู่ของงู ซึ่งสามารถจำแนกสายพันธุ์ของงูในแต่ละเขตพื้นที่ประเทศไทยตามรายงานดังนี้

ภาคเหนือ ได้แก่ งูเห่า งูจงอาง งูทับสมิงคลาและงูสามเหลี่ยม

ภาคกลาง ได้แก่ งูเห่า งูเขียวหางไหม้ งูจงอางและงูสามเหลี่ยม

ภาคตะวันออกเฉียงเหนือ ได้แก่ งูทับสมิงคลา

ภาคใต้ ได้แก่ งูจงอาง งูแมวเซาและงูกะปะ

งูไม่มีพิษที่สามารถพบได้บ่อย ได้แก่ งูแสงอาทิตย์ งูทางมะพร้าว งูเขียวพระอินทร์ งูเหลือม งูหลาม งูสิงหางลาย งูลายสอ งูปล้องฉนวนสร้อยเหลือง งูก้นขบ งูปี่แก้วลายแต้ม งูคุด งูงอดไทยและงูสิงบ้าน (กรมควบคุมโรค, 2562)

จากข้อมูลที่รายงานว่าร้อยละ 69.24 ของผู้ที่ถูกงูกัดในประเทศไทยไม่สามารถระบุชนิดของงูได้ จึงได้มีการพัฒนาและทดลองโมเดลหลากหลายรูปแบบเพื่อค้นหาโมเดลที่เหมาะสมที่สุดในการตรวจจับและจำแนกชนิดของงู โดยแบ่งการทำงานออกเป็น 2 ส่วนหลัก ได้แก่ 1 โมเดลสำหรับจำแนกงูออกจากวัตถุอื่น เช่น สายยาง โดยใช้ YOLOv5 และ YOLOv11 ซึ่งเป็นอัลกอริทึมที่ออกแบบมาเพื่อการตรวจจับวัตถุในภาพ (Object Detection) ที่มีประสิทธิภาพและแม่นยำสูง และ 2) โมเดลสำหรับจำแนกสายพันธุ์งูโดยเฉพาะ โดยใช้เทคนิค Deep Learning ที่หลากหลาย ได้แก่ ได้แก่ YOLOv5, YOLOv8,YOLOv11,MobileNetV2และ EfficientNet เพื่อนำมาเปรียบเทียบประสิทธิภาพและเลือกโมเดลที่มีความแม่นยำและเหมาะสมที่สุด นอกจากนี้ เพื่อให้การใช้งานโมเดลสะดวกและเข้าถึงได้ง่ายสำหรับผู้ใช้งาน จึงมีการเชื่อมต่อโมเดลที่พัฒนาแล้วกับ LINE application ซึ่งมีผู้ใช้งานในประเทศไทยมากถึง 56 ล้านคน การเชื่อมต่อนี้ช่วยให้ผู้ใช้งานสามารถรับข้อมูลเกี่ยวกับสายพันธุ์งูได้สะดวกโดยไม่ต้องดาวน์โหลดแอปพลิเคชันเพิ่มเติม

1.2 ปัญหา

1.2.1 ข้อผิดพลาดในการรักษา

ไม่สามารถระบุชนิดของงูที่กัดอาจทำให้การเลือกใช้เซรุ่มผิดประเภท ซึ่งส่งผลกระทบต่อการวินิจฉัยในการรักษาและอาจทำให้เกิดภาวะแทรกซ้อนรุนแรง

1.2.2 ความล่าช้าในการตอบสนองทางการแพทย์

การระบุชนิดของงูที่กัดได้ช้าอาจทำให้เกิดความล่าช้าในการให้การรักษา ซึ่งมีความสำคัญต่อการป้องกันอาการพิษที่รุนแรง

1.2.3 ความเสี่ยงในการเกิดผลกระทบที่รุนแรง

ความล่าช้าในการระบุชนิดของงูอาจทำให้ผู้ป่วยต้องเผชิญกับความเสี่ยงจากการได้รับการรักษาที่ไม่เหมาะสม ซึ่งอาจนำไปสู่ความเสียหายถาวร

1.2.4 ขาดการศึกษาข้อมูลที่เพียงพอ

ขาดการศึกษาและข้อมูลที่เพียงพอเกี่ยวกับชนิดของงูในพื้นที่ทำให้ชุมชนหรือเจ้าหน้าที่ไม่สามารถตอบสนองได้อย่างทันท่วงที

1.2.5 ความรู้และการฝึกอบรมไม่เพียงพอ

ผู้ที่มีความเสี่ยงอาจขาดความรู้ในการระบุชนิดของงูและการปฐมพยาบาลเบื้องต้น ซึ่งอาจส่งผลต่อความปลอดภัยของผู้ป่วย

1.2.6 การรับมือกับความกลัวและความวิตกกังวล

การไม่ทราบชนิดของงูที่กัดอาจเพิ่มความกลัว ความวิตกกังวลของผู้ป่วยและครอบครัว ซึ่งอาจส่งผลต่อความสามารถในการตัดสินใจ การดำเนินการในกรณีฉุกเฉิน

1.3 วัตถุประสงค์

1.3.1 เพื่อพัฒนาโมเดลที่สามารถตรวจจับและจำแนกงูในภาพอย่างแม่นยำ โดยเฉพาะในพื้นที่เสี่ยงที่มีการถูกงูกัดบ่อยครั้ง

1.3.2 เพื่อเปรียบเทียบประสิทธิภาพของโมเดล Deep Learning ต่าง ๆ ได้แก่ YOLOv5,YOLOv11 ในการตรวจจับและจำแนกสายพันธุ์งู YOLOv8,MobileNetV2, EfficientNet ในการจำแนกสายพันธุ์งู

1.3.3 เพื่อสร้างระบบ LINE Chatbot ที่เชื่อมต่อกับโมเดล AI เพื่อให้ข้อมูลสายพันธุ์งูได้สะดวกและรวดเร็ว

1.3.4 เพื่อให้ผู้ใช้งานสามารถเข้าถึงข้อมูลเกี่ยวกับงูได้สะดวกและรวดเร็ว

1.3.5 เพื่อลดความเสี่ยงและเพิ่มความปลอดภัยในการจัดการกับการถูกงูกัด

1.3.6 เพื่อเสริมสร้างความรู้ การตระหนักเกี่ยวกับการป้องกัน และการจัดการกับงูกัด

1.4 ขอบเขตการดำเนินงาน

1.4.1 LINE Chatbot สามารถจำแนกสายพันธุ์งูได้ 10 สายพันธุ์

1.4.2 LINE Chatbot สามารถตรวจจับงูภายในภาพที่ผู้ใช้งานส่งเข้ามาได้

1.4.3 LINE Chatbot สามารถแนะนำวิธีการปฐมพยาบาลเบื้องต้นเมื่อถูกงูกัดได้

1.4.4 LINE Chatbot สามารถให้เบอร์โทรติดต่อหน่วยงานที่เกี่ยวข้องได้

1.4.5 LINE Chatbot สามารถแนะนำเว็ปไซต์ให้ความรู้เกี่ยวกับงูได้

1.4.6 เว็ปไซต์ สามารถดูภาพที่ผู้ใช้ส่งรูปเข้ามาได้

1.5 ขั้นตอนการดำเนินงาน

การพัฒนาระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงูในครั้งนี้ มีระบบที่ต้องพัฒนา 4 ระบบได้แก่ ระบบการตรวจจับ และ จำแนกสายพันธุ์งู ระบบแชทบอทไลน์ และ เว็บไซต์ โดยผู้จัดทำได้วางแผนการดำเนินงานดังนี้

1.5.1 การจัดเตรียมข้อมูลรูปภาพ

1.ข้อมูลรูปภาพที่ใช้ในงานโครงการ ผู้จัดทำได้ดำเนินการค้นหารูปภาพจากแหล่งข้อมูลบนอินเทอร์เน็ตรวมถึงถ่ายภาพงูแต่ละสายพันธุ์

2.กำหนดป้ายกำกับ ผู้จัดทำได้ดำเนินการกำหนดป้ายกำกับ (Labeling) โดย Roboflow

3.สร้างชุดข้อมูลภาพเพิ่มเติม (Data Augmentation) ผ่านแพลตฟอร์ม Roboflow

4.แบ่งสัดส่วนรูปภาพ ผู้จัดทำได้ดำเนินการแบ่งชุดข้อมูลเพื่อใช้ในการฝึกฝนโมเดลให้เกิดการรู้จำ (Training) ตรวจสอบความถูกต้อง (Validation) และทดสอบระบบ (Testing) โดยแบ่งเป็นอัตราส่วนร้อยละ 70, 20 และ 10 ตามลำดับ

1.5.2 การพัฒนาระบบการตรวจจับงู

1.ติดตั้งอัลกอริทึม YOLOv5 และ YOLOv11

2.ฝึกฝนระบบการตรวจจับงูให้เกิดการรู้จำ (training) ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 2 วัตถุ (2 class) ได้แก่ งูและสายยาง จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย YOLOv5 และ YOLOv11

3.การวิเคราะห์ข้อมูลการทำงานของระบบการตรวจจับงูด้วย Confusion Matrix

1.5.3 การพัฒนาระบบการจำแนกสายพันธุ์งู

1.ติดตั้งอัลกอริทึม YOLOv5 ,YOLOv8 ,YOLOv11 ,MobileNetV2 และ EfficientNet

2.ทดสอบอัลกอริทึม เพื่อให้แน่ใจว่าอัลกอริทึมนั้นสามารถใช้งานได้

3.ฝึกฝนระบบการจำแนกสายพันธุ์งูให้เกิดการรู้จำ (training) ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 10 วัตถุ (10 class) จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย YOLOv5 ,YOLOv8 ,YOLOv11 ,MobileNetV2 และ EfficientNet

4.การวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งู Confusion Matrix

1.5.4 การพัฒนาLINE Chatbot

1. การวิเคราะห์ภาพงู ในการวิเคราะห์ภาพงู จะนำตัวโมเดลกับ LINE Chatbot เชื่อมต่อกัน

2. Rich Menu จะมีตัวเลือกสำหรับการตอบกลับ 4 รายการ ได้แก่ การแนะนำโมเดล การปฐมพยาบาลเบื้องต้น การแสดงหมายเลขโทรศัพท์ฉุกเฉิน และเว็บไซต์ให้ความรู้เกี่ยวกับสายพันธุ์งู

3. ข้อความตอบกลับอัตโนมัติ จะตอบกลับโดยจะตอบกลับตามคีย์เวิร์ดที่ได้กำหนดไว้

4. Card-based messages สร้าง card ใน Card-based messages เลือกประเภทของการ์ดที่จะแสดงรูปภาพ

1.5.5 การพัฒนา Web Application สำหรับดูข้อมูลการวิเคราะห์ของโมเดลจำแนกสายพันธุ์งู

1.ออกแบบ Backend โดยใช้ Fast API เพื่อรองรับการอัปโหลดข้อมูล

2.พัฒนา Frontend ด้วย vue.js โดยผู้ใช้สามารถ

- แสดงรูปภาพที่ผู้ใช้ส่งมาเพื่อแสดงและตรวจสอบภาพของงูที่ผู้ใช้ส่งมาได้สะดวกมากยิ่งขึ้น

- ลบข้อมูล

ตารางที่ 1 แผนการดำเนินงานโครงงาน

แผนงาน	ธันวาคม 2567				มกราคม 2568				กุมภาพันธ์ 2568				มีนาคม 2568			
ในแต่ละสัปดาห์	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1. รวมรวบข้อมูลรูปภาพที่ใช้ในงานโครงการ	←	-	-	-	-	-	-	→								
	←	-	-	-	-	-	→									
2. การกำหนดป้ายกำกับ			←	-	-	-	-	-	-	-	→					
			←	→			←	→								
3. การสร้างชุดข้อมูลภาพเพิ่มเติมและการแบ่งสัดส่วนรูปภาพ					←	-	-	-	-	-	-	→				
					←	→			←	→						
4. การติดตั้งและทดสอบอัลกอริทึม	←	-	-	→						←	-	-	-	-	-	→
	←	→								←	-	-	-	-	→	
5. การฝึกฝนระบบการตรวจจับงูให้เกิดการรู้จำ				←	-	-	-	-	-	-	-	-	-	-	-	→
					←	→			←	→						
6. การทดสอบการทำงานและวิเคราะห์การทำงานของระบบการตรวจจับงู					←	-	-	-	-	-	-	-	-	-	-	→
					←	→			←	→						
7. การฝึกฝนระบบการจำแนกสายพันธุ์งูให้เกิดการรู้จำ					←	-	-	-	-	-	-	-	-	-	-	→
							←	→								
8. การทดสอบการทำงานและวิเคราะห์ของระบบการจำแนกสายพันธุ์งู					←	-	-	-	-	-	-	-	-	-	-	→
							←	→								
9. การพัฒนาLINE Chatbot ในการวิเคราะห์ภาพงู							←	-	-	-	-	-	-	-	-	→
										←	→					
10. การตั้งค่าRich Menu และ ข้อความตอบกลับอัตโนมัติ	←	-	-	→												
	←	→														
11. การพัฒนาเว็บไซต์ฝั่ง backend									←	-	-	-	-	-	→	
									←	→						

แผนงาน ในแต่ละสัปดาห์	ธันวาคม 2567				มกราคม 2568				กุมภาพันธ์ 2568				มีนาคม 2568			
	1	2	3	4		1	2	3	4		1	2	3	4		1
12. การพัฒนาเว็บไซต์ฝั่ง frontend									◄----	----►					----►	
13. การจัดทำเอกสาร					◄----	----	----	----	----	----	----	----	----	----	----	----►
					◄----	----	----	----	----	----►						

หมายเหตุ ◄——► หมายถึง ระยะเวลาที่ปฏิบัติงานจริง

◄---► หมายถึง ระยะเวลาที่วางแผนไว้

1.6 ประโยชน์ที่ได้รับ

1.6.1 LINE Chatbot ที่สามารถวิเคราะห์เพื่อจำแนกสายพันธุ์ของงูได้

1.6.2 LINE Chatbot ที่สามารถให้ข้อมูลเบื้องต้นเกี่ยวกับสายพันธุ์งูที่พบได้

1.6.3 LINE Chatbot ที่สามารถแนะนำวิธีการปฐมพยาบาลเบื้องต้นเมื่อถูกงูกัดได้

1.6.4 LINE Chatbot ที่สามารถแสดงเบอร์โทรหน่วยงานที่เกี่ยวข้องเมื่อพบเจองู หรือถูกงูกัด

บทที่ 2
ทฤษฎี และเทคนิคที่เกี่ยวข้อง

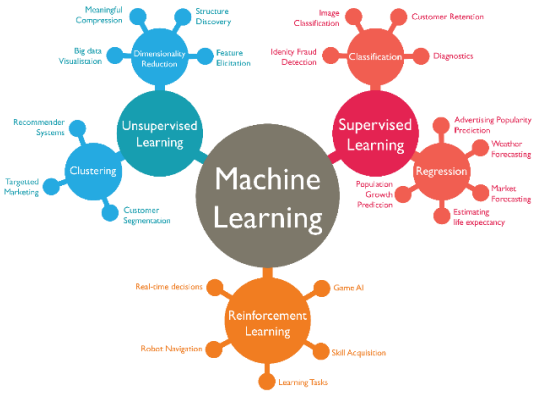
ในการจัดทำโครงงานสหกิจศึกษานี้ ผู้จัดทำได้ทำการศึกษาค้นคว้าแนวคิด ทฤษฎี เทคโนโลยี และเครื่องมือต่าง ๆ เพื่อนำมาประยุกต์ใช้ในการพัฒนาโครงงาน ประกอบด้วย

2.1 ทฤษฎีที่เกี่ยวข้อง

ในส่วนของทฤษฎีที่เกี่ยวข้อง ผู้พัฒนาได้นำเสนอทฤษฎีที่จะนำมาใช้ในการพัฒนาระบบแชทบอท ดังนี้

2.1.1 Machine Learning

Machine Learning คือ การทำให้คอมพิวเตอร์ สามารถเรียนรู้สิ่งต่าง ๆ การพัฒนาการทำงานให้ดีขึ้นได้ด้วยตัวเองจากข้อมูล และสภาพแวดล้อมที่ได้รับจากการเรียนรู้ของระบบ โดยไม่ต้องใช้มนุษย์ในการกำกับหรือเขียนโปรแกรมเพิ่มเติม และไม่ว่าในอนาคตมันจะมีข้อมูลรูปแบบใหม่ ๆ ที่เกิดขึ้นมา มนุษย์ก็ไม่จำเป็นที่จะต้องไปนั่งเขียนโปรแกรมใหม่ เพราะคอมพิวเตอร์สามารถตีความและตอบสนองได้ด้วยตัวเองข้อมูลเท่านั้น ที่เหลือเครื่องจัดการเอง โดย Machine Learning มี 3 ประเภท



ภาพที่ 1 ประเภทของ Machine Learning

ที่มา : https://pkomiske.com/talk/harvardlunchtalk2017/harvardlunchtalk2017.pdf

1. การเรียนรู้ที่มีการควบคุม (Supervised Learning) : เป็นการเรียนรู้ที่เครื่องจักรจะต้องอาศัยข้อมูลใน การฝึกฝน เปรียบเสมือนกับการเรียนการสอนของเด็กเล็ก โดยจำเป็นจะต้องอาศัยชุดข้อมูลต่าง ๆ ซึ่งประกอบไปด้วยชุดของข้อมูลและชุดผลลัพธ์ของข้อมูลที่ต้องการ โดยผลที่ได้จากการเรียนรู้คือ Machine Learning สามารถคาดคะเนผลลัพธ์ที่เกิดขึ้นจากการได้รับข้อมูล

2. การเรียนรู้ที่ไม่มีการควบคุม (Unsupervised Learning) : เป็นการเรียนรู้ที่ให้เครื่องจักรนั้นสามารถเรียนรู้ได้ด้วยตนเอง โดยไม่ต้องมีค่าเป้าหมายของแต่ละข้อมูล ซึ่งวิธีการคือมนุษย์จะเป็นผู้ใส่ข้อมูลต่าง ๆ และกำหนดสิ่งที่ต้องการจากข้อมูลเหล่านั้น โดยให้เครื่องจักรวิเคราะห์จากการจำแนกและสร้างแบบแผนจากข้อมูลที่ได้รับมา

3. การเรียนรู้เชิงเสริม (Reinforcement Learning) : เป็นการเรียนรู้สิ่งต่าง ๆ จากการลองผิดลองถูก ภายใต้แนวคิดที่ว่าจะเลือกกระทำสิ่งที่ทำให้ได้ผลลัพธ์มากที่สุด โดยทำการเรียนรู้จากการ

ลองผิดลองถูกในสถานการณ์ในอดีตหรือระบบจำลองและพยายามที่จะพัฒนาระบบการตัดสินใจของตัวเองให้ดีขึ้นเรื่อย ๆ โดยที่อาจจะพัฒนาด้วยการพยายามสร้างแบบจำลองสถานการณ์ต่าง ๆ (Disrupt Technology Venture, n.d.)

จากภาพที่ 10 จะศึกษาต่อในประเภท Supervised Learning โดยแบ่งได้ 2 ประเภท

1.1 Classification เป็นการจำแนกข้อมูลออกเป็นประเภทต่างๆ ตามที่ Label ได้กำหนดไว้ โดย Machine Learning ประเภท Classification จะให้คำตอบเป็น Label / Class เท่านั้น ไม่สามารถให้คำตอบที่นอกเหนือจาก Label ในชุดฝึกฝน มีทั้งหมด 4 ประเภท

1.1.1 การจำแนกแบบไบนารี (Binary Classification) คือ การเปรียบเทียบตัวแปรที่แบ่งเพียง 2 หมวดหมู่เท่านั้น

1.1.2 การจำแนกประเภทหลายคลาส (Multi-Class Classification) คือ การเปรียบเทียบตัวแปรที่แบ่งมากกว่า 2 หมวดหมู่

1.1.3 การจำแนกประเภทหลายเลเบล (Multi-Label Classification) คือ การเลเบล หรือติดฉลากว่าในรูปนั้น ๆ โดยไม่จำกัดแค่หมวดหมู่เดียว

1.1.4 การจำแนกแบบข้อมูลไม่เท่าเทียม (Imbalanced Classification) คือ เป็นปัญหาที่เกิดจากข้อมูลที่ไม่เท่าเทียม (Imbalanced dataset) ตัวอย่าง ข้อมูลการทุจริต โดยข้อมูลส่วนใหญ่ย่อมเป็นข้อมูลที่จัดว่า “ไม่ทุจริต” และจะมีเปอร์เซ็นต์น้อยที่จัดว่าเป็น “ทุจริต” เป็นต้น

1.2 Regression คือการนำ Input เข้าไปฝึกฝนและให้คำตอบออกมาเป็นตัวเลขเท่านั้น คำตอบไม่สามารถออกมาเป็น Label/Class ได้ มีทั้งหมด 4 ประเภท

1.2.1 Linear Regression คือ การศึกษาความสัมพันธ์ระหว่างตัวแปรตั้งแต่ 2 ตัวขึ้นไป โดยเป็นความสัมพันธ์แบบเชิงเส้น ซึ่ง 2 ตัวแปรก็คือ ประมาณการและตัวตอบสนอง

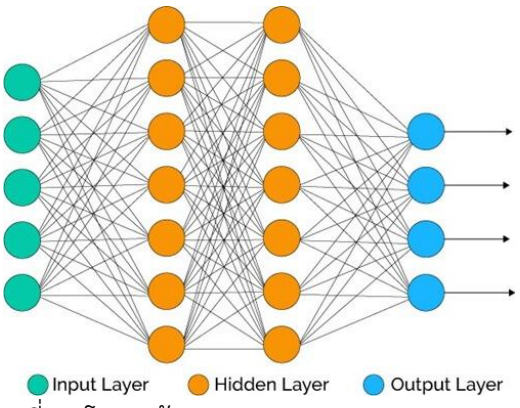
1.2.2 Logistic Regression คือ ใช้เพื่อแก้ปัญหาการทำนาย Target Variable ที่มีสอง Class หรือที่เรียกว่า Binary Classification

1.2.3 Polynomial Regression คือ เทคนิคเมื่อมีความสัมพันธ์ข้อมูลเชิงเส้นโค้ง เทคนิคนี้จะสร้างแบบจำลองค่าที่คาดหวังของตัวแปรตาม (Y) เทียบกับตัวแปรอิสระ (x) ด้วยวิธีกำลังสองน้อยที่สุด (Least-Aquares)

1.2.4 Ridge regression คือ ตัวแปรหลายตัว จะถูกใช้เมื่อเกิด Multicollinearity ซึ่งอาจทำให้วิธีกำลังสองน้อยที่สุด (Least-Squares) ไม่เอนเอียง จึงต้องเพิ่มความเอนเอียงในระดับหนึ่งในค่าประมาณ เพื่อช่วยลดข้อผิดพลาดมาตรฐาน (Standard Error) (Achieve Space, 2020)

2.1.2 Deep Learning

Deep Learning จะทำงานเลียนแบบการทำงานของโครงข่ายประสาทของมนุษย์ (Neural Networks) คือ โมเดลทางคณิตศาสตร์หรือโมเดลทางคอมพิวเตอร์สำหรับประมวลผลสารสนเทศด้วยการคำนวณแบบคอนเนคชันนิสต์ (Connectionist) แนวคิดเริ่มต้นของเทคนิคนี้ได้มาจากการศึกษาโครงข่ายไฟฟ้าชีวภาพ (Bioelectric Network) ในสมอง ซึ่งประกอบด้วย เซลล์ประสาท (Neurons) และ จุดประสานประสาท (Synapses) ตามโมเดลดังนี้ ข่ายงานประสาทเกิดจากการเชื่อมต่อระหว่างเซลล์ประสาทจนเป็นเครือข่ายที่ทำงานร่วมกัน



ภาพที่ 2 โครงสร้างของ Neural Network

ที่มา : https://www.tandfonline.com/doi/full/10.1080/10095020.2020.1718003#d1e134

จากภาพที่ 27 Neural Network จะมีส่วนประกอบ 3 อย่างดังนี้

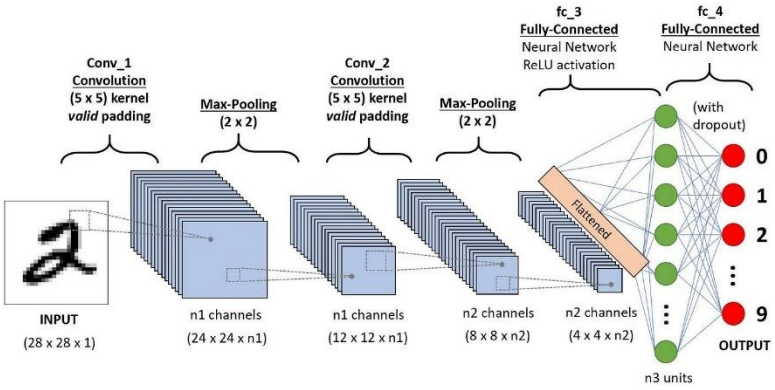
1. Neurons คือ ก้อนกลมๆ ข้างในจะต่างกันตาม Layer ที่มันอยู่ โดยถ้าเป็น Input ข้างในตัวมันก็จะมีข้อมูลที่รับมา แต่ถ้าเป็น Hidden Layer ก็จะมีสมการที่ช่วยในการคำนวณเพื่อทำนายว่าเป็นคลาสอะไร หรือคำนวณแบบถดถอย (Regression) แต่ถ้าเป็น Output ก็จะเป็นตัวที่บ่งบอกว่าเป็นคลาสอะไรนั้นเอง

2. Input Layer มีหน้าที่ในการรับข้อมูลเข้ามาในโครงข่ายประสาทโดย Input Layer จะเพียงชั้นเดียวเท่านั้นและมีหน้าส่งข้อมูลไปยังชั้นถัดไป (Hidden Layer)

3. Hidden Layer มีหน้าที่รับข้อมูลจาก Layer ก่อนหน้า Hidden Layer สามารถมีจำนวนมากกว่า 1 ได้ และโดยพื้นฐาน ถ้ายิ่งต้องการความแม่นยำที่มากขึ้นจะต้องเพิ่มจำนวนชั้นของ Hidden Layer และจำนวน Neurons ให้มากขึ้นก็จะช่วยได้

4. Output Layer มีหน้าที่รอรับค่าจาก Hidden Layer อันสุดท้าย โดยในชั้น Output นั้นแต่ละ Neurons จะมีค่าน้ำหนักของคลาสอยู่เช่น มีประเภทของ Output ทั้งหมดหมด 2 แบบคือ แมว กับ หมาเพราะฉะนั้น Output Layer จะมี Neurons 2 ตัว ตัวแรกอาจจะเป็นหมา Neurons ตัวที่สองจะเป็นแมว โดยเมื่อข้อมูลผ่าน Hidden Layer ไปสู่ Output ไปแล้ว Neurons ทั้ง 2 ตัวจะมีค่าข้างในไม่เท่ากัน

ในปัจจุบันมีอัลกอริทึมย่อย ๆ ของ Neural Network มาก โดยในโครงงานนี้จะใช้อัลกอริทึมโครงข่ายประสาทแบบคอนโวลูชัน(Convolutional Neural Network : CNN)

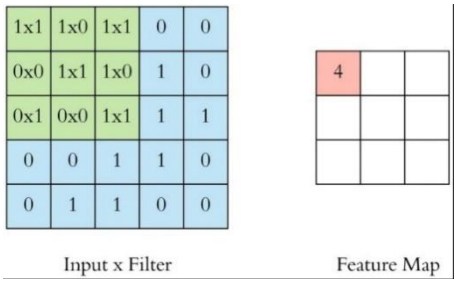


ภาพที่ 3 Convolutional Neural Network

ที่มา : https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

CNN จะใช้ Convolution Layer มาประกอบกับ Layer ชนิดอื่น เช่น Pooling Layer แล้วนำกลุ่ม Layer ดังกล่าวมาซ้อนต่อๆ กัน โดยอาจเปลี่ยน Hyperparameter บางอย่าง เช่นขนาดของ Filter Layer (ซึ่งเป็นส่วนหนึ่งของ Convolution Layer) และจำนวน Channel ของ Layer วิธีการนำเอาส่วนต่างๆ มาประกอบกันนี้ เรียกว่าเป็นโครงสร้าง (Architecture) ของ CNN ซึ่งมีหลายแบบ เช่น LeNet AlexNet VGG ResNet InceptionและNetwork เป็นต้น สถาปัตยกรรมของ CNN มีดังนี้

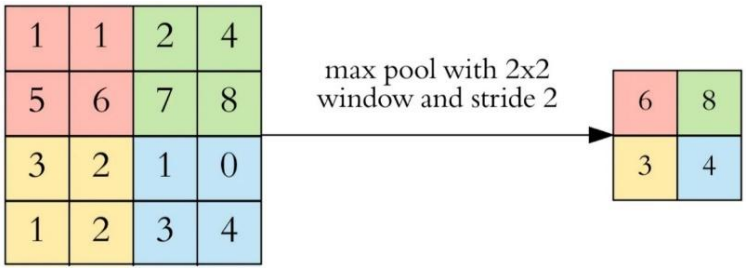
1. Convolution Layer ใช้ในการสกัดฟีเจอร์จากภาพ โดยการใช้ Filters หรือ Kernels ซึ่งเป็นชุดของน้ำหนักที่เรียนรู้ได้ (Learnable Weights) เพื่อคำนวณค่าในแต่ละพิกเซลของภาพ ผลลัพธ์ที่ได้จะเป็น Feature Map ที่แสดงลักษณะเฉพาะของภาพ เช่น ขอบ ข้อความหรือรูปทรง



ภาพที่ 4 Feature Map

ที่มา : https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5/

2. Pooling Layer ใช้เพื่อลดขนาดของ Feature Map และลดการคำนวณ โดยการเลือกค่าที่สำคัญ เช่น ค่าเฉลี่ยหรือค่าสูงสุด จากพื้นที่เล็กๆ ของ Feature Map ช่วยลดการทำงานของโมเดลและทำให้โมเดลมีความทนทานต่อการเปลี่ยนแปลงเล็กน้อยในภาพ



ภาพที่ 5 Pooling Layer

ที่มา : https://www.geeksforgeeks.org/introduction-convolution-neural-network/

3. Fully Connected Layer เป็นชั้นสุดท้ายที่ใช้ในการเชื่อมต่อฟีเจอร์ที่ได้จาก Convolutionและ Pooling Layers กับการจำแนกประเภทสุดท้ายหรือการคาดการณ์ผลลัพธ์(StackPython,2020)

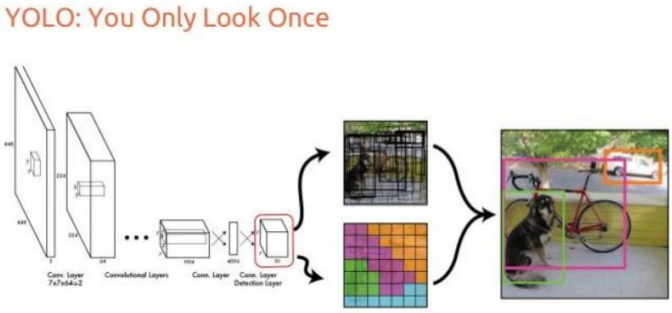
2.1.3 Object Detection

Object Detection หรือ เทคโนโลยีตรวจจับวัตถุ คือ เทคโนโลยีที่ใช้ในการระบุและค้นหาวัตถุในภาพหรือวิดีโอ โดยอาศัยเทคนิคการวิเคราะห์ข้อมูลจากกล้องวงจรปิดและการประมวลผลภาพเพื่อหาตำแหน่งและประเภทของวัตถุที่มีอยู่ เช่น มนุษย์ สัตว์ รถยนต์ อาคารและอื่น ๆ หลักการทำงานของ Object Detection ได้แก่

1. Object Classification การจำแนกประเภทของวัตถุในภาพ เช่น ว่าภาพนั้นเป็นของมนุษย์ สัตว์ หรือรถยนต์

2. Object Localization การระบุและสร้างกรอบขอบเขต (Bounding Box) รอบ ๆ วัตถุที่ตรวจจับในภาพ รวมถึงการให้ตำแหน่งที่แน่นอน

3. Detection Model โมเดลที่ใช้ในการตรวจจับวัตถุ เช่น YOLO (You Only Look Once) SSD (Single Shot MultiBox Detector) และ Faster R-CNN



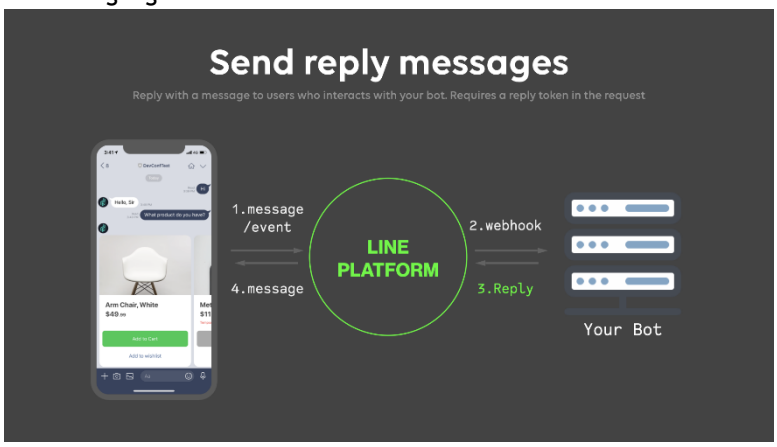
ภาพที่ 6 ภาพรวมของกระบวนการทำงาน YOLO

ที่มา : https://medium.com/towards-data-science/yolov6-next-generation-object-detection-review-and-comparison-c02e515dc45f

ในข้อที่ 3 โมเดลที่ใช้นำมาในการตรวจจับวัตถุในโครงงานนี้คือ YOLOv5 หรือ You Only Look Once Version 5 คือ Realtime Object Detection Model ที่มีความโดดเด่นเรื่องความเร็วและความถูกต้องหลักการของมันคือ ถ้ามีรูปหมาน้อย จักรยานและรถบรรทุกอยู่ด้านหลังแบบนี้ มันก็จะพยายาม Rectangle Object เหล่านั้นไว้ โดยหาจุดกึ่งกลางของแต่ละ Object แล้วค่อยครอบ

Box เอาไว้และบอกออกมาว่าสิ่งนั้นคืออะไร โดยมี Model พื้นฐานอยู่แล้วประมาณ 80 Classes ที่ถูกเทรนเอาไว้ และสามารถบอกได้ด้วยว่าความน่าจะเป็นมีเท่าไหร่ จาก Model ที่มี (Kunakorntum, 2023)

2.1.4 LINE Messaging API



ภาพที่ 7 การทำงานของ LINE Messaging API

ที่มา : https://linedevth.line.me/th/messaging-api?utm_source=chatgpt.com

การใช้ LINE Messaging API เพื่อสร้างแชทบอทนั้นจะเชื่อมโยง LINE Chatbot กับแพลตฟอร์มไลน์ ซึ่งจะทำงานอยู่เบื้องหลังบัญชี LINE Official Account โดยต้องใช้ในโหมดบอท

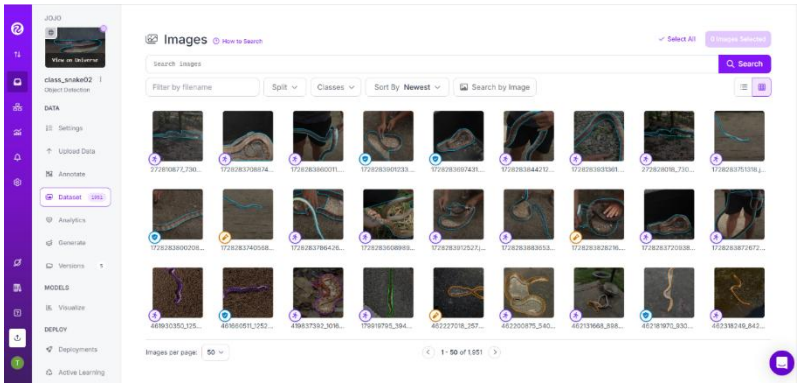
หลักการทำงานตัว LINE Messaging API นั้นจะยอมรับหรืออนุญาตให้ส่งข้อมูลกับเซิร์ฟเวอร์ LINE Chatbotและแพลตฟอร์มไลน์คำขอนั้นจะถูกส่งผ่าน HTTOS ในรูปแบบ JSON

- ผู้ใช้จะส่งข้อความไปยัง LINE Official Account
- แพลตฟอร์มไลน์ จะส่ง Webhook URL ของเซิร์ฟเวอร์LINE Chatbot
- มีการตอบกลับของเซิร์ฟเวอร์ไปยังผู้ใช้ผ่าน แพลตฟอร์มไลน์ LINE Messaging API
- Reply Messages คือการตอบกลับข้อความไปยัง ผู้ใช้ก็คือคนที่ใช้งานกับ LINE Official Account
- Push Messages คือการส่งข้อความไปยัง ผู้ใช้งาน โดยตรง เมื่อไหร่ก็ตามที่ผู้ใช้ต้องการ (LINE Developers, n.d.)

2.2 เครื่องมือที่ใช้สำหรับการพัฒนา

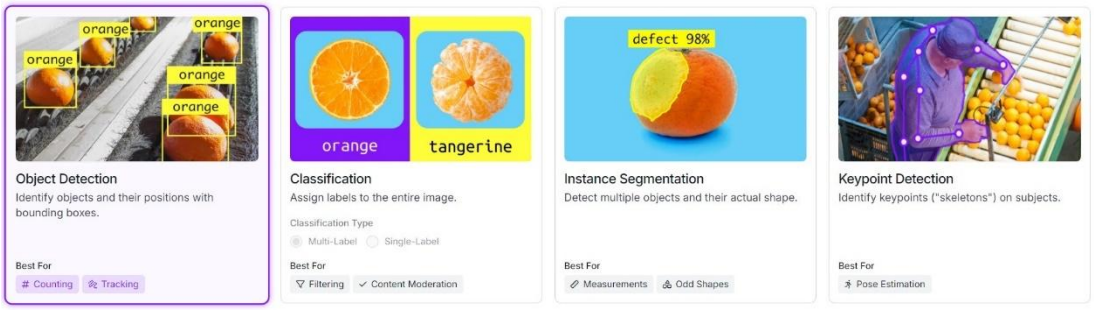
ในส่วนของเครื่องมือที่ใช้สำหรับการพัฒนา ทั้งกระบวนการประมวลผลทางภา ภาษา โปรแกรมมิ่งที่ใช้พัฒนา และ อื่นๆ ดังนี้

2.2.1 Roboflow



ภาพที่ 8 หน้าจอการทำงาน Roboflow

Roboflow เป็นเครื่องมือสำหรับการทำ Labelling ที่สามารถเตรียม Dataset ของงานในกลุ่ม Computer Vision ได้พร้อมกันหลายคน (Collaborative) นอกจากมีฟีเจอร์ในการทำงานร่วมกันแล้ว Roboflow ยังมีฟีเจอร์ในการทำ image augmentation และ tool ที่ใช้ export dataset ยังมีความสะดวก ใช้งานได้ง่าย โดย Roboflow สามารถเตรียม dataset ได้ 4 ประเภท ดังนี้



ภาพที่ 9 ประเภทของ Roboflow

1. Image Classification คือ การแยกแยะและจำแนกภาพเป็น กลุ่ม หรือ หมวดหมู่ต่างๆ

2. Object Detection คือ การสอนให้คอมพิวเตอร์ทำหน้าที่เสมือนดวงตา ที่สามารถรับรู้ได้ว่าในรูปภาพที่แสดงอยู่ มีวัตถุอะไรบ้างที่เราสนใจ และบอกตำแหน่งของวัตถุต่างๆเหล่านั้น ว่าตั้งอยู่ตรงไหนของภาพ

3. Semantic Segmentation คือ จำแนกว่า Pixel หลายล้าน Pixel แต่ละจุด คืออะไร จะได้ผลออกมาเป็นแบ่งเป็นพื้นที่สีต่าง ๆ ซึ่งแต่ละสีหมายความถึงลักษณะที่แตกต่างกัน

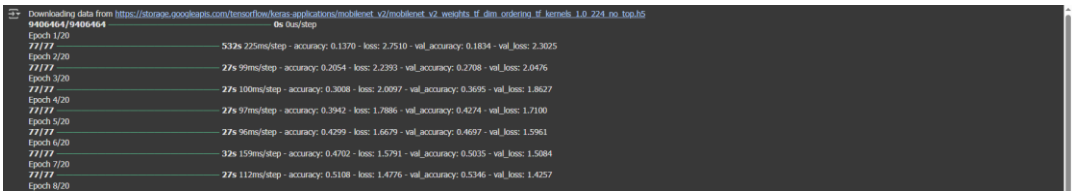
4. Key Point Detection คือ เทคนิคนี้ใช้ในการตรวจจับจุดสำคัญของวัตถุไปพร้อมกับการระบุตำแหน่งของจุดสำคัญนั้นๆ (Roboflow, n.d.).

2.2.2 YOLO

ภาพที่ 10 หน้าจอการฝึกโมเดล YOLO

YOLOv คือ Realtime Object Detection Model ที่มีความโดดเด่นเรื่องความเร็วและความถูกต้องหลักการของมันคือ ถ้ามีรูปหมาน้อย, จักรยาน และรถบรรทุกอยู่ด้านหลังแบบนี้ มันก็จะพยายาม Rectangle Object เหล่านั้นไว้ (โดยหาจุดกึ่งกลางของแต่ละ Object แล้วค่อยครอบ Box เอาไว้) และบอกออกมาว่าสิ่งนั้นคืออะไร โดยมี Model พื้นฐานอยู่แล้วประมาณ 80 classes ที่ถูกเทรนเอาไว้ และสามารถบอกได้ด้วยว่าความน่าจะเป็นมีเท่าไหร่ จาก Model ที่มี (Redmon et al, 2016)

2.2.3 MobileNetV2



ภาพที่ 11 หน้าจอการฝึกโมเดล MobileNetV2

ใน MobileNetV2 ได้มีการแนะนำโมดูลที่ดียิ่งขึ้นด้วยโครงสร้าง inverted residual และครั้งนี้ได้มีการตัด non-linearities ในเลเยอร์ที่แคบออกไป ด้วย MobileNetV2 เป็นโครงสร้างพื้นฐานสำหรับการดึงคุณลักษณะ ทำให้สามารถทำงานได้อย่างยอดเยี่ยมในงาน object detection และ semantic segmentation

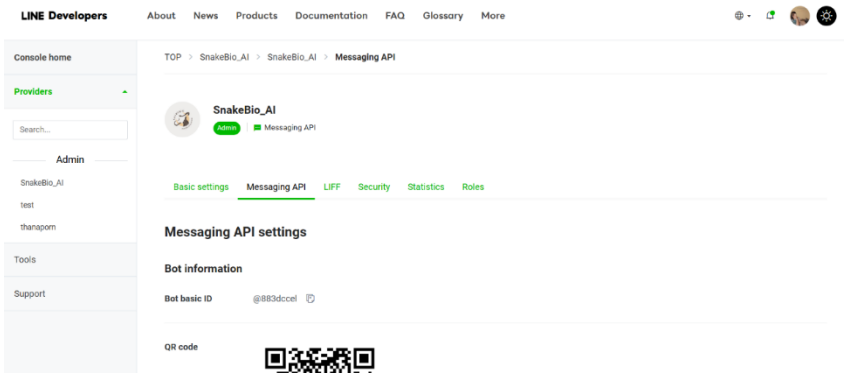
2.2.4 EfficientNet

ภาพที่ 12 หน้าจอการฝึกโมเดล EfficientNet

โครงสร้างของ EfficientNet ใช้ mobile inverted bottleneck convolution ซึ่งคล้ายกับ MobileNet V2 แต่มีขนาดใหญ่ขึ้น เนื่องจากมีจำนวน FLOPS (floating point operations per second) เพิ่มขึ้น จากโมเดลพื้นฐานนี้ เราจึงทำการ ขยายขนาด (scaling up) เพื่อสร้างตระกูลโมเดล EfficientNets

Machine อื่นๆ

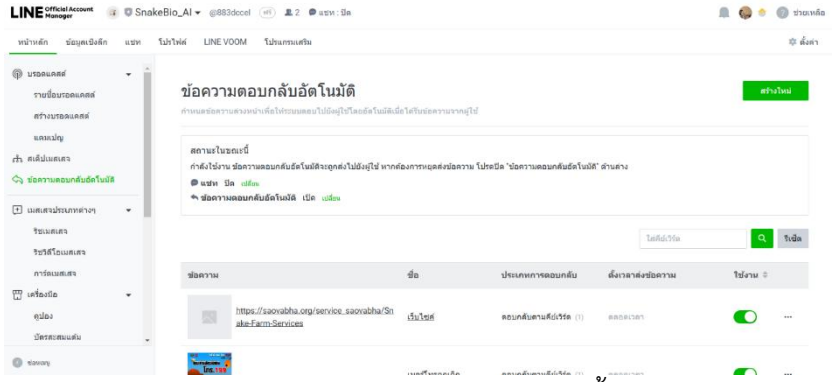
2.2.7 LINE Developer



ภาพที่ 13 หน้าจอการทำงาน LINE Developer ในการตั้งค่า Messaging API

LINE Developers คือคอนโซลทางเทคนิคสำหรับการจัดการ LINE OA ซึ่งช่วยให้บริษัทต่างๆ สามารถตั้งค่าทรัพยากรการพัฒนาต่างๆ ที่เกี่ยวข้องกับ LINE OA เช่น การตั้งค่าสำหรับข้อความพุชอัตโนมัติหรือข้อมูลการรับรองความถูกต้องที่จำเป็นสำหรับเพื่อนใหม่ LINE Developers แตกต่างจากแพลตฟอร์ม LINE OA (Official Account Manager) แพลตฟอร์ม LINE OA เป็นที่บริษัทสามารถจัดการการสื่อสารกับเพื่อนๆ ได้โดยตรง(LINE Developers, n.d.)

2.2.8 LINE Official Account

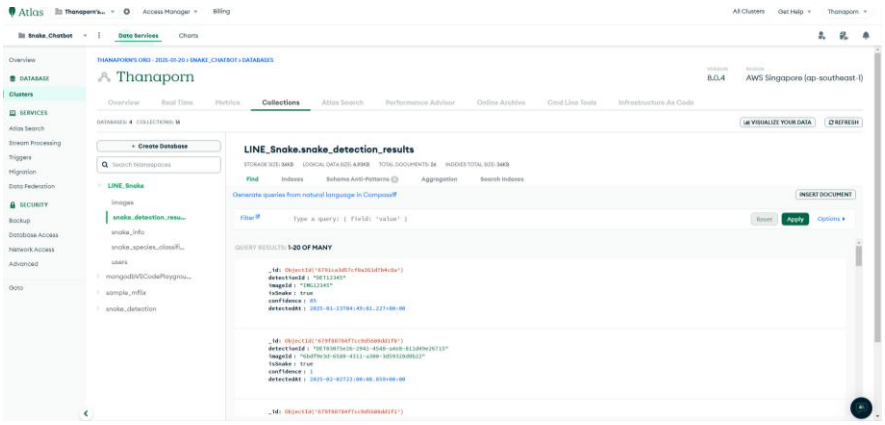


ภาพที่ 14 หน้าจอการทำงาน LINE Official Account ในการตั้งค่าการตอบกลับอัตโนมัติ

เป็นบัญชี LINE สำหรับธุรกิจ โดยสามารถทำทุกอย่างได้เหมือนกับบัญชี LINE ส่วนตัว ทั้งในส่วนของการส่งข้อความ ส่งรูปภาพ ส่งวิดีโอหรือสติกเกอร์ต่าง ๆ แต่สิ่งที่ทำให้ LINE Official Account เหมาะกับแบรนด์หรือธุรกิจนั้นคือฟังก์ชันหลากหลายที่ช่วยให้แบรนด์พูดคุยและแจ้งข่าวสารให้กับผู้ติดตามได้โดยตรงหรือแม้กระทั่งการออกแบบคอนเทนต์ให้ตรงกับกลุ่มเป้าหมาย โดยในโครงงานนี้ได้มีการนำฟีเจอร์บ้างอย่างมาใช้ได้แก่

1. Rich Menu คือ เมื่อผู้ใช้งานกดเข้ามาในแชทไลน์ ก็จะพบกับเมนูลัดด้านล่าง เรียกว่า LINE Rich Menu ซึ่งเป็นระบบที่สามารถตั้งค่าปุ่มให้ผู้ใช้งานสามารถเข้าถึงข้อมูลต่างๆได้ทันที โดยสามารถได้สูงสุดถึง 6 เมนู และสามารถกำหนดได้ว่า ต้องการให้ผู้ใช้งานกดแล้วไปที่เมนูหรือช่องทางไหน เช่น ปุ่มสำหรับสอบถามข้อมูล หรือ กดเลือกดูสินค้า ได้ผ่านตรง LINE Rich Menu ได้เลย (Sellsuki,2021)

2.2.9 MongoDB



ภาพที่ 15 หน้าจอการทำงาน Mongo DB

MongoDB เป็นระบบฐานข้อมูลประเภท NoSQL (Not Only SQL) ที่มีความแตกต่างจากฐานข้อมูลประเภท SQL ซึ่งใช้รูปแบบการจัดเก็บข้อมูลแบบตาราง (Relational Database) โดย NoSQL มีความยืดหยุ่นในการจัดการข้อมูลมากกว่า MongoDB ใช้รูปแบบการจัดเก็บข้อมูลแบบ Document-based (รูปแบบเอกสาร) ซึ่งมีการจัดโครงสร้างข้อมูลเป็นลำดับชั้น 3 ระดับ ดังนี้

1. Database (ฐานข้อมูล) : เป็นลำดับชั้นที่ใหญ่ที่สุด ทำหน้าที่เป็นคลังข้อมูลที่ใช้สำหรับแบ่งข้อมูลออกเป็นส่วน ๆ ฐานข้อมูลใน MongoDB สามารถใช้เพื่อจัดเก็บข้อมูลของระบบทั้งหมด และยังสามารถกำหนดสิทธิ์การเข้าถึงข้อมูลได้ ไม่ว่าจะเป็นสิทธิ์ของผู้ใช้หรือซอฟต์แวร์ที่ต้องการเข้าถึงฐานข้อมูลดังกล่าว
2. Collection (แฟ้มข้อมูล) : เป็นลำดับชั้นรองลงมา ทำหน้าที่คล้ายกับโฟลเดอร์ที่รวบรวมเอกสารประเภทเดียวกันไว้ด้วยกัน การจัดเก็บข้อมูลในรูปแบบนี้ช่วยให้เกิดความเป็นระเบียบและง่ายต่อการเข้าถึงและบริหารจัดการ
3. Document (เอกสาร) : เป็นหน่วยข้อมูลที่เล็กที่สุดใน MongoDB แต่ละเอกสารจะถูกเก็บอยู่ในแฟ้มข้อมูล (Collection) โดยเอกสารแต่ละฉบับมีลักษณะเป็นโครงสร้าง JSON ที่เก็บข้อมูลต่าง ๆ ไว้ และแต่ละเอกสารจะมีหมายเลขระบุเฉพาะที่เรียกว่า ObjectId เพื่อใช้ในการอ้างอิงหรือเรียกดูข้อมูล เอกสารสามารถถูกเรียกใช้ทีละชุดหรือทีละเอกสารผ่านการกรองข้อมูล (filter) ตามเงื่อนไขที่ต้องการ

MongoDB ได้รับการออกแบบมาให้สามารถจัดการกับข้อมูลที่มีโครงสร้างไม่แน่นอน (Unstructured Data) ได้อย่างมีประสิทธิภาพ เหมาะสำหรับงานที่ต้องการความยืดหยุ่นสูง เช่น การจัดเก็บข้อมูลแอปพลิเคชันที่เปลี่ยนแปลงบ่อยหรือการจัดการข้อมูลที่มีลักษณะแตกต่างกันในชุดข้อมูลเดียวกัน(MongoDB,n.d)

2.3 งานวิจัยที่เกี่ยวข้อง

2.3.1 แอปพลิเคชันจำแนกสายพันธุ์งูที่พบในไทยด้วย Deep Learnin

งานวิจัยนี้นำเสนอการพัฒนาแอปพลิเคชันสำหรับจำแนกสายพันธุ์งูในประเทศไทยโดยใช้เทคโนโลยี Deep Learning เพื่อช่วยให้ผู้ใช้งานสามารถระบุสายพันธุ์งูได้อย่างถูกต้อง พร้อมให้ข้อมูลเกี่ยวกับลักษณะและวิธีการปฐมพยาบาลเมื่อถูกงูกัด โมเดลที่ใช้ในการวิจัยประกอบด้วย MobileNetV2 และ InceptionV3 โดย MobileNetV2 ถูกเลือกเนื่องจากมีขนาดโมเดลเล็กและเหมาะสมกับการใช้งานบนอุปกรณ์มือถือ

ชุดข้อมูลที่ใช้ประกอบด้วยภาพงู 20 สายพันธุ์ จำนวน 4,069 รูป โดยมีการใช้เทคนิค Data Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล และประเมินผลด้วยค่าความแม่นยำ (Accuracy), ค่าความระลึก (Recall), และ F1-Score ผลการทดลองแสดงให้เห็นว่า MobileNetV2 มีค่า Accuracy สูงสุด 66% ซึ่งสูงกว่า InceptionV3 ที่มีค่า Accuracy 65%

อ้างอิงจาก : http://digital.csmsu.net:8080/library/handle/123456789/193ฤ

2.3.2 YODE-FEIM: An Enhanced YOLOv5s Algorithm for Snake Detection in Wild Environments

วัตถุประสงค์เพื่อพัฒนาอัลกอริทึม YOLOv5s สำหรับการตรวจจับงูในสภาพแวดล้อมธรรมชาติโดยปรับปรุงด้วยการใช้ FasterNet block และ C3 module รวมถึงกลไก EMA attention และ RT-DETR detection head เพื่อเร่งการเรียนรู้และความแม่นยำ โมเดลถูกทดสอบ กับชุดข้อมูล ChineseSnake และให้ผลลัพธ์ความแม่นยำถึง 92.7 %

ผลการทดลองพบว่าโมเดลที่ปรับปรุงนี้สามารถตรวจจับงูได้อย่างแม่นยำที่ 92.7% และค่าความแม่นยำเฉลี่ย (mAP) ที่ 90 % โดยงานวิจัยนี้ชี้ให้เห็นถึงความสามารถในการใช้งานโมเดลในสภาพแวดล้อมธรรมชาติอย่างมีประสิทธิภาพ

อ้างอิงจาก : https://ijsea.com/archive/volume13/issue10/IJSEA13101016.pdf

2.3.3 Object Detection and Classification Based on YOLO-V5 with Improved Maritime Dataset

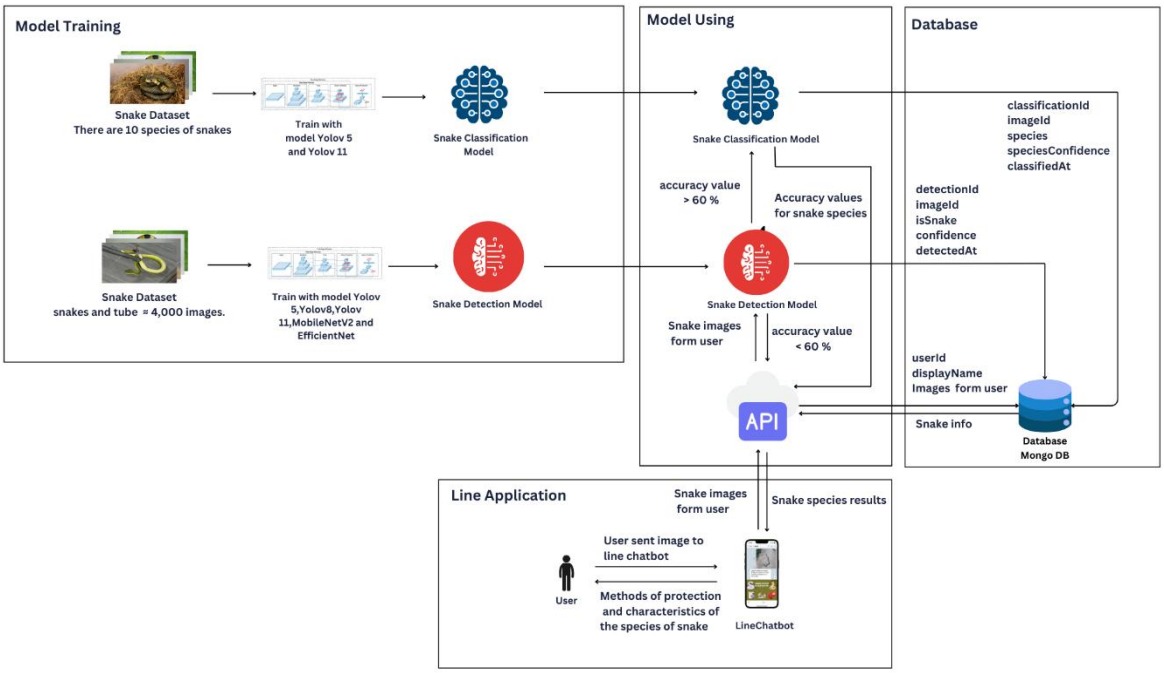
การพัฒนาชุดข้อมูล Singapore Maritime Dataset (SMD) ซึ่งเป็นชุดข้อมูลวิดีโอที่มีการติดป้ายกำกับ เพื่อใช้ในงาน Deep Neural Networks (DNN) สำหรับการตรวจจับและจำแนกวัตถุในสภาพแวดล้อมทางทะเล งานวิจัยปรับปรุงชุดข้อมูลให้มีความแม่นยำสูงขึ้นและเรียกว่า SMD-Plus โดยแก้ไขปัญหา เช่น การระบุขอบเขตวัตถุที่คลาดเคลื่อน และปัญหาความไม่สมดุลของคลาส

โดยแก้ไขการระบุป้ายกำกับและขอบเขตของวัตถุให้แม่นยำยิ่งขึ้น ผสานคลาสที่มีลักษณะคล้ายกัน เช่น รวมคลาส "Speed Boat" และ "Boat" นอกจากนี้ ยังได้นำเทคนิค Copy & Paste Augmentation แบบออนไลน์มาใช้ เพื่อเพิ่มจำนวนข้อมูลในคลาสที่มีตัวอย่างน้อย เพื่อลดปัญหาความไม่สมดุลของคลาส และใช้เทคนิค Mix-up เพื่อผสมภาพและป้ายกำกับ ช่วยแก้ปัญหาป้ายกำกับที่คลาดเคลื่อน การประเมินผลดำเนินการด้วยโมเดลพื้นฐาน YOLO-V5 โดยเปรียบเทียบประสิทธิภาพกับ YOLO-V4 บน SMD และ SMD-Plus ซึ่งแสดงให้เห็นถึงความสามารถในการปรับปรุงความแม่นยำในการตรวจจับและจำแนกวัตถุอย่างมีนัยสำคัญ

อ้างอิงจาก : https://www.mdpi.com/2077-1312/10/3/377

บทที่ 3
การออกแบบและพัฒนาระบบ

3.1 System Overview



ภาพที่ 16 แสดงภาพรวมของการทำงานระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู

จากภาพที่ 16 เป็นภาพรวมในการดำเนินงานของ ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู โดยจะแบ่งเป็น 2 ส่วน โดยแต่ละส่วนมีรายละเอียดดังต่อไปนี้

1. Model Training

จากภาพในส่วนของ Model Training เป็นขั้นตอนการพัฒนา Model ที่ใช้สำหรับจำแนกสายพันธุ์งูเพื่อนำไปใช้ใน LINE Application โดยก่อนที่จะพัฒนาโมเดลดังกล่าว จำเป็นต้องจัดเตรียม Dataset ของงูและสายยาง ซึ่งได้รวบรวมข้อมูลจากหลากหลายเว็บไซต์ หลังจากนั้นจึงนำชุดข้อมูลนี้ไปใช้ในการฝึกโมเดล (Train Model) ด้วยอัลกอริทึม YOLOv5 และ YOLOv11 เพื่อทำการตรวจสอบว่าในภาพมีงูอยู่หรือไม่ และระบุตำแหน่งของงูภายในภาพ

จัดเตรียม Dataset ของงูแต่ละสายพันธุ์ซึ่งได้รวบรวมข้อมูลจากหลากหลายเว็บไซต์ หลังจากนั้นจึงนำชุดข้อมูลนี้ไปใช้ในการฝึกโมเดล (Train Model) ด้วยอัลกอริทึม YOLOv5,YOLOv11 เพื่อจำแนกสายพันธุ์ของงูในภาพที่ตรวจพบ

2. Model Using

จากภาพในส่วนของ Model Training เป็นขั้นตอนการใช้งานในการตรวจจับและจำแนกสายพันธุ์งูในภาพที่ผู้ใช้ส่งเข้ามาใน LINE Chatbot ซึ่งจะประกอบไปด้วย 2 ขั้นตอนหลัก ได้แก่

1.การใช้งาน Model Snake Detection (การตรวจจับงู) ในขั้นตอนนี้, เมื่อผู้ใช้ส่งภาพของงูเข้ามาผ่าน LINE Chatbot ระบบจะนำภาพนั้นไปประมวลผลด้วยโมเดล Snake Detection ที่ใช้ YOLOv5

และ YOLOv11 เพื่อตรวจสอบว่าในภาพมีงูอยู่หรือไม่ ถ้าภาพนั้นมีงูและมีความมั่นใจ (confidence) มากกว่า 60% ระบบจะส่งข้อมูลการตรวจจับไปยังโมเดลถัดไป 2.การใช้งาน Model Snake Classification (การจำแนกสายพันธุ์งู)

เมื่อโมเดลตรวจจับงูในภาพได้แล้ว โมเดล Snake Classification ที่ฝึกมาแล้วจะทำการจำแนกว่างูในภาพนั้นเป็นสายพันธุ์ใด หลังจากการจำแนกสายพันธุ์เสร็จสิ้น ผลลัพธ์จะถูกส่งไปยัง ฐานข้อมูล เพื่อเก็บข้อมูลสำหรับการติดตามและการใช้งานในอนาคต และแสดงผลลัพธ์ให้กับผู้ใช้ทราบ โดยจะมีข้อมูลเพิ่มเติมเกี่ยวกับสายพันธุ์งูและวิธีการป้องกันงูชนิดนั้น ๆ

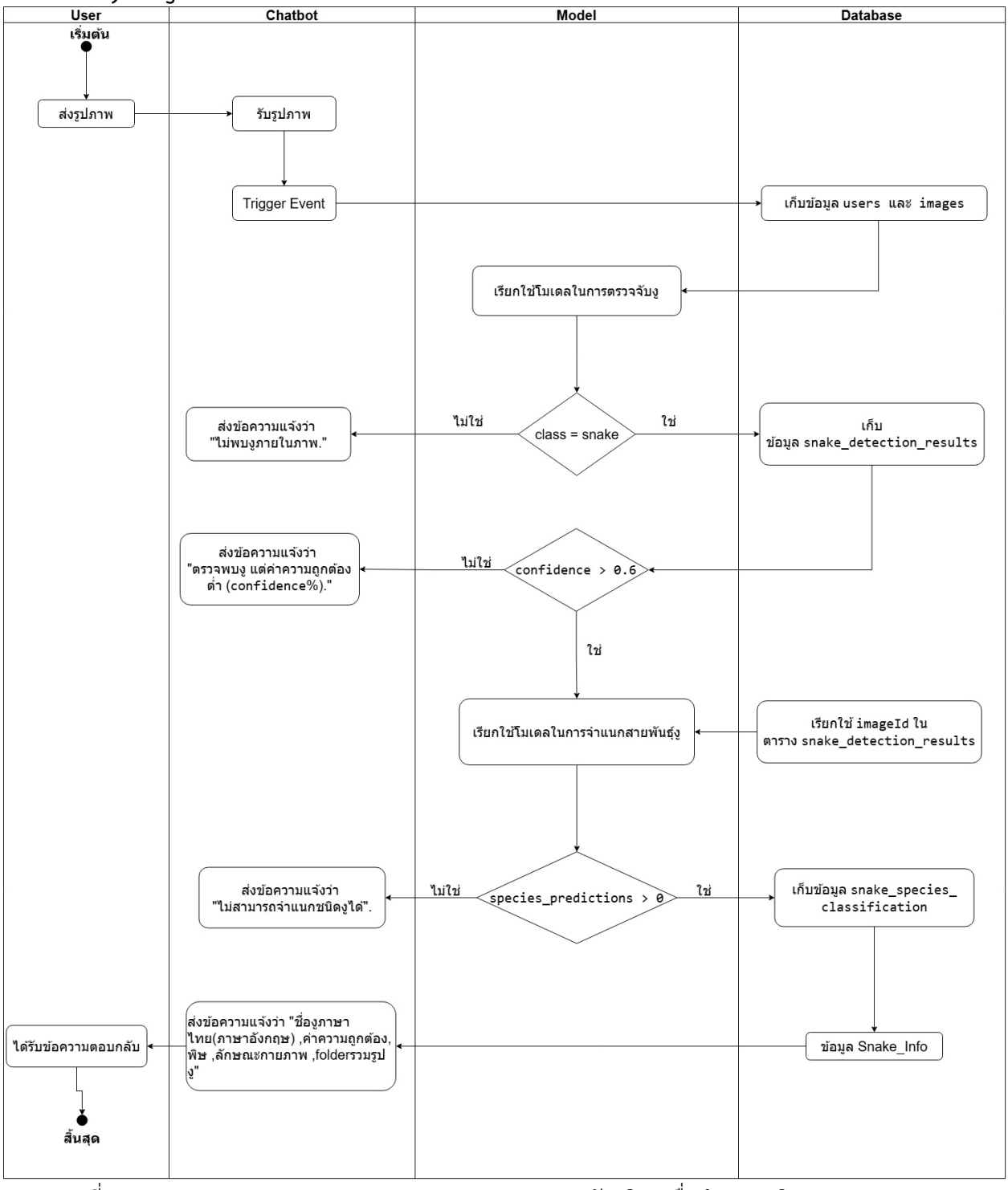
3.Database

ในระบบ Snake Detection และ Classification ข้อมูลที่ได้จากการตรวจจับและจำแนกงูจะถูกบันทึกไว้ในฐานข้อมูลเพื่อให้สามารถเข้าถึงและใช้งานได้ในอนาคต ฐานข้อมูลนี้จะช่วยจัดระเบียบข้อมูลและทำให้สามารถตรวจสอบประวัติการใช้งานได้อย่างมีประสิทธิภาพ

4.LINE Application

จากภาพในส่วนของ LINE Application โดยขั้นตอนในการใช้งาน Model Using คือผู้ใช้งานต้องถ่ายรูปภาพงูหรือเลือกรูปภาพงูจากแกลเลอรี่เมื่อผู้ใช้งานส่งภาพมา ระบบจะส่งภาพดังกล่าวไปยัง API ที่เชื่อมกับ Model Using จะทำการประมวลผลภาพโดย ตรวจสอบว่ามีงูในภาพหรือไม่ หากพบงูในภาพ จะทำการจำแนกสายพันธุ์ของงู และจะส่งผลลัพธ์ที่ได้จัดเก็บผลลัพธ์ในฐานข้อมูล MongoDBและส่งกลับมาที่ LINE Chatbot มีการแสดงข้อมูลได้แก่ชื่อสายพันธุ์งูที่พบ พิษ ลักษณะทางกายภาพ และ ลิงค์ไฟล์ภาพรวบรวมรูปงู

3.2 Activity diagram



ภาพที่ 17 แผนภาพ Activity diagram ของ ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู

อธิบาย Activity diagram ของ ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู

1. User

ผู้ใช้เริ่มต้นกระบวนการโดยการ ส่งรูปภาพ ไปยัง Chatbot

2. Chatbot

Chatbot รับรูปภาพ และทำการ Trigger Event เพื่อเรียกใช้โมเดลสำหรับการตรวจจับงู

3. Model

3.1 เก็บข้อมูล ของผู้ใช้และรูปภาพลงใน Database

3.2 เรียกใช้โมเดลตรวจจับงู เพื่อตรวจสอบว่าภาพมีงูหรือไม่

3.3 หาก class ≠ snake Chatbot แจ้งกลับว่า "ไม่พบงูในภาพ"

หาก class = snake บันทึกผลลัพธ์ในตาราง snake_detection_results

3.4 การตรวจสอบความแม่นยำ (Confidence Score)

- หาก confidence ≤ 0.6 แจ้งกลับว่า "ตรวจพบงู แต่ค่าความถูกต้องต่ำ (confidence%)"

- หาก confidence > 0.6 โมเดลจะทำการจำแนกสายพันธุ์งู

3.5 การจำแนกสายพันธุ์งู

- หากไม่มีการจำแนกสายพันธุ์ได้ (species_predictions ≤ 0) แจ้งกลับว่า "ไม่สามารถจำแนกสายพันธุ์ได้"

- หากสามารถจำแนกได้ (species_predictions > 0) บันทึกข้อมูลลงใน snake_species_classification และดึงข้อมูลสายพันธุ์จาก Snake_Info

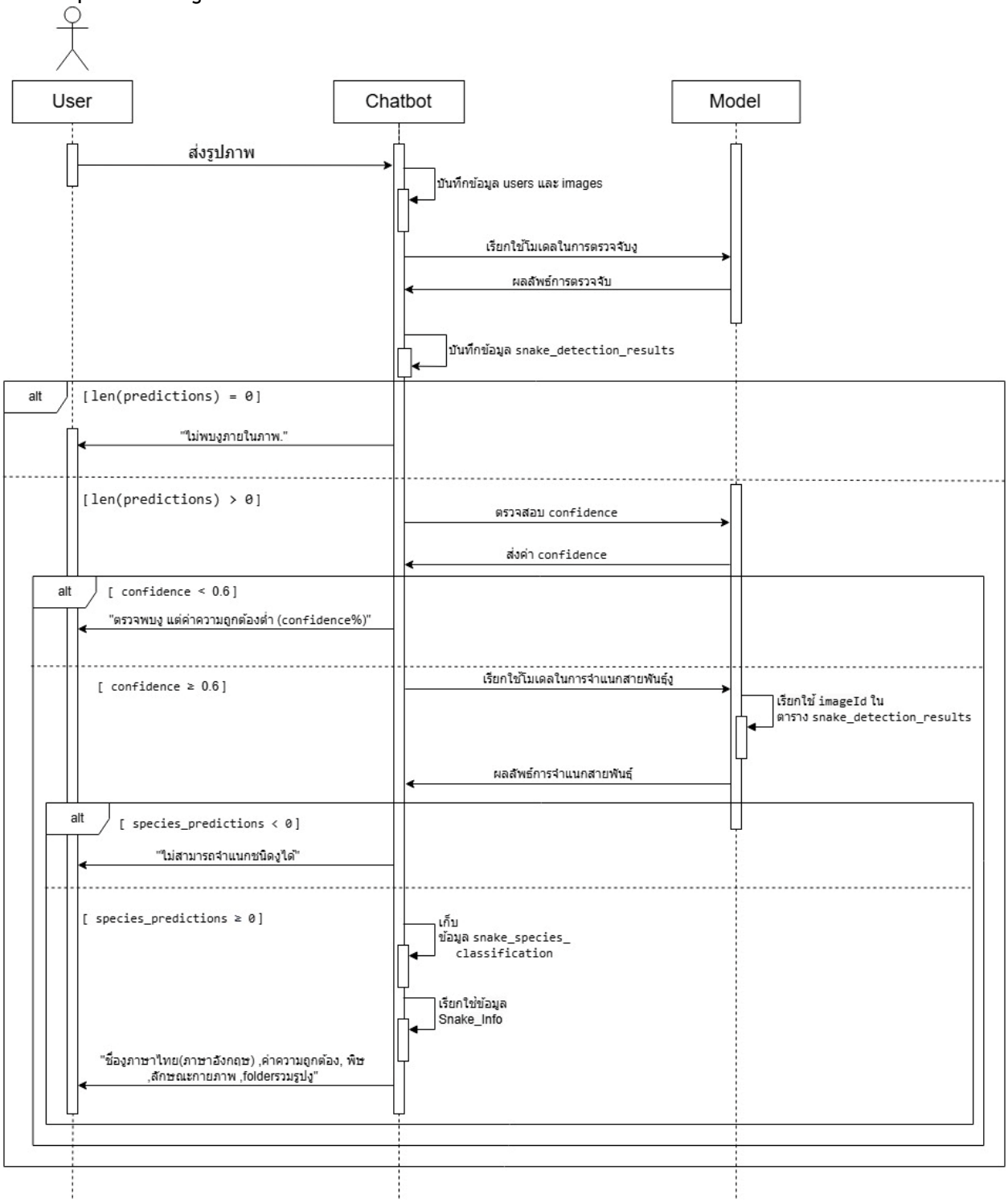
4. Chatbot

ส่งข้อมูลกลับไปยังผู้ใช้ รวมถึงชื่อสายพันธุ์งู (ไทย & อังกฤษ) ค่าความถูกต้องของการจำแนกลักษณะของงู และลิงก์ไปยังแหล่งข้อมูลเพิ่มเติม (เช่น Folder รูปงู)

5. User

สิ้นสุดกระบวนการ

3.3 Sequence diagram



ภาพที่ 18 แผนภาพ Sequence diagram ของ ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู

3.4 Data Dictionary

ตารางที่ 2 Data Dictionary(users)

Data Dictionary(users)			
Field Name	Data type	Description	Example
userId	ObjectId	รหัสประจำตัวผู้ใช้แบบเฉพาะเจาะจง	Ue2078ab699f628fe98459ba25e47bfe7
displayName	String	ชื่อที่แสดงของผู้ใช้งาน	P
createdAt	Date	วันที่และเวลาที่ผู้ใช้ถูกสร้างข้อมูล	09 February 2025, 01:30

ตารางที่ 3 Data Dictionary(images)

Data Dictionary(images)			
Field Name	Data type	Description	Example
imageId	ObjectId	รหัสประจำตัวรูปภาพแบบเฉพาะเจาะจง	c8d981a9-b23d-44a3-a9b3-137cd2f49cfc
userId	String	รหัสประจำตัวผู้ใช้ที่อัปโหลดรูปภาพ	Ue2078ab699f628fe98459ba25e47bfe7
imageUrl	String	URL ของรูปภาพ	Binary.createFromBase64('/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAx...', 0)
uploadedAt	Date	วันที่และเวลาที่รูปภาพถูกอัปโหลด	09 February 2025, 01:30

ตารางที่ 4 Data Dictionary(snake_detection_results)

Data Dictionary(snake_detection_results)			
Field Name	Data type	Description	Example
detectionId	ObjectId	รหัสประจำตัวผลการตรวจจับ	DETa832614a-69c6-4719-b44e-1a0b676e1e1e
imageId	String	รหัสประจำตัวรูปภาพที่ทำการตรวจจับ	c8d981a9-b23d-44a3-a9b3-137cd2f49cfc
isSnake	Boolean	ผลลัพธ์ว่าคือ "งู" หรือไม่ (true = งู)	true
confidence	Integer	ความมั่นใจของโมเดลในผลลัพธ์ (%)	0.6392170190811157
detectedAt	Date	วันที่และเวลาที่ทำการตรวจจับ	09 February 2025, 01:30

ตารางที่ 5 ตารางที่ 15 Data Dictionary(snake_species_classification)

Data Dictionary(snake_species_classification)			
Field Name	Data type	Description	Example
classificationId	ObjectId	รหัสประจำตัวผลการจำแนกสายพันธุ์	CLASS80b2cf30-9741-4538-8186-7ab0342414ec
imageId	String	รหัสประจำตัวรูปภาพที่ทำการจำแนกสายพันธุ์	c8d981a9-b23d-44a3-a9b3-137cd2f49cfc
species	String	ชื่อสายพันธุ์ของงูที่จำแนกได้	2
speciesConfidence	Integer	ความมั่นใจของโมเดลในผลลัพธ์ (%)	0.6353339552879333
classifiedAt	Date	วันที่และเวลาที่ทำการจำแนกสายพันธุ์	09 February 2025, 01:30

ตารางที่ 6 Data Dictionary(Information Snake)

Data Dictionary(Information Snake)			
Field Name	Data type	Description	Example
Class_id	Integer	รหัสที่ใช้แทนแต่ละสายพันธุ์งู (ID)	0
Species_name	String	ชื่อของสายพันธุ์งูในภาษาไทย	งูสามเหลี่ยม
Eng_name	String	ชื่อของสายพันธุ์งูในภาษาอังกฤษ	Banded Krait
Poison_info	String	ข้อมูลเกี่ยวกับประเภทของพิษที่งูมี หรือระบุว่าไม่มีพิษ	ไม่มีพิษ
Additional_info	String	ข้อมูลลักษณะของงูเพิ่มเติม เช่น สี ลาย ตัวอย่างลักษณะทางกายภาพ	ลำตัวมีสีดำสลับเหลืองเป็นปล้องขนาดใกล้เคียงกันตลอดตัวทั้งส่วนบนและส่วนท้อง...
Folder_link	URL	ลิงก์ไปยังโฟลเดอร์ที่เก็บรูปภาพของงูแต่ละสายพันธุ์	https://drive.google.com/drive/folders/1zBEzb31UfDkYXDU14VPjyhnnRAv1AnAJ?usp=sharing

ตารางที่ 7 Data Dictionary(Comment Collection)

Data Dictionary(Comment Collection)			
Field Name	Data type	Description	Example
userId	ObjectId	รหัสประจำตัวผู้ใช้แบบเฉพาะเจาะจง	Ue2078ab699f628fe98459ba25e47bfe7
comment	String	การบอกสายพันธุ์ที่ถูกต้อง	แก้ไขสายพันธุ์ของงู
commentAt	Date	วันที่และเวลาที่ส่งข้อความเข้ามา	21 March 2025, 03:17

3.5 การจัดเตรียมข้อมูลรูปภาพ

1.ข้อมูลรูปภาพที่ใช้ในงานโครงการ

ผู้จัดทำได้ดำเนินการค้นหารูปภาพจากแหล่งข้อมูลบนอินเทอร์เน็ต รวมถึงถ่ายภาพงูแต่ละสายพันธุ์โดยตรง ณ สวนงู สถานเสาวภา สภากาชาดไทย เพื่อให้ได้ข้อมูลที่หลากหลายและครอบคลุมมากที่สุดรูปภาพที่รวบรวมจากแหล่งต่าง ๆ ถูกจัดเก็บและจำแนกตามสายพันธุ์ เพื่อนำไปใช้ในการฝึกโมเดล โดยรายละเอียดของรูปภาพที่รวบรวมสามารถดูได้ใน ตารางที่ 7 และ 8

ตารางที่ 8 ข้อมูลรูปภาพที่รวบรวมใช้สำหรับโมเดลในการตรวจจับงู

ลำดับ	ประเภทภาพ	จำนวน(ภาพ)
1	งู	1653
2	สายยาง	1152
	รวม	2805

ตารางที่ 9 ข้อมูลรูปภาพที่รวบรวมใช้สำหรับโมเดลในการจำแนกสายพันธุ์งู

ลำดับ	ชื่อสายพันธุ์งู	จำนวน(ภาพ)
1	งูสามเหลี่ยม	253
2	งูสิงหางลาย	353
3	งูเขียวพระอินทร์	309
4	งูจงอาง	274
5	งูทับสมิงคลา	298
6	งูกะปะ	253
7	งูเห่า	300
8	งูเขียวหางไหม้	309
9	งูเหลือม	281
10	งูแมวเซา	235
	รวม	2865

2.การกำหนดป้ายกำกับ

การกำหนดป้ายกำกับ (Labeling) เป็นขั้นตอนสำคัญในกระบวนการเตรียมข้อมูลสำหรับการฝึกโมเดลปัญญาประดิษฐ์ (AI) ด้านการรู้จำภาพ (Image Recognition) โดย Roboflow เป็นแพลตฟอร์มที่ใช้งานผ่านเว็บเบราว์เซอร์ ซึ่งช่วยให้กระบวนการนี้มีประสิทธิภาพและสะดวกยิ่งขึ้น

Roboflow มีเครื่องมือที่เรียกว่า Smart Polygon ซึ่งช่วยในการกำหนดบริเวณของวัตถุในภาพได้อย่างแม่นยำ โดยเฉพาะการจำแนก งู หรือ สายยาง และระบุสายพันธุ์ของงู ได้อย่างละเอียด ซึ่งช่วยเพิ่มความแม่นยำในการฝึกโมเดลและลดข้อผิดพลาดในการจำแนกประเภทของวัตถุ โดยรายละเอียดของ Label ที่ใช้สามารถดูได้ใน ตารางที่ 9 และ 10 ซึ่งแสดงการแบ่งประเภทของข้อมูลสำหรับโมเดลที่ใช้ในการตรวจจับงู และ โมเดลที่ใช้ในการตรวจจับงู

ตารางที่ 10 รายการป้ายกำกับสำหรับการฝึกโมเดลในการตรวจจับงู

ลำดับ	ประเภทภาพ	Class name
1	งู	snake
2	สายยาง	tube

ตารางที่ 11 รายการป้ายกำกับสำหรับการฝึกโมเดลในการจำแนกสายพันธุ์งู

ลำดับ	ชื่อสายพันธุ์งู	Class name
1	งูสามเหลี่ยม	Banded Krait
2	งูสิงหางลาย	Banded Rat Snake
3	งูเขียวพระอินทร์	Golden Flying Snake
4	งูจงอาง	King cobra
5	งูทับสมิงคลา	Malayan Krait
6	งูกะปะ	Malayan Pitviper
7	งูเห่า	Monocellate Cobra
8	งูเขียวหางไหม้	Pope-s Pit Viper
9	งูเหลือม	Reticulated Python
10	งูแมวเซา	Siamese Russell-s Viper

3.การสร้างชุดข้อมูลภาพเพิ่มเติม (Data Augmentation)

ในขั้นตอนนี้ได้มีการเปลี่ยนขนาดภาพให้เป็น 640 x 640 pixelsด้วยการใช้เทคนิค Data Augmentation ดังแสดงในตารางที่ 5 และ 6 ผ่านแพลตฟอร์ม Roboflow ซึ่งเป็นแพลตฟอร์มที่ใช้งานผ่านเว็บบราวเซอร์ ชุดข้อมูลที่ใช้สำหรับฝึกฝนโมเดลมีทั้ง 2 โมเดลได้แก่ 1.ชุดข้อมูลสำหรับฝึกฝนโมเดลในการตรวจจับงู จำนวน 4327 ภาพ 2.ชุดข้อมูลสำหรับฝึกฝนโมเดลในการจำแนกสายพันธุ์งู จำนวน 3516 ภาพ สามารถดูได้ในตารางที่ 11 และ 12

ตารางที่ 12 การตั้งค่า Data Augmentation สำหรับโมเดลในการตรวจจับงู

ลำดับ	เทคนิคที่ใช้	การตั้งค่า
1	90° Rotate	Clockwise, Counter-Clockwise, Upside Down
2	Grayscale	15%
3	Hue	Between -15° and +15°
4	Noise	Up to 1.01% of pixels

ตารางที่ 13 การตั้งค่า Data Augmentation สำหรับโมเดลในการจำแนกสายพันธุ์งู

ลำดับ	เทคนิคที่ใช้	การตั้งค่า
1	Flip	Horizontal, Vertical
2	90° Rotate	Clockwise, Counter-Clockwise, Upside Down
3	Grayscale	15%
4	Hue	Between -15° and +15°
5	Brightness	Between -15° and +15°

ตารางที่ 14 ข้อมูลรูปภาพที่ฝึกฝนโมเดลในการตรวจจับงู

ลำดับ	ประเภทภาพ	จำนวน(ภาพ)
1	งู	2556
2	สายยาง	1771
	รวม	4327

ตารางที่ 15 ข้อมูลรูปภาพที่ฝึกฝนโมเดลในการจำแนกสายพันธุ์งู

ลำดับ	ชื่อสายพันธุ์งู	จำนวน(ภาพ)
1	งูสามเหลี่ยม	335
2	งูสิงหางลาย	360
3	งูเขียวพระอินทร์	361
4	งูจงอาง	340
5	งูทับสมิงคลา	340
6	งูกะปะ	360
7	งูเห่า	361
8	งูเขียวหางไหม้	361
9	งูเหลือม	362
10	งูแมวเซา	336
	รวม	3516

4.การแบ่งสัดส่วนรูปภาพ

จากการสร้างชุดข้อมูลเพิ่มเติมเสร็จเรียบร้อย ผู้จัดทำจึงได้ดำเนินการแบ่งชุดข้อมูลเพื่อใช้ในการฝึกฝนโมเดลให้เกิดการรู้จำ (Training) ตรวจสอบความถูกต้อง (Validation) และทดสอบระบบ (Testing) โดยแบ่งเป็นอัตราส่วนร้อยละ 70, 20 และ 10 ตามลำดับ ดังแสดงในตารางที่ 9 และ 10

ตารางที่ 16 การแบ่งสัดส่วนรูปภาพที่ฝึกฝนโมเดลในการตรวจจับงู

ลำดับ	การใช้งานรูปภาพ	ร้อยละ	จำนวน(ภาพ)
1	สำหรับใช้ฝึกฝนโมเดล (Training)	70	3044
2	สำหรับใช้ตรวจสอบความถูกต้อง (Validation)	20	848
3	สำหรับใช้ทดสอบระบบ (Testing)	10	435
	รวม	100	4327

ตารางที่ 17 การแบ่งสัดส่วนรูปภาพที่ฝึกฝนโมเดลในการจำแนกสายพันธุ์งู

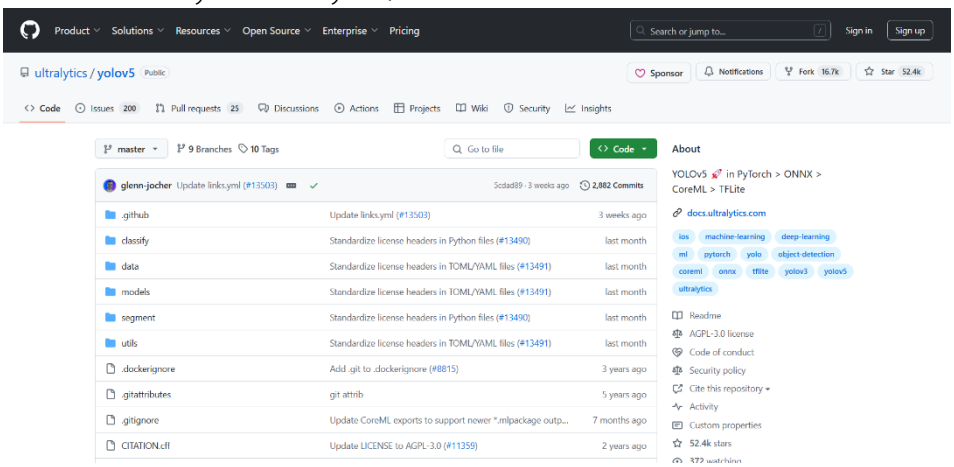
ลำดับ	การใช้งานรูปภาพ	ร้อยละ	จำนวน(ภาพ)
1	สำหรับใช้ฝึกฝนโมเดล (Training)	70	2444
2	สำหรับใช้ตรวจสอบความถูกต้อง (Validation)	20	709
3	สำหรับใช้ทดสอบระบบ (Testing)	10	363
	รวม	100	3516

3.6 การพัฒนา Model

3.6.1 การพัฒนาระบบการตรวจจับงู

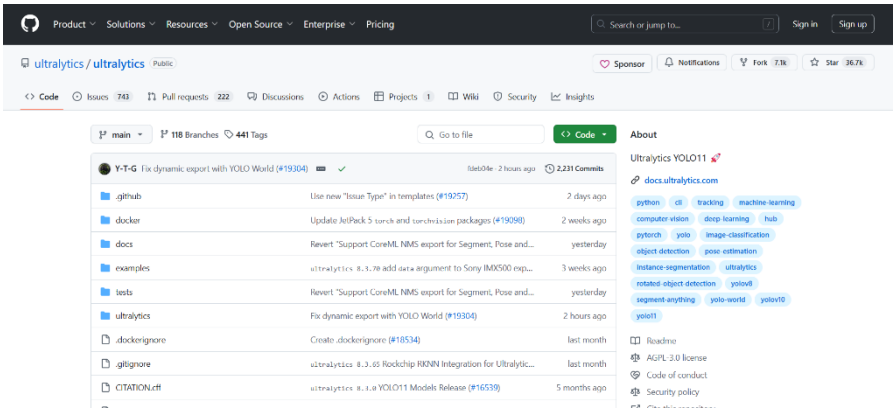
1.การติดตั้งอัลกอริทึม

YOLOv5 และ YOLOv11 เป็นอัลกอริทึมประเภทโอเพ่นซอส (open source) ที่ใช้สำหรับการตรวจจับวัตถุ และ จำแนกสายพันธุ์ โดยผู้พัฒนาได้เปิดให้ผู้ที่ต้องการใช้งานสามารถดาวน์โหลดได้จากเว็บไซต์ Github (https://github.com/ultralytics/yolov5 และ https://github.com/ultralytics/ultralytics) ดังแสดงในภาพที่ 21 และ 22



ภาพที่ 19 เว็บไซต์ github สำหรับดาวน์โหลด YOLOv5

ที่มา : https://github.com/ultralytics/yolov5



ภาพที่ 20 เว็บไซต์ github สำหรับดาวน์โหลด YOLOv11

ที่มา : https://github.com/ultralytics/ultralytics

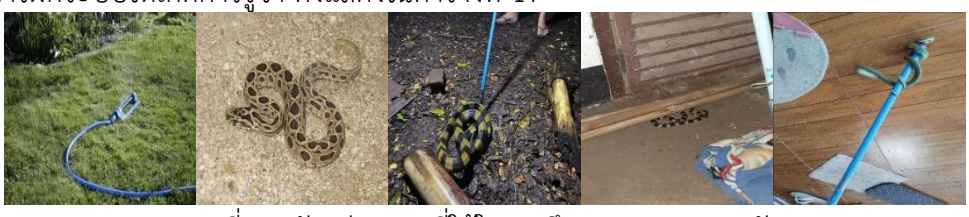
2.การฝึกฝนระบบการตรวจจับงูให้เกิดการรู้จำ (training)

ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 2 วัตถุ (2 class) ได้แก่ งูและสายยาง จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย YOLOv5 และ YOLOv11 การฝึกระบบดำเนินการตามเงื่อนไขดังนี้

- โมเดล YOLOv5 ทำการฝึกทั้งหมด 100 รอบ (100 epochs)

- โมเดล YOLOv11 ทำการฝึกทั้งหมด 40 รอบ (40 epochs)

ตัวอย่างภาพที่ใช้ในการฝึกระบบให้เกิดการรู้จำแสดงดังภาพที่ 23 และได้ออกแบบตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ ดังแสดงในตารางที่ 17



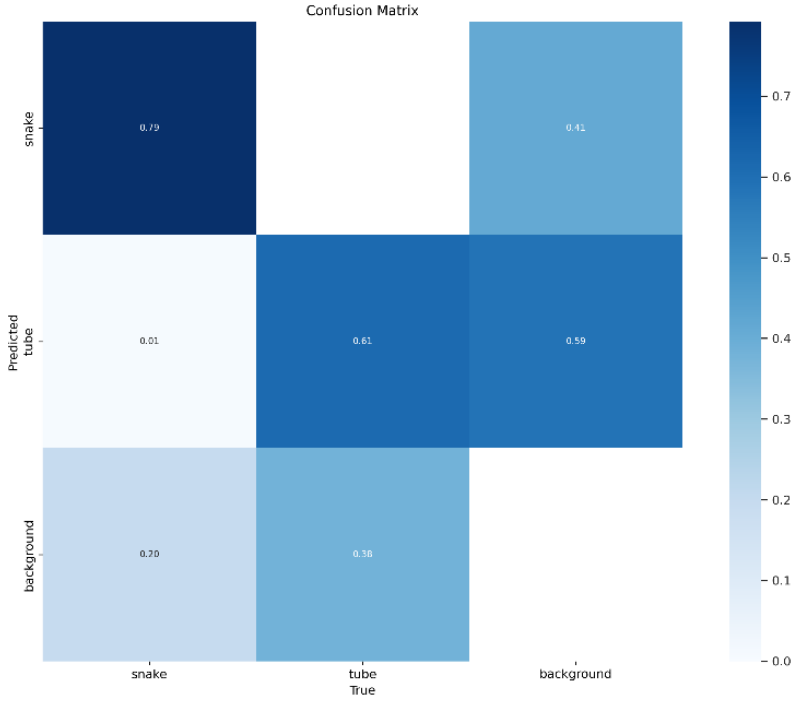
ภาพที่ 21 ตัวอย่างภาพที่ใช้ในการฝึกระบบการตรวจจับงู

ตารางที่ 18 ตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ(training)

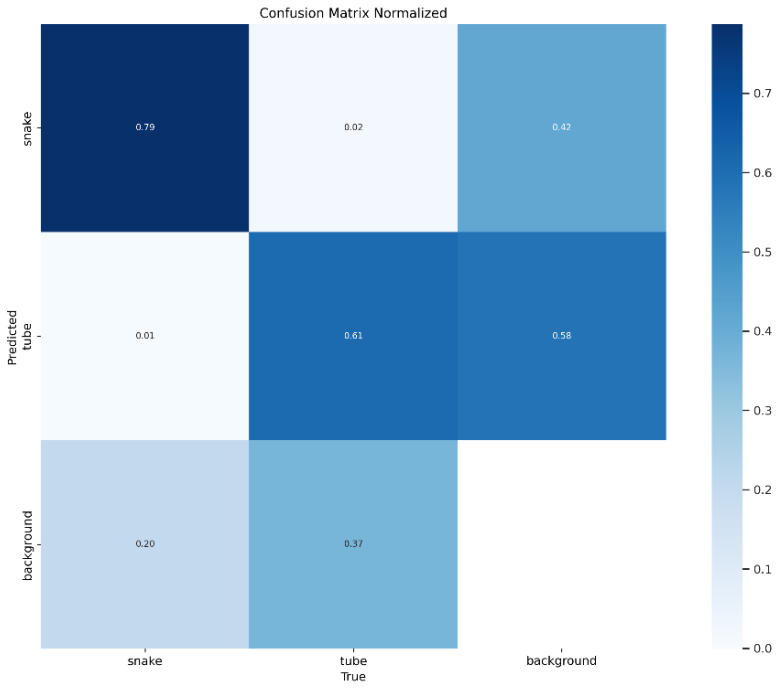
ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	mAP50	mAP50-95
1	YOLOv5	63m.	100	0.806	0.683	0.731	0.568
2	YOLOv11	48m	40	0.783	0.655	0.731	0.654

3.การวิเคราะห์ข้อมูลการทำงานของระบบการตรวจจับงูด้วย Confusion Matrix

จากผลการทดสอบระบบผู้วิจัยได้ทำการวิเคราะห์ข้อมูลด้วย Confusion Matrixโดยทำการแยกภาพภาพงูและสายยางทั้งหมดจำนวน 435 ภาพจากนั้นทำการบันทึกผลการทดสอบดังแสดงในภาพที่ 24 และ 25



ภาพที่ 22 Confusion Matrix YOLOv5 สำหรับวิเคราะห์ข้อมูลการทำงานของการตรวจจับงู



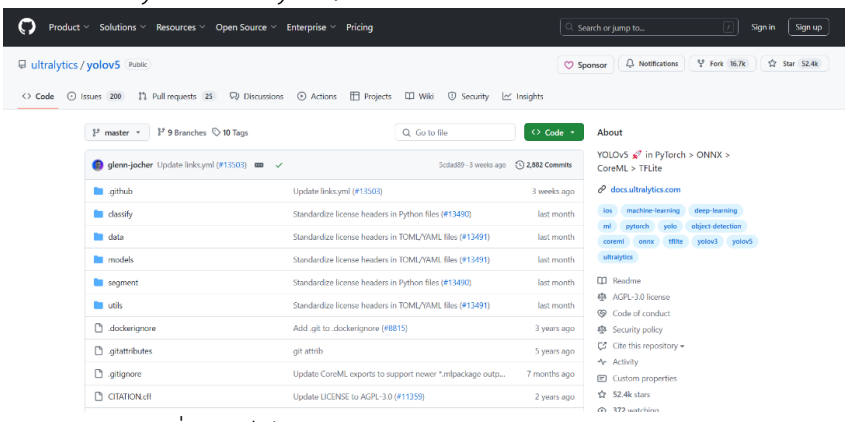
ภาพที่ 23 Confusion Matrix YOLOv11 สำหรับวิเคราะห์ข้อมูลการทำงานของการตรวจจับงู

3.6.2 การพัฒนาระบบการจำแนกสายพันธุ์งู

- **การพัฒนา Yolo Model**

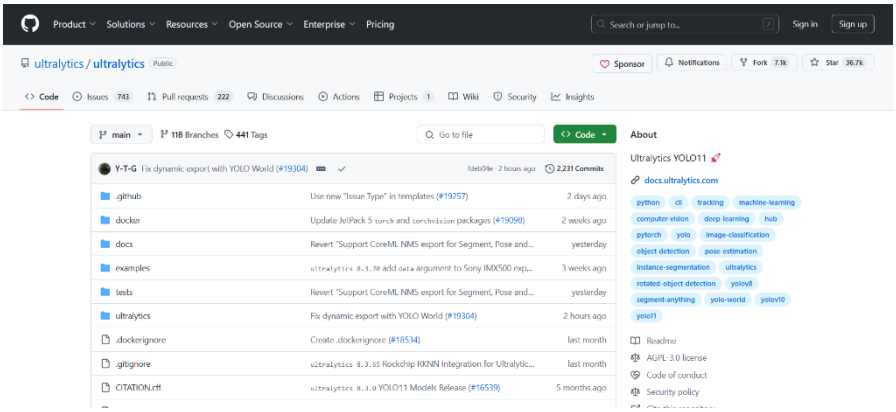
1.การติดตั้งอัลกอริทึม

YOLOv5, YOLOv8 และ YOLOv11 เป็นอัลกอริทึมประเภทโอเพ่นซอส (open source) ที่ใช้สำหรับการตรวจจับวัตถุ และ จำแนกสายพันธุ์ โดยผู้พัฒนาได้เปิดให้ผู้ที่ต้องการใช้งานสามารถดาวน์โหลดได้จากเว็บไซต์ Github (https://github.com/ultralytics/yolov5 และ https://github.com/ultralytics/ultralytics) ดังแสดงในภาพที่ 26 และ 27



ภาพที่ 24 เว็บไซต์ github สำหรับดาวน์โหลด YOLOv5

ที่มา :_https://github.com/ultralytics/yolov5



ภาพที่ 25 เว็บไซต์ github สำหรับดาวน์โหลด YOLOv8 และYOLOv11

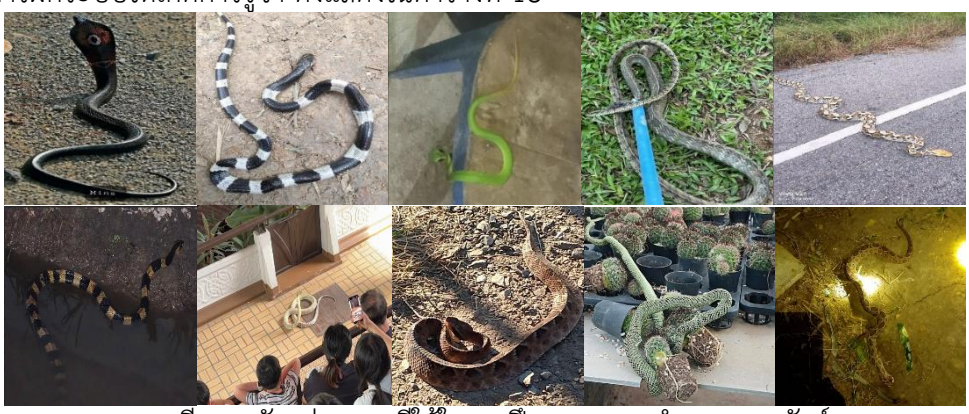
ที่มา : https://github.com/ultralytics/ultralytics

2.การฝึกฝนระบบการจำแนกสายพันธุ์งูให้เกิดการรู้จำ (training)

ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 10 วัตถุ (10 class) ได้แก่ งูสามเหลี่ยม,งูสิงหางลาย,งูเขียวพระอินทร์,งูจงอาง,งูทับสมิงคลา,งูกะปะ,งูเห่า,งูเขียวหางไหม้,งูเหลือม และ งูแมวเซา จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย YOLOv5,YOLOv8และ YOLOv11 การฝึกระบบดำเนินการตามเงื่อนไขดังนี้

- โมเดล YOLOv5 ทำการฝึกทั้งหมด 100 รอบ (100 epochs)
- โมเดล YOLOv8 ทำการฝึกทั้งหมด 150 รอบ (100 epochs)
- โมเดล YOLOv11 ทำการฝึกทั้งหมด 40 รอบ (40 epochs)

ตัวอย่างภาพที่ใช้ในการฝึกระบบให้เกิดการรู้จำแสดงดังภาพที่ 28 และได้ออกแบบตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ ดังแสดงในตารางที่ 18



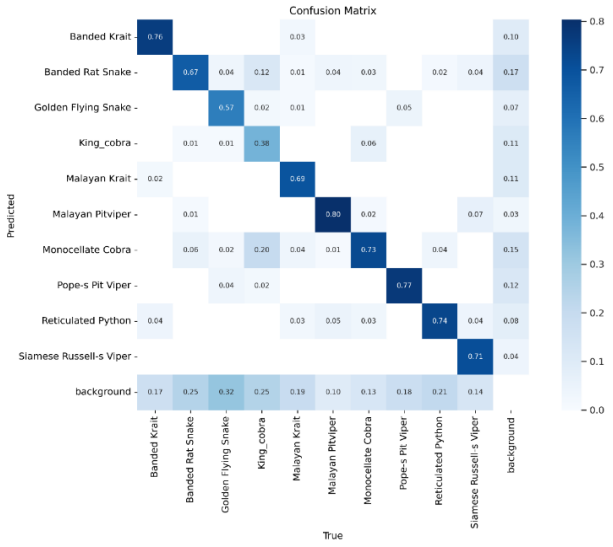
ภาพที่ 26 ตัวอย่างภาพที่ใช้ในการฝึกระบบการจำแนกสายพันธุ์งู

ตารางที่ 19 ตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ(training)

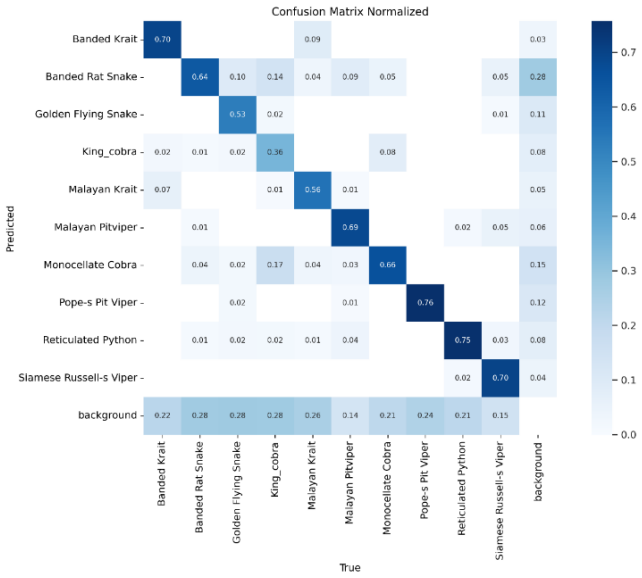
ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	F1-Score
1	YOLOv5	37m	50	0.769	0.663	0.711
2	YOLOv8	48m	150	0.746	0.587	0.667
3	YOLOv11	20m	40	0.745	0.581	0.654

3. การวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งูด้วย Confusion Matrix

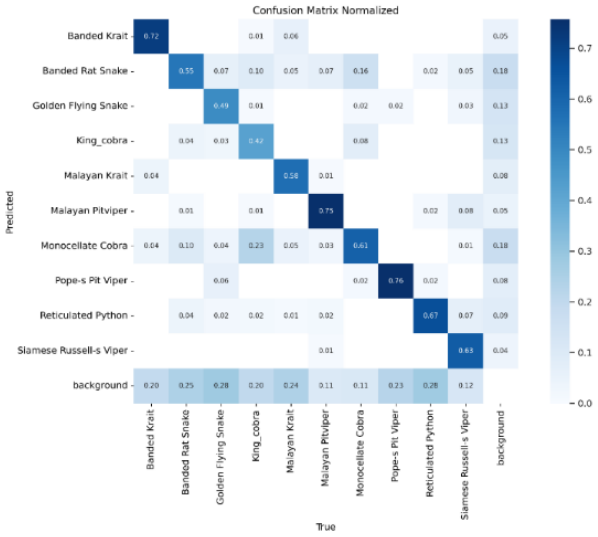
จากผลการทดสอบระบบผู้วิจัยได้ทำการวิเคราะห์ข้อมูลด้วย Confusion Matrixโดยทำการแยกภาพภาพงูและสายยางทั้งหมดจำนวน 363 ภาพจากนั้นทำการบันทึกผลการทดสอบดังแสดงในภาพที่ 29 ,30 และ 31



ภาพที่ 27 Confusion Matrix YOLOv5 สำหรับวิเคราะห์ข้อมูลการทำงานของการจำแนกสายพันธุ์งู



ภาพที่ 28 Confusion Matrix YOLOv8 สำหรับวิเคราะห์ข้อมูลการทำงานของการจำแนกสายพันธุ์งู

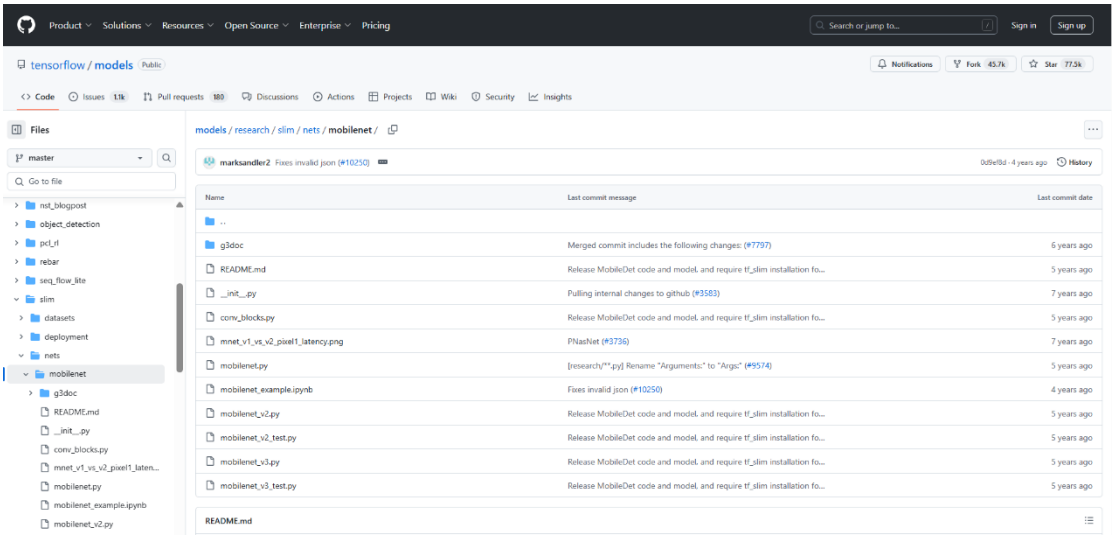


ภาพที่ 29 Confusion Matrix YOLOv11 สำหรับวิเคราะห์ข้อมูลการทำงานของการจำแนกสายพันธุ์งู

- **การพัฒนา MobileNetV2 Model**

1.การติดตั้งอัลกอริทึม

MobileNetV2 เป็นโมเดลประเภท Convolutional Neural Network (CNN) ที่ถูกออกแบบมาเพื่อ จำแนกประเภทของภาพ (Image Classification) โดยพัฒนาโดย Google และเผยแพร่ในรูปแบบ โอเพ่นซอร์ส (open source) บนแพลตฟอร์ม บนแพลตฟอร์ม TensorFlow และ Keras สามารถดาวน์โหลดโมเดลและโค้ดตัวอย่างได้จาก GitHub(https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet) ดังแสดงในภาพที่ 30



ภาพที่ 30 เว็บไซต์ github สำหรับดาวน์โหลด MobileNetV2

2.การเตรียมข้อมูล

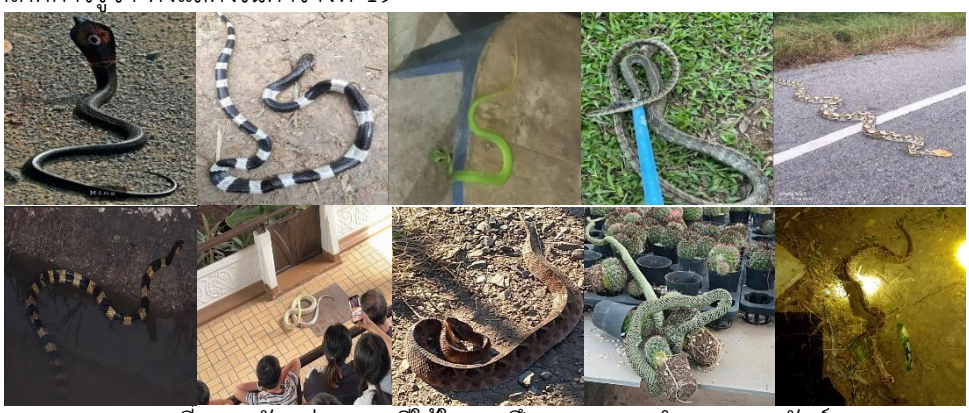
ในการเตรียมข้อมูลสำหรับการฝึกโมเดล Deep Learning โดยการโหลดข้อมูลจากไฟล์ JSON และทำการปรับ category_id ให้มีการเรียงลำดับใหม่อย่างต่อเนื่อง เนื่องจากข้อมูลจาก COCO JSON อาจใช้ category_id ที่ไม่ต่อเนื่องกัน นอกจากนี้ควรทำการโหลดภาพจากที่เก็บข้อมูลและแปลงเป็น TensorFlow Dataset โดยปรับขนาดของภาพให้เป็น 224x224 พิกเซล และทำการ One-Hot Encoding ให้กับ label เพื่อให้เหมาะสมกับการฝึกโมเดล

```python
# โหลดรูปภาพ
def load_image_and_label(image_id):
    image_id = int(image_id.numpy())
    img_path = os.path.join(IMAGE_DIRS["train"], train_images[image_id])

    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)) / 255.0

    labels = [ann["category_id"] for ann in train_annotations if ann["image_id"] == image_id]
    labels = [0] if len(labels) == 0 else labels
    one_hot_label = tf.keras.utils.to_categorical(labels[0], num_classes)

    return img.astype(np.float32), one_hot_label.astype(np.float32)

# สร้าง TensorFlow Dataset
def create_tf_dataset(image_dict):
    image_ids = list(image_dict.keys())
    dataset = tf.data.Dataset.from_tensor_slices(image_ids)

    dataset = dataset.map(lambda id: tf.py_function(
        load_image_and_label,
        [id],
        [tf.float32, tf.float32]
    ), num_parallel_calls=tf.data.AUTOTUNE)

    dataset = dataset.map(lambda img, lbl: (tf.ensure_shape(img, (224, 224, 3)), tf.ensure_shape(lbl, (num_classes,)))).batch(32).shuffle(100).prefetch(tf.data.AUTOTUNE)

    return dataset
```

ภาพที่ 31 การเตรียมข้อมูลสำหรับ MobileNetV2

3.การทดสอบการทำงานของระบบการจำแนกสายพันธุ์งู

ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 10 วัตถุ (10 class) ได้แก่ งูสามเหลี่ยม,งูสิงหางลาย,งูเขียวพระอินทร์,งูจงอาง,งูทับสมิงคลา,งูกะปะ,งูเห่า,งูเขียวหางไหม้,งูเหลือม และ งูแมวเซา จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย ทำการฝึกทั้งหมด 20 รอบ (20 epochs) ตัวอย่างภาพที่ใช้ในการฝึกระบบให้เกิดการรู้จำแสดงดังภาพที่ 34 และได้ออกแบบตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ ดังแสดงในตารางที่ 19



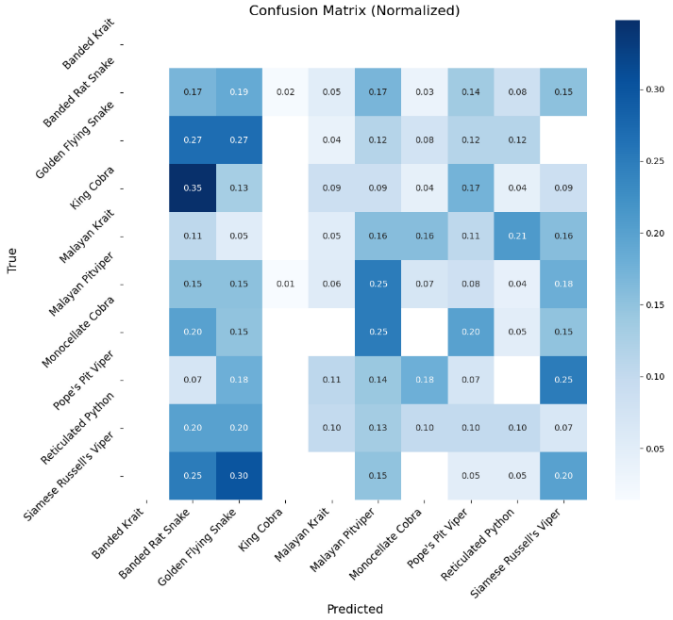
ภาพที่ 32 ตัวอย่างภาพที่ใช้ในการฝึกระบบการจำแนกสายพันธุ์งู

ตารางที่ 20 ตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ(training)

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	F1-Score
1	MobileNetV2	25m	20	0.760	0.710	0.730

4.การวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งูด้วย Confusion Matrix

จากผลการทดสอบระบบผู้วิจัยได้ทำการวิเคราะห์ข้อมูลด้วย Confusion Matrixโดยทำการแยกภาพภาพงูและสายยางทั้งหมดจำนวน 363 ภาพจากนั้นทำการบันทึกผลการทดสอบดังแสดงในภาพที่ 32

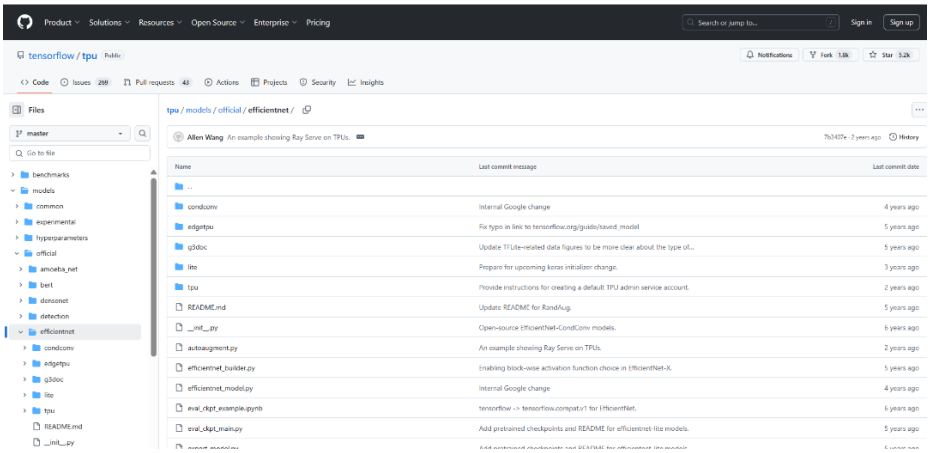


ภาพที่ 33 Confusion Matrix MobileNetV2 สำหรับวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งู

- **การพัฒนา EfficientNet Model**

1.การติดตั้งอัลกอริทึม

EfficientNet เป็นโมเดลประเภท Convolutional Neural Network (CNN) ที่ถูกออกแบบมาเพื่อ จำแนกประเภทของภาพ (Image Classification) โดยพัฒนาโดย Google และเผยแพร่ในรูปแบบโอเพ่นซอร์ส (open source) บนแพลตฟอร์ม TensorFlow และ PyTorch สามารถดาวน์โหลดโมเดลและโค้ดตัวอย่างได้จากGitHub ดังแสดงในภาพที่ 34



ภาพที่ 34 เว็บไซต์ github สำหรับดาวน์โหลด EfficientNet

ที่มา : https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

2.การเตรียมข้อมูล

สำหรับการฝึกโมเดล Deep Learning โดยมีการโหลดและประมวลผลข้อมูลที่อยู่ในรูปแบบ COCO (Common Objects in Context) ก่อนนำไปใช้กับโมเดล TensorFlow/Keras โดยมีการกำหนดค่าดังนี้

- กำหนดค่าพื้นฐานของโมเดล เช่น IMG_SIZE, BATCH_SIZE, และ NUM_CLASSES

- กำหนดพาธของไฟล์ COCO และรูปภาพ ได้แก่ dataset_dir, coco_file, และ image_dir
- โหลดไฟล์ Annotation ของ COCO เพื่อนำข้อมูล JSON มาสร้างชุดข้อมูล
- ฟังก์ชันโหลดข้อมูลจาก COCO โดยแปลงภาพเป็นอาร์เรย์และทำ One-Hot Encoding สำหรับคลาส
- สร้าง TensorFlow Dataset และจัดรูปแบบข้อมูลให้พร้อมใช้งานโดยใช้ shuffle() และ batch()

```
IMG_SIZE = 640
BATCH_SIZE = 16
NUM_CLASSES = 10

dataset_dir = "/content/drive/MyDrive/EfficientNet/dataset_v.2/train"
coco_file = os.path.join(dataset_dir, "_annotations.coco.json")
image_dir = os.path.join(dataset_dir, "images")

coco = COCO(coco_file)

category_ids = coco.getCatIds()
categories = coco.loadCats(category_ids)
class_names = {cat["id"]: cat["name"] for cat in categories}

def load_coco_data(coco, image_dir, img_size=IMG_SIZE):
    image_ids = coco.getImgIds()
    X, y = [], []

    for img_id in image_ids:
        img_info = coco.loadImgs(img_id)[0]
        img_path = os.path.join(image_dir, img_info["file_name"])

        if not os.path.exists(img_path):
            print(f"❌ ไม่พบรูปภาพ: {img_path}")
            continue

        print(f"✅ กำลังโหลด: {img_path}")
        img = load_img(img_path, target_size=(img_size, img_size))
        img_array = img_to_array(img) / 255.0
        X.append(img_array)
        ann_ids = coco.getAnnIds(imgIds=img_id)
        anns = coco.loadAnns(ann_ids)
        if anns:
            class_id = anns[0]["category_id"]
            y.append(class_id)
        else:
            y.append(0)
    X = np.array(X)
    y = np.array(y)
    y = to_categorical(y, num_classes=len(class_names))

    return X, y

X_train, y_train = load_coco_data(coco, image_dir)

train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
train_dataset = train_dataset.shuffle(1000).batch(BATCH_SIZE)
```

ภาพที่ 35 การเตรียมข้อมูลสำหรับ EfficientNet

3.การฝึกฝนระบบการตรวจจับงูให้เกิดการรู้จำ (training)

ผู้จัดทำได้การแบ่งประเภทของวัตถุในภาพเพื่อฝึกให้ระบบเกิดการรู้จำทั้งหมด 10 วัตถุ (10 class) ได้แก่ งูสามเหลี่ยม,งูสิงหางลาย,งูเขียวพระอินทร์,งูจงอาง,งูทับสมิงคลา,งูกะปะ,งูเห่า,งูเขียวหางไหม้,งูเหลือม และ งูแมวเซา จากนั้นทำการฝึกระบบให้เกิดการรู้จำด้วย ทำการฝึกทั้งหมด 41 รอบ (41 epochs) ตัวอย่างภาพที่ใช้ในการฝึกระบบให้เกิดการรู้จำแสดงดังภาพที่ 38 และได้ออกแบบตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ ดังแสดงในตารางที่ 20

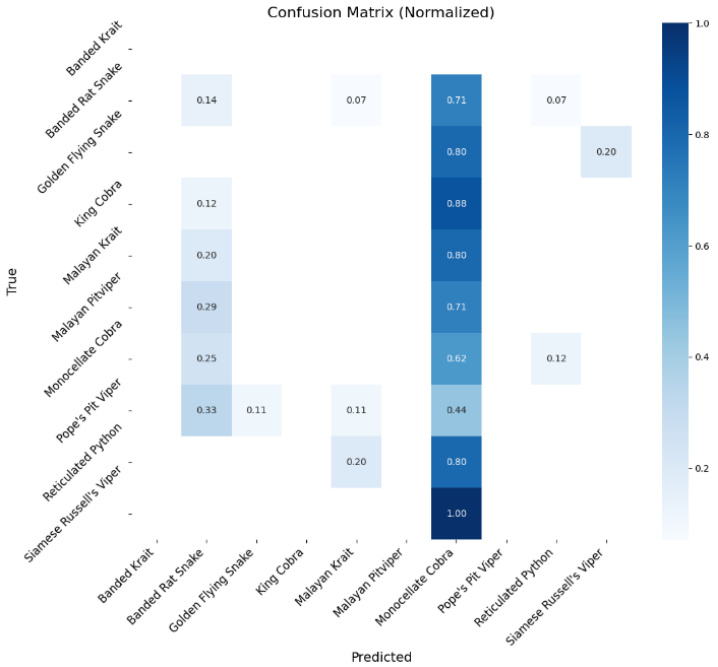
ภาพที่ 36 ตัวอย่างภาพที่ใช้ในการฝึกระบบการจำแนกสายพันธุ์งู

ตารางที่ 21 ตารางสำหรับบันทึกผลการฝึกระบบให้เกิดการรู้จำ(training)

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	F1-Score
1	EfficientNet	30m	40	0.520	0.521	0.487

4.การวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งูด้วย Confusion Matrix

จากผลการทดสอบระบบผู้วิจัยได้ทำการวิเคราะห์ข้อมูลด้วย Confusion Matrixโดยทำการแยกภาพภาพงูและสายยางทั้งหมดจำนวน 363 ภาพจากนั้นทำการบันทึกผลการทดสอบดังแสดงในภาพที่ 25



ภาพที่ 37 Confusion Matrix EfficientNet สำหรับวิเคราะห์ข้อมูลการทำงานของระบบการจำแนกสายพันธุ์งู

3.7 การพัฒนาLINE Chatbot

ได้ลงทะเบียนและตั้งค่าLINE Chatbot ใน LINE Developers โดยตั้งค่าการทำงานหลักได้ดังนี้

1. การวิเคราะห์ภาพงู

ในการวิเคราะห์ภาพงู จะนำตัวโมเดลกับ LINE Chatbot เชื่อมต่อกันโดยใช้ Channel Secret และ Channel Access Token เป็นตัวเชื่อมต่อโดยจะใช้ภาษาไพทอน (Python) ด้วยโปรแกรม Visual Studio Code

2. Rich Menu

ภายในเมนู Rich Menu จะมีตัวเลือกสำหรับการตอบกลับทั้งหมด 4 รายการ ได้แก่ การแนะนำโมเดล การปฐมพยาบาลเบื้องต้น การแสดงหมายเลขโทรศัพท์ฉุกเฉิน และเว็บไซต์ให้ความรู้เกี่ยวกับสายพันธุ์งู โดยการตอบกลับใน 3 ตัวเลือกแรกจะเป็นการส่งข้อความ ซึ่งข้อความนั้นจะถูกใช้เป็นคีย์เวิร์ดสำหรับการตอบกลับอัตโนมัติ ส่วนในตัวเลือกสุดท้ายจะเป็นการเปิดเว็บไซต์โดยตรงทันที

3. ข้อความตอบกลับอัตโนมัติ

การตอบกลับข้อความอัตโนมัติ จะตอบกลับโดยจะตอบกลับตามคีย์เวิร์ดที่ได้กำหนดไว้ โดยการตอบกลับสามารถตอบกลับได้หลายประเภท ได้แก่ ข้อความ รูป ริชเมสเสจ การ์ดเมสเสจ วิดีโอคูปอง ริชวิดีโอเมจเสจ สติกเกอร์ ข้อความเสียง และแบบสอบถาม

4. Card-based messages

สร้าง card ใน Card-based messages เลือกประเภทของ การ์ดที่จะแสดงรูปภาพ จากนั้นเพิ่ม Message content ใน Greeting messenge เพิ่มให้แสดง การ์ดเมื่อกดเข้ามาใน ไลน์แชทบอท

3.8 การพัฒนา Web Site สำหรับดูข้อมูลการวิเคราะห์ของโมเดลจำแนกสายพันธุ์งู

1.ออกแบบ Backend โดยใช้ Fast API เพื่อรองรับการอัปโหลดข้อมูล

2.พัฒนา Frontend ด้วย vue.js โดยผู้ใช้สามารถ

- แสดงรูปภาพที่ผู้ใช้ส่งมาเพื่อแสดงและตรวจสอบภาพของงูที่ผู้ใช้ส่งมาได้สะดวกมากยิ่งขึ้น
- ลบข้อมูล

3.9 การวัดประสิทธิภาพ Model

ตารางที่ 22 ผลการฝึกฝนโมเดลในการตรวจจับงูให้เกิดการรู้จำ(Training)

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	mAP50	mAP50-95
1	YOLOv5	63m.	100	0.806	0.683	0.731	0.568
2	YOLOv11	48m	40	0.783	0.655	0.731	0.654

ตารางที่ 23 ตารางเปรียบเทียบการฝึกฝนโมเดลในการจำแนกสายพันธุ์งูให้เกิดการรู้จำ(Training)

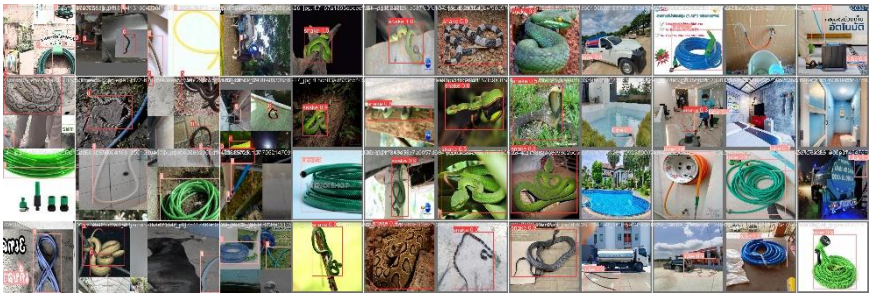
	YOLOv5	YOLOv8	YOLOv11	MobileNetV2	EfficientNet
งูสามเหลี่ยม	0.76	0.70	0.72	0.35	0.71
งูสิงหางลาย	0.67	0.64	0.55	0.27	0.07
งูเขียวพระอินทร์	0.57	0.53	0.58	0.25	0.80
งูจงอาง	0.38	0.36	0.41	0.30	0.88
งูทับสมิงคลา	0.69	0.56	0.61	0.11	0.80
งูกะปะ	0.80	0.69	0.62	0.15	0.71
งูเห่า	0.73	0.66	0.47	0.25	0.62
งูเขียวหางไหม้	0.77	0.76	0.71	0.18	0.44
งูเหลือม	0.74	0.75	0.67	0.13	0.80
งูแมวเซา	0.71	0.70	0.73	0.20	1.00

บทที่ 4
การทดลองและวิเคราะห์ผล

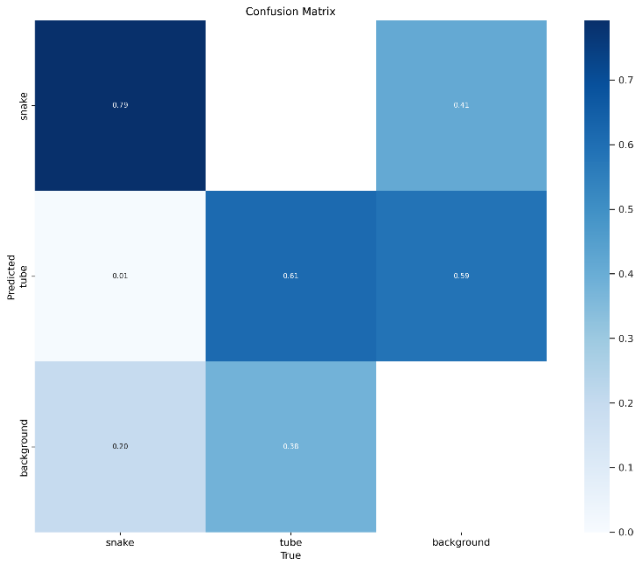
4.1 ผลการฝึกฝนระบบให้เกิดการรู้จำ(Training)

4.1.1 ผลการฝึกฝนโมเดลในการตรวจจับงู

ชุดข้อมูลสำหรับฝึกฝนโมเดลในการตรวจจับงู จำนวน 4327 รูป ได้ดำเนินการแบ่งชุดข้อมูลเพื่อใช้ในการฝึกฝนโมเดลให้เกิดการรู้จำ (Training) ตรวจสอบความถูกต้อง (Validation) และทดสอบระบบ(Testing) โดยแบ่งเป็นอัตราส่วนร้อยละ 70, 20 และ 10 ตามลำดับ แล้วใช้คำสั่งให้ทำการฝึกระบบ (Training) ด้วย YOLOv5 จำนวน 100 รอบ YOLOv11 จำนวน 40 รอบ เพื่อให้ได้ผลการรู้จำที่ดีที่สุด โดยภาพตัวอย่างการตีกรอบรอบวัตถุ, Confusion Matrix ผลการทดสอบโมเดลตรวจจับงูและผลการฝึกฝนโมเดลในการตรวจจับงูให้เกิดการรู้จำ แสดงดังภาพ 36,37,38,39 และ ตารางที่ 23

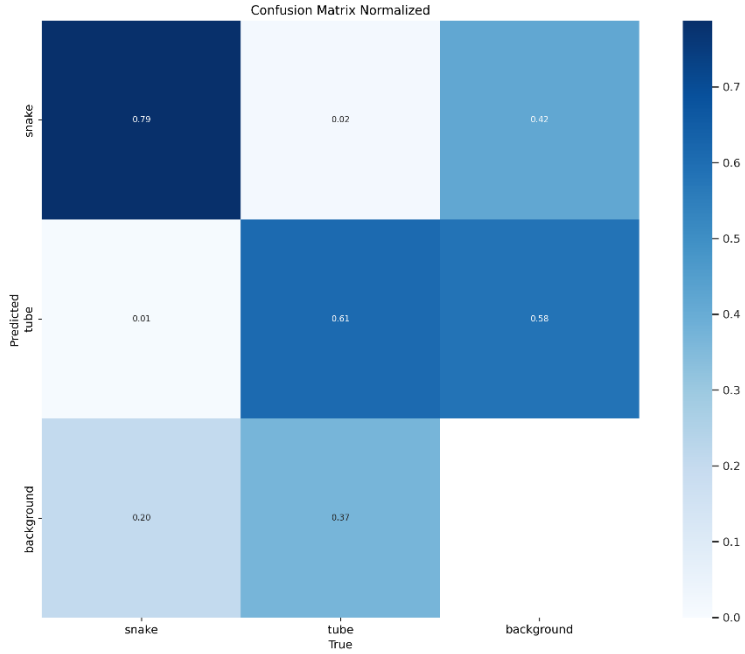


ภาพที่ 38 ภาพตัวอย่างการตีกรอบรอบวัตถุ ด้วย YOLOv5



ภาพที่ 39 Confusion Matrix ผลการทดสอบโมเดลตรวจจับงูด้วย YOLOv5

ภาพที่ 40 ภาพตัวอย่างการตีกรอบรอบวัตถุ ด้วย YOLOv11



ภาพที่ 41 Confusion Matrix ผลการทดสอบโมเดลตรวจจับงูด้วย YOLOv11

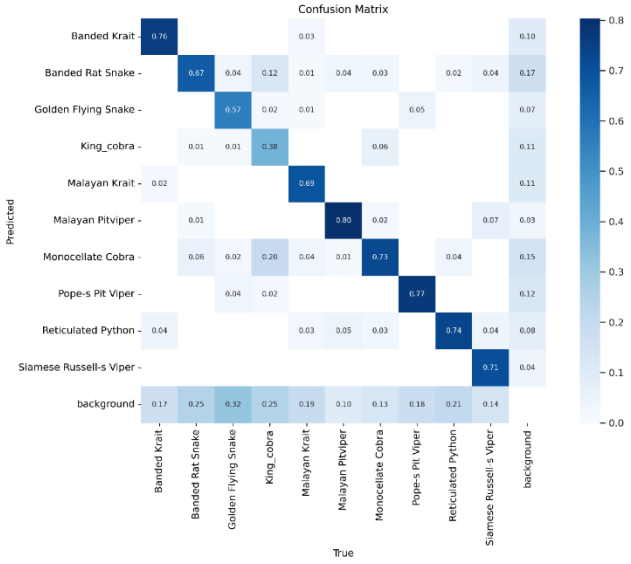
ตารางที่ 24 ผลการฝึกฝนโมเดลในการตรวจจับงูให้เกิดการรู้จำ(Training)

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	mAP50	mAP50-95
1	YOLOv5	63m.	100	0.806	0.683	0.731	0.568
2	YOLOv11	48m	40	0.783	0.655	0.731	0.654

4.2.2 ผลการฝึกฝนโมเดลในการจำแนกสายพันธุ์งู

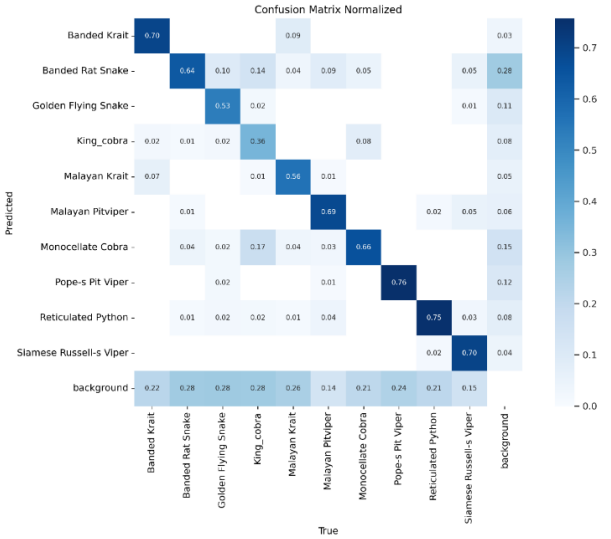
ชุดข้อมูลสำหรับฝึกฝนโมเดลในการจำแนกสายพันธุ์งู จำนวน 3516 รูป ได้ดำเนินการแบ่งชุดข้อมูลเพื่อใช้ในการฝึกฝนโมเดลให้เกิดการรู้จำ (Training) ตรวจสอบความถูกต้อง (Validation) และทดสอบระบบ(Testing) โดยแบ่งเป็นอัตราส่วนร้อยละ 70, 20 และ 10 ตามลำดับ แล้วใช้คำสั่งให้ทำการฝึกระบบ (training) ด้วย YOLOv5 จำนวน 100 รอบ YOLOv8 จำนวน 150 รอบ YOLOv11 จำนวน 40 รอบ EfficientNet จำนวน 40 รอบ MobileNetV2 จำนวน 20 รอบ เพื่อให้ได้ผลการรู้จำที่ดีที่สุด โดยภาพตัวอย่างการตีกรอบรอบวัตถุ, Confusion Matrix ผลการทดสอบโมเดลตรวจจับงูและ ผลการฝึกฝนโมเดลในการจำแนกสายพันธุ์งูให้เกิดการรู้จำ แสดงดังภาพ 40,41,42,43,44,45,46,47 และ 48

ภาพที่ 42 ภาพตัวอย่างการตีกรอบรอบวัตถุ ด้วย YOLOv5



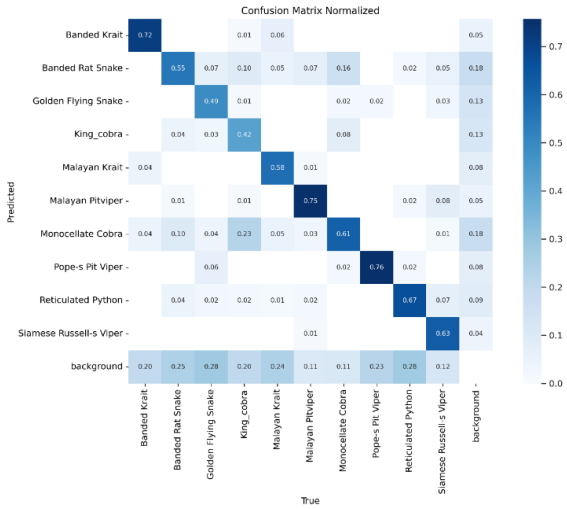
ภาพที่ 43 Confusion Matrix ผลการทดสอบโมเดลจำแนกสายพันธุ์งูด้วย YOLOv5

ภาพที่ 44 ภาพตัวอย่างการตีกรอบรอบวัตถุ ด้วย YOLOv8

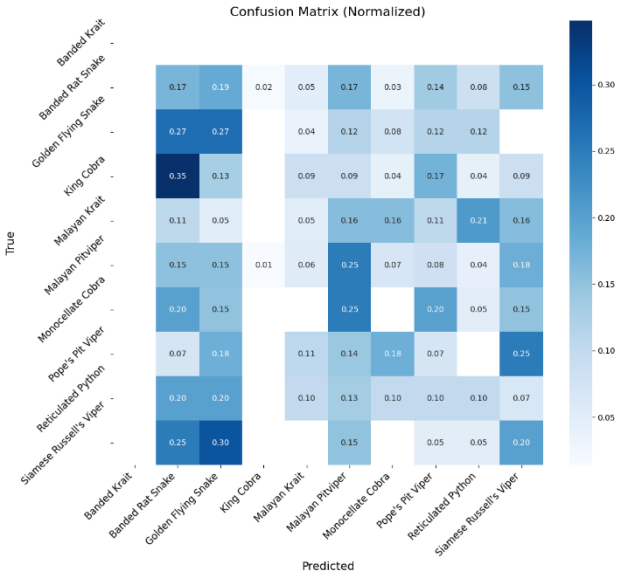


ภาพที่ 45 Confusion Matrix ผลการทดสอบโมเดลจำแนกสายพันธุ์งูด้วย YOLOv8

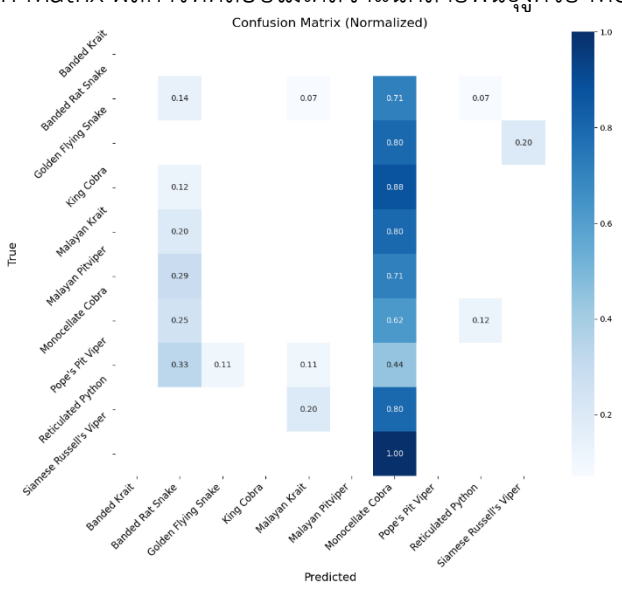
ภาพที่ 46 ภาพตัวอย่างการตีกรอบรอบวัตถุ ด้วย YOLOv11



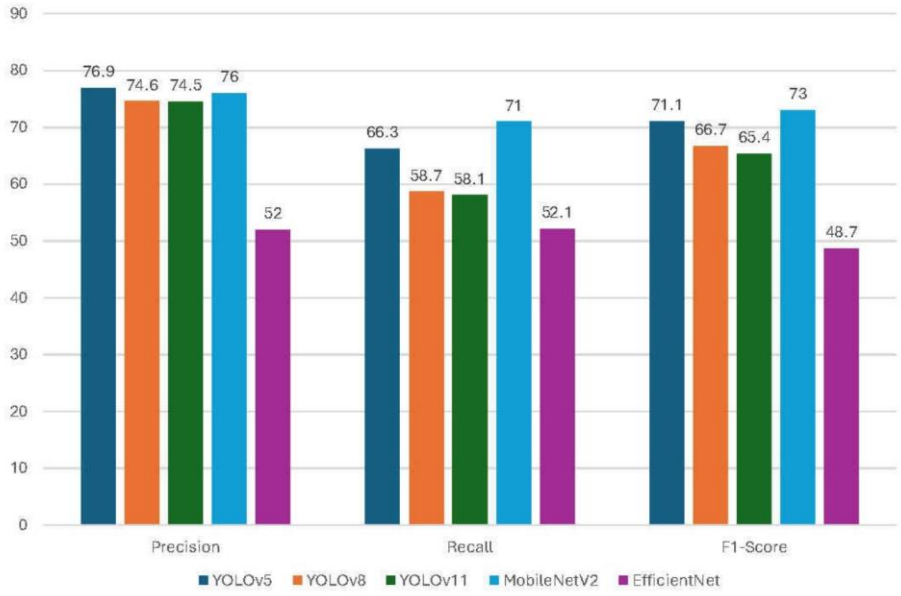
ภาพที่ 47 Confusion Matrix ผลการทดสอบโมเดลจำแนกสายพันธุ์งูด้วย YOLOv11



ภาพที่ 48 Confusion Matrix ผลการทดสอบโมเดลจำแนกสายพันธุ์งูด้วย MobileNetV2

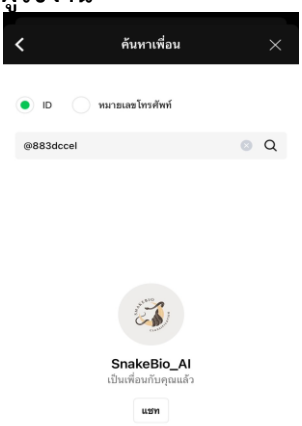


ภาพที่ 49 Confusion Matrix ผลการทดสอบโมเดลจำแนกสายพันธุ์งูด้วย EfficientNet



ภาพที่ 50 ผลการฝึกฝนโมเดลในการจำแนกสายพันธุ์งูให้เกิดการรู้จำ(Training)

4.2 การออกแบบส่วนติดต่อกับผู้ใช้งาน



ภาพที่ 51 หน้าจอค้นหาบัญชี SnakeBio_AI โดยใช้ ID LINE ในการค้นหา

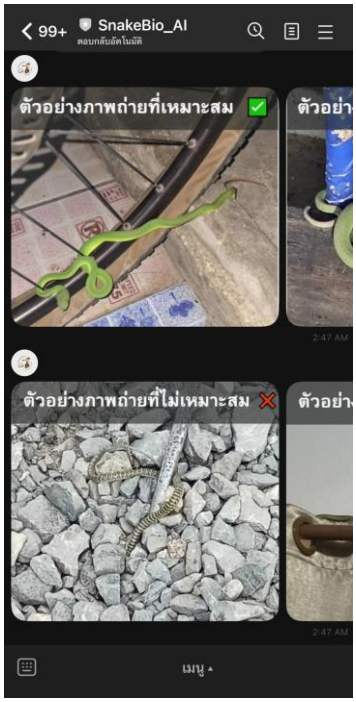
จากภาพที่ 49 แสดงหน้าจอค้นหาบัญชีไลน์ในการวิเคราะห์จำแนกสายพันธุ์งู โดยใช้ชื่อว่า SnakeBio_AI โดยผู้ใช้สามารถค้นหาได้จากช่องการค้นหาด้วย LINE ID

ภาพที่ 52 หน้าจอไลน์แชทบอท ทักทายเพื่อนใหม่

จากภาพที่ 52 เมื่อผู้ใช้งานเพิ่มเพื่อนสำเร็จระบบจะส่งข้อความต้อนรับเพื่อนใหม่อัตโนมัติ

ภาพที่ 53 หน้าจอไลน์แชทบอท Rich menu

จากภาพที่ 53 ระบบจะแสดงหน้า Rich menu ขึ้นมาโดยอัตโนมัติ ผู้ใช้งานสามารถเลือกใช้ตามเมนูที่ปรากฏได้ว่าต้องการข้อมูลเกี่ยวกับด้านไหน

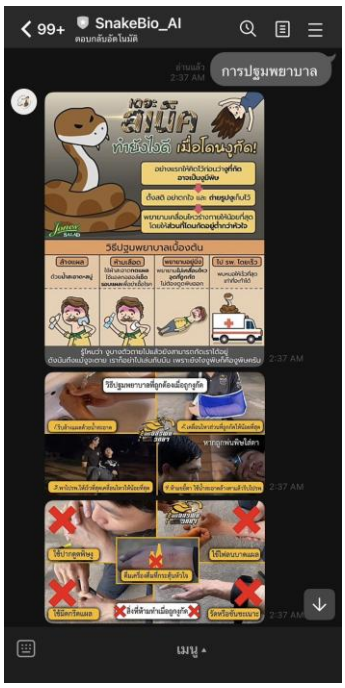


ภาพที่ 54 หน้าจอแสดงตัวอย่างรูปภาพที่เหมาะสมสำหรับการส่ง

จากภาพที่ 54 ระบบจะแสดงหน้าจอ ตัวอย่างรูปภาพที่เหมาะสม และไม่เหมาะสม ให้ผู้ใช้ได้ทราบว่าควรส่งรูปภาพแบบใด ให้ไลน์แชทบอท

ภาพที่ 55 หน้าจอแสดงเว็บไซต์ของสถานเสาวภา สภากาชาดไทย

จากภาพที่ 55 ระบบจะแสดงหน้า เว็บไซต์ของสถานเสาวภา สภากาชาดไทย เกี่ยวกับสายพันธุ์งูที่มีความสำคัญทางสาธารณสุข



ภาพที่ 56 หน้าจอแสดงการปฐมพยาบาลเบื้องต้น

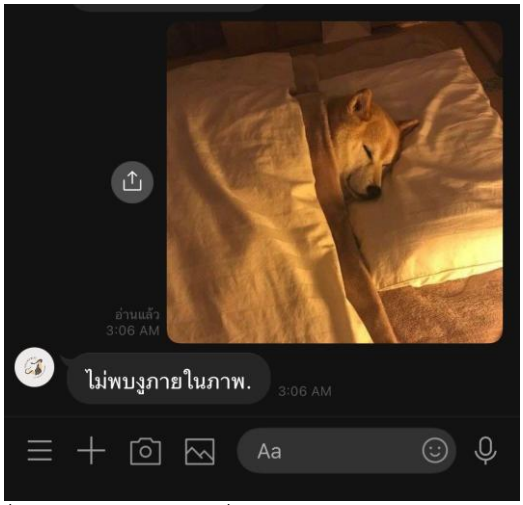
จากภาพที่ 56 แสดงหน้าจอหลังจากผู้ใช้งานเลือกการปฐมพยาบาลเบื้องต้น โดยจะแสดงขั้นตอนการปฐมพยาบาลเบื้องต้นเมื่อถูกงูกัด

ภาพที่ 57 หน้าจอแสดงเบอร์โทรฉุกเฉิน

จากภาพที่ 57 แสดงหน้าจอหลังจากผู้ใช้งานเลือกเบอร์โทรฉุกเฉิน โดยจะแสดงเบอร์โทรฉุกเฉินเมื่อพบเจองูและ เบอร์โทรหน่วยงานต่างๆ

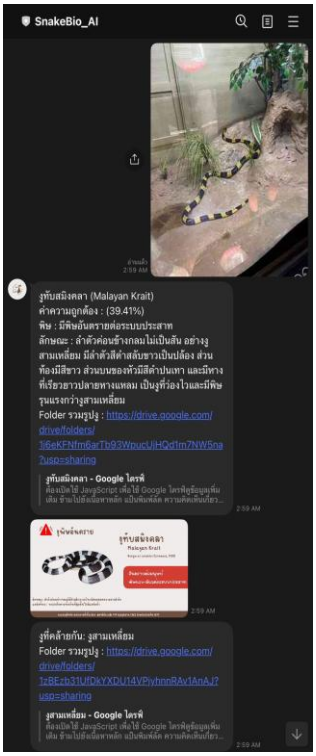
ภาพที่ 58 หน้าจอแสดงเมื่อวิเคราะห์สายพันธุ์งูภายในภาพสำเร็จ

จากภาพที่ 58 แสดงหน้าจอหลังจากผู้ใช้งานส่งภาพเข้ามา และเมื่อระบบได้วิเคราะห์สายพันธุ์งูสำเร็จ จะมีการแสดงข้อมูลได้แก่ชื่อสายพันธุ์งูที่พบ พิษ ลักษณะทางกายภาพ และ ลิงค์ไฟล์ภาพรวบรวมรูปงูชนิดนั้นๆ และ รูปภาพรายละเอียด



ภาพที่ 59 หน้าจอแสดงเมื่อไม่สามารถตรวจจับงูได้ในภาพ

จากภาพที่ 59 แสดงหน้าจอหลังจากผู้ใช้งานส่งภาพเข้ามา และทางระบบไม่สามารถตรวจจับงูได้



ภาพที่ 60 หน้าจอแสดงเมื่อวิเคราะห์สายพันธุ์งูภายในภาพสำเร็จกรณีมีสายพันธุ์งูที่คล้ายกัน

จากภาพที่ 60 แสดงหน้าจอหลังจากผู้ใช้งานส่งภาพเข้ามา และเมื่อระบบได้วิเคราะห์สายพันธุ์งูสำเร็จ จะมีการแสดงข้อมูลได้แก่ชื่อสายพันธุ์งูที่พบ พิษ ลักษณะทางกายภาพ และ ลิงค์ไฟล์ภาพรวบรวมรูปงูชนิดนั้น รูปรายละเอียด และข้อมูลงูที่มีลวดลาย หรือ ลักษณะ คล้ายคลึงกัน โดยจะมี ชื่อสายพันธุ์ และลิงค์ไฟล์ภาพรวบรวมรูปงูชนิดนั้น

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

ในบทนี้กล่าวถึงสรุปผลการดำเนินงานของโครงงานวิทยาการคอมพิวเตอร์ สามารถแบ่งออกเป็น 2 ส่วน ดังนี้

5.1 สรุปผลการดำเนินงาน

โครงการนี้เป็นการพัฒนา 2 โมเดลหลัก ได้แก่ โมเดลในการตรวจจับงู และ โมเดลในการจำแนกสายพันธุ์งู โดยมีผลการทดสอบระบบ ดังนี้

1.โมเดลในการตรวจจับงู โมเดลนี้ได้รับการฝึกด้วยอัลกอริทึม YOLOv5 และ YOLOv11 โดยมีการกำหนดป้ายกำกับ (Label) จำนวน 2 ป้าย ได้แก่ "งู" และ "สายยาง" เมื่อพิจารณาผลการทดสอบผ่าน Confusion Matrix พบว่าโมเดลที่ฝึกด้วยอัลกอริทึม YOLOv5 และ YOLOv11 สามารถจำแนกงูได้อย่างถูกต้องที่ร้อยละ และ สามารถจำแนกสายยางได้อย่างถูกต้องที่ร้อยละ นอกจากนี้ มีการประเมินประสิทธิภาพของโมเดลด้วยตัวชี้วัดอื่นๆดังตารางที่ :

ตารางที่ 24 ผลการฝึกฝนโมเดลในการตรวจจับงูให้เกิดการรู้จำ(Training)

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	mAP50	mAP50-95
1	YOLOv5	63m.	100	0.806	0.683	0.731	0.568
2	YOLOv11	48m	40	0.783	0.655	0.731	0.654

2.โมเดลในการจำแนกสายพันธุ์งู ได้รับการฝึกด้วยอัลกอริทึม YOLOv5 ,YOLOv11 ,EfficientNet และ MobileNetV2 จำนวน โดยมีการกำหนดป้ายกำกับ (Label) จำนวน 10 ป้าย ได้แก่ งูสามเหลี่ยม, งูสิงหางลาย, งูเขียวพระอินทร์, งูจงอาง, งูทับสมิงคลา, งูกะปะ, งูเห่า, งูเขียวหางไหม้, งูเหลือม และงูแมวเซา เมื่อพิจารณาผลการทดสอบผ่าน Confusion Matrix พบว่าโมเดลสามารถจำแนกสายพันธุ์งูได้อย่างถูกต้องดังตารางที่ 30

ตารางที่ 26 ผลการฝึกฝนโมเดลในการจำแนกสายพันธุ์งู

	YOLOv5	YOLOv8	YOLOv11	MobileNetV2	EfficientNet
งูสามเหลี่ยม	0.76	0.70	0.72	0.35	0.71
งูสิงหางลาย	0.67	0.64	0.55	0.27	0.07
งู เขียวพระอินทร์	0.57	0.53	0.58	0.25	0.80
งูจงอาง	0.38	0.36	0.41	0.30	0.88
งูทับสมิงคลา	0.69	0.56	0.61	0.11	0.80
งูกะปะ	0.80	0.69	0.62	0.15	0.71
งูเห่า	0.73	0.66	0.47	0.25	0.62
งูเขียวหางไหม้	0.77	0.76	0.71	0.18	0.44
งูเหลือม	0.74	0.75	0.67	0.13	0.80
งูแมวเซา	0.71	0.70	0.73	0.20	1.00

ผลการประเมินประสิทธิภาพของโมเดลนี้ในการฝึกฝนด้วยอัลกอริทึม
YOLOv5 ,YOLOv11 ,EfficientNet และ MobileNetV2 ได้ผลลัพธ์ดังตารางที่ 31

ตารางที่ 27 ผลการฝึกฝนโมเดลในการจำแนกสายพันธุ์งู

ลำดับ	โมเดล	เวลาที่ใช้	Epoch	Precision	Recall	F1-Score
1	EfficientNet	30m	40	0.520	0.521	0.487
2	MobileNetV2	25m	20	0.760	0.710	0.730
3	YOLOv5	37m	50	0.769	0.663	0.711
4	YOLOv8	48m	150	0.746	0.587	0.667
5	YOLOv11	20m	40	0.745	0.581	0.654

สรุปจากทั้ง 2 โมเดลที่ได้พัฒนา ได้ข้อสรุปว่า โมเดลในการตรวจจับงู และ โมเดลในการจำแนกสายพันธุ์งู เลือกใช้ YOLOv5 ที่มีค่าความถูกต้องในการตรวจจับงู 0.806 และ ค่าความถูกต้องในการจำแนกสายพันธุ์งู 0.769 ทั้งสองโมเดลได้ถูกนำไปเชื่อมต่อกับระบบ LINE เพื่อให้สามารถจำแนกสายพันธุ์งูจากรูปภาพที่ส่งมาได้อย่างมีประสิทธิภาพ นอกจากนี้ ระบบยังได้เพิ่ม Rich Menu ที่มีฟังก์ชันเพิ่มเติมเพื่ออำนวยความสะดวกแก่ผู้ใช้ โดยประกอบด้วย

1. การเข้าถึงข้อมูล การปฐมพยาบาลเบื้องต้น
2. แสดง หมายเลขโทรศัพท์ฉุกเฉิน
3. ลิงก์ไปยัง เว็บไซต์สวนงู ของสถานเสาวภา สภากาชาดไทย

การเพิ่มฟังก์ชันเหล่านี้ใน Rich Menu ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลที่สำคัญและจำเป็นได้อย่างรวดเร็วและง่ายดาย.

5.2 ปัญหาและอุปสรรคที่พบ

1. การรวบรวมข้อมูลภาพ - ข้อมูลภาพงูมีจำนวนจำกัด และบางสายพันธุ์มีภาพน้อย ทำให้โมเดลเรียนรู้ได้ไม่ดี
2. คุณภาพของข้อมูล - ภาพที่ใช้ฝึกโมเดลบางรูปมีความคมชัดไม่เพียงพอ หรือมีฉากหลังที่ซับซ้อน ทำให้จำแนกได้ยาก
3. การเทรนข้อมูล - พื้นที่ที่ใช้สำหรับการเทรนข้อมูลใน Google Collab บางครั้งมีจำนวนไม่พอ ทำให้ต้อซื้อเพิ่มบ่อยครั้ง
4. การตอบกลับผู้ใช้งาน - การตอบกลับผู้ใช้งานไม่สามารถส่งรูปภาพที่วิเคราะห์ให้กับผู้ใช้งานได้เนื่องจากไม่ใช่URLภายนอก

5.3 แนวทางการพัฒนาต่อ

1. การรองรับหลายแพลตฟอร์ม - อาจต้องพัฒนาให้รองรับ API ได้หลายแพลตฟอร์มมากขึ้น
2. รองรับหลายภาษา - หากต้องการให้ใช้งานได้ทั่วโลก อาจต้องรองรับการแปลภาษาอัตโนมัติ
3. เพิ่มสายพันธุ์งู - เพื่อให้ใช้ได้หลายประเทศมากยิ่งขึ้นอาจจะต้องเพิ่มสายพันธุ์ของงู
4. สามารถป้อนข้อมูลเพิ่มเติม - ให้ผู้ใช้สามารถป้อนข้อมูลเพิ่มเติม เช่น สี ลวดลาย หรือสถานที่พบ

5.4 ข้อเสนอแนะ

1.ควรทำการตรวจสอบและคัดกรองชุดข้อมูลให้มีความหลากหลายและคุณภาพสูงยิ่งขึ้น โดยเฉพาะภาพที่แสดงตำแหน่งงูที่มีความซับซ้อน เช่น งูที่ซ่อนอยู่บางส่วน หรือภาพที่มีสิ่งรบกวน (Noise) มาก อาจช่วยให้โมเดลเรียนรู้ได้ดีขึ้น

2.ข้อผิดพลาดที่ระบบจำแนกผิดเป็น Background ปัญหาหนึ่งที่พบคือการจำแนกผิดเป็น Background ในบางกรณี การพิจารณาปรับปรุงชุดข้อมูลและการกำหนดขอบเขตของภาพที่ใช้ในการฝึกอาจช่วยลดปัญหานี้ได้

3.เพิ่มการตอบสนองที่ละเอียดและครอบคลุมมากขึ้นในกรณีที่ผู้ใช้ส่งภาพที่ไม่สามารถระบุสายพันธุ์งูได้ ให้แสดงคำแนะนำเบื้องต้นหรือแนะนำให้ผู้ใช้ตรวจสอบกับผู้เชี่ยวชาญ

4.เพิ่มฟังก์ชันการค้นหาหรือการสอบถามเกี่ยวกับงูแต่ละสายพันธุ์ผ่าน Rich Menu เพื่อให้ผู้ใช้สามารถหาข้อมูลเพิ่มเติมได้ง่ายขึ้น

5.ควรทำการศึกษา Feedback จากผู้ใช้จริงเพื่อหาจุดที่ควรปรับปรุงเพิ่มเติม โดยเฉพาะในการใช้งานฟีเจอร์ที่มีใน Rich Menu และการตอบสนองของโมเดล

เอกสารอ้างอิง

[1] กรมควบคุมโรค, "Annual Epidemiological Surveilance Report 2019," 2019.

[2] W. W. Fankam-ai, Suphachai; Kitiva, Sahawut; Ketui, Nongnuch, "Development of Mask Classification and Wearing Detection Program with Image Processing Technology," *TNI Journal of Engineering and Technology,* vol. 11, no. 1, 2023.

[3] H. L. Yimeng Xia, "YODE-FEIM: An Enhanced YOLOv5s Algorithm for Snake Detection in Wild Environments," *International Journal of Science and Engineering Applications,* vol. 13, no. 10, 2024, doi: 10.7753/ijsea1310.1016.

ภาคผนวก

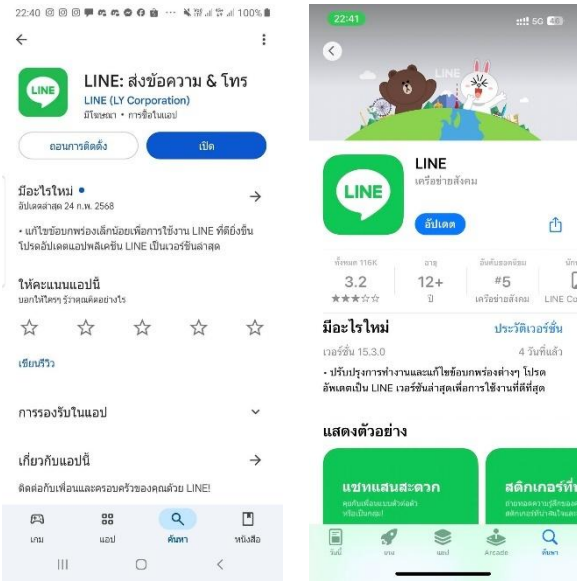
ภาคผนวก ก

คู่มือการใช้งานระบบ

ในการใช้งานระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู มี 2 ส่วน ได้แก่ส่วนของผู้ใช้งาน และ ส่วนของแอดมิน

ส่วนของผู้ใช้งาน

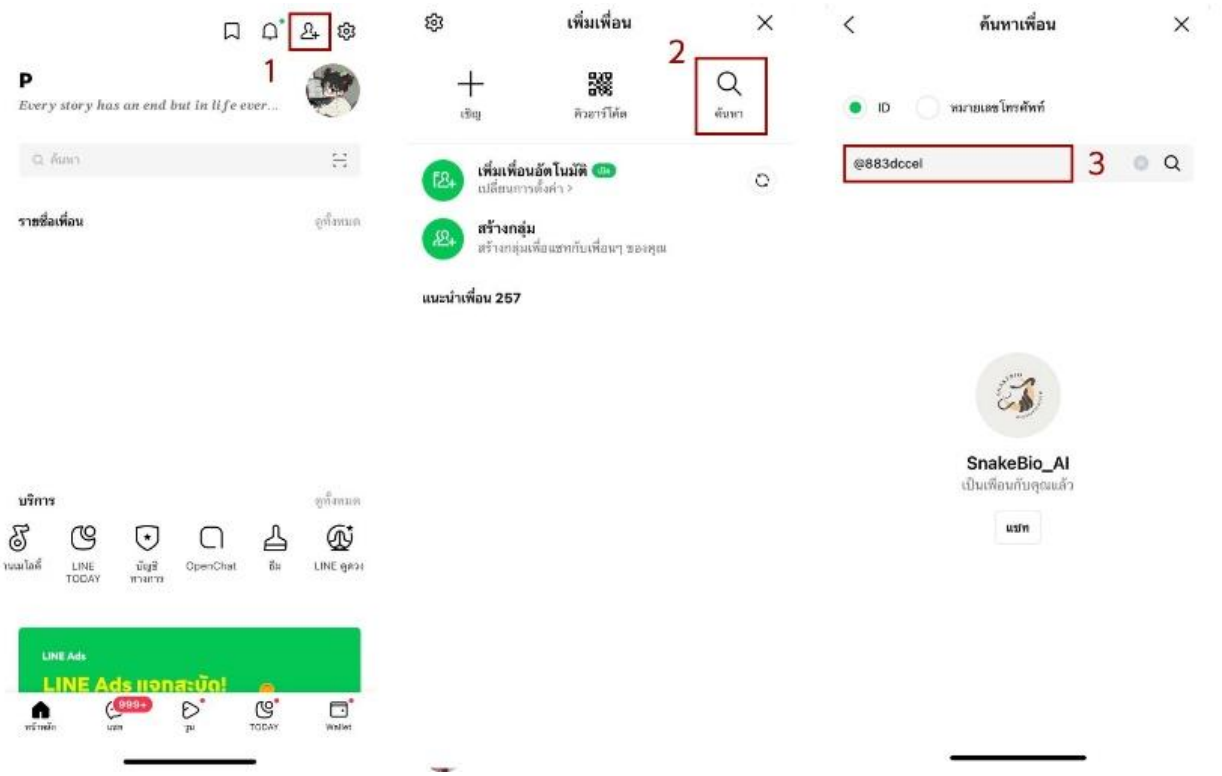
1.ผู้ใช้งานสามารถใช้งานระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงูผ่านแอพพลิเคชั่น LINE ซึ่งสามารถดาวน์โหลดได้ทาง Google Play สำหรับอุปกรณ์ Android และ App Store สำหรับอุปกรณ์ iOS



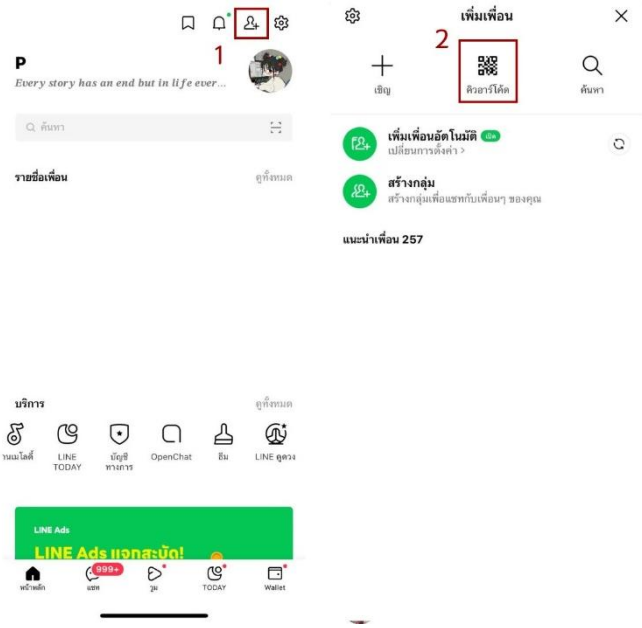
ภาพที่ ก.1 หน้าจอค้นหาแอพพลิเคชั่น LINE ทาง Google Play และ App Store

จากภาพที่ ก.1 เมื่อดาวน์โหลดและติดตั้งแอพพลิเคชั่นเสร็จเรียบร้อย สามารถสมัครบัญชี LINE ใหม่ต้องใช้หมายเลขโทรศัพท์ หรือ อีเมล โดยสามารถทำตามวิดีโอดังนี้ https://youtu.be/npq_I7pXmQ4?si=hburRrCk7SJeCfUj

2.เมื่อสมัครบัญชี LINE เสร็จสิ้น ผู้ใช้งานสามารถค้นหาบัญชี LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู เพื่อเพิ่มเป็นเพื่อนและเริ่มใช้งานได้ 2 วิธี ดังภาพ ก.2 และ ก.3



ภาพที่ ก.2 ขั้นตอนในการค้นหา แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู ด้วย ID LINE



ภาพที่ ก.3 ขั้นตอนในการค้นหา แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู ด้วย QR code

ภาพที่ ก.4 QR code แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู

จากภาพ ก.2 จะแสดงขั้นตอนในการค้นหาบัญชี LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู โดยมีรายละเอียดดังนี้

1.เมื่อเข้าสู่แอพพลิเคชั่น LINE จะสามารถเพิ่มเพื่อนด้วย ID LINE โดยสามารถกดเลือกไอคอนมุมขวาบน อันที่ 3

2.เมื่อกดไอคอนเข้ามาจะเจอ 3 เมนู ได้แก่ 1.เชิญ 2.คิวอาร์โค้ด 3.ค้นหา โดยผู้ใช้งานกดเลือกเมนู ค้นหา ฝั่งขวามือ

3.เมื่อกดเลือก ค้นหา จะเจอให้เลือกให้การค้นหาเพื่อน 2 ตัวเลือก ได้แก่ 1.ID 2.หมายเลขโทรศัพท์ โดยผู้ใช้งานกดเลือก ID และกรอก ID LINE ดังนี้ @883dccel เมื่อกรอกเสร็จสิ้น ผู้ใช้งานกดไอคอน ค้นหา จะแสดงดังภาพ ก.2

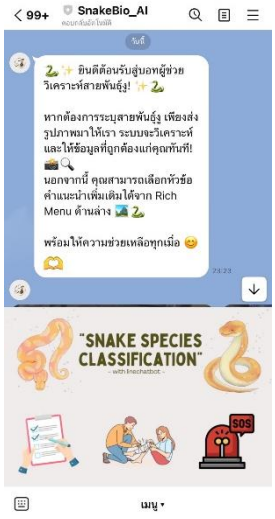
จากภาพ ก.3 จะแสดงขั้นตอนในการค้นหาบัญชี LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู โดยมีรายละเอียดดังนี้

1.เมื่อเข้าสู่แอพพลิเคชั่น LINE จะสามารถเพิ่มเพื่อนด้วย ID LINE โดยสามารถกดเลือกไอคอนมุมขวาบน อันที่ 3

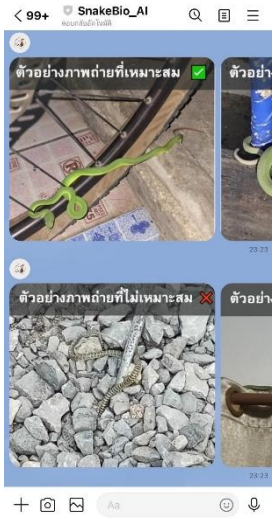
2.เมื่อกดไอคอนเข้ามาจะเจอ 3 เมนู ได้แก่ 1.เชิญ 2.คิวอาร์โค้ด 3.ค้นหา โดยผู้ใช้งานกดเลือกเมนู คิวอาร์โค้ด ตรงกลาง

3.สแกนคิวอาร์โค้ด ในภาพที่ ก.4

3.หน้าจอไลน์แชทบอท ทักทายเพื่อนใหม่



ภาพที่ ก.5 หน้าจอไลน์แชทบอท ทักทายเพื่อนใหม่



ภาพที่ ก.6 หน้าจอแสดงตัวอย่างรูปภาพที่เหมาะสมสำหรับการส่ง

4.ผู้ใช้งานสามารถใช้ไลน์แชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู โดยสามารถส่งรูปภาพงูที่ผู้ใช้งานต้องการทราบสายพันธุ์

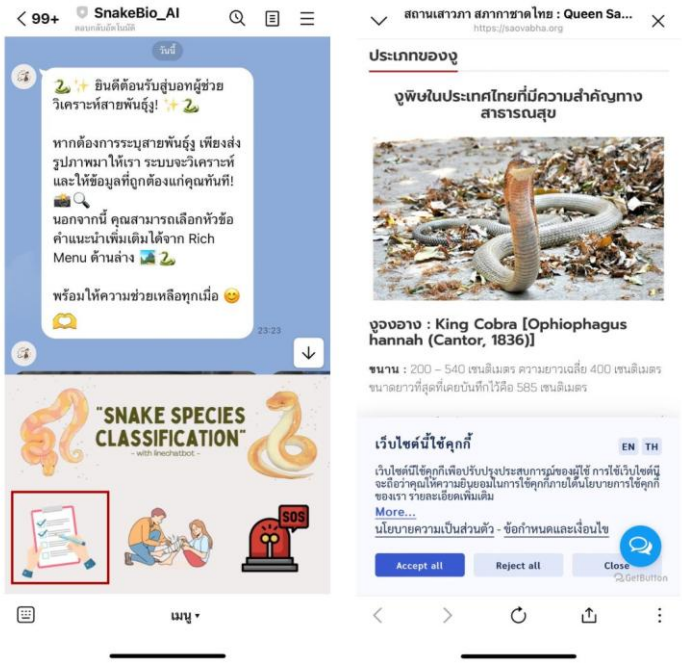
ภาพที่ ก.7 หน้าจอแสดงเมื่อวิเคราะห์สายพันธุ์งูภายในภาพ

5.ส่วนของRich Menu

ภาพที่ ก.8 หน้าจอไลน์แชทบอท Rich menu

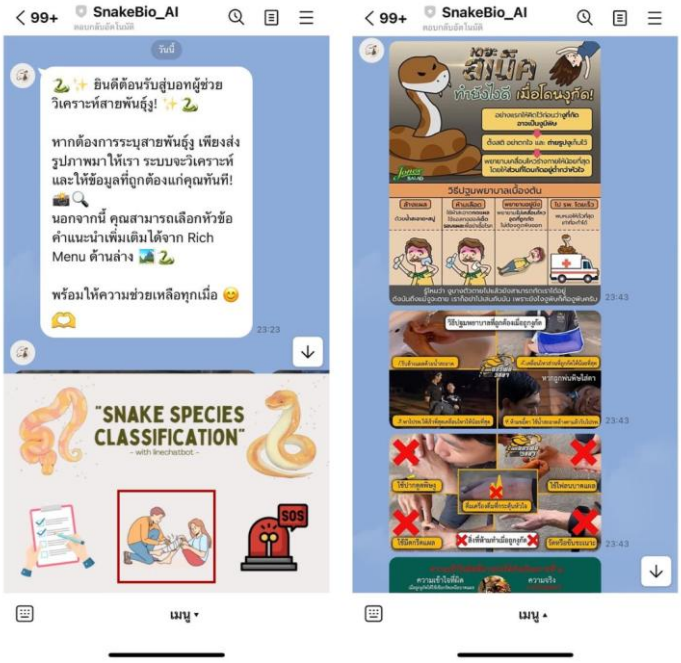
จากภาพ ก.8 ระบบจะแสดงหน้า Rich menu ขึ้นมาในอัตโนมัติ โดยผู้ใช้งานสามารถเลือกใช้ตามเมนูที่ปรากฏได้ว่าต้องการข้อมูลเกี่ยวกับด้านใด โดยจะมี 3 ด้าน ได้แก่

1.การนำไปสู่เว็บไซต์ของสถานเสาวภา สภากาชาดไทย เกี่ยวกับสายพันธุ์งูที่มีความสำคัญทางสาธารณสุข ดังภาพที่ ก.9



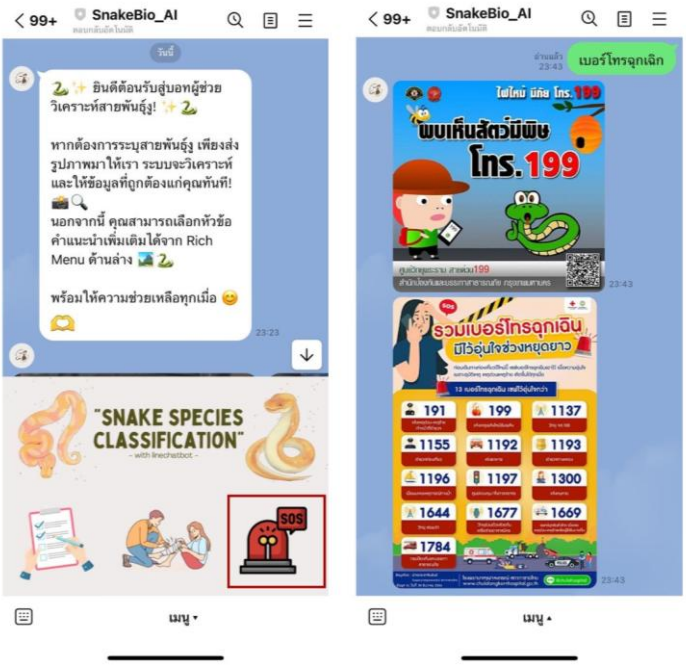
ภาพที่ ก.9 ขั้นตอนในการแสดงเว็บไซต์ของสถานเสาวภา สภากาชาดไทย

2.การปฐมพยาบาลเบื้องต้น โดยจะแสดงขั้นตอนการปฐมพยาบาลเบื้องต้นเมื่อถูกงูกัด ดังภาพที่ ก.10



ภาพ ก.10 ขั้นตอนในการปฐมพยาบาลเบื้องต้น

3.เบอร์โทรฉุกเฉิก โดยจะแสดงเบอร์โทรฉุกเฉิกเมื่อพบเจองู และ เบอร์โทรหน่วยงานต่างๆ ดังภาพ ก.11



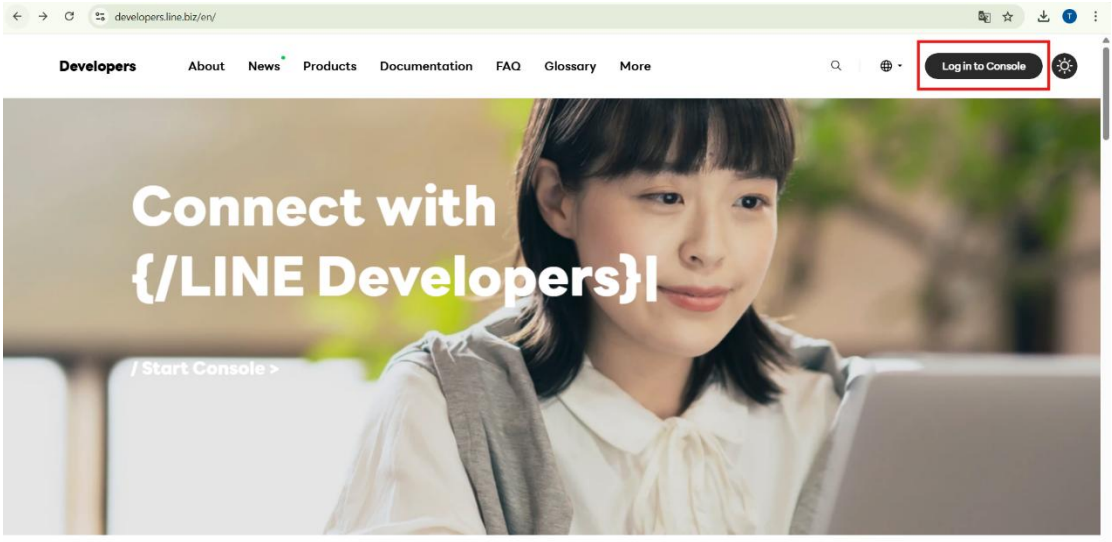
ภาพ ก.11 ขั้นตอนในการแสดงเบอร์โทรฉุกเฉิน

ส่วนของแอดมิน

ในส่วนของแอดมิน จะแบ่งออกเป็น 2 ส่วน ได้แก่ 1. line developers 2. line official account manager 3.Web site

1.line developers แอดมินสามารถแก้ไขการทำงานต่างๆได้ภายในเว็บไซค์ https://developers.line.biz/en/

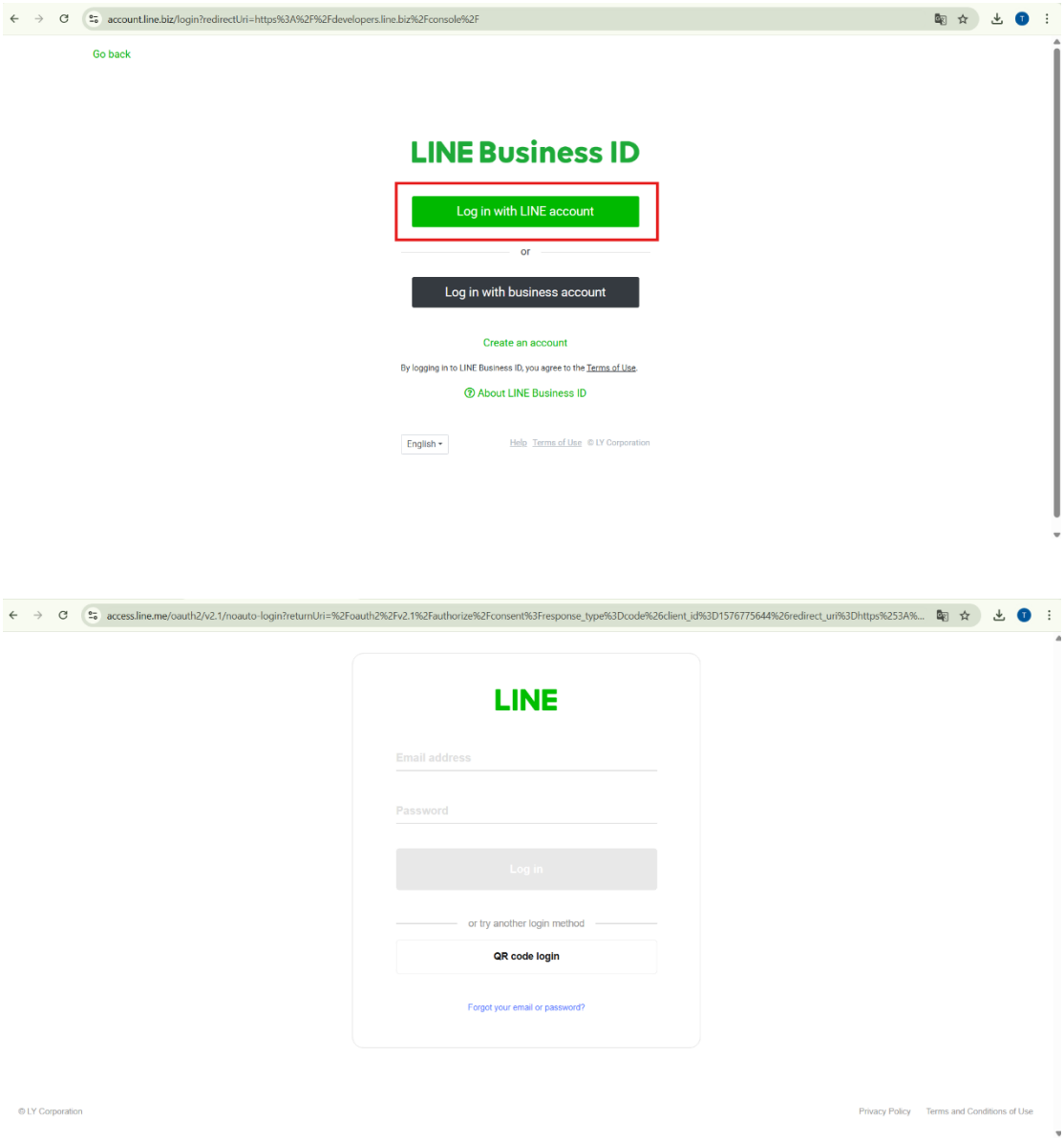
1.1 การเข้าสู่หน้าเว็บไซค์ line developers



ภาพที่ ก.12 หน้าเว็บไซค์ Developers

จากภาพที่ก.12 แอดมินเมนู Log in Console เพื่อเข้าสู่ระบบ

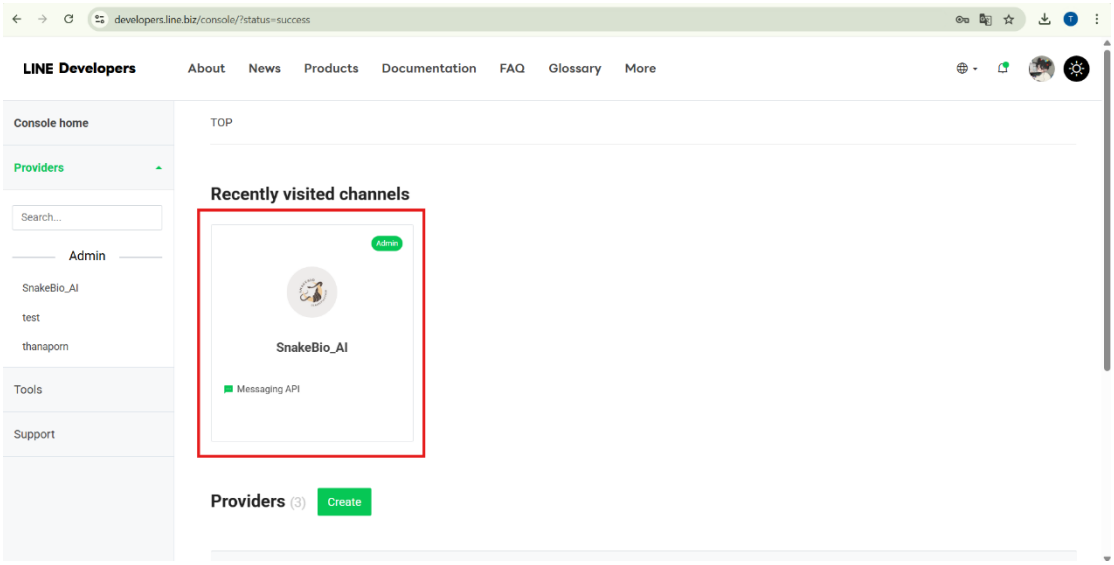
1.2 การเข้าสู่ระบบ



ภาพที่ ก.13 หน้าเว็บไซค์ Log in Console

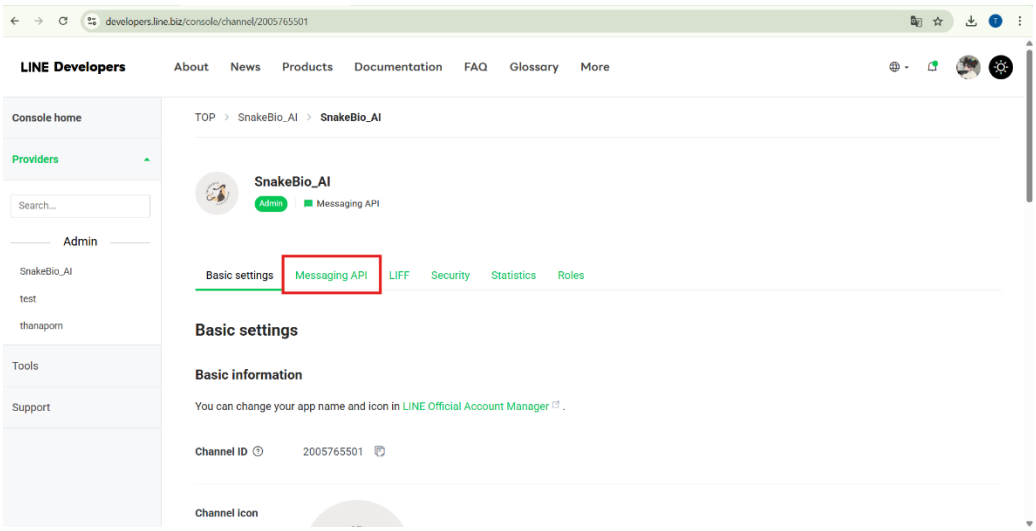
จากภาพที่ ก.13 แอดมินสามารถ Log in Console ได้ด้วย LINE ที่เชื่อมกับ LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู โดยกรอก Email และ รหัสผ่าน

1.3 การตั้งค่า LINE Developers



ภาพที่ ก.14 หน้าเว็บไซค์ LINE Developers

จากภาพที่ ก.14 เมื่อ Log in เสร็จสินจะแสดงหน้าดังภาพที่ ก.14 โดยจะแสดงLINE ที่เราได้เป็นแอดมิน และมีสิทธิในการแก้ไขการทำงานต่างๆ อย่างในภาพ จะเป็นไลน์แชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู หรือ SnakeBio_AI โดนแอดมินสามารถตั้งค่าการทำงานต่างๆ โดยคลิกที่กรอบสีแดง จะแสดงหน้าเว็บไซค์ดังภาพที่ ก.15



ภาพที่ ก.15 หน้าLINE Developers Dashboard

จากภาพที่ ก.15 ภาพนี้แสดง LINE Developers Dashboard ในส่วนของการตั้งค่าช่องทาง (Channel) สำหรับ SnakeBio_AI โดยแบ่งออกเป็น 3 ส่วน ได้แก่

1.แถบเมนูด้านซ้าย (Sidebar) : แสดงรายการ Providers ที่คุณมีสิทธิ์เข้าถึง เช่น SnakeBio_AI, test, Thanaporn และ มีหมวด Tools และ Support สำหรับเข้าถึงเครื่องมือและการสนับสนุน

2.ส่วนเนื้อหาหลัก โดยมีแท็บการตั้งค่าหลัก ได้แก่

2.1 Basic settings (การตั้งค่าพื้นฐาน)

2.2 Messaging API (API สำหรับการส่งข้อความและโต้ตอบ)

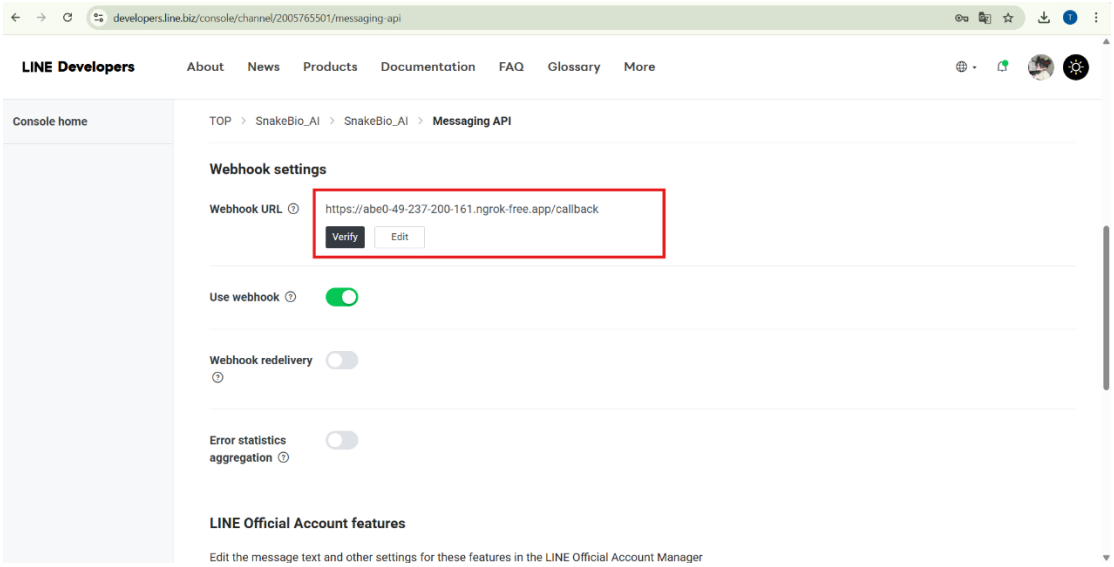
2.3 LIFF (LINE Front-end Framework)

2.4 Security (การรักษาความปลอดภัย)

2.5 Statistics (สถิติการใช้งาน)

2.6 Roles (การจัดการบทบาทผู้ดูแล)

ในส่วนนี้เราจะสนใจในส่วนของเนื้อหลัก โดยทชจะไปเลือก ที่แท็บการตั้งค่า Messaging API ที่กรอบสีแดงได้วงเอาไว้ เมื่อกดเลือก Messaging API จะแสดงหน้าเว็บไซค์ดังภาพที่ ก.16

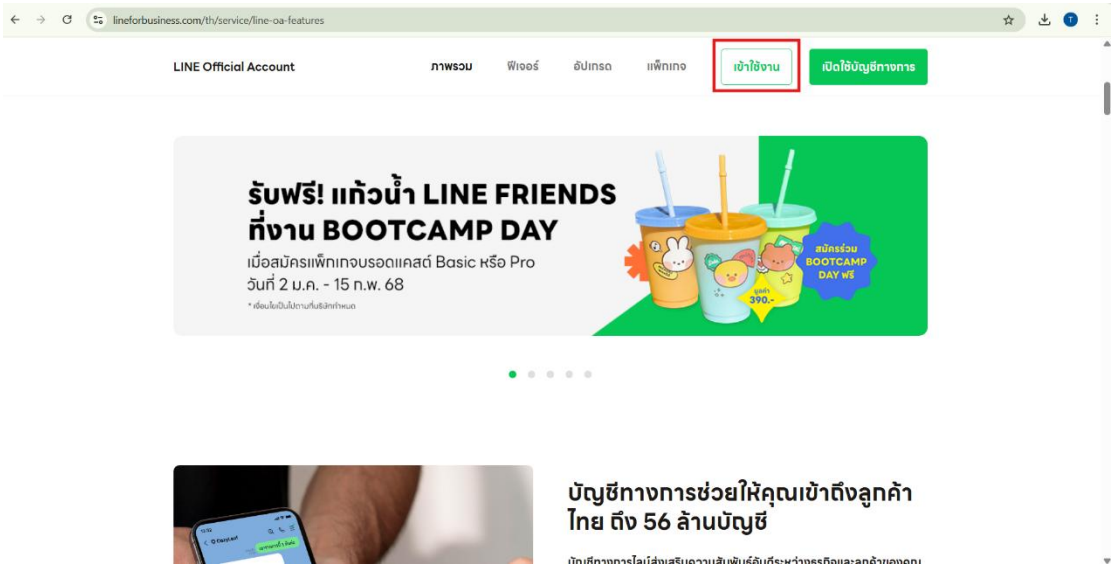


ภาพที่ ก.16 หน้า Webhook Settings ภายใน LINE Developers Console

จากภาพที่ ก.16 จะแสดงหน้า Webhook Settings ภายใน LINE Developers Console สำหรับช่องทาง (Channel) SnakeBio_AI ที่ใช้ Messaging API โดยในส่วนนี้ แอดมินสามารถเปลี่ยนแปลง public link ที่ได้เชื่อมกับการตั้งค่าการทำงานของโมเดลในการตรวจจับและจำแนกชนิดของงู

2. line official account manager แอดมินสามารถแก้ไขการทำงานต่างๆได้ภายในเว็บไซค์ https://lineforbusiness.com/th/service/line-oa-features

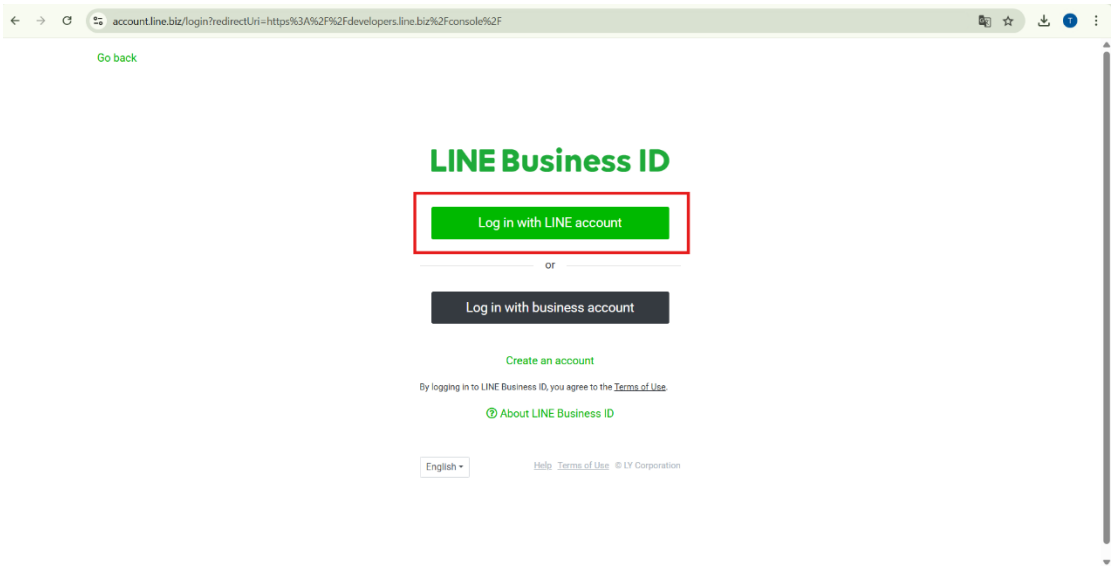
2.1 การเข้าสู่หน้าเว็บไซค์ line official account manager

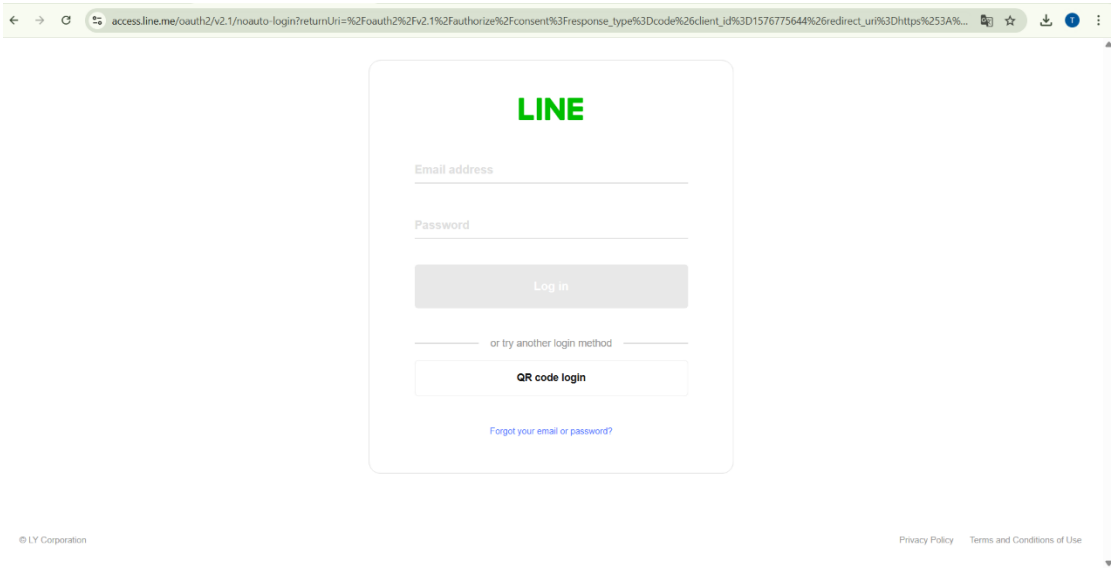


ภาพที่ ก.17 หน้าเว็บไซค์ line official account manager

จากภาพที่ก.17 แอดมินเมนู เข้าใช้งาน เพื่อเข้าสู่ระบบ

2.2 การเข้าสู่ระบบ

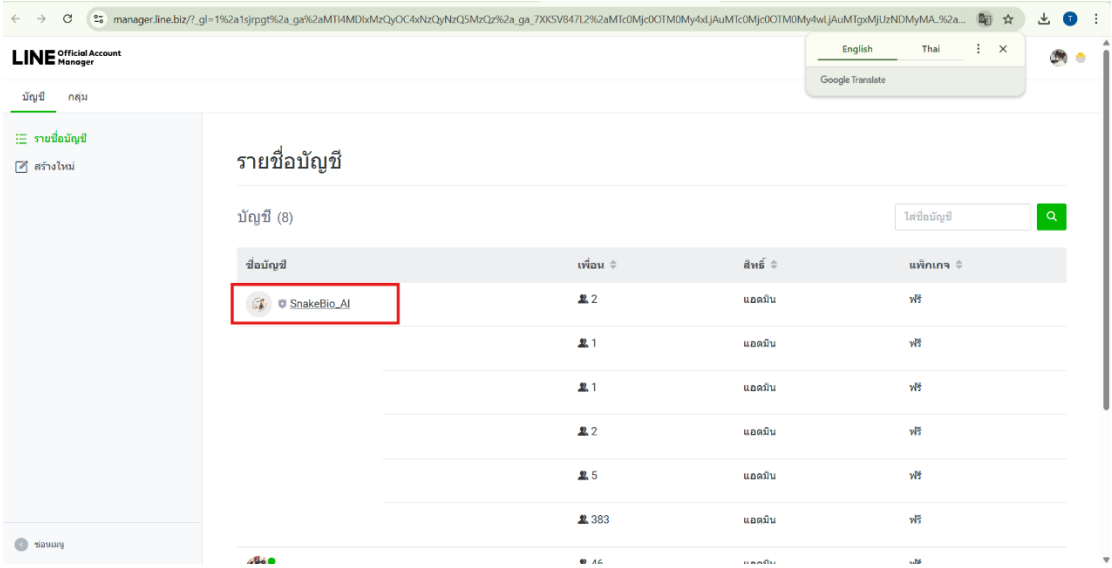




ภาพที่ ก.18 หน้าเว็บไซค์ Log in line official account

จากภาพที่ ก.18 แอดมินสามารถ Log in Console ได้ด้วย LINE ที่เชื่อมกับ LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู โดยกรอก Email และ รหัสผ่าน

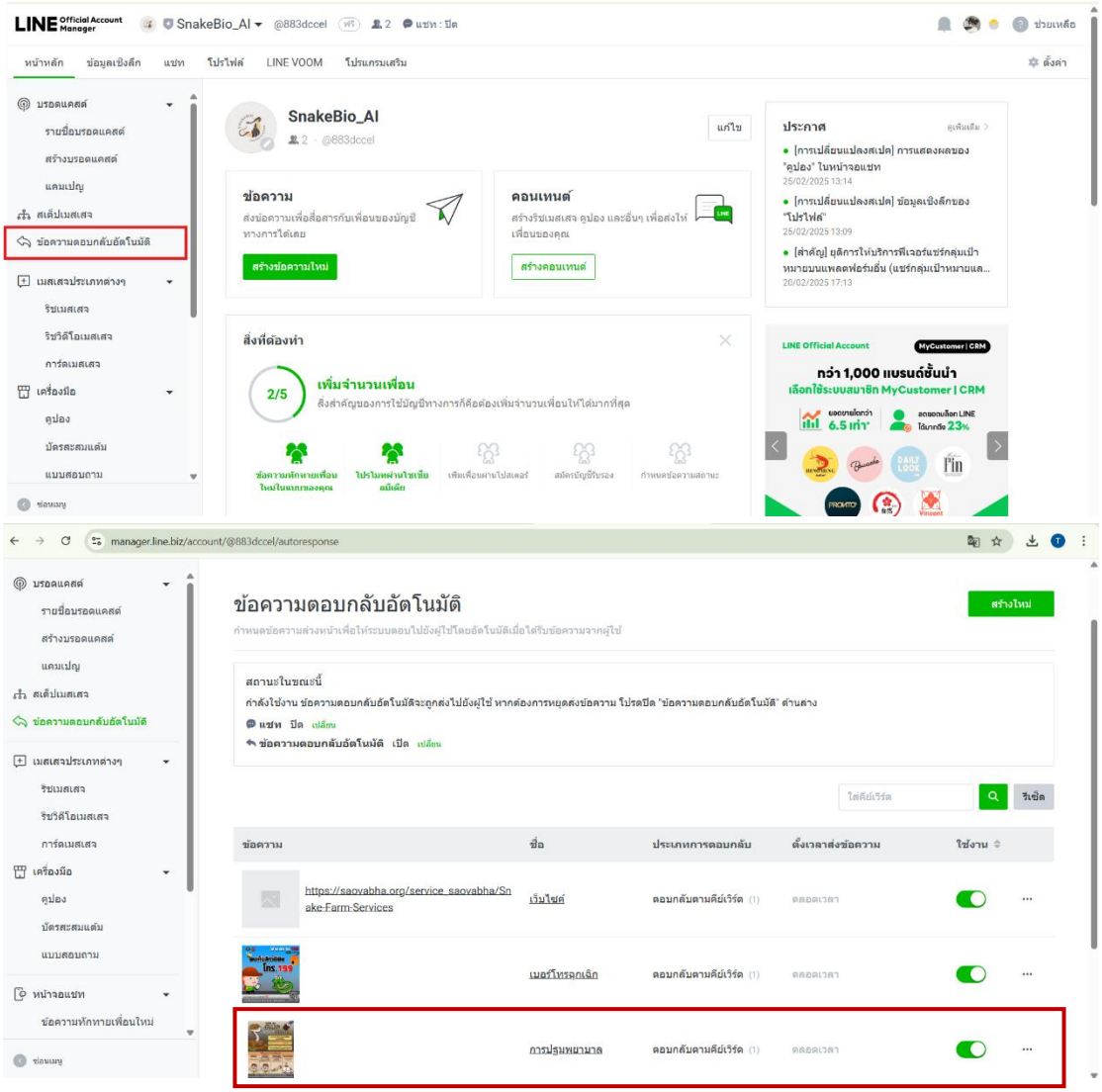
2.3 การตั้งค่า Line official account manager

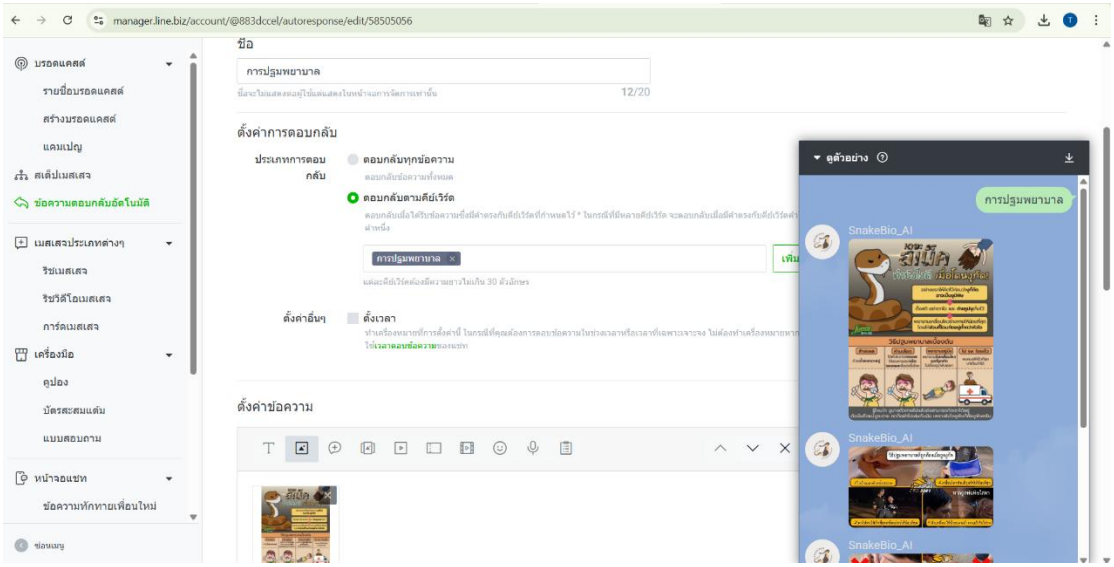


ภาพที่ ก.19 หน้าเว็บไซค์ Line official account manager

จากภาพที่ ก.19 เมื่อ Log in เสร็จสินจะแสดงหน้าดังภาพที่ ก.19 โดยจะแสดงLINE ที่เราได้เป็นแอดมิน และมีสิทธิในการแก้ไขการทำงานต่างๆ อย่างในภาพ จะเป็นไลน์แชทบอทอัจฉริยะเพื่อ

จำแนกชนิดของงู หรือ SnakeBio_AI โดนแอดมินสามารถตั้งค่าการทำงานต่างๆ โดยคลิกที่กรอบสีแดง จะแสดงหน้าเว็บไซค์ดังภาพที่ ก.20





ภาพที่ ก.20 การตั้งค่าข้อความตอบกลับอัตโนมัติ

จากภาพที่ ก.20. จะเป็นขั้นตอนการตั้งค่าข้อความตอบกลับอัตโนมัติ โดยเริ่มแรก กดเลือกในแท็บเมนูมุมซ้ายมือ ที่มีชื่อว่า “ข้อความตอบกลับอัตโนมัติ”เมื่อกดเลือเมนูเสร็จสิ้นจะข้อความตอบกลับอัตโนมัติที่ได้ตั่งค่าเอาไว้ หากต้องการแก้ไข ให้กดคลิกเลือกในส่วนที่ต้องการแก้ไข อย่างในตัวอย่างนี้จะแก้ไขในส่วนของ การปฐมพยาบาลเบื้องต้น เมื่อกดเลือกเสร็จสิ้น จะแสดงหน้าตามภาพที่ ก. โดยจะแสดงการตั้งค่าต่างๆโดยจะมี 3 อย่าง ได้แก่

1.ประเภทการตอบกลับ โดยจะแบ่งออกเป็น

2 ประเภท คือ ตอบกลับทุกข้อความ และ ตอบกลับตามคีย์เวิร์ด 2.การตั่งค่าอื่นๆ ในส่วนนี้จะสามารถตั้งค่าเวลาในการตอบกลับได้

3.การตั้งค่าข้อความในส่วนนี้จะมีข้อความตอบกลับไปยังผู้ใช้งาน โดยสามารถตอบได้ทั้งหมด 10 อย่าง ได้แก่

3.1 ข้อความตัวอักษร (Text Message)

3.2 รูปภาพ (Image Message)

3.3 Rich Message

3.4 Flex Message

3.5 วิดีโอ (Video Message)

3.6 Rich Video Message

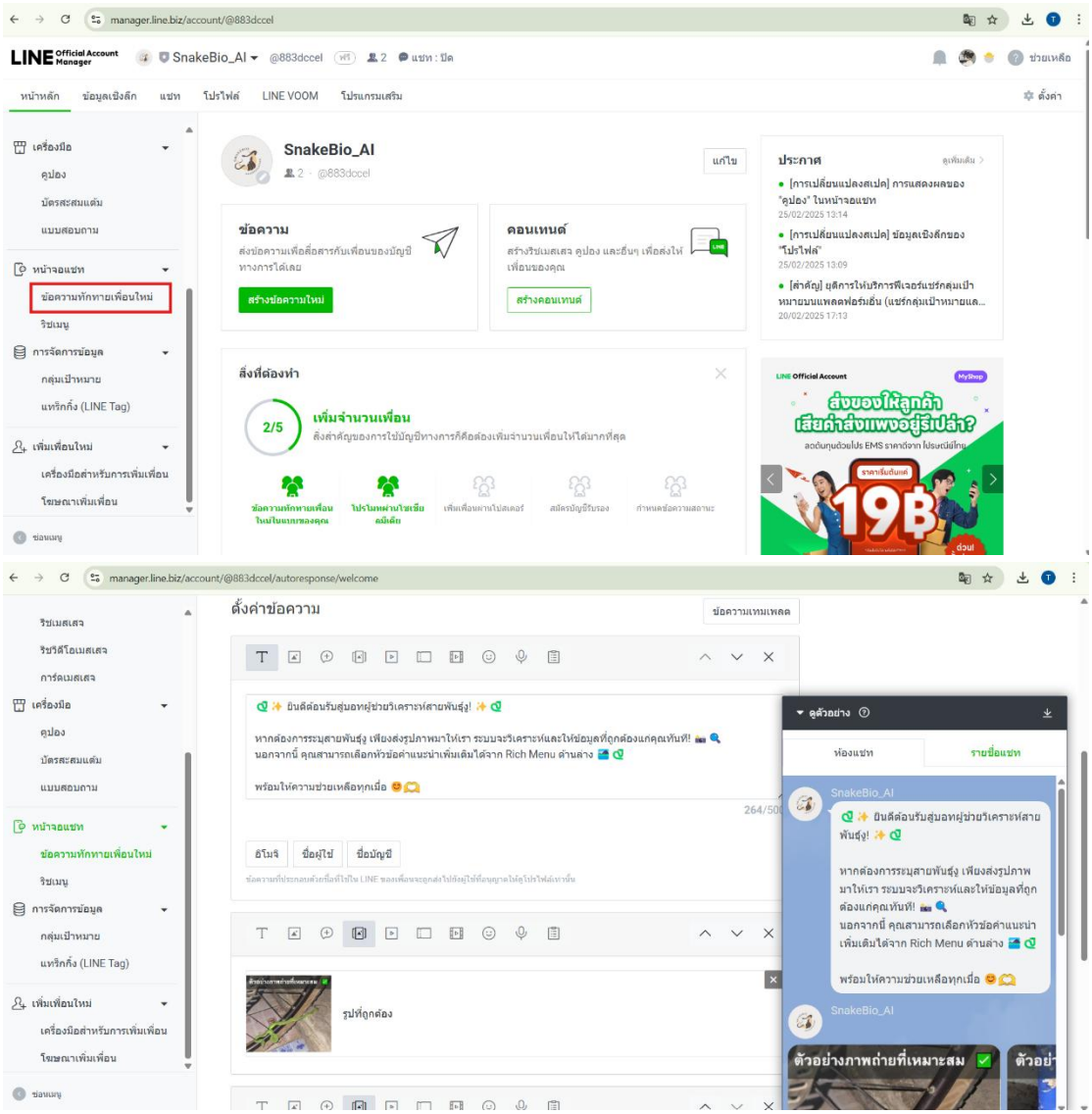
3.7 สติกเกอร์ (Sticker Message)

3.8 สติกเกอร์ (Sticker Message)

3.9 ไฟล์เสียง (Audio Message)

3.10 ปุ่มตัวเลือก (Template Message: Buttons, Confirm, Carousel)

โดยในการตั้งค่าสำหรับการปฐมพยาบาลนั้นจะเป็นการตอบกลับด้วยรูปภาพที่ได้มีรายละเอียดเกี่ยวกับการปฐมพยาบาลเรียบร้อย



ภาพที่ ก.21 การตั้งค่าข้อความทักทายเพื่อนใหม่

จากภาพที่ ก.21 จะเป็นขั้นตอนการตั้งค่าข้อความทักทายเพื่อนใหม่ โดยเริ่มแรก กดเลือกในแท็บเมนูมุมซ้ายมือ ที่มีชื่อว่า “ข้อความทักทายเพื่อนใหม่”เมื่อกดเลือกเมนูเสร็จสิ้นจะแสดงข้อความทักทายเพื่อนใหม่ที่ได้ตั้งค่าเอาไว้ โดยการตั้งค่าข้อความในส่วนนี้จะมีข้อความตอบกลับไปยังผู้ใช้งาน โดยสามารถตอบได้ทั้งหมด 10 อย่าง ได้แก่

1. ข้อความตัวอักษร (Text Message)
2. รูปภาพ (Image Message)

3. Rich Message

4. Flex Message

5. วิดีโอ (Video Message)

6. Rich Video Message

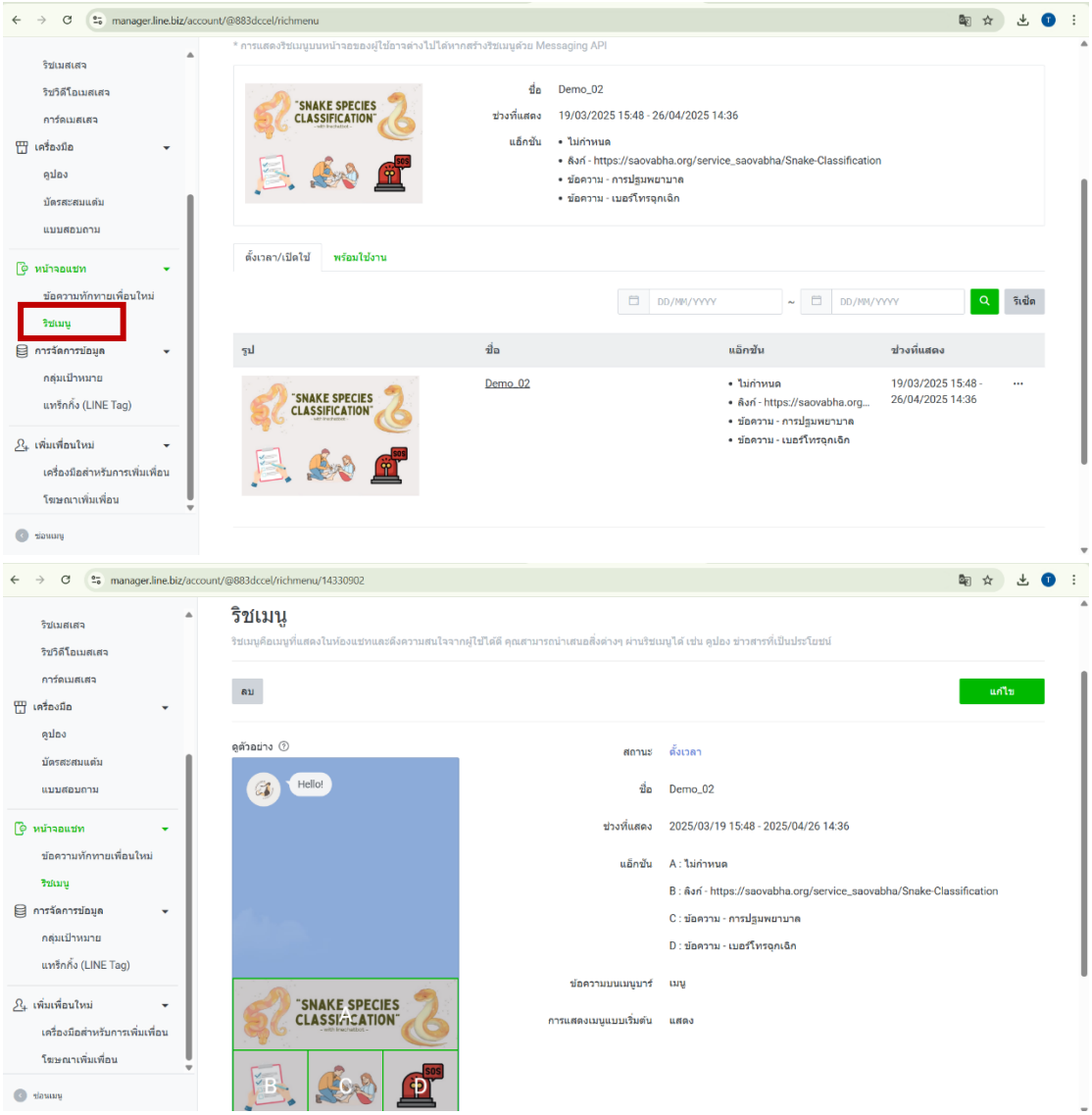
7. สติกเกอร์ (Sticker Message)

8. สติกเกอร์ (Sticker Message)

9. ไฟล์เสียง (Audio Message)

10. ปุ่มตัวเลือก (Template Message: Buttons, Confirm, Carousel)

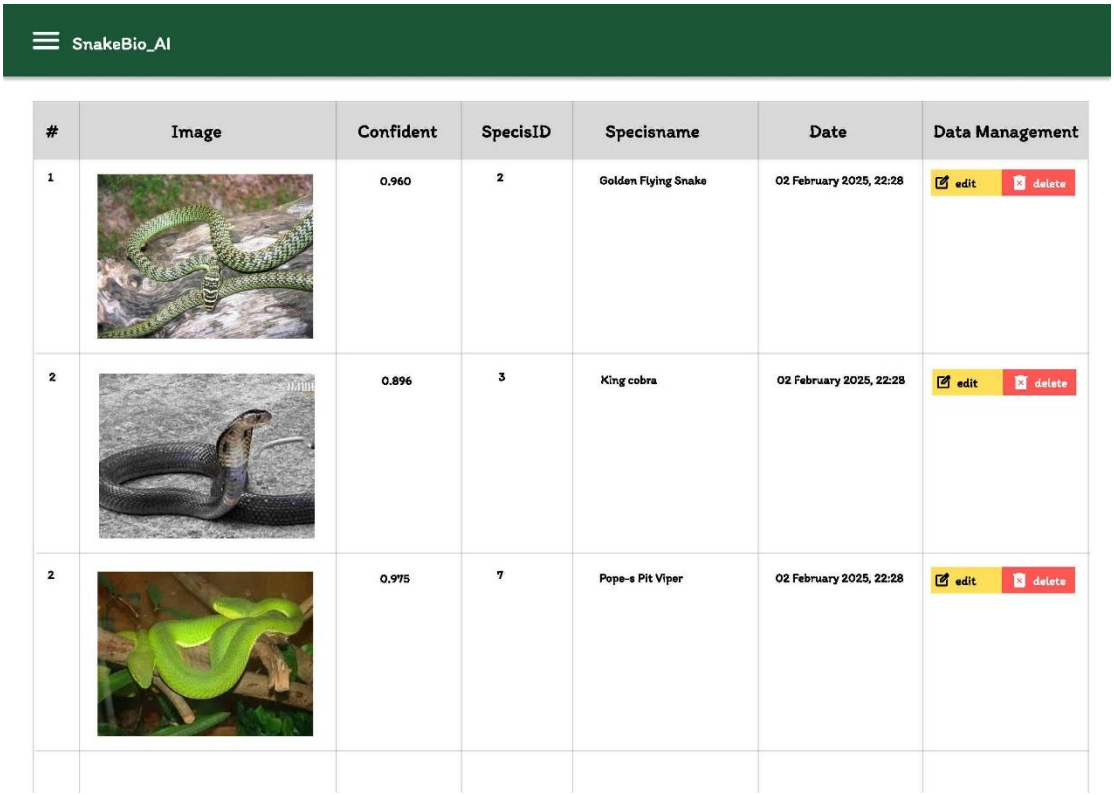
นอกเหนือจากนี้ยังสามารถกดเลือกอิโมจิ,ผู้ใช้ และ ชื่อบัญชีได้



ภาพที่ ก.22 การตั้งค่าริชเมนู

จากภาพที่ ก.22 จะเป็นขั้นตอนการตั้งค่าริชเมนูโดยเริ่มแรก กดเลือกในแท็บเมนูมุมซ้ายมือ ที่มีชื่อว่า “ริชเมนู”เมื่อกดเลือกเมนูเสร็จสิ้นจะแสดงริชเมนูที่ได้ตั้งค่าเอาไว้ โดยการตั้งค่าริชเมนูในส่วนนี้จะสามารถแก้ไขริชเมนูที่มีอยู่หรือจะสร้างใหม่ก็ได้ โดยในส่วนของหน้าการตั้งค่าจะแสดงรายละเอียดดังนี้ 1.สถานะ 2.ชื่อ 3.ช่วงที่แสดง 4.แอ็กชัน 5.ข้อความบนเมนูบาร์ 6.การแสดงข้อมูลเริ่มต้น

3.Web site แอดมินสามารถดูผลการวิเคราะห์สายพันธุ์จากการที่ผู้ใช้งานส่งผ่านมาทางไลน์ได้



ภาพที่ ก.23 หน้า Web site

จากภาพที่ ก.23 จะเป็นหน้าแสดงหน้า Web site สามารถดูผลการวิเคราะห์สายพันธุ์จากการที่ผู้ใช้งานส่งผ่านมาทางไลน์ได้อีกทั้งยังสามารถกดเลือกแสดงเฉพาะสายพันธุ์ที่สนใจ โดยกดเลือกที่ ไอคอน เบอร์เกอร์ จะแสดงสายพันธุ์ทั้ง 10 สายพันธุ์ให้เลือก

ภาคผนวก ข

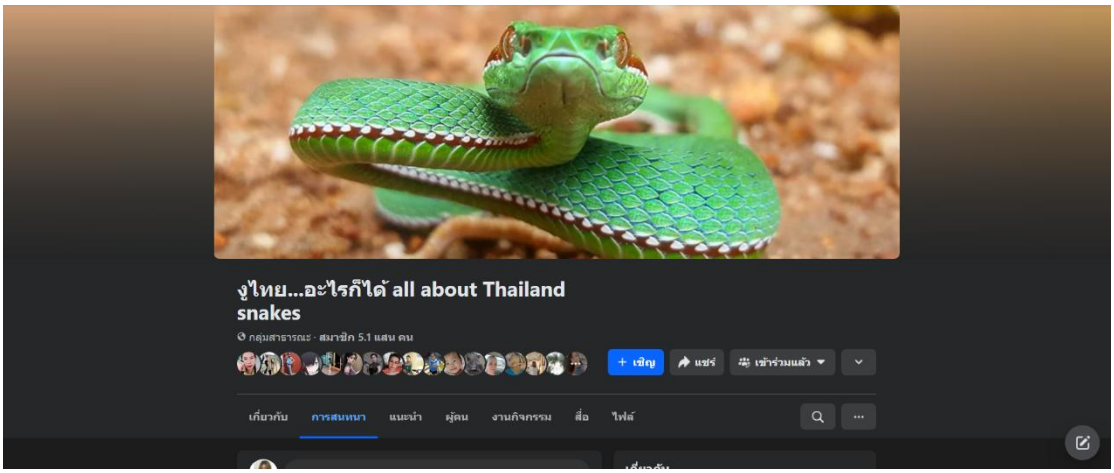
การเชื่อมต่อและโปรแกรมส่วนสำคัญของระบบ

ระบบแชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู(Snake species classification on line chatbot) ถูกพัฒนาขึ้นโดยใช้สถาปัตยกรรมแบบ Two-Stage AI Model Pipeline ในรูปแบบ Client-Server ซึ่งมีการใช้เทคนิค Deep Learning และการประมวลผลแบบกระจายบน Cloud เพื่อให้บริการผ่าน LINE Chatbot

1. การเทรนโมเดล

1.1 การจัดเตรียมชุดข้อมูล(Dataset)

การรวบรวมข้อมูลเกี่ยวกับท่าทางของงูและสายพันธุ์ต่าง ๆ เพื่อใช้ในการฝึกโมเดล AI ในโครงงานได้รวมรวบผ่าน google,กลุ่ม Facebook และ สวนงู สถานเสาวภา สภากาชาดไทย



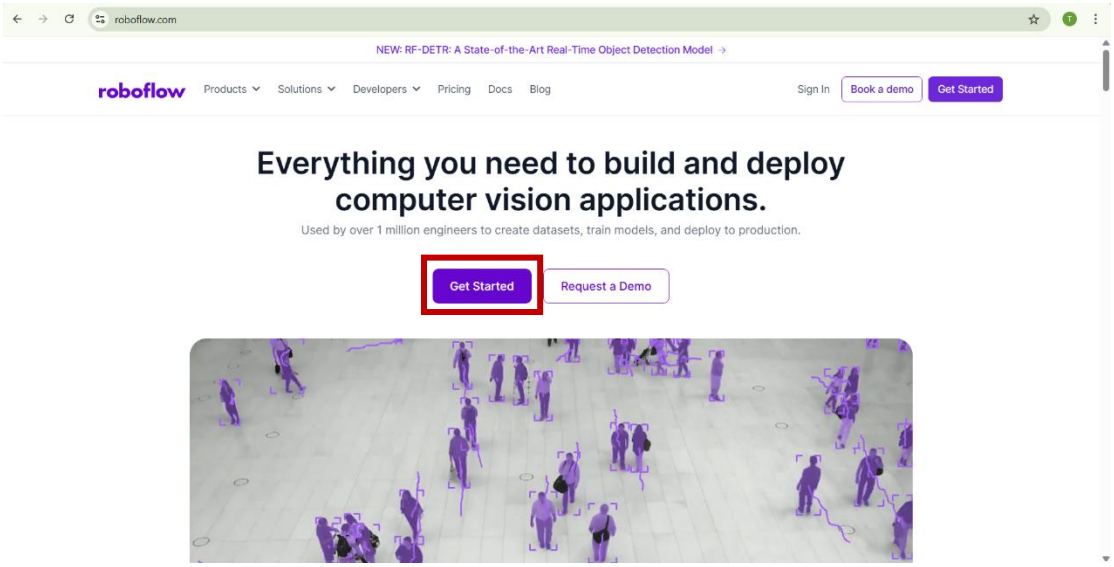
ภาพที่ ข.1 หน้าจอ กลุ่ม Facebook งูไทย...อะไรก็ได้ all about Thailand snakes

ที่มา : https://web.facebook.com/share/g/1Bq5TC43Ue/

1.2 การกำหนดป้ายกำกับในแต่ละภาพ

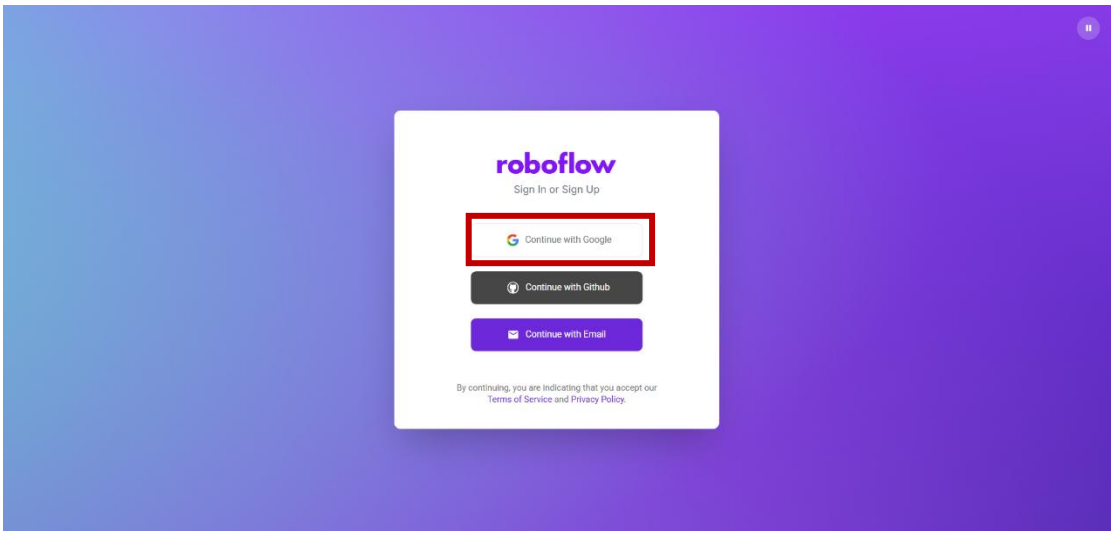
การกำหนดป้ายกำกับ โดย Roboflow เป็นแพลตฟอร์มที่ใช้งานผ่านเว็บบราว์เซอร์ https://roboflow.com/

1.2.1 การเข้าสู่ระบบ Roboflow

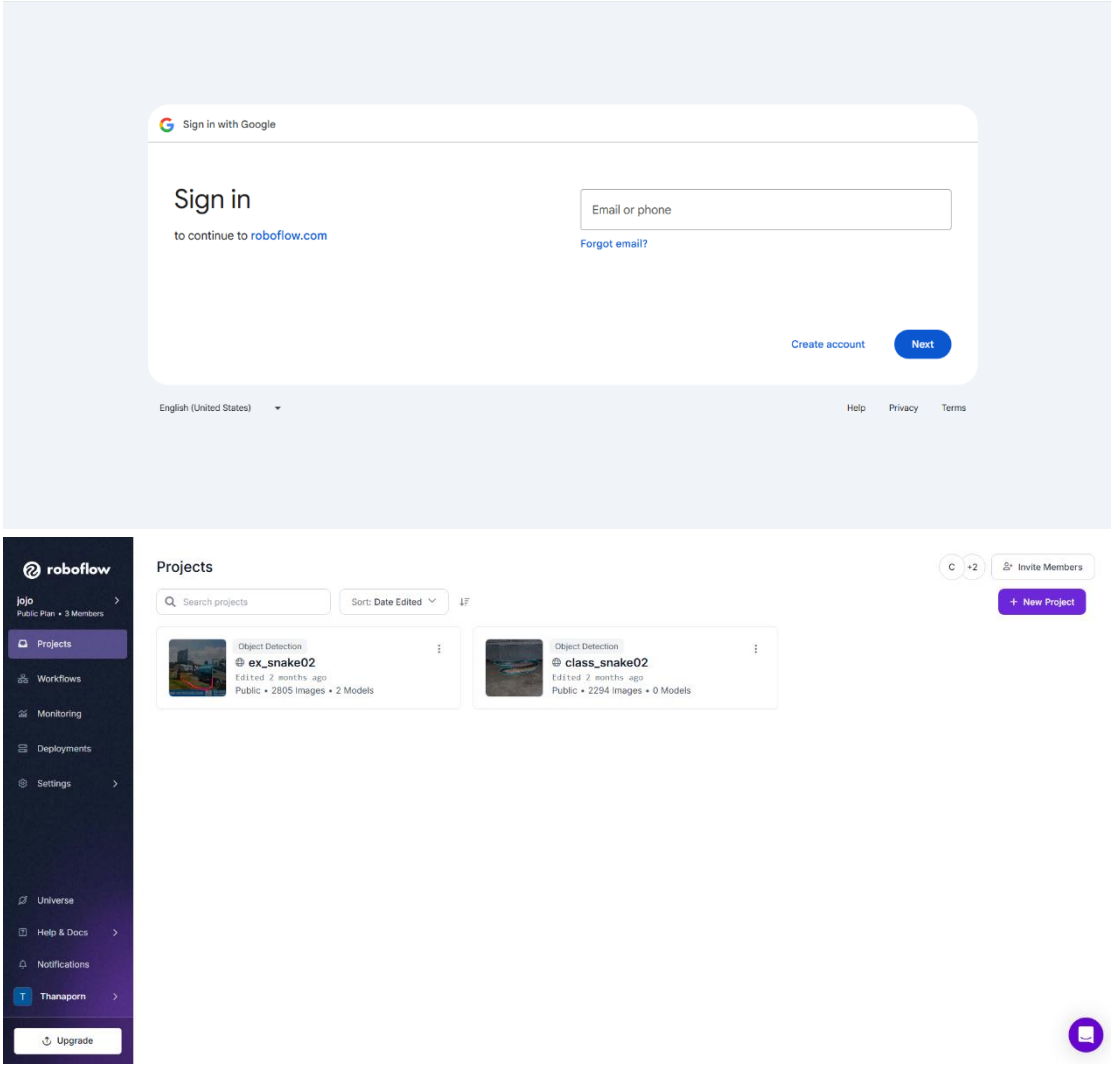


ภาพที่ ข.2 หน้าเว็บบราว์เซอร์ของ roboflow

จากภาพที่ ข. จะเป็นหน้าเว็บบราว์เซอร์ของ roboflow โดยแสดงดังภาพที่ ข. ให้กดคลิกที่กรอบสีแดง เพื่อlog in เข้าใช้งาน ดังภาพที่ ข.3



ภาพที่ ข.3 หน้า log in ของ roboflow

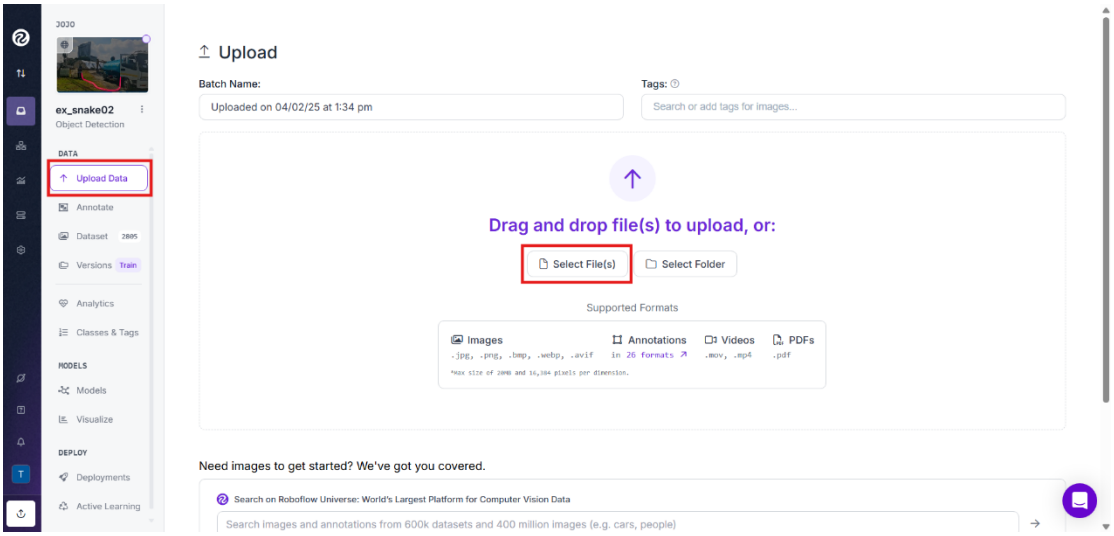


ภาพที่ ข.4 หน้าจอของแพลตฟอร์ม Roboflow

จากภาพที่ ข.3 จะเป็นหน้า log in โดยมีให้เลือกทั้งหมด 3 ตัวเลือก ได้แก่ 1.Google 2.Github 3.Email ในโครงงานนี้เราจะlog in ด้วย google โดยเมื่อคลิกเลือกเรียบร้อยให้กรอก email และ password ของ google ที่ได้เชื่อมไว้ เมื่อเสร็จสิ้นจะแสดงดังภาพที่ ข.4 โดยในภาพที่ ข.4 จะแสดงหน้าจอของแพลตฟอร์ม Roboflow ที่แสดงโปรเจกต์ที่มีอยู่ในบัญชีของผู้ใช้ ได้แก่ 1.ex_snake02 คือ โปรเจกต์ในการทำdataset สำหรับโมเดลในการตรวจจับงู 2.class_snake02 คือ โปรเจกต์ในการทำdataset สำหรับโมเดลในการจำแนกสายพันธุ์งู นอกจากนี้ในเมนูด้านซ้าย ผู้ใช้สามารถเข้าถึงส่วนต่างๆ เช่น Projects, Workflows, Monitoring, Deployments, Settings และอื่นๆ นอกจากนี้ ยังมีปุ่ม "New Project" สำหรับสร้างโปรเจกต์ใหม่ และตัวเลือก "Invite Members" สำหรับเชิญสมาชิกเข้าร่วมทีม

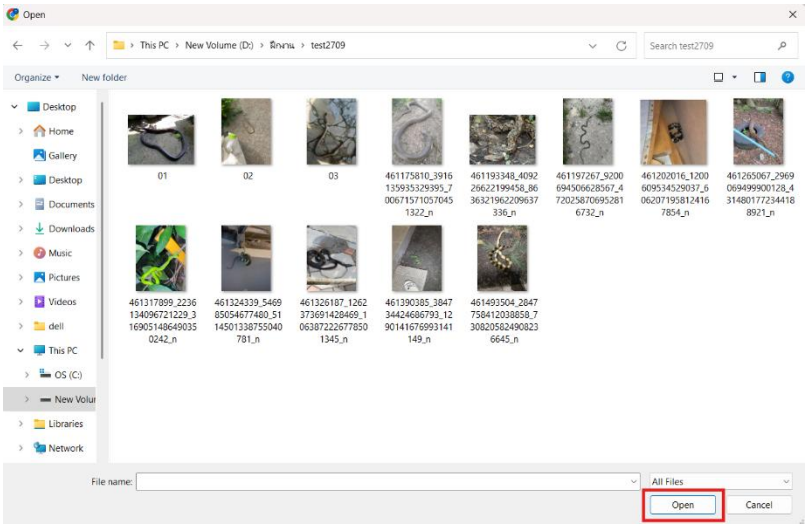
1.2.2 การอัปโหลดรูปภาพลง Roboflow

เริ่มแรกให้กดเลือกโปรเจกต์ที่ต้องการอัปโหลดรูปภาพ โดยในตัวอย่างนี้จะเลือกโปรเจกต์ที่มีชื่อว่า ex_snake02 เมื่อคลิกเข้าไปจะได้ดังภาพที่ ข.5



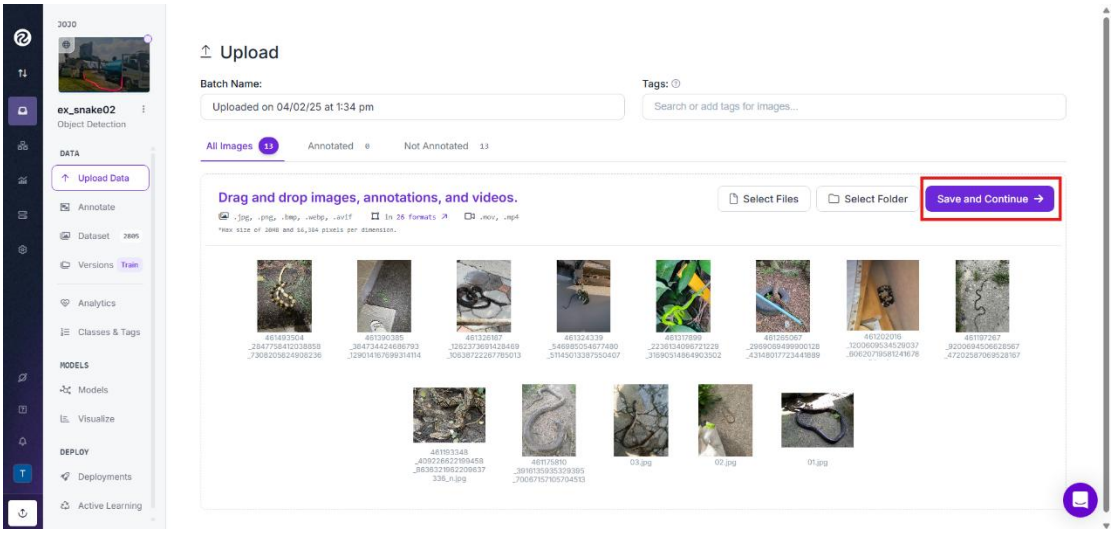
ภาพที่ ข.5 หน้าจอในการUpload Data

จากภาพที่ ข.5 จะแสดงรายละเอียดต่างๆภายในโปรเจกต์ โดยสิ่งที่ต้องทำอย่างแรกคือ กดเลือกเมนูที่มีชื่อว่า “Upload Data” และเมื่อกดเลือกเมนูเสร็จเรียบร้อยจะมีปุ่ม ให้เลือกระหว่าง Select File(s) และ Select Folder โดยเราจะเลือก Select File(s) เพื่อที่จะเลือกรูปภาพที่ต้องการอัปโหลดลงมา เมื่อกดคลิกปุ่ม Select File(s) จะขึ้นหน้า pop up ตามภาพที่ ข.6



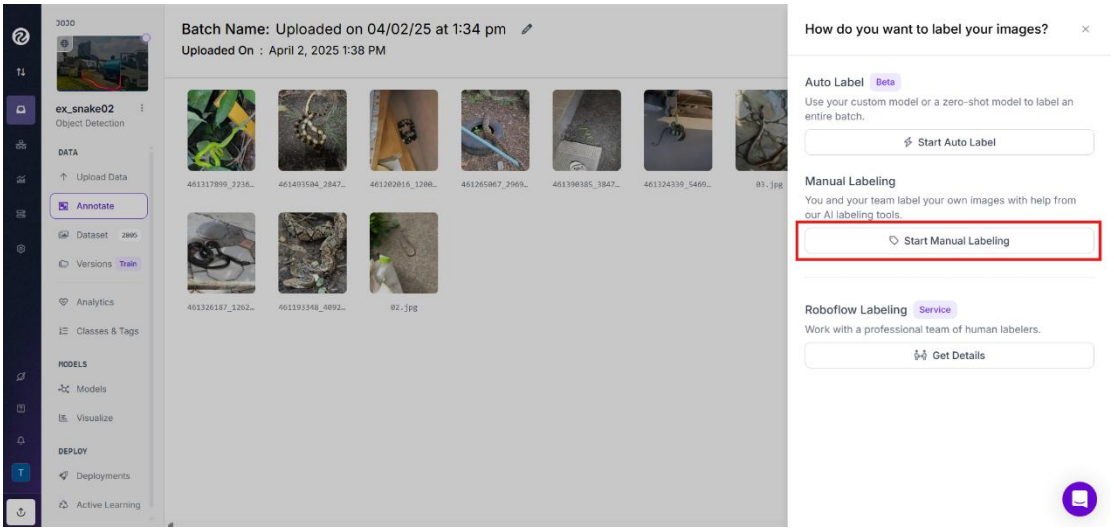
ภาพที่ ข.6 หน้าต่าง Open File Dialog

จากภาพที่ ข.6 ให้เราเลือกโฟลเดอร์ที่เราได้เก็บรุปภาพที่จะนำมาทำ dataset ในตัวอย่างเราจะเก็บรูปภาพไว้ที่ D:\ฝึกงาน\test2709 และกดเลือกรูปภาพที่ต้องการอัปโหลด และกดปุ่ม Open เมื่อกดเสร็จสิ้น จะมีการนำรูปภาพที่เราเลือก มาอัปโหลดลง roboflow ดังภาพที่ ข.7



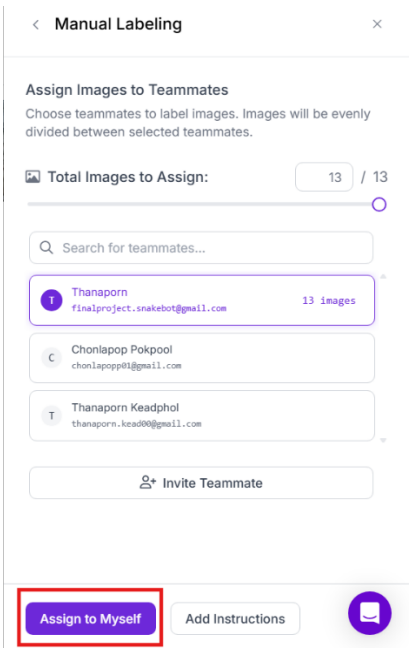
ภาพที่ ข.7 หน้าจอแสดงข้อมูลอัปโหลด

จากภาพที่ ข.7 ให้ตรวจสอบว่ารูปภาพที่จะอัปโหลดครบถ้วนหรือไม่ หากไม่ครบถ้วนและต้องการเพิ่ม สามารถกดปุ่ม Select Flies เพื่อที่กดเลือกรูปภาพเพิ่มเติมได้ แต่หากครบถ้วนสมบูรณ์แล้ว ให้กดปุ่ม Save and Continue เพื่อที่จะ save รูปภาพลง roboflow จะได้ดังภาพที่ ข.8



ภาพที่ ข.8 Sidebar How do you want to label your images?

จากภาพที่ ข.8 Roboflow เราต้องการที่จะติดป้ายกำกับให้กับรูปภาพอย่าง โดยมี 3 ตัวเลือก 1.Auto Label จะใช้โมเดล AI เพื่อติดป้ายกำกับอัตโนมัติ 2.Manual Labeling ผู้ใช้ต้องติดป้ายกำกับด้วยตัวเอง (ปุ่ม Start Manual Labeling) 3.Roboflow Labeling (Service) จะทำการจ้างทีมงานของ Roboflow ให้ช่วยติดป้ายกำกับ ในโครงงานนี้เราจะเลือกใช้ Manual Labeling เมื่อกดเลือกเสร็จสิ้นจะขึ้นดังภาพที่ ข.9

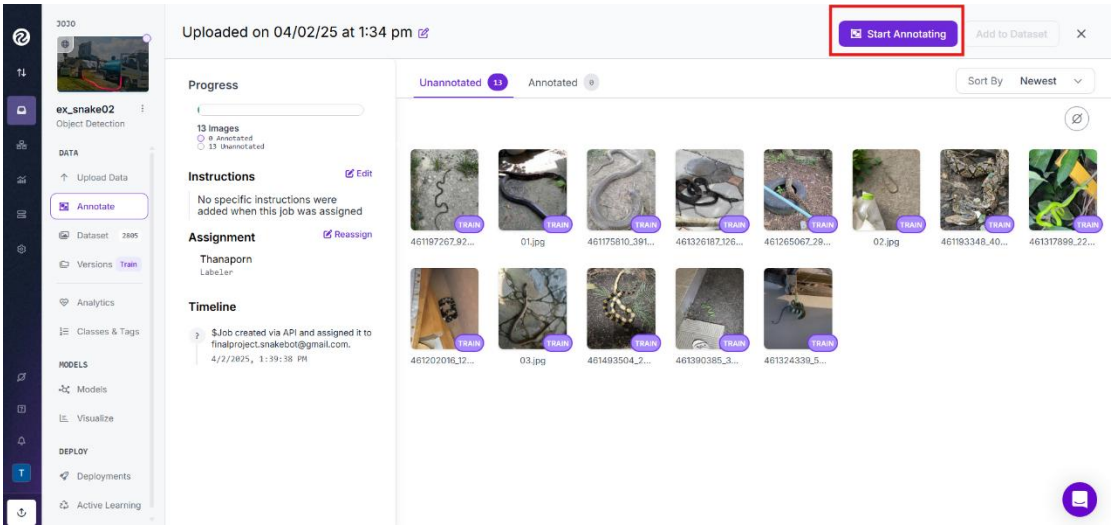


ภาพที่ ข.9 Sidebar Manual Labeling

จากภาพที่ ข.9 จะแสดงรายละเอียดโดยจะมีจำนวนรูปภาพที่ต้องทำการติดป้ายกำกับ และยังสามารถกดเลือกได้ว่าจะแจกจ่ายให้กับคนในทีมทำกี่รูป เมื่อเสร็จสิ้นกดปุ่ม Assign to Myself ถือเป็นการเสร็จสิ้นในการอัปโหลดรูปภาพลง roboflow เรียบร้อย

1.2.3 การกำหนดป้ายกำกับในแต่ละภาพ

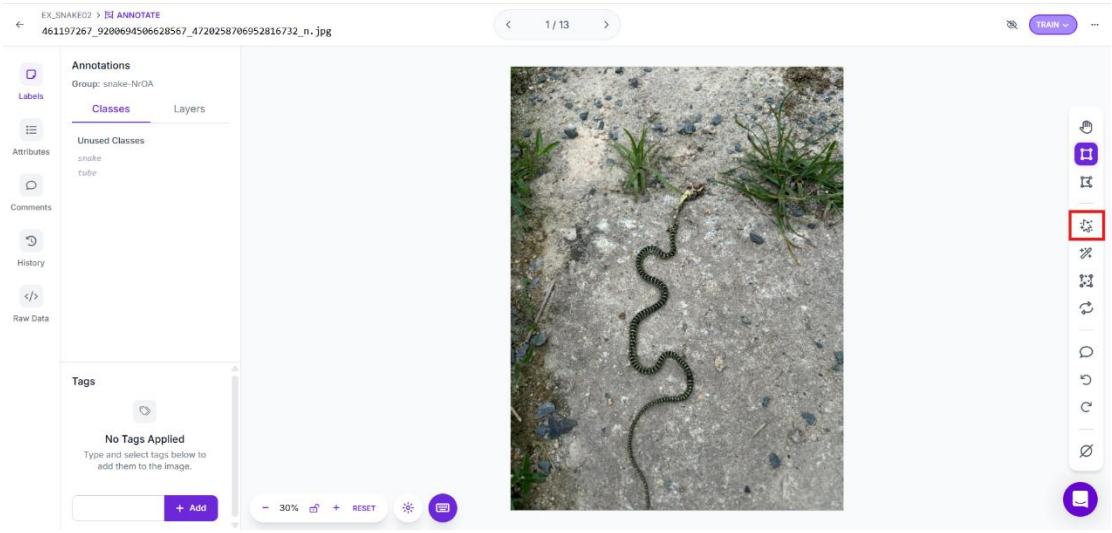
เมื่อเราทำขั้นตอน 1.2.2 เสร็จสิ้นจะแสดงหน้าจอดังภาพที่ ข.10



ภาพที่ ข.10 หน้าจอแสดงรายละเอียดข้อมูลที่จะทำการ Annotate

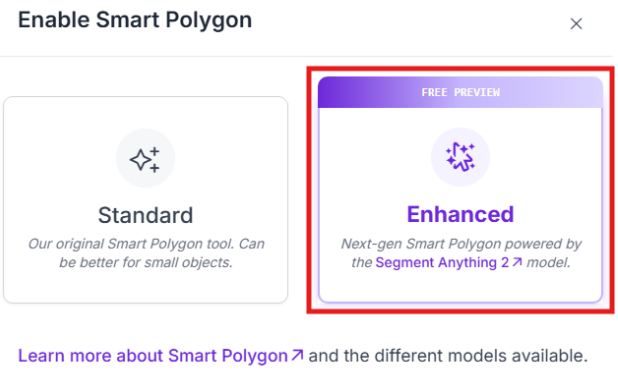
จากภาพที่ ข.10 จะแสดงรายละเอียดข้อมูลที่จะทำการ Annotate โดยให้ดูที่แถบเมนูจะมี 2 แถบเมนู คือ Unannotated และ Annotated หมายถึงรูปภาพที่ได้ทำการกำหนด label หรือยัง โดย

หากยังจะอยู่ใน Unannotated แต่หากกำหนดเรียบร้อยจะอยู่ใน Annotated โดยในภาพรูปภาพยังไม่ได้กำหนด label จะอยู่ใน Unannotated จำนวน 13 รูป โดยหากต้องการเริ่มต้นการกำหนด label ให้กดปุ่ม Start Annotated จะขึ้นแสดงหน้าจอดังภาพที่ ข.11



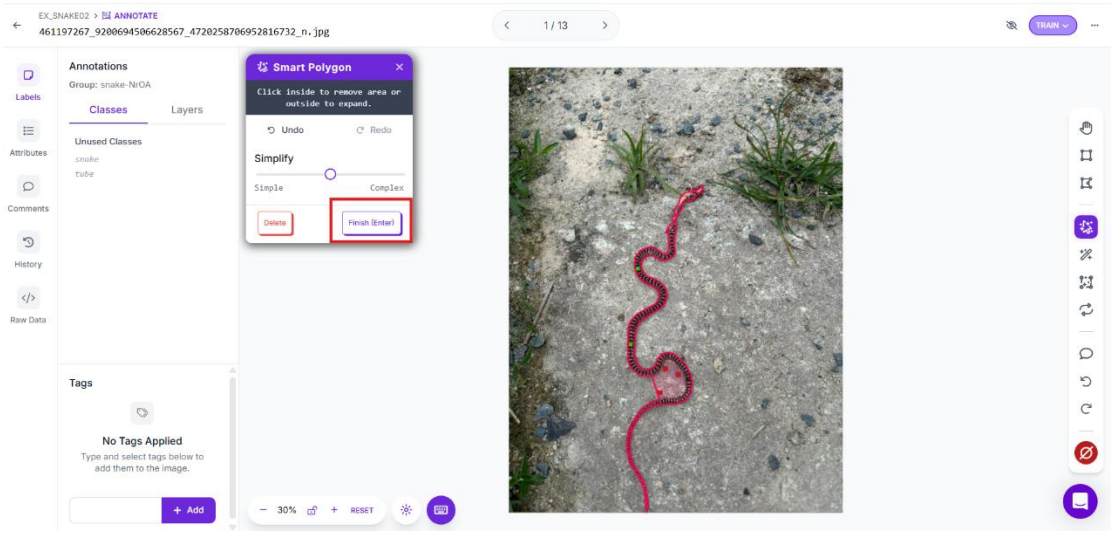
ภาพที่ ข.11 หน้าจอในการกำหนด label

จากภาพที่ ข.11 จะเป็นหน้าจอในการกำหนด label โดยจะมีแถบเมนูอยู่ฝั่งขวามือ โดยในโครงงานนี้เราจะในเครื่องมือที่มีชื่อว่า Smart Polygon (AI Labeling) หรือ  เมื่อเราคลิกที่ตัวเครื่องมือจะมีpop up ขึ้นมาดังภาพที่ ข.12



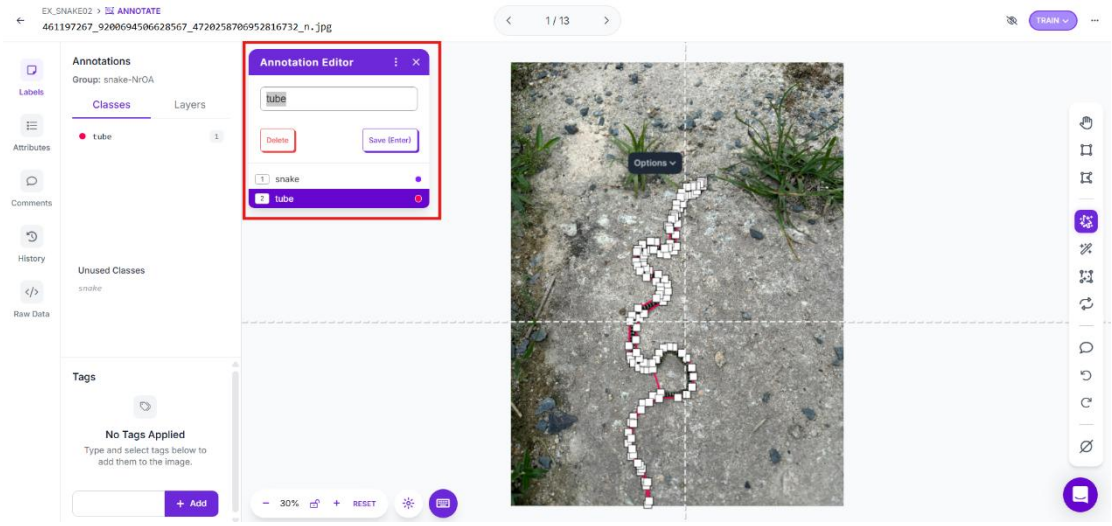
ภาพที่ ข.12 Pop up Enable Smart Polygon

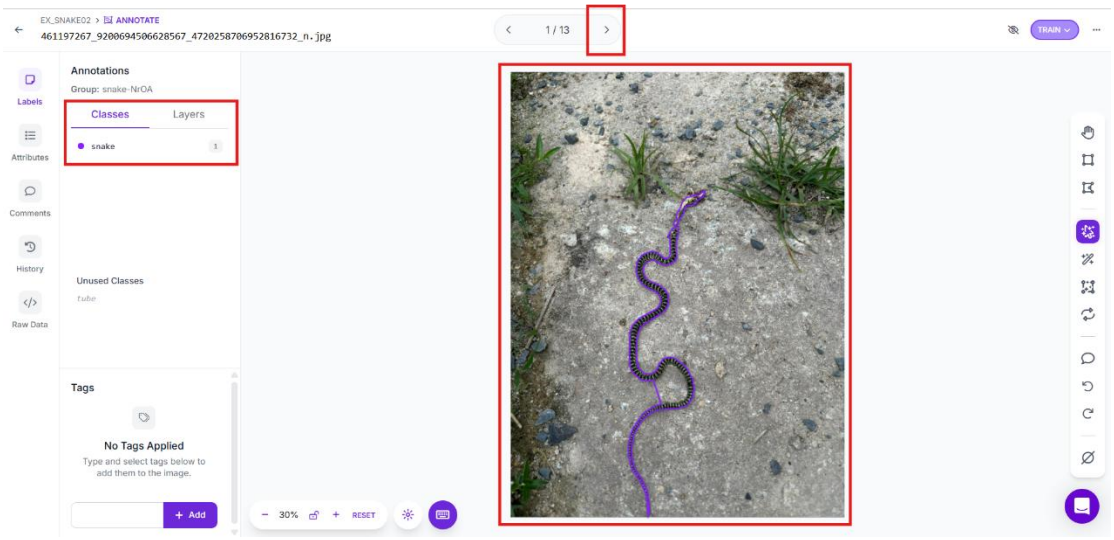
จากภาพที่ ข.12 ให้เราเลือก Enhanced เป็นเครื่องมือช่วย Annotate อัตโนมัติ ใน Roboflow โดยใช้ AI เพื่อสร้างขอบเขตรูปหลายเหลี่ยมรอบวัตถุ (Polygon) แทนการวาด Bounding Box แบบธรรมดา เมื่อเราคลิกเลือก Enhanced เสร็จสิ้น คลิกส่วนไหนของรูปภาพก็ได้ที่เราต้องการกำหนด label เพื่อติดป้ายกำกับว่าเป็น snake หรือ tube เมื่อเสร็จจะได้ดังภาพที่ ข.1.3



ภาพที่ ข.13 หน้าจอที่ได้ทำการกำหนด label

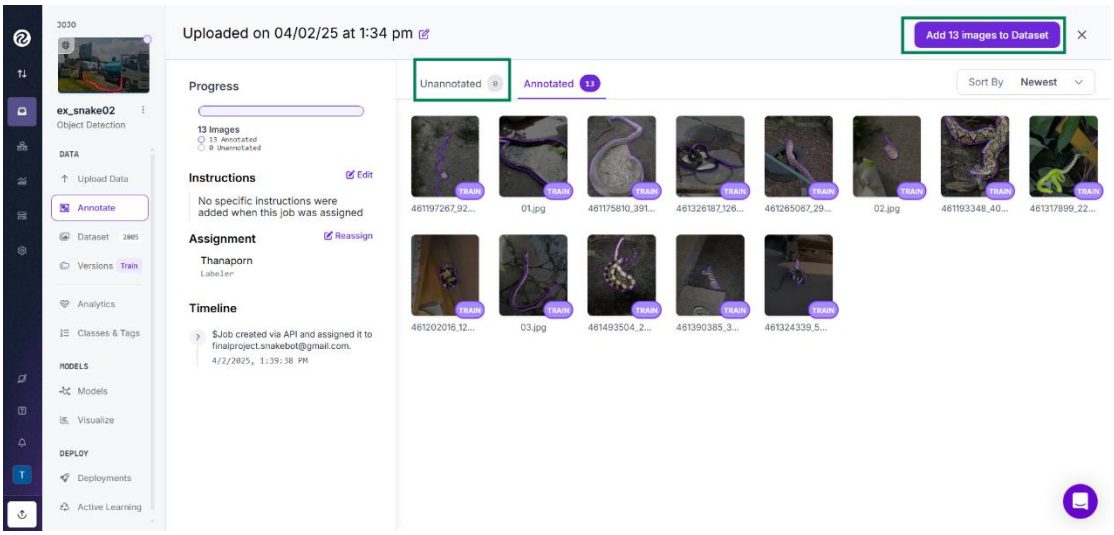
จากภาพที่ ข.13 เมื่อได้คลิกส่วนไหนของรูปภาพก็ได้ที่เราต้องการกำหนด label เสร็จสิ้นให้กดปุ่ม Finish หรือ Enter และเราสามารถเลือกclass ได้ว่าจะกำหนดเป็นclass อะไร อย่างในภาพที่ ข.13 เราจะเลือกเป็น snake เมื่อเลือกเสร็จสิ้นให้กด save ถือว่าเป็นเสร็จสิ้นในการกำหนดป้ายกำกับ





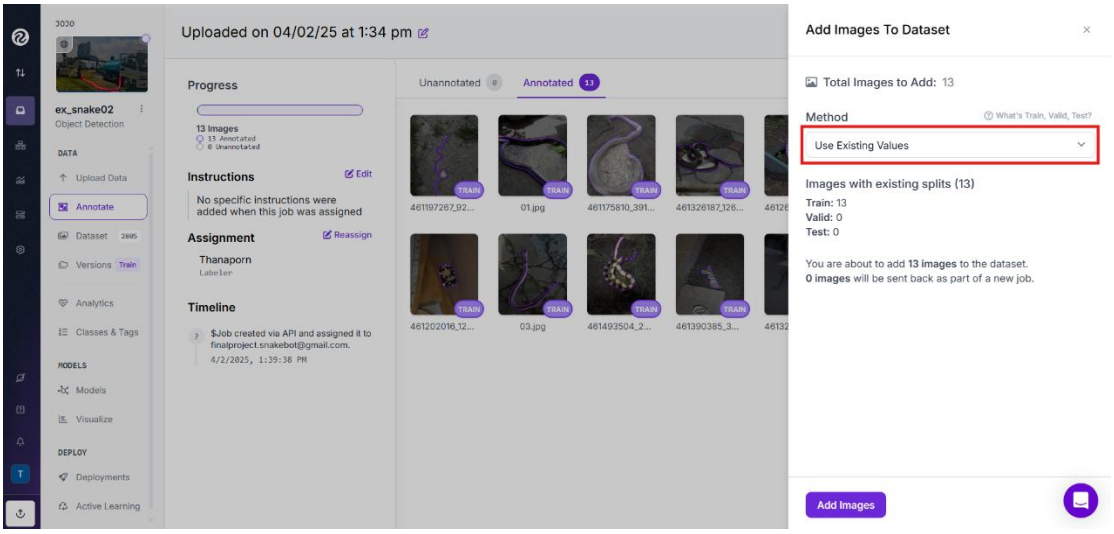
ภาพที่ ข.14 หน้าจอที่ได้ทำการกำหนด label

หลังจากที่เราได้ทำการกำหนดป้ายกำกับทุกรูปภาพเสร็จสิ้น จะแสดงดังภาพที่ ข.15



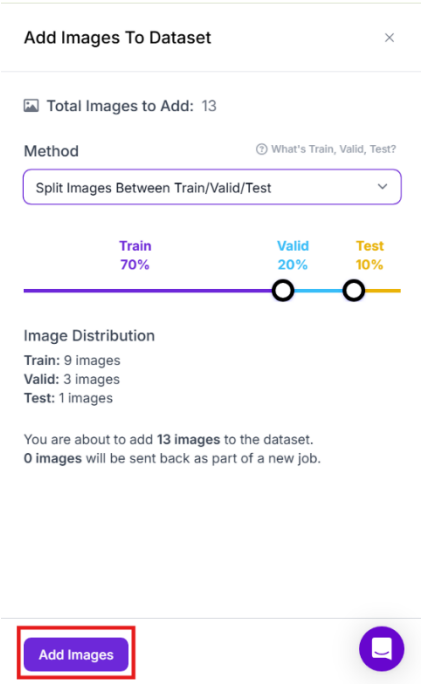
ภาพที่ ข.15 หน้าจอแสดงรายละเอียดข้อมูลที่ทำการ Annotate

จากภาพที่ ข.15 จะแสดงรายละเอียดข้อมูลที่ทำการ Annotate โดยให้ดูที่แถบเมนูจะมี 2 แถบเมนู คือ Unannotated และ Annotated หมายถึงรูปภาพที่ได้ทำการกำหนด label หรือยัง โดยหากได้ทำการกำหนด label เสร็จสิ้นเรียบร้อย ในส่วน Unannotated ของควรที่จะ 0 และ ส่วนของ Annotated 13 รูปดังภาพ เมื่อตรวจสอบเรียบร้อยว่าได้ทำการกำหนด label เสร็จสิ้นให้กดปุ่ม Add 13 images to Dataset จะมีsidebar ขึ้นดังภาพที่ ข.16



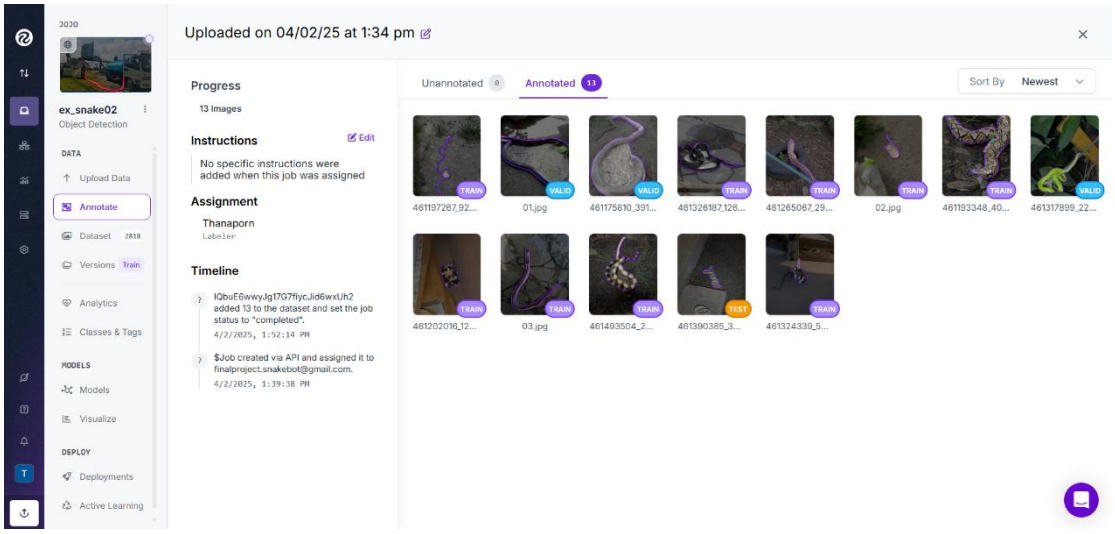
ภาพที่ ข.16 Sidebar Add Images To Dataset

จากภาพที่ ข.16 ใน Sidebar สามารถเลือกได้ว่ารูปภาพที่มีการกำหนด Label แล้วจะถูกเพิ่มลงในชุดข้อมูลใด โดยมีตัวเลือกดังนี้ 1.Split Images Between Train/Valid/Test 2.Add All Images to Training Set 3. Add All Images to Validation Set 4. Add All Images to Testing Set .สำหรับโปรเจกต์นี้ เราเลือกตัวเลือก "Split Images Between Train/Valid/Test" และกำหนดอัตราส่วนเป็น 70% Train / 20% Validation / 10% Test ดังภาพที่ ข.17



ภาพที่ ข.17 Sidebar Add Images To Dataset

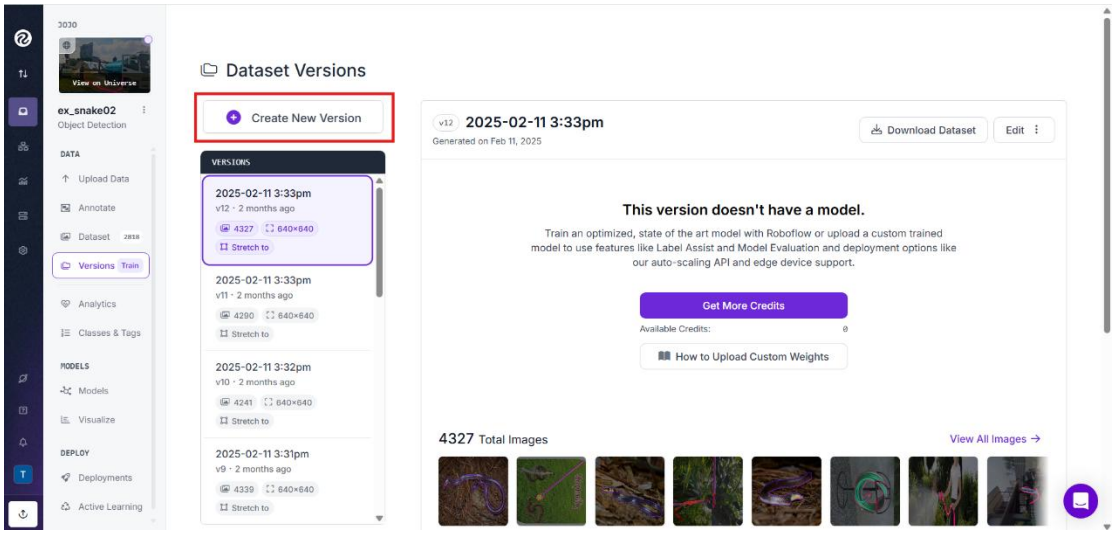
จากภาพที่ ข.17 เมื่อกำหนดอัตราส่วนตามภาพเรียบร้อยแล้ว ให้คลิกปุ่ม 'Add Images' เพื่อบันทึกลงใน Dataset จะได้ดังภาพที่ ข.18



ภาพที่ ข.18 หน้าจอแสงรายละเอียดAnnotateที่ได้กำหนดDataset

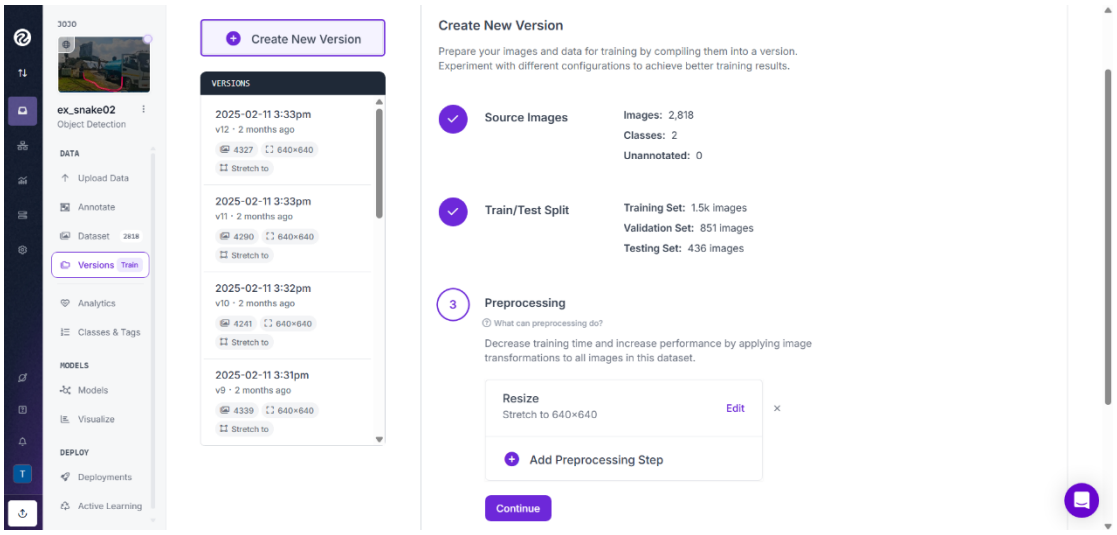
1.3 การสร้างชุดข้อมูล

ในขั้นตอนนี้ให้กดเลือกเมนู “Versions” จะได้ดังภาพที่ ข.19



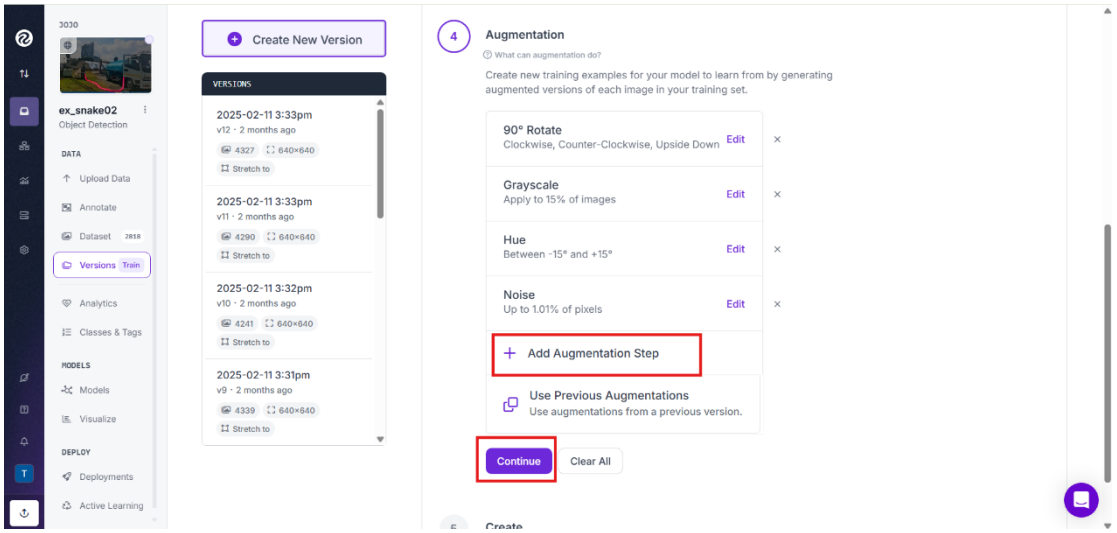
ภาพที่ ข.19 หน้าจอแสดง Dataset Versions

จากภาพที่ ข.19 จะสร้างเวอร์ชันใหม่ของชุดข้อมูล โดยคลิกที่ปุ่ม Create New Version จะมี pop up ขึ้นดังภาพที่ ข.20



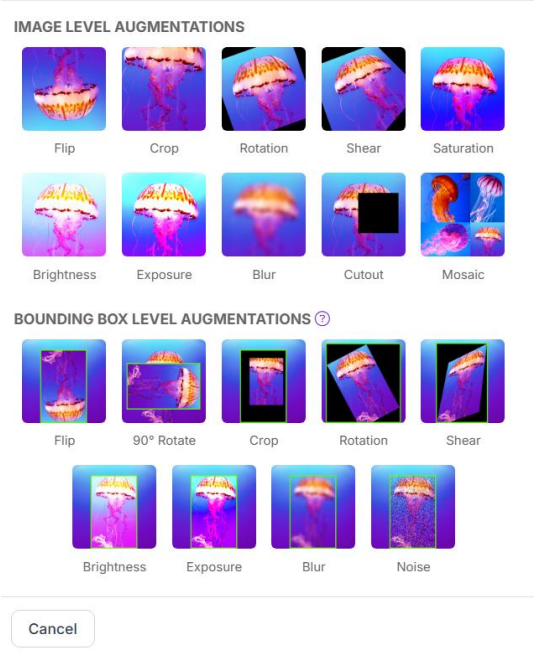
ภาพที่ ข.20 หน้าจอ Create New Version

จากภาพที่ ข.20 จะแสดงรายละเอียดของ Dataset ที่จะสร้างขึ้นใหม่ โดยในภาพจะมี 3 ส่วน ได้แก่ 1.Source Images จะแสดงจำนวนรูปภาพทั้งหมดและจำนวนClass 2.Train/Test Split จะบอกรายละเอียดว่าในชุดข้อมูล ทั้ง 3 set มีจำนวนเท่าไร 3.Perprocessing จะทำการ resize รูปภาพให้มีขนาด 640 x 640 เมื่อเสร็จสิ้นให้กดปุ่ม Continue เมื่อเลื่อนลงมาจะได้ดังภาพที่ ข.21

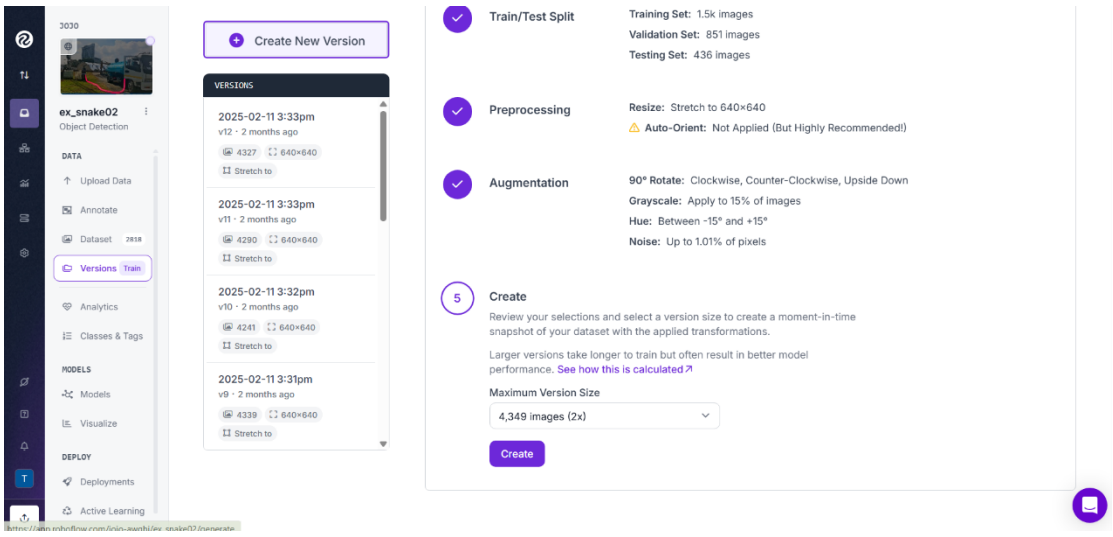


ภาพที่ ข.21 หน้าจอ Create New Version ขั้นตอน Augmentation

จากภาพที่ ข.21 จะเป็นส่วนที่ 4 โดยในส่วนนี้เราจะเลือกได้ว่าเราจะทำ Augmentation ใดบ้าง โดยสร้างคลิกที่ปุ่ม “Add Augmentation Step” จะแสดงดังภาพที่ ข.22 เมื่อเสร็จสิ้นให้กดปุ่ม Continue เมื่อเลื่อนลงมาจะได้ดังภาพที่ ข.23

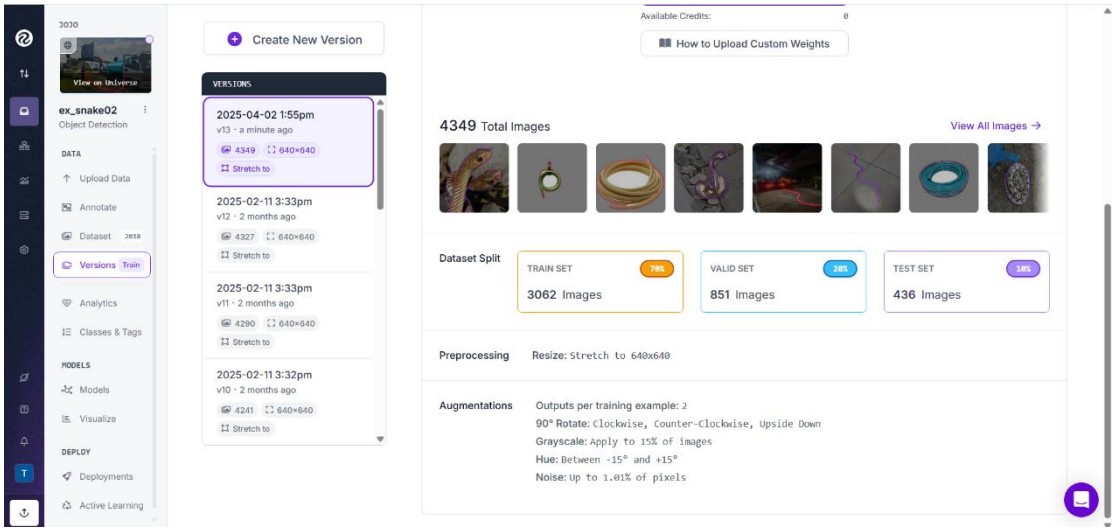


ภาพที่ ข.22 Augmentation Options



ภาพที่ ข.23 หน้าจอ Create New Version ขั้นตอน Create

จากภาพที่ ข.23 จะเป็นส่วนที่ 5 โดยในส่วนนี้เราจะเลือกได้ว่าเราสามารถเลือกขนาดสูงสุดของเวอร์ชันข้อมูลที่ต้องการสร้างโดยในโปรเจกต์นี้ ขนาดสูงสุดที่เลือกได้ในตัวอย่างนี้คือ 4,349 ภาพ (2x) ซึ่งหมายถึงการขยายชุดข้อมูลโดยการใช้ Augmentation เพื่อเพิ่มจำนวนภาพจากต้นฉบับ เมื่อเสร็จสิ้นให้กดปุ่ม Create จะได้ดังภาพที่ ข.24

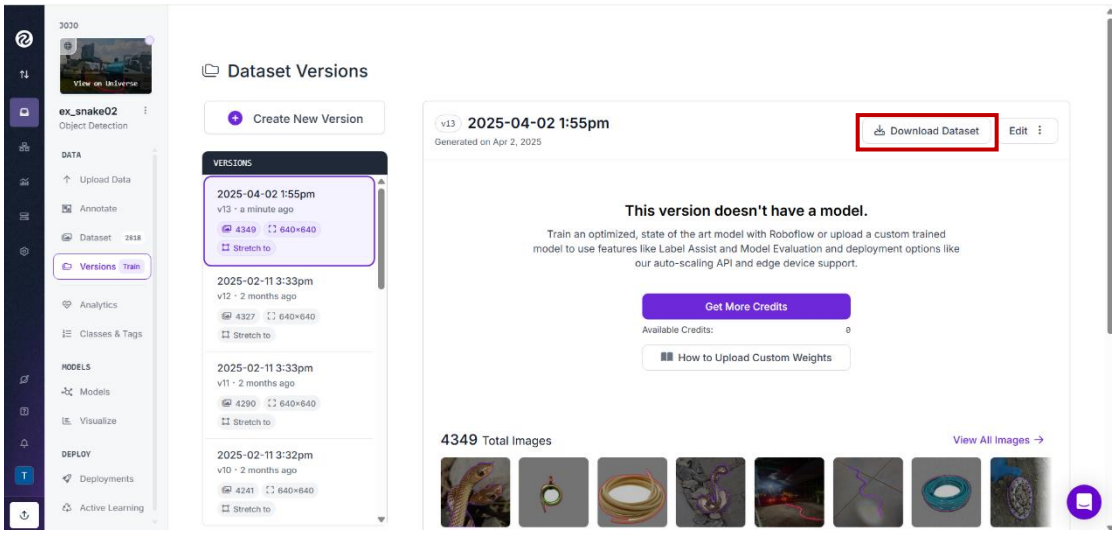


ภาพที่ ข.24 หน้าจอแสดงรายละเอียด Dataset Version

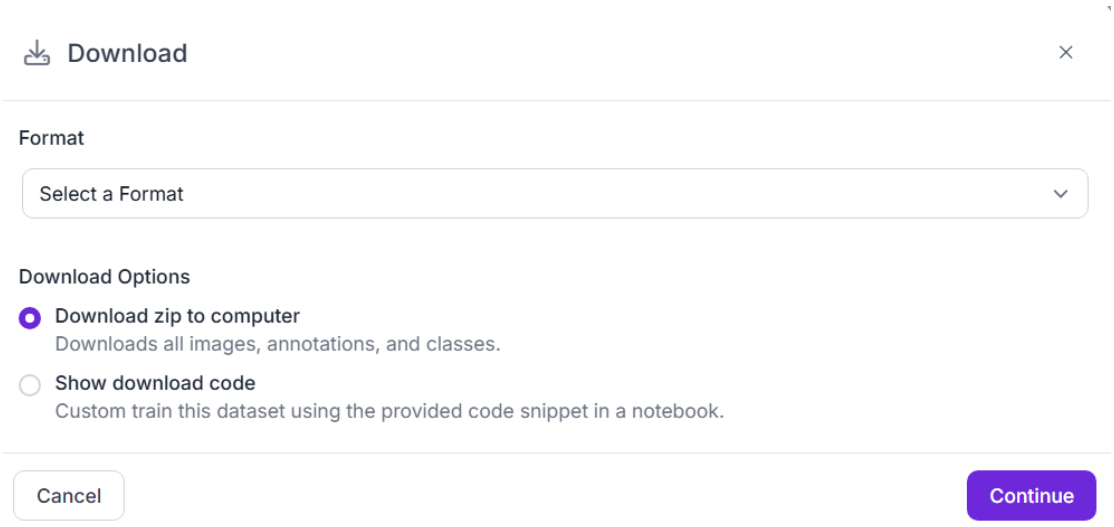
จากภาพที่ ข.24 จะเป็นหน้าแสดงรายละเอียดของชุดข้อมูลที่เราได้ทำการสร้างเมื่อสักครู่

1.4 การดาวน์โหลด Dataset

การดาวน์โหลด Dataset ให้คลิกเลือกปุ่ม “Download Dataset” ดังภาพที่ ข.25 จะขึ้นpop up ดังภาพที่ ข.26



ภาพที่ ข.25 ขั้นตอนการ Download Dataset



ภาพที่ ข.26 Pop up เลือกประเภทในการ Download Dataset

จากภาพที่ ข.26 แบ่งเป็น 2 ส่วน 1.Format ให้เลือกตามตารางที่ ข.1 และ 2. Download Options ให้เลือก “Download zip to compute” และคลิกปุ่ม “Continue” ถือว่าเสร็จสิ้น

ตารางที่ ข.1 ตารางแสดงFormat ในการ Download Dataset

ลำดับ	โมเดล	Format
1	YOLOv5	YOLOv5Pytorh
2	YOLOv8	YOLOv8
3	YOLOv11	YOLOv11
4	MobileNetV2	COCO
5	EfficientNet	COCO

2.การพัฒนาโมเดล

กระบวนการพัฒนาโมเดลดำเนินการบนแพลตฟอร์ม Google Colab ผ่านทางเว็บไซต์ https://colab.research.google.com/

2.1 การพัฒนาโมเดลด้วย YOLOv5

2.1.1 ทำการเชื่อมต่อกับ Google Drive

ภาพที่ ข.27 Code connect Google Drive

2.1.2 Clone git Yolov5

```
[ ] %cd /content/drive/MyDrive/democlass_snakeyolov5
```

```
/content/drive/MyDrive/democlass_snakeyolov5
```

```
!git clone https://github.com/ultralytics/yolov5  # clone
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 17270, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 17270 (delta 0), reused 0 (delta 0), pack-reused 17269 (from 2)
Receiving objects: 100% (17270/17270), 16.11 MiB | 8.91 MiB/s, done.
Resolving deltas: 100% (11861/11861), done.
Updating files: 100% (146/146), done.
```

ภาพที่ ข.28 Code Clone git Yolov5

2.1.3. ติดตั้ง requirements.txt

```
[ ] %cd /content/drive/MyDrive/democlass_snakeyolov5/yolov5
    %pip install -qr requirements.txt  # install dependencies
```

```
/content/drive/MyDrive/democlass_snakeyolov5/yolov5
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 61.3 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 42.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 28.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 11.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 66.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 914.9/914.9 kB 43.4 MB/s eta 0:00:00
```

ภาพที่ ข.29 Code Install requirements.txt

2.1.4 การฝึกโมเดล

```
[ ] %cd /content/drive/MyDrive/democlass_snakeyolov5/yolov5
    !python train.py --img 640 --epochs 50 --data data.yaml --cfg /models/yolov5x.yaml --weights 'yolov5x.pt'
```

```
Validating runs/train/exp10/weights/best.pt...
Fusing layers...
YOLOv5x summary: 322 layers, 86233975 parameters, 0 gradients, 203.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 23/23 [00:08<00:00,  2.80it/s]
                   all        709        880      0.769      0.663      0.721      0.582
           Banded Krait        709         46      0.798      0.687      0.776      0.574
       Banded Rat Snake        709         69      0.603      0.662      0.592      0.476
    Golden Flying Snake        709        223       0.85      0.508      0.683      0.534
             King cobra        709        104      0.741      0.384      0.539      0.446
         Malayan Krait         709         78      0.842      0.685      0.774      0.601
      Malayan Pitviper         709        102      0.926      0.784      0.869      0.737
     Monocellate Cobra         709         62      0.487      0.694       0.57      0.479
       Pope-s Pit Viper        709         66      0.825      0.773      0.817      0.675
    Reticulated Python         709         57      0.675      0.737      0.751       0.59
Siamese Russell-s Viper         709         73      0.946      0.719      0.843      0.713
Results saved to runs/train/exp10
```

ภาพที่ ข.30 Code Train Model YOLOv5

2.2 การพัฒนาโมเดลด้วย YOLOv8

2.2.1 ทำการเชื่อมต่อกับ Google Drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

ภาพที่ ข.31 Code connect Google Drive

2.2.2 Clone git YOLOv8

```
[ ] %cd /content/drive/MyDrive/SnakeClass_yolo8
```

```
/content/drive/MyDrive/SnakeClass_yolo8
```

```
[ ] !git clone https://github.com/ultralytics/ultralytics.git
```

```
Cloning into 'ultralytics'...
remote: Enumerating objects: 52199, done.
remote: Total 52199 (delta 0), reused 0 (delta 0), pack-reused 52199 (from 1)
Receiving objects: 100% (52199/52199), 29.68 MiB | 11.03 MiB/s, done.
Resolving deltas: 100% (38636/38636), done.
Updating files: 100% (705/705), done.
```

ภาพที่ ข.32 Code Clone git YOLOv8

2.2.3 ติดตั้ง Ultralytics

```
[ ] %cd /content/drive/MyDrive/SnakeClass_yolo8
    %pip install ultralytics
```

```
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 59.4 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.3 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 9.4 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 36.9 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 16.6 MB/s eta 0:00:00
```

ภาพที่ ข.33 Code Install ultralytics

2.2.4 การฝึกโมเดล

```
[ ] from ultralytics import YOLO

    # โหลดโมเดล YOLOv8
    model = YOLO("yolov8n.pt")

    # ฝึกโมเดลพร้อมกับการหยุดเมื่อไม่มีการปรับปรุง (early stopping)
    train_results = model.train(
        data="/content/drive/MyDrive/SnakeClass_yolo8/data.yaml",  # พาธของไฟล์ YAML ที่เก็บข้อมูล
        epochs=150,  # จำนวน epochs ที่จะฝึก
        imgsz=640,  # ขนาดของภาพที่ใช้ในการฝึก
        device="cuda",  # กำหนดอุปกรณ์ที่ใช้ เช่น device=0 หรือ device=cpu
    )

    # ประเมินผลการทำงานของโมเดลบนชุดข้อมูล validation
    metrics = model.val()

    # ทำนายการตรวจจับวัตถุในภาพ
    results = model("/content/drive/MyDrive/SnakeClass_yolo8/dataset/test/images/12932783_264655617209053_6210877979916900034_n_jpg.rf.49a0522415b9a72cdb7013666bf5bd19.jpg")
    results[0].show()

    # ส่งออกโมเดลในรูปแบบ ONNX
    path = model.export(format="onnx")

    # ส่งคืนพาธที่ใช้ในการส่งออกโมเดล
```

ภาพที่ ข.34 Code Train Model YOLOv8

```
Validating runs/detect/train7/weights/best.pt...
Ultralytics 8.3.93 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
Model summary (fused): 72 layers, 3,007,598 parameters, 0 gradients, 8.1 GFLOPs
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:04<00:00,  5.44it/s]
                     all        709        880      0.745      0.586      0.647      0.501
            Banded Krait         39         46      0.746      0.609      0.655      0.456
         Banded Rat Snake         53         69       0.43      0.638      0.492      0.392
     Golden Flying Snake        166        223      0.887      0.493      0.703      0.515
              King_cobra         85        104      0.743      0.337      0.445      0.327
           Malayan Krait         59         78      0.832      0.513      0.641      0.487
        Malayan Pitviper         95        102       0.91      0.627       0.81      0.637
      Monocellate Cobra         54         62      0.543      0.645      0.525      0.406
        Pope-s Pit Viper         49         66      0.738      0.621      0.695      0.559
     Reticulated Python         47         57      0.708      0.702      0.734      0.573
Siamese Russell-s Viper         69         73      0.908      0.675      0.771      0.657
Speed: 0.2ms preprocess, 0.5ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to runs/detect/train7
Ultralytics 8.3.93 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
Model summary (fused): 72 layers, 3,007,598 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /content/drive/MyDrive/SnakeClass_yolo8/dataset/valid/labels.cache... 709 images, 1 backgrounds, 0 corrupt: 100%|██████████| 709/709 [00:00<?, ?it/s]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 45/45 [00:08<00:00,  5.56it/s]
                     all        709        880      0.725      0.593      0.647        0.5
            Banded Krait         39         46      0.739      0.615      0.655      0.455
         Banded Rat Snake         53         69      0.414      0.652      0.491      0.392
     Golden Flying Snake        166        223      0.876      0.505      0.703      0.515
              King_cobra         85        104      0.683      0.337      0.444      0.327
           Malayan Krait         59         78      0.801      0.513       0.64      0.486
        Malayan Pitviper         95        102      0.903       0.64       0.81      0.637
      Monocellate Cobra         54         62      0.503      0.645      0.525      0.402
        Pope-s Pit Viper         49         66      0.723      0.621      0.695      0.558
     Reticulated Python         47         57      0.698      0.719      0.734      0.573
Siamese Russell-s Viper         69         73      0.909       0.68      0.771      0.657
Speed: 0.4ms preprocess, 0.9ms inference, 0.0ms loss, 1.2ms postprocess per image
Results saved to runs/detect/train72
```

ภาพที่ ข.35 ผลลัพธ์ Train Model YOLOv8

2.3 การพัฒนาโมเดลด้วย YOLOv11

2.3.1 การเชื่อมต่อกับ Google Drive

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

```
Mounted at /content/drive
```

ภาพที่ ข.36 Code connect Google Drive

2.3.2 ติดตั้ง Ultralytics

```
[ ] %cd /content/drive/MyDrive/SnakeClassifica_Yolov11
```

```
/content/drive/.shortcut-targets-by-id/1IDw7nxeMrjda6hMmb0NhmtyoGR_cF5yt/SnakeClassifica_Yolov11
```

```
[ ] %pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.3.75-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: numpy<=2.1.1,>=1.23.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (3.10.0)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (4.11.0.86)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (11.1.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (2.5.1+cu124)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (0.20.1+cu124)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.11/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (0.13.2)
```

ภาพที่ ข.37 Install Ultralytics

2.3.3 การฝึกฝนโมเดล

```
[ ] from ultralytics import YOLO

    # Load a model
    model = YOLO("yolo11n.pt")

    # Train the model
    train_results = model.train(
        data="/content/drive/MyDrive/SnakeClassifica_Yolov11/datasetv.2/data.yaml",  # path to dataset YAML
        epochs=40,  # number of training epochs
        imgsz=640,  # training image size
        device="cuda",  # device to run on, i.e. device=0 or device=0,1,2,3 or device=cpu
    )

    # Evaluate model performance on the validation set
    metrics = model.val()

    # Perform object detection on an image
    results = model("/content/drive/MyDrive/SnakeClassifica_Yolov11/datasetv.2/test/images/12805871_262626920745256_8500042655933008195_n_jpg.rf.9afeb3ac117b798e54452d157f0458d4.jpg")
    results[0].show()

    # Export the model to ONNX format
    path = model.export(format="onnx")  # return path to exported model
```

ภาพที่ ข.38 Code Tarin Model YOLOv11

```
Validating runs/detect/train3/weights/best.pt...
Ultralytics 8.3.74 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLO11n summary (fused): 238 layers, 2,584,102 parameters, 0 gradients, 6.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 23/23 [00:04<00:00,  5.60it/s]
                   all        709        880      0.745      0.581      0.651      0.519
          Banded Krait         39         46      0.756      0.652      0.687      0.543
      Banded Rat Snake         53         69      0.487      0.507       0.47      0.348
   Golden Flying Snake        166        223       0.91      0.501      0.706      0.525
            King_cobra         85        104      0.761      0.375      0.511      0.426
         Malayan Krait         59         78      0.846      0.551      0.665      0.517
      Malayan Pitviper         95        102      0.911      0.702      0.836      0.678
     Monocellate Cobra         54         62       0.46      0.577      0.522      0.429
       Pope-s Pit Viper         49         66      0.676      0.727      0.714      0.582
    Reticulated Python         47         57      0.725      0.614      0.666      0.511
Siamese Russell-s Viper         69         73      0.916      0.598      0.736      0.634
Speed: 0.1ms preprocess, 0.6ms inference, 0.0ms loss, 1.0ms postprocess per image
Results saved to runs/detect/train3
Ultralytics 8.3.74 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLO11n summary (fused): 238 layers, 2,584,102 parameters, 0 gradients, 6.3 GFLOPs
val: Scanning /content/drive/.shortcut-targets-by-id/1IDw7nxeMrjda6hMmb0NhmtyoGR_cF5yt/SnakeClassifica_Yolov11/datasetv.2/valid/labels.cache... 709 images, 1 backgrounds, 0 corrupt: 100%|████████| 709/7
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 45/45 [00:07<00:00,  6.01it/s]
                   all        709        880      0.745      0.582      0.651       0.52
          Banded Krait         39         46      0.754      0.652      0.687      0.545
      Banded Rat Snake         53         69      0.483      0.507       0.47      0.349
   Golden Flying Snake        166        223      0.911      0.503      0.706      0.526
            King_cobra         85        104       0.76      0.375      0.511      0.424
         Malayan Krait         59         78      0.843      0.552      0.665      0.516
      Malayan Pitviper         95        102      0.911      0.703      0.836      0.679
     Monocellate Cobra         54         62      0.461      0.581      0.521      0.429
       Pope-s Pit Viper         49         66      0.675      0.727      0.714      0.582
    Reticulated Python         47         57      0.733      0.626      0.666      0.511
Siamese Russell-s Viper         69         73      0.916      0.598      0.736       0.64
Speed: 0.5ms preprocess, 1.2ms inference, 0.0ms loss, 1.1ms postprocess per image
Results saved to runs/detect/train32
```

ภาพที่ ข.39 ผลลัพธ์ Train Model YOLOv11

2.4 การพัฒนาโมเดลด้วย EfficientNet

2.4.1.การเชื่อมต่อกับ Google Drive

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

ภาพที่ ข.40 Code connect Google Drive

2.4.2 ติดตั้ง Tensorflow และ -U efficientnet

```
[ ] %cd /content/drive/MyDrive/EfficientNet

/content/drive/MyDrive/EfficientNet
```

```
[ ] %cd /content/drive/MyDrive/EfficientNet

    # ติดตั้ง TensorFlow และ EfficientNet
    !pip install tensorflow
    !pip install -U efficientnet

    # นำเข้าไลบรารี
    import tensorflow as tf
    import efficientnet.tfkeras as efn
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    import os

/content/drive/MyDrive/EfficientNet
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
```

ภาพที่ ข.41 Code Install Tensorflow and -U efficientnet

2.4.3 การติดตั้ง Pycocotools

```
[ ] !pip install pycocotools

Requirement already satisfied: pycocotools in /usr/local/lib/python3.11/dist-packages (2.0.8)
Requirement already satisfied: matplotlib>=2.1.0 in /usr/local/lib/python3.11/dist-packages (from pycocotools) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from pycocotools) (1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=2.1.0->pycocotools) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib>=2.1.0->pycocotools) (1.17.0)
```

ภาพที่ ข.42 Code Install Pycocotools

2.4.4 การฝึกฝนโมเดล

```
[ ] import os
    import json
    import tensorflow as tf
    import numpy as np
    from pycocotools.coco import COCO
    from tensorflow.keras.utils import to_categorical
    from tensorflow.keras.preprocessing.image import load_img, img_to_array

    IMG_SIZE = 640
    BATCH_SIZE = 16
    NUM_CLASSES = 10

    dataset_dir = "/content/drive/MyDrive/EfficientNet/dataset_v.2/train"
    coco_file = os.path.join(dataset_dir, "_annotations.coco.json")
    image_dir = os.path.join(dataset_dir, "images")

    coco = COCO(coco_file)

    category_ids = coco.getCatIds()
    categories = coco.loadCats(category_ids)
    class_names = {cat["id"]: cat["name"] for cat in categories}
```

```
    def load_coco_data(coco, image_dir, img_size=IMG_SIZE):
        image_ids = coco.getImgIds()
        X, y = [], []
        for img_id in image_ids:
            img_info = coco.loadImgs(img_id)[0]
            img_path = os.path.join(image_dir, img_info["file_name"])

            if not os.path.exists(img_path):
                print(f"❌ ไม่พบรูปภาพ: {img_path}")
                continue

            print(f"✅ กำลังโหลด: {img_path}")
            img = load_img(img_path, target_size=(img_size, img_size))
            img_array = img_to_array(img) / 255.0
            X.append(img_array)
            ann_ids = coco.getAnnIds(imgIds=img_id)
            anns = coco.loadAnns(ann_ids)
            if anns:
                class_id = anns[0]["category_id"]
                y.append(class_id)
            else:
                y.append(0)
        X = np.array(X)
        y = np.array(y)
        y = to_categorical(y, num_classes=len(class_names))
        return X, y
    X_train, y_train = load_coco_data(coco, image_dir)
    train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train))
    train_dataset = train_dataset.shuffle(1000).batch(BATCH_SIZE)
```

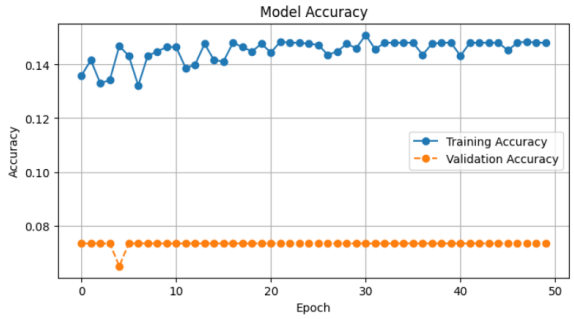
```
    from tensorflow.keras.applications import EfficientNetB0
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
    import matplotlib.pyplot as plt
    import json
    import os
    from google.colab import drive

    # ✅ 1. เชื่อม Google Drive
    drive.mount('/content/drive')

    # กำหนด path สำหรับบันทึกไฟล์
    SAVE_PATH = "/content/drive/MyDrive/EfficientNet/run"
    os.makedirs(SAVE_PATH, exist_ok=True)

    # ✅ 2. โหลด EfficientNetB0 (ไม่รวม Fully Connected Layer)
    base_model = EfficientNetB0(weights="imagenet", include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

    # แช่แข็ง layer ทั้งหมดของ EfficientNet
    base_model.trainable = False

    # ✅ 3. สร้าง Fully Connected Layer ใหม่
    x = GlobalAveragePooling2D()(base_model.output)
    x = Dense(256, activation="relu")(x)
    x = Dropout(0.3)(x)
    output_layer = Dense(len(class_names), activation="softmax")(x)
```

```
    # สร้างโมเดลสุดท้าย
    model = Model(inputs=base_model.input, outputs=output_layer)

    # ✅ 4. คอมไพล์โมเดล
    model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

    # ✅ 5. เทรนโมเดล พร้อมเก็บค่า history
    history = model.fit(train_dataset, epochs=50, batch_size=BATCH_SIZE, validation_data=val_dataset)

    # ✅ 6. บันทึกโมเดลและ weights ลง Google Drive
    model.save(SAVE_PATH + "efficientnet_model.keras")  # แนะนำใช้ .keras แทน .h5
    model.save_weights(SAVE_PATH + "efficientnet.weights.h5")  # ✅ ต้องใช้ .weights.h5

    # ✅ 7. บันทึกค่าประวัติการเทรนเป็น JSON
    with open(SAVE_PATH + "history.json", "w") as f:
        json.dump(history.history, f)

    # ✅ 8. สร้างและบันทึกกราฟ Accuracy & Loss
    # Accuracy
    plt.figure(figsize=(8, 4))
    plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o', linestyle='dashed')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.title('Model Accuracy')
    plt.grid()
    plt.savefig(SAVE_PATH + "accuracy_plot.png")  # บันทึกกราฟ Accuracy
    plt.show()
```

```
    # Loss
    plt.figure(figsize=(8, 4))
    plt.plot(history.history['loss'], label='Training Loss', marker='o', color='red')
    plt.plot(history.history['val_loss'], label='Validation Loss', marker='o', linestyle='dashed', color='orange')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.title('Model Loss')
    plt.grid()
    plt.savefig(SAVE_PATH + "loss_plot.png")  # บันทึกกราฟ Loss
    plt.show()
```

ภาพที่ ข.43 Code Train Model EfficientNet

```
Epoch 46/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 8s 52ms/step - accuracy: 0.1463 - loss: 2.2484 - val_accuracy: 0.0733 - val_loss: 2.5025
Epoch 47/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 8s 51ms/step - accuracy: 0.1443 - loss: 2.2457 - val_accuracy: 0.0733 - val_loss: 2.4543
Epoch 48/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 8s 52ms/step - accuracy: 0.1448 - loss: 2.2580 - val_accuracy: 0.0733 - val_loss: 2.5149
Epoch 49/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 8s 51ms/step - accuracy: 0.1542 - loss: 2.2425 - val_accuracy: 0.0733 - val_loss: 2.5372
Epoch 50/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 8s 51ms/step - accuracy: 0.1395 - loss: 2.2599 - val_accuracy: 0.0733 - val_loss: 2.5306
```



ภาพที่ ข.44 ผลลัพธ์ Train Model EfficientNet

2.5 การพัฒนาโมเดลด้วย MobileNetV2

2.5.1 การเชื่อมต่อ Google Drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

ภาพที่ ข.45 Code connect Google Drive

2.การติดตั้ง roboflow

```
[ ] %cd /content/drive/MyDrive/Mobilenet/classifica
```

```
/content/drive/.shortcut-targets-by-id/1ZM8ZpFn42VLIHsZPX9JwcJu7Zzv0Yviu/Mobilenet/classifica
```

```
[ ] !pip install roboflow

    from roboflow import Roboflow
    rf = Roboflow(api_key="t9PtKMrafFZXjBozVnm5")
    project = rf.workspace("jojo-awqhj").project("ex_snake02")
    version = project.version(12)
    dataset = version.download("coco")
```

```
Collecting roboflow
  Downloading roboflow-1.1.54-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from roboflow) (2025.1.31)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.11/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.4.8)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from roboflow) (3.10.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.26.4)
Collecting opencv-python-headless==4.10.0.84 (from roboflow)
  Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.11/dist-packages (from roboflow) (11.1.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.8.2)
```

ภาพที่ ข.46 Code Install roboflow

3.การฝึกฝนโมเดล

```
[ ] import os
    import json
    import cv2
    import numpy as np
    import tensorflow as tf
    import matplotlib.pyplot as plt
    from sklearn.metrics import precision_score, recall_score, classification_report, average_precision_score

    # โฟลเดอร์ dataset
    DATASET_DIR = "class_snake02-10"
    ANNOTATION_FILES = {
        "train": f"{DATASET_DIR}/train/_annotations.coco.json",
        "valid": f"{DATASET_DIR}/valid/_annotations.coco.json",
        "test": f"{DATASET_DIR}/test/_annotations.coco.json"
    }
    IMAGE_DIRS = {
        "train": f"{DATASET_DIR}/train",
        "valid": f"{DATASET_DIR}/valid",
        "test": f"{DATASET_DIR}/test"
    }
```

```python
# โหลด dataset แก้ category_id ให้เรียงลำดับต่อเนื่อง
def load_coco_dataset(split):
    with open(ANNOTATION_FILES[split], "r") as f:
        coco_data = json.load(f)

    images = {img["id"]: img["file_name"] for img in coco_data["images"]}
    annotations = coco_data["annotations"]

    # แก้ category_id ให้ต่อเนื่อง
    categories = {cat["id"]: cat["name"] for cat in coco_data["categories"]}
    category_mapping = {old_id: new_id for new_id, old_id in enumerate(sorted(categories.keys()))}

    # อัปเดต category_id ใน annotations
    for ann in annotations:
        ann["category_id"] = category_mapping[ann["category_id"]]

    num_classes = len(categories)
    categories = {new_id: categories[old_id] for old_id, new_id in category_mapping.items()}

    return images, annotations, categories, num_classes

# โหลด dataset ใหม่
train_images, train_annotations, categories, num_classes = load_coco_dataset("train")
valid_images, valid_annotations, _, _ = load_coco_dataset("valid")
test_images, test_annotations, _, _ = load_coco_dataset("test")
```

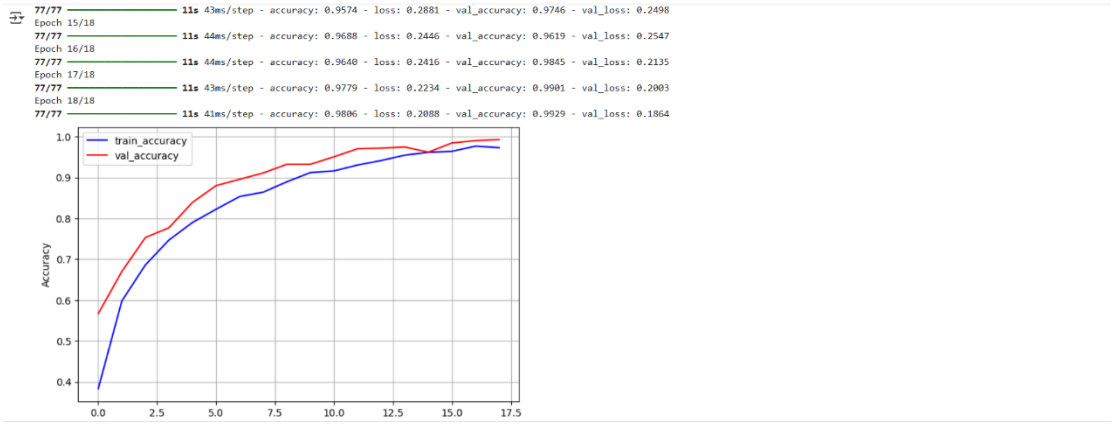
```python
# โหลดรูปภาพ
def load_image_and_label(image_id):
    image_id = int(image_id.numpy())
    img_path = os.path.join(IMAGE_DIRS["train"], train_images[image_id])

    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)) / 255.0

    labels = [ann["category_id"] for ann in train_annotations if ann["image_id"] == image_id]
    labels = [0] if len(labels) == 0 else labels
    one_hot_label = tf.keras.utils.to_categorical(labels[0], num_classes)

    return img.astype(np.float32), one_hot_label.astype(np.float32)

# สร้าง TensorFlow Dataset
def create_tf_dataset(image_dict):
    image_ids = list(image_dict.keys())
    dataset = tf.data.Dataset.from_tensor_slices(image_ids)

    dataset = dataset.map(lambda id: tf.py_function(
        load_image_and_label,
        [id],
        [tf.float32, tf.float32]
    ), num_parallel_calls=tf.data.AUTOTUNE)

    dataset = dataset.map(lambda img, lbl: (tf.ensure_shape(img, (224, 224, 3)), tf.ensure_shape(lbl, (num_classes,)))).batch(32).shuffle(100).prefetch(tf.data.AUTOTUNE)

    return dataset
```

```python
train_dataset = create_tf_dataset(train_images)
valid_dataset = create_tf_dataset(valid_images)
test_dataset = create_tf_dataset(test_images)

# โหลด MobileNetV2
base_model = tf.keras.applications.MobileNetV2(input_shape=(224, 224, 3), include_top=False, weights='imagenet')
base_model.trainable = False

# Layer สำหรับ classification
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
outputs = tf.keras.layers.Dense(num_classes, activation="softmax")(x)

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

# คอมไพล์โมเดล
# กำหนดค่า learning rate ใหม่
learning_rate = 0.0001  # ปรับค่าได้ตามต้องการ

# ใช้ Adam optimizer พร้อมกำหนด learning rate
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)

# คอมไพล์โมเดลใหม่
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])


# เทรนโมเดล
history = model.fit(train_dataset, epochs=18, validation_data=valid_dataset)
```

```python
# พล็อตกราฟ
def plot_metrics(history):
    metrics = ["accuracy", "loss"]
    for metric in metrics:
        plt.figure(figsize=(8, 5))
        plt.plot(history.history[metric], label=f"train_{metric}", color="b")
        plt.plot(history.history[f"val_{metric}"], label=f"val_{metric}", color="r")
        plt.xlabel("Epoch")
        plt.ylabel(metric.capitalize())
        plt.legend()
        plt.grid(True)
        plt.show()

plot_metrics(history)

# แก้ evaluate_snake_model()
def evaluate_snake_model(model, test_dataset):
    y_true, y_pred = [], []
    for images, labels in test_dataset:
        preds = model.predict(images)
        y_true.extend(np.argmax(labels.numpy(), axis=1))
        y_pred.extend(np.argmax(preds, axis=1))

    # ตรวจสอบจำนวนคลาสให้ตรงกัน
    if len(set(y_true)) > num_classes:
        print(f"Warning: พบคลาสที่ไม่อยู่ใน num_classes ({num_classes})")

    # ระบุ labels เพื่อให้ตรงกับจำนวนคลาสจริงที่มีใน y_true และ y_pred
    labels = list(range(num_classes))
    print(classification_report(y_true, y_pred, labels=labels, target_names=list(categories.values())[:num_classes], zero_division=0))
```

```python
# บันทึกโมเดล
SAVE_DIR = "/content/drive/MyDrive/Mobilenet/classifica/newresult"
os.makedirs(SAVE_DIR, exist_ok=True)
model.save(f"{SAVE_DIR}/mobilenetv2_model.keras")
model.save(f"{SAVE_DIR}/mobilenetv2_model.h5")

# โหลดโมเดลกลับมาใช้งาน
from tensorflow.keras.models import load_model

model = load_model(f"{SAVE_DIR}/mobilenetv2_model.h5")
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

# ประเมินผลโมเดลบนชุดทดสอบ
evaluate_snake_model(model, test_dataset)
```

ภาพที่ ข.47 Code Train Model MobileNetV2



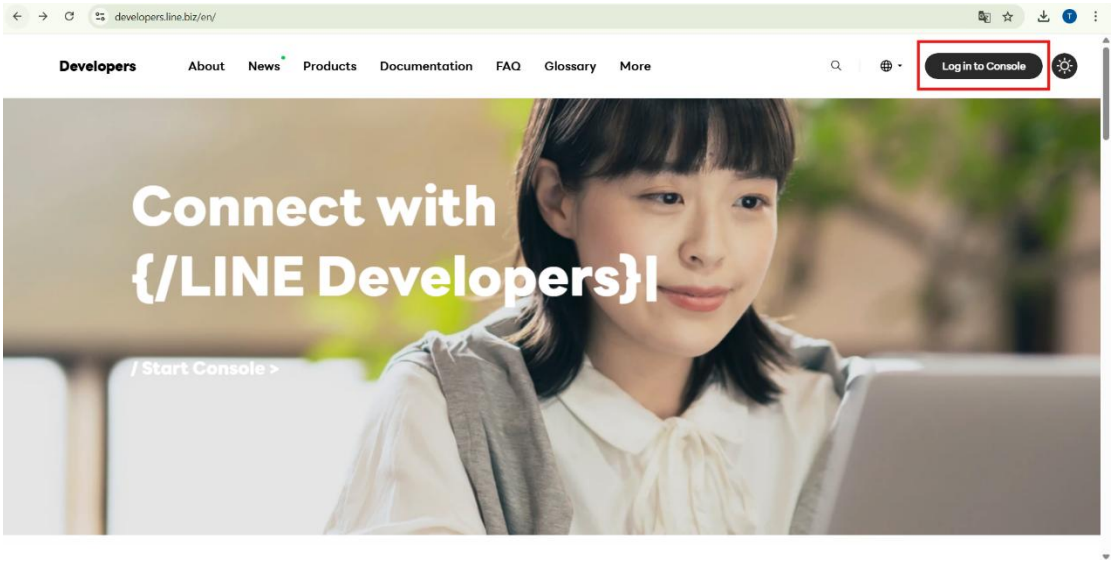
ภาพที่ ข.48 ผลลัพธ์ Train Model MobileNetV2

3.การเชื่อมต่อระบบLINE Chatbot

ในการเชื่อมต่อกับ LINE Chatbot จะแบ่งการทำงานหลัก 2 ส่วน .line developers และ VS Code

3.1.**line developers**

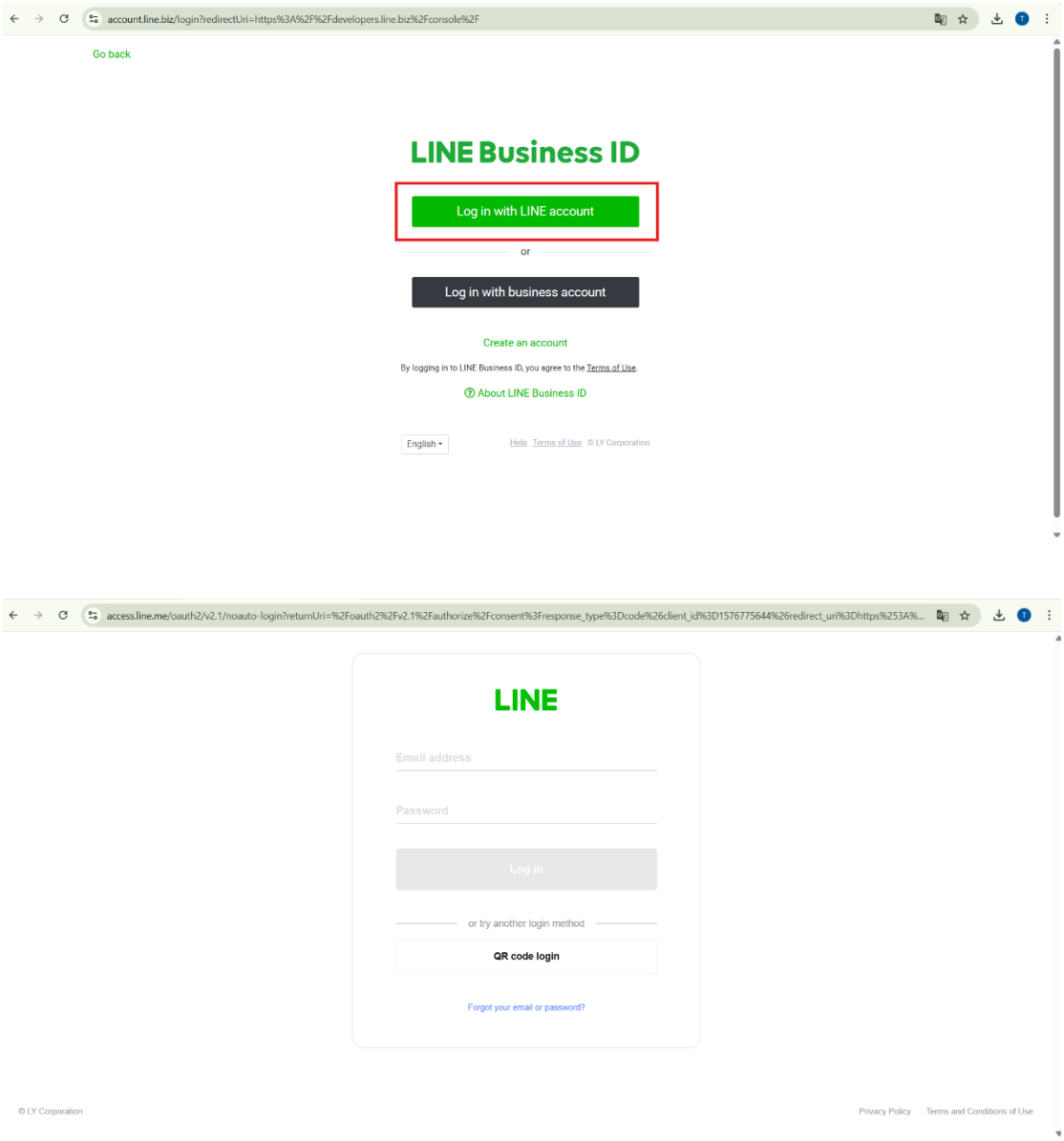
3.1.1 การเข้าสู่หน้าเว็บไซค์ line developers จากhttps://developers.line.biz



ภาพที่ ข.49 หน้าเว็บไซค์ Developers

จากภาพที่ข.49 แอดมินเมนู Log in Console เพื่อเข้าสู่ระบบ

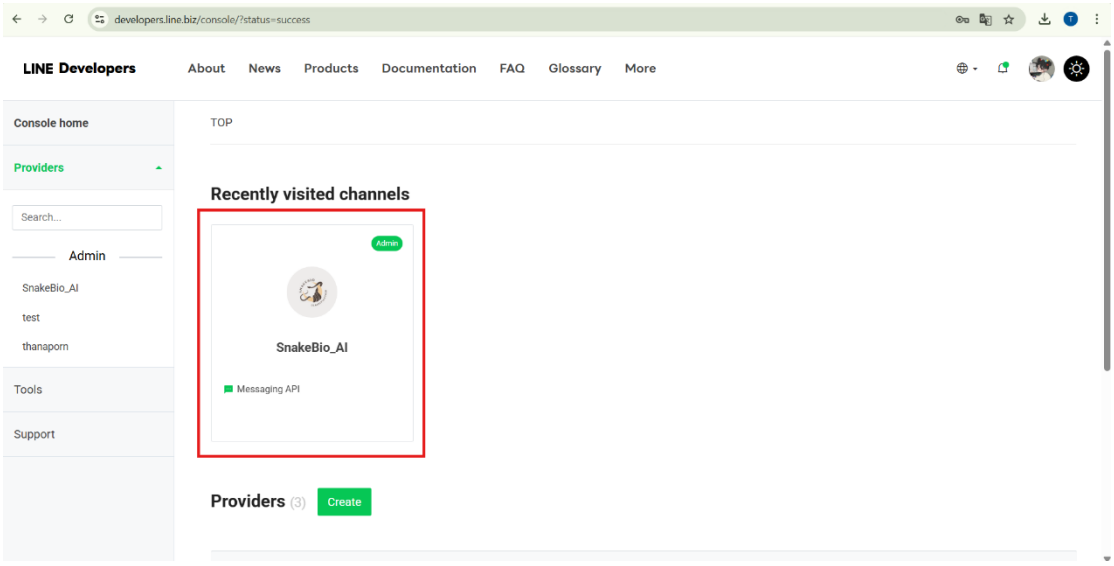
3.1.2. การเข้าสู่ระบบ



ภาพที่ ข.50 หน้าเว็บไซค์ Log in Console

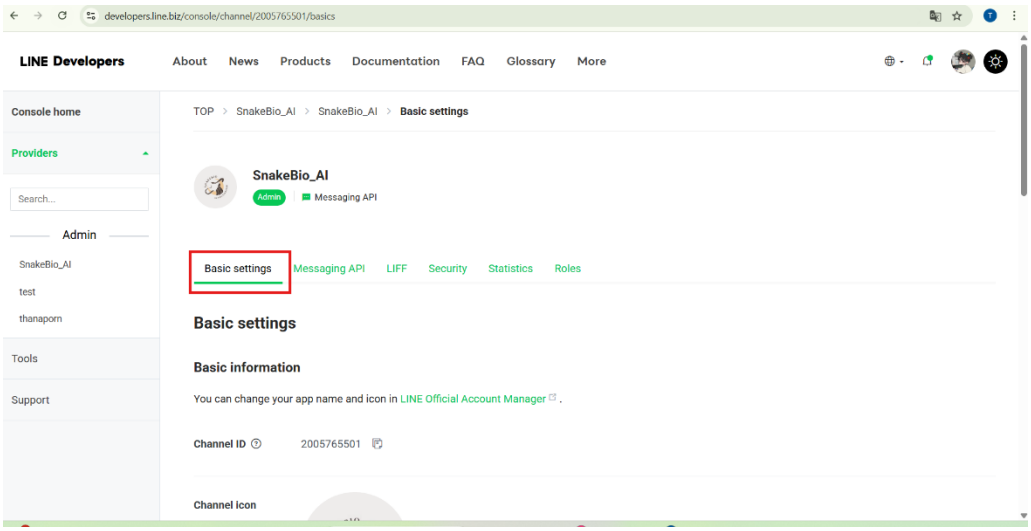
จากภาพที่ ข.50 สามารถ Log in Console ได้ด้วย LINE ที่เชื่อมกับ LINE แชทบอทอัจฉริยะสำหรับจำแนกชนิดของงู โดยกรอก Email และ รหัสผ่าน

3.1.3. การตั้งค่า LINE Developers



ภาพที่ ข.51 หน้าเว็บไซค์ LINE Developers

จากภาพที่ ข.51 เมื่อ Log in เสร็จสินจะแสดงหน้าดังภาพที่ ก. โดยจะแสดงLINE ที่เราได้เป็นแอดมินและมีสิทธิในการแก้ไขการทำงานต่างๆ อย่างในภาพ จะเป็นไลน์แชทบอทอัจฉริยะเพื่อจำแนกชนิดของงู หรือ SnakeBio_AI โดนแอดมินสามารถตั้งค่าการทำงานต่างๆ โดยคลิกที่กรอบสีแดง จะแสดงหน้าเว็บไซค์ดังภาพที่ ข.52



ภาพที่ ข.52 หน้าLINE Developers Dashboard

จากภาพที่ ข.52 ภาพนี้แสดง LINE Developers Dashboard ในส่วนของการตั้งค่าช่องทาง (Channel) สำหรับ SnakeBio_AI โดยแบ่งออกเป็น 3 ส่วน ได้แก่

1.แถบเมนูด้านซ้าย (Sidebar) : แสดงรายการ Providers ที่คุณมีสิทธิ์เข้าถึง เช่น SnakeBio_AI, test, Thanaporn และ มีหมวด Tools และ Support สำหรับเข้าถึงเครื่องมือและการสนับสนุน

2.ส่วนเนื้อหาหลัก โดยมีแท็บการตั้งค่าหลัก ได้แก่

2.1 Basic settings (การตั้งค่าพื้นฐาน)

2.2 Messaging API (API สำหรับการส่งข้อความและโต้ตอบ)

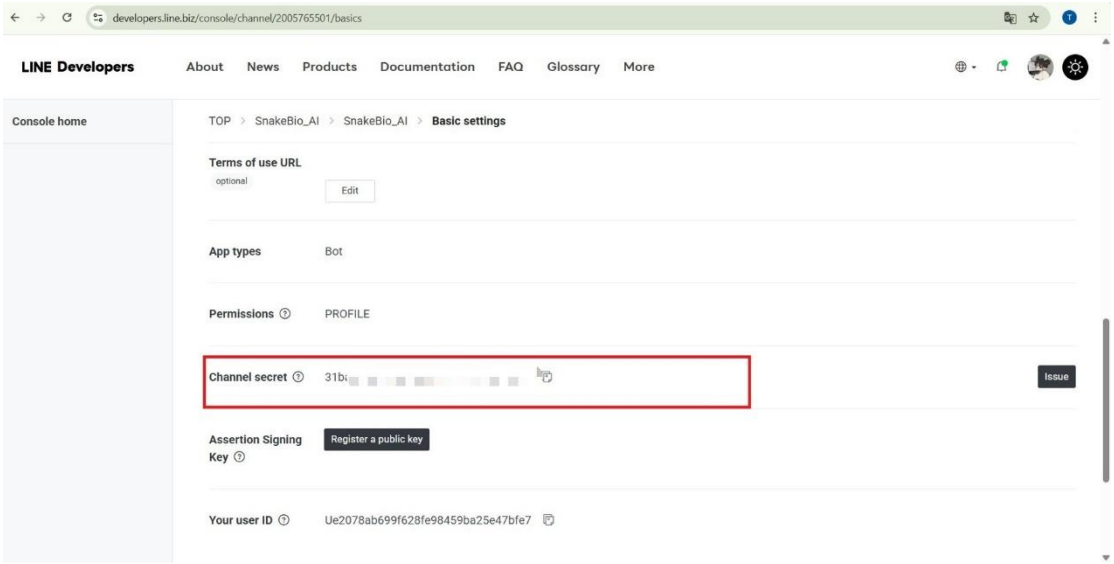
2.3 LIFF (LINE Front-end Framework)

2.4 Security (การรักษาความปลอดภัย)

2.5 Statistics (สถิติการใช้งาน)

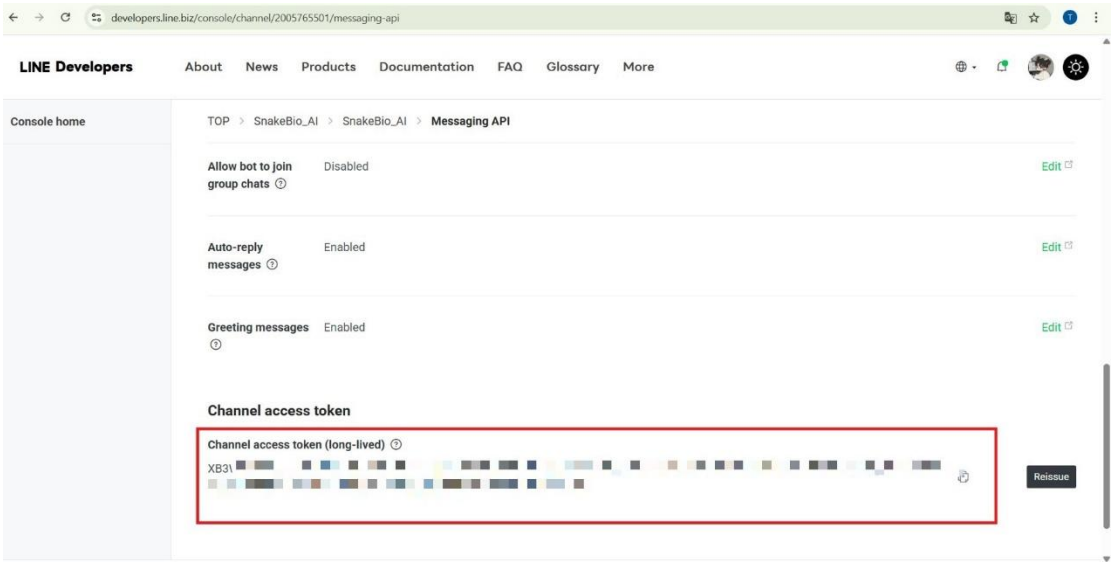
2.6 Roles (การจัดการบทบาทผู้ดูแล)

ในส่วนนี้เราจะสนใจในส่วนของเนื้อหลัก โดยจะไปเลือก ที่แท็บการตั้งค่า Messaging API ที่กรอบสีแดงได้วงเอาไว้ เมื่อกดเลือก Basic settings จะแสดงหน้าเว็บไซค์ดังภาพที่ ข.53



ภาพที่ ข.53 หน้าจอแสดงหน้า Channel secret ภายใน LINE Developers Console

จากภาพที่ ข.53 จะแสดงหน้า Channel secret ภายใน LINE Developers Console สำหรับเชื่อมกับการตั้งค่าการทำงานของโมเดลในการตรวจจับและจำแนกชนิดของงู ถัดมาเมื่อกดเลือก Messaging API จะแสดงหน้าเว็บไซค์ดังภาพที่ ข.54



ภาพที่ ข.54 หน้าจอแสดงหน้า Channel access token ภายใน LINE Developers Console

จากภาพที่ ข.54 จะแสดงหน้า Channel access token ภายใน LINE Developers Console สำหรับเชื่อมกับการตั้งค่าการทำงานของโมเดลในการตรวจจับและจำแนกชนิดของงู

3.2 Vscode

3.2.1.สร้างไฟล์ .env

ภาพที ข.55 Code การกำหนดค่า **LINE**_Channel secret และ **LINE**_Channel access token

3.2.2. LINE API Configuration ในไฟล์ main.py

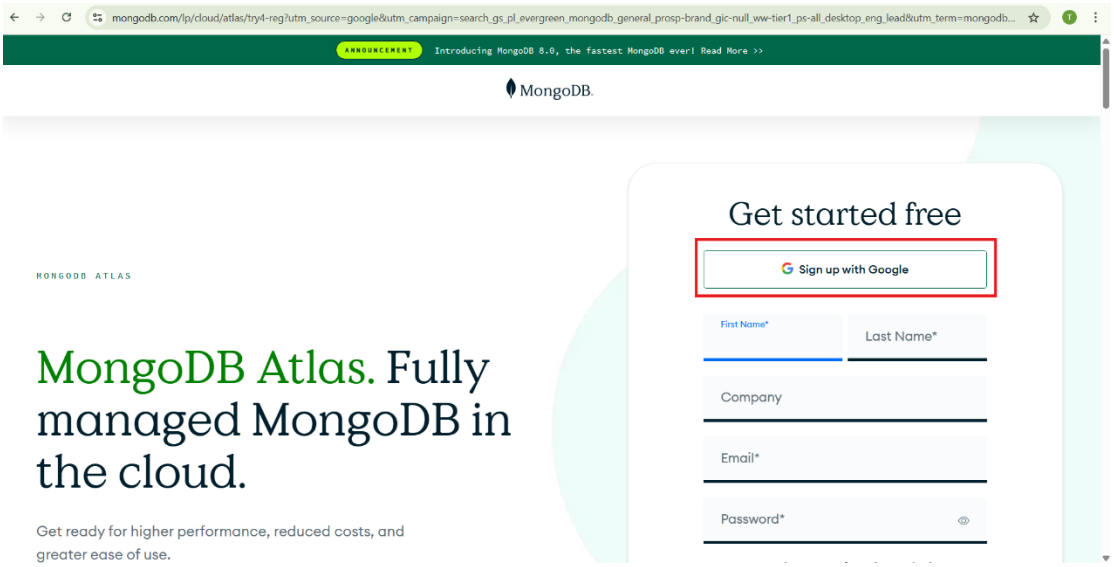
```
channel_secret = os.getenv('LINE_CHANNEL_SECRET', None)
channel_access_token = os.getenv('LINE_CHANNEL_ACCESS_TOKEN', None)

if not channel_secret or not channel_access_token:
    raise Exception("Please set LINE_CHANNEL_SECRET and LINE_CHANNEL_ACCESS_TOKEN in your .env file")

# LINE API Configuration
line_bot_api = LineBotApi(channel_access_token, timeout=5)
handler = WebhookHandler(channel_secret)
```

ภาพที่ ข.56 Code . LINE API Configuration

4. การเชื่อมต่อฐานข้อมูล MongoDB

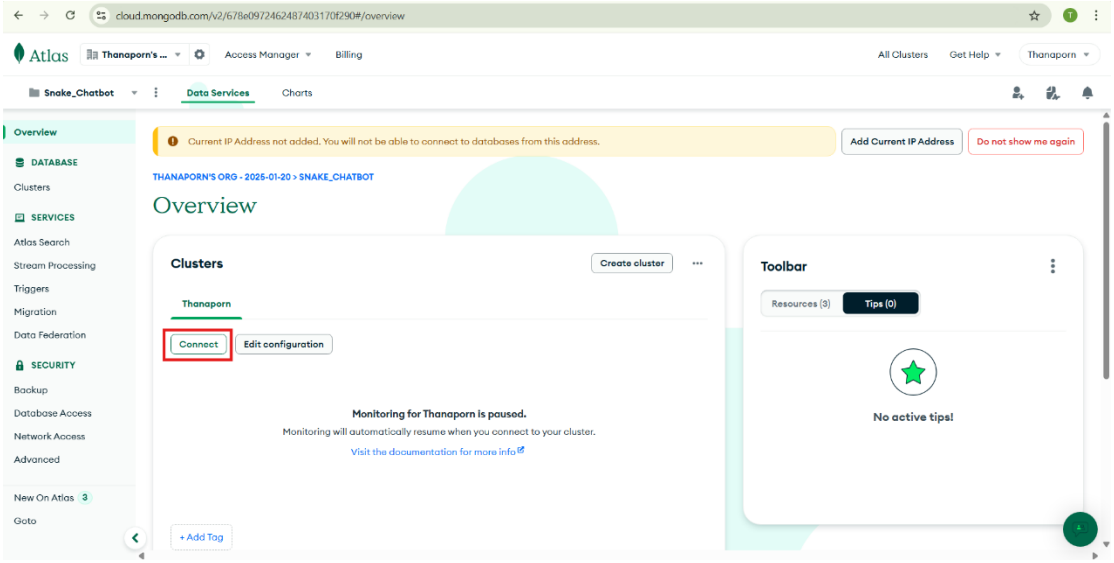
4.1 การเชื่อมต่อฐานข้อมูล MongoDB กับ VSCode

4.1.1.การเข้าสู่หน้าเว็บไซค์ MongoDB จาก https://www.mongodb.com/



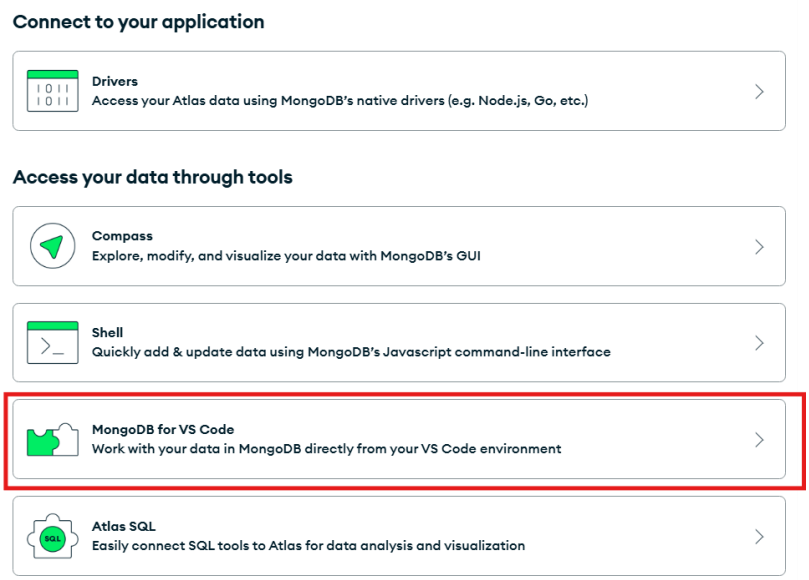
ภาพที่ ข.57 หน้าเว็บไซค์ MongoDB

จากภาพที่ ข.57 คลิกเลือก sign up with Google เมื่อ sign up เสร็จสิ้น ได้ดังภาพที่ ข.58

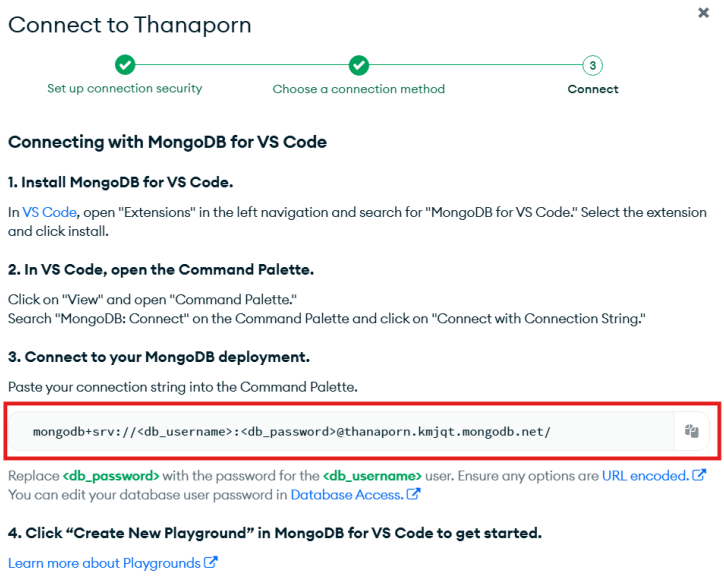


ภาพที่ ข.58 หน้าจอ MongoDB Atlas

จากภาพที่ ข. หน้าจอ MongoDB Atlas ซึ่งเป็นบริการฐานข้อมูลแบบคลาวด์ของMongoDB โดยจะทำการเชื่อมต่อกับ VScode จะต้องคลิกที่ปุ่ม Connect กรอบสีแดง จะขึ้น pop up ดังภาพที่ ข.



ภาพที่ ข.59 Pop up Access your data through tools

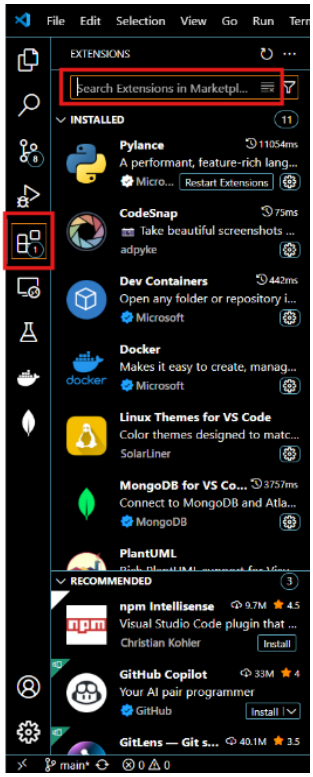


ภาพที่ ข.60 Pop up Connect with MongoDB for VS code

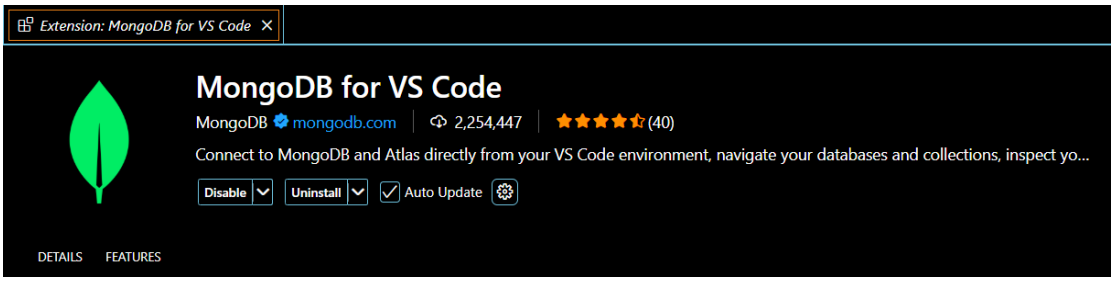
จากภาพที่ ข.59 ในการเชื่อมต่อกับ VScode ให้เลือกตัวเลือก “MongoDB for VScode” เมื่อคลิกเลือกจะได้ดังภาพที่ ข.60 ให้ Copy ในกรอบสีแดง เพื่อนำไปเชื่อมกับ VScode หลังจากนี้ทำใน VScode

4.1.2. การเชื่อมต่อฐานข้อมูล MongoDB ใน VSCode

เปิดไฟล์โครงงานใน VScode และไปที่ Extensions และ ค้นหาคำว่า “MongoDB for VS Code.” ดังภาพที่ ข.61 และ ข.62

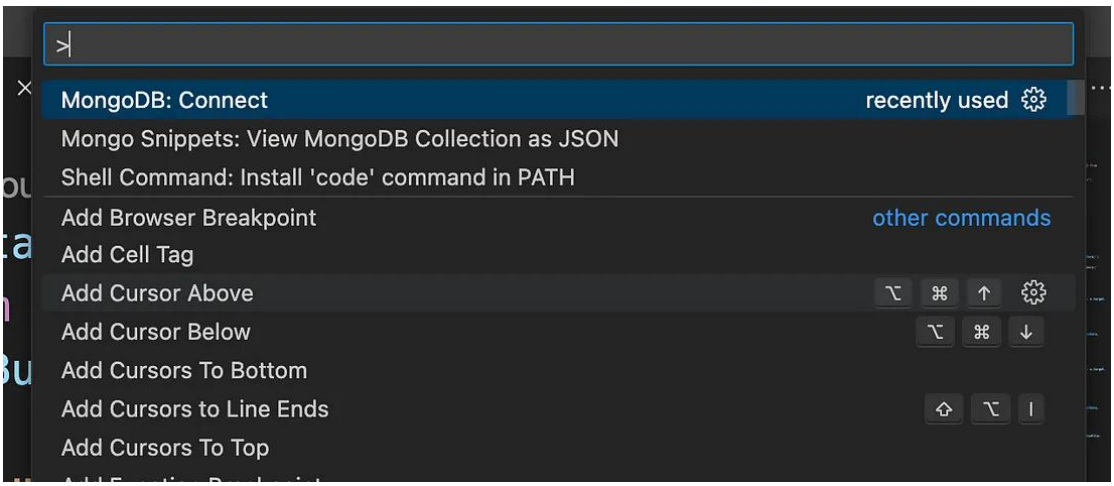


ภาพที่ ข.61 Menu Extensions



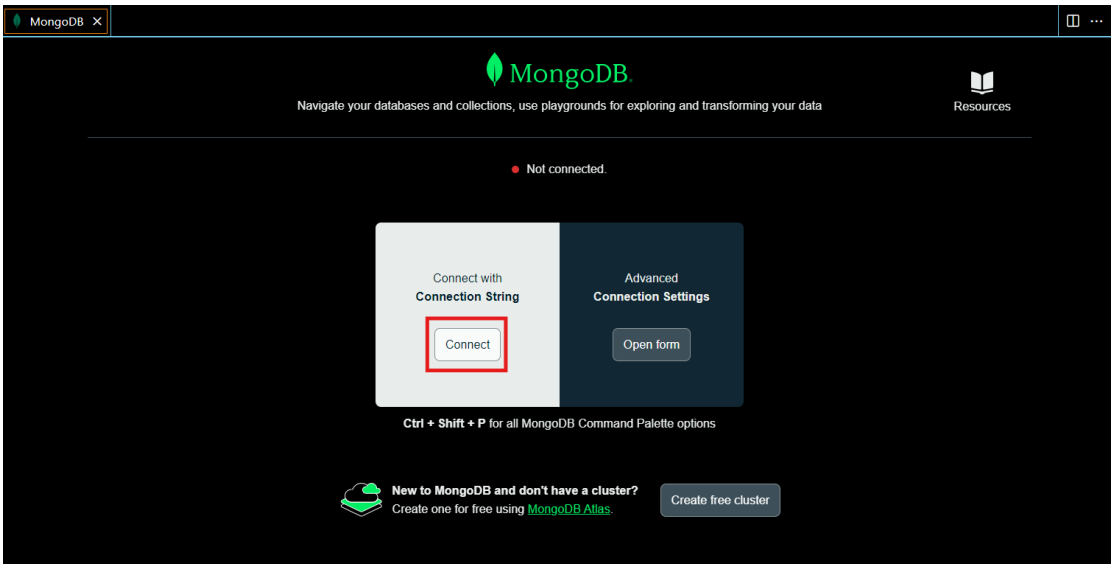
ภาพที่ ข.62 Install MongoDB for VS Code

จากภาพที่ ข.62 เมื่อติดตั้งเสร็จสิ้น ให้ไปที่เมนู “View” และ คลิกเลือก “Command Palette”จะได้ดังภาพที่ ข.63

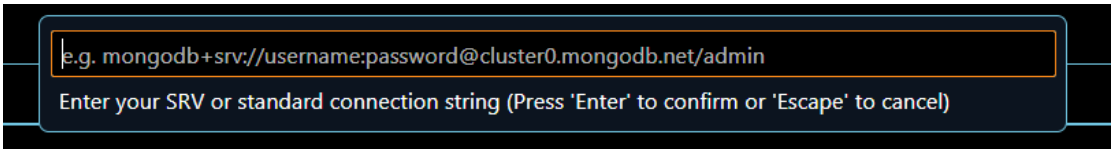


ภาพที่ ข.63 Command Palette

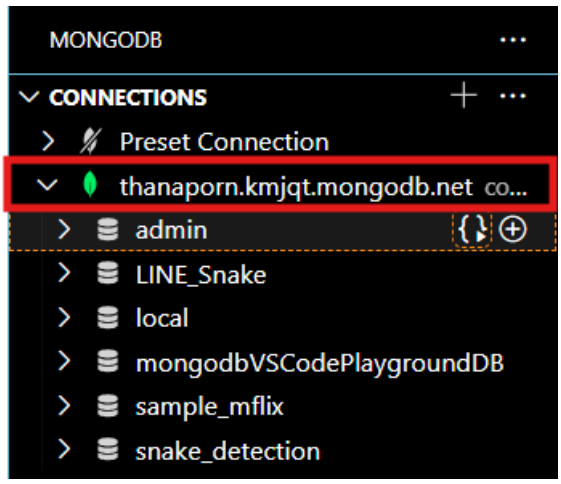
จากภาพที่ ข.63 เมื่อขึ้นดังภาพ ให้คลิกเลือก “MongoDB: Connect” จะได้ดังภาพที่ ข. ให้คลิกเลือก “Connect with Connection String.” และให้กรองจากข้อมูลที่ได้ทำการ Copy จาก MongoDB เอาไว้ เมื่อเสร็จสิ้นจะได้ดังภาพที่ ข.64



ภาพที่ ข.64 Connect with Connection String



ภาพที่ ข.65 Standard connection string



ภาพที่ ข.66 Connect Mongodb Successfully

4.1.3.การเรียกใช้ฐานข้อมูลMongoDB

ในการเรียกใช้ฐานข้อมูลMongoDB จะใช้คำสั่งในบรรทัดที่ 6 และ 7 ใน ภาพที่ ข.67

```python
from pymongo import MongoClient
from datetime import datetime
import uuid

# เชื่อมต่อ MongoDB
client = MongoClient('mongodb+srv://                                                    .net/?retryWrites=true&w=majority&appName=          ')
db = client['LINE_Snake']

# ฟังก์ชันแปลงวันที่ให้เป็นรูปแบบที่อ่านง่าย
def format_date(date):
    return date.strftime("%d %B %Y, %H:%M")

# ฟังก์ชันเก็บข้อมูลผู้ใช้
def store_user_data(user_id, display_name):
    user_data = {
        "userId": user_id,
        "displayName": display_name,
        "createdAt": format_date(datetime.now())  # แปลงวันที่เป็นรูปแบบที่อ่านง่าย
    }
    db.users.insert_one(user_data)

# ฟังก์ชันเก็บข้อมูลรูปภาพ
def store_image_data(image_id, user_id, image_url):
    image_data = {
        "imageId": image_id,
        "userId": user_id,
        "imageUrl": image_url,
        "uploadedAt": format_date(datetime.now())  # แปลงวันที่เป็นรูปแบบที่อ่านง่าย
    }
    db.images.insert_one(image_data)

# ฟังก์ชันเก็บผลการตรวจจับงู
def store_detection_result(image_id, is_snake, confidence):
    detection_data = {
        "detectionId": f"DET{str(uuid.uuid4())}",
        "imageId": image_id,
        "isSnake": is_snake,
        "confidence": confidence,
        "detectedAt": format_date(datetime.now())  # แปลงวันที่เป็นรูปแบบที่อ่านง่าย
    }
    db.snake_detection_results.insert_one(detection_data)

# ฟังก์ชันเก็บผลการจำแนกสายพันธุ์งู
def store_species_classification(image_id, species, species_confidence,species_image_id):
    classification_data = {
        "classificationId": f"CLASS{str(uuid.uuid4())}",
        "imageId": image_id,
        "species": species,
        "speciesConfidence": species_confidence,
        "speciesImageId": species_image_id,  # บันทึก imageId ของภาพที่ถูกครอบ
        "classifiedAt": format_date(datetime.now())  # แปลงวันที่เป็นรูปแบบที่อ่านง่าย
    }
    db.snake_species_classification.insert_one(classification_data)

def get_species_info_from_mongo(species_class):
    client = MongoClient('mongodb+srv://thanapornkead00:123456789snake@thanaporn.kmjqt.mongodb.net/?retryWrites=true&w=majority&appName=Thanaporn')
    db = client['LINE_Snake']
    collection = db["snake_info"]

    species_info = collection.find_one({"species": species_class})
    client.close()

    if species_info:
        return (
            species_info["eng_name"],
            species_info["species_name"],
            species_info["poison_info"],
            species_info["additional_info"],
            species_info["folder_link"],
            species_info["pic_info"],
            species_info.get("similar_snakes", None),
            species_info.get("similar_folder", None)
        )
    return None, None, None, None, None, None, None, None


def store_comment_collection(user_id, comment):
    comment_data = {
        "userId": user_id,
        "comment": comment,
        "commentAt": format_date(datetime.now())
    }
    db.comment_collection.insert_one(comment_data)

```

ภาพที่ ข.67 Code MongoDB

5.การตั้งค่าการทำงาน Line Chatbot

ในส่วนนี้จะเป็นการเรียกใช้โมเดลในการวิเคราะห์รูปภาพหลังจากที่ผู้ใช้งานส่งรูปภาพเข้ามาผ่านไลน์ โดยมีรายละเอียดดังนี้

5.1 การดึงข้อมูลผู้ใช้จาก LINE

ขั้นตอนนี้จะทำก็ต่อเมื่อผู้ใช้งานได้ส่งรูปภาพเข้ามาภายในline@ โดยยจะมีการดึงข้อมูลผู้ใช้งานจาก LINE ได้แก่ ชื่อไลน์,รูปภาพ หลังจากนั้นจะทำการเก็บรูปภาพที่ผู้ใช้งานส่งเข้ามาโดยบันทึกลง GridFS ดังภาพที่ ข.68

```python
@handler.add(MessageEvent, message=ImageMessage)
def handle_image_message(event):
    message_id = event.message.id
    user_id = event.source.user_id
    profile = line_bot_api.get_profile(user_id)
    display_name = profile.display_name
    store_user_data(user_id, display_name)


    # ดึงภาพจาก LINE
    message_content = line_bot_api.get_message_content(message_id)
    img = Image.open(io.BytesIO(message_content.content)).convert('RGB')
    img_np = np.array(img)
    img_bgr = img_np[:, :, ::-1].copy()  # Convert RGB to BGR

    # บันทึกรูปภาพลง GridFS
    img_io = io.BytesIO()
    img.save(img_io, format="JPEG")
    img_io.seek(0)
    image_id = fs.put(img_io, filename=f"{message_id}.jpg", user_id=user_id, uploaded_at=datetime.datetime.utcnow())
    print(f"Image uploaded successfully with ID: {image_id}")

    image_data = {
        "imageId": image_id,
        "userId": user_id,
        "uploadedAt": datetime.datetime.utcnow()
    }
    db.images.insert_one(image_data)

```

ภาพที่ ข.68 Code ดึงข้อมูลผู้ใช้จาก LINE

5.2 การตรวจจับงูภายในภาพ

ในขั้นตอนนี้จะทำการนำรูปภาพที่ผู้ใช้งานส่งเข้ามา นำไปวิเคราะห์ว่าภายในภาพนั้นมีงูหรือไหม หากตรวจพบงูจะทำต่อใน 5.3 แต่ตรวจไม่พบงูจะแสดงข้อความไปยังหาผู้ใช้งาน “ไม่พบงูภายในภาพ” ดังภาพที่ ข.69

```python
# ตรวจจับงูหรือสายยาง
    results = model_snake_or_hose(img_bgr)
    predictions = results.xyxy[0].cpu().numpy()
    messages = []

    if len(predictions) == 0:
        messages = (TextSendMessage(text="ไม่พบงูภายในภาพ."))
        return

    detected_objects = []
    image_id = str(uuid.uuid4())  # Create a unique ID for the image

```

ภาพที่ ข.69 Code การตรวจจับงูภายในภาพ

5.3 การวิเคราะห์สายพันธุ์งู

ขั้นตอนนี้จะทำการวิเคราะห์สายพันธุ์งูจากบริเวณรูปภาพที่ได้ทำการตรวจจับงูจากขั้นตอนที่ 5.2 โดย เมื่อวิเคราะห์เสร็จสิ้น จะแบ่งได้ 2 กรณี ดังนี้

5.3.1. สามารถวิเคราะห์สายพันธุ์ของงูภายในรูปภาพได้จะทำการตอบกลับผู้ใช้งาน โดยจะมีการบอกรายละเอียดดังนี้ 1.ชื่อสายพันธุ์งูภาษาไทยและภาษาอังกฤษ 2.ค่าความถูกต้องที่วิเคราะห์ได้ 3.ลักษณะพิษ 4.ลักษณะกายภาพ 5.Folderที่รวบรวมรูปภาพงู 6.งูที่มีลักษณะที่คล้ายกัน หลังจากตอบกลับจะมีการสอบถามผู้ใช้งานว่า “โมเดลทำนายถูกต้องหรือไหม” หากไม่ถูกต้องผู้ใช้งานสามารถพิมพ์สายพันธุ์งูที่ถูกต้องได้ ดังภาพที่ ข.70

```python
# ตรวจสอบสายพันธุ์งู
        species_results = model_snake_species(snake_img)
        species_predictions = species_results.xyxy[0].cpu().numpy()

        if len(species_predictions) > 0:
            species_class = int(species_predictions[0][5])
            species_confidence = float(species_predictions[0][4])  # Confidence ของการทำนายสายพันธุ์งู


            # ● แปลงรูปที่ตรวจจับงูได้เป็นไฟล์สำหรับบันทึก
            snake_img_pil = Image.fromarray(snake_img)  # แปลง NumPy array เป็น PIL Image
            snake_img_io = io.BytesIO()
            snake_img_pil.save(snake_img_io, format="JPEG")
            snake_img_io.seek(0)

            # ● บันทึกรูปภาพของงูลง GridFS
            species_image_id = fs.put(snake_img_io, filename=f"species_{str(uuid.uuid4())}.jpg", uploaded_at=datetime.datetime.utcnow())

            store_species_classification(image_id, species_class, species_confidence, species_image_id)
            # ดึงข้อมูลจาก MongoDB
            eng_name, species_name, poison_info, additional_info, folder_link, pic_info, similar_snakes, similar_folder = get_species_info_from_mongo(species_class)

            if species_name:
                send_quick_reply(event, species_name,eng_name, species_confidence, poison_info, additional_info, folder_link, pic_info, similar_snakes, similar_folder)
            else:
                line_bot_api.reply_message(
                    event.reply_token,
                    TextSendMessage(text="ตรวจพบงูภายในภาพ แต่ไม่สามารถระบุชนิดงูได้.")
                )
```

ภาพที่ ข.70 Code วิเคราะห์สายพันธุ์ของงูภายในรูปภาพได้

```python
line_bot_api = LineBotApi(channel_access_token, timeout=5)
handler = WebhookHandler(channel_secret)

def send_quick_reply(event, species_name, eng_name,species_confidence, poison_info, additional_info, folder_link, pic_info, similar_snakes, similar_folder):
    # ข้อความหลักเกี่ยวกับงู
    messages = [
        TextSendMessage(
            text=(f"{species_name} ({eng_name})\n"
                  f"ค่าความถูกต้อง : ({species_confidence * 100:.2f}%)\n"
                  f"พิษ : {poison_info}\n"
                  f"ลักษณะ : {additional_info}\n"
                  f"Folder รวมรูปงู : {folder_link}")
        ),
        ImageSendMessage(
            original_content_url=pic_info,
            preview_image_url=pic_info
        )
    ]

    # หากมีงูที่คล้ายกัน
    if similar_snakes:
        messages.append(
            TextSendMessage(
                text=(f"งูที่คล้ายกัน: {similar_snakes}\n"
                      f"Folder รวมรูปงู : {similar_folder}")
            )
        )

    # เพิ่มข้อความ Quick Reply หลังจากแสดงข้อมูล
    quick_reply_buttons = [
        QuickReplyButton(action=MessageAction(label="ถูกต้อง", text="ขอบคุณที่ใช้บริการ")),
        QuickReplyButton(action=MessageAction(label="ต้องการแก้ไขข้อมูล", text="ต้องการแก้ไขสายพันธุ์งู"))
    ]

    message = TextSendMessage(
        text="โมเดลทำนายถูกต้องหรือไหม",
        quick_reply=QuickReply(items=quick_reply_buttons)
    )

    messages.append(message)

    # ส่งข้อความทั้งหมด
    line_bot_api.reply_message(event.reply_token, messages)

```

ภาพที่ ข.71 Code ตั้งค่าการตอบกลับ

```python
@handler.add(MessageEvent, message=TextMessage)  # ✅ ถูกต้อง
def handle_message(event):
    user_id = event.source.user_id  # ดึง userId จาก LINE
    comment = event.message.text.strip()  # กำหนดให้ `comment` เป็นข้อความที่ผู้ใช้ส่งเข้ามา (ตัดช่องว่างก่อนและหลัง)

    # กรณีผู้ใช้พิมพ์ข้อความ "กรุณาพิมพ์ชื่อสายพันธุ์ที่ถูกต้อง"
    if comment == "ต้องการแก้ไขสายพันธุ์งู":
        line_bot_api.reply_message(
            event.reply_token,
            TextSendMessage(text="กรุณาพิมพ์ชื่อสายพันธุ์ที่ถูกต้อง")
        )
    else:
        # บันทึกข้อความที่ผู้ใช้พิมพ์หลังจาก "กรุณาพิมพ์ชื่อสายพันธุ์ที่ถูกต้อง"
        try:
            store_comment_collection(user_id,comment)  # บันทึกข้อมูลลงฐานข้อมูล
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text="บันทึกข้อมูลเรียบร้อย ขอบคุณที่ใช้บริการ!")
            )
        except Exception as e:
            # หากเกิดข้อผิดพลาดในการบันทึกข้อมูล
            line_bot_api.reply_message(
                event.reply_token,
                TextSendMessage(text=f"เกิดข้อผิดพลาดในการบันทึกข้อมูล: {str(e)}")
            )
```

ภาพที่ ข.72 Code ตั้งค่าการบันทึกcommentลงฐานข้อมูล

5.3.2. ไม่สามารถวิเคราะห์สายพันธุ์งูภายในรูปภาพได้จะทำการตอบกลับผู้ใช้งานว่า “ตรวจพบงูภายในภาพ แต่ไม่สามารถระบุชนิดงูได้” ดังภาพที่ ข.73

```
else:
                    line_bot_api.reply_message(
                        event.reply_token,
                        TextSendMessage(text="ตรวจพบงูภายในภาพ แต่ไม่สามารถระบุชนิดงูได้.")
                    )
```

ภาพที่ ข.73 Code ตั้งค่าเมื่อไม่สามารถวิเคราะห์สายพันธุ์ของงูภายในรูปภาพได้

6.การพัฒนา Web Site

ในขั้นตอนการพัฒนา Web Site จะออกเป็น 2 ส่วน ได้แก่

6.1 Frontend

ในส่วนของ fronted เราออกเป็น 2 ส่วน คือ 1.Sidebar 2.เนื้อหา โดยในส่วนของ Sidebar จะตั้งค่าการทำงานให้ส่งค่าไปยังตัวแปล “items” เพื่อแสดงข้อมูลตามค่าที่ส่งเข้ามา โดยค่าที่ส่งเข้ามาจะหมายถึงรหัสเลขสายพันธุ์ของงูแต่ละชนิด

```
<v-app-bar color="#1C5739">
      <v-app-bar-nav-icon variant="text" @click.stop="drawer = !drawer"></v-app-bar-nav-icon>
      <v-toolbar-title>SnakeBio_AI</v-toolbar-title>
      <v-spacer></v-spacer>
    </v-app-bar>

    <!-- Sidebar -->
    <v-navigation-drawer v-model="drawer" temporary>
      <v-list>
        <v-list-item v-for="item in items" :key="item.value" @click="filterImages(item.value)">
          {{ item.title }}
        </v-list-item>
      </v-list>
    </v-navigation-drawer>
```

ภาพที่ ข.74 Code Sidebar

```
items: [
        { title: "All Species", value: null },
        { title: "Banded Krait", value: 0 },
        { title: "Banded Rat Snake", value: 1 },
        { title: "Golden Flying Snake", value: 2 },
        { title: "King cobra", value: 3 },
        { title: "Malayan Krait", value: 4 },
        { title: "Malayan Pitviper", value: 5 },
        { title: "Monocellate Cobra", value: 6 },
        { title: "Pope's Pit Viper", value: 7 },
        { title: "Reticulated Python", value: 8 },
        { title: "Siamese Russell's Viper", value: 9 },
      ],
```

ภาพที่ ข.75 Code การกำหนดค่าในSidebar

```
computed: {
    filteredImages() {
      return this.selectedSpeciesID === null
        ? this.image_list
        : this.image_list.filter((img) => img.speciesID === this.selectedSpeciesID);
    },
  },
```

ภาพที่ ข.76 Code การนำไปค่าไปเชื่อมกับspeciesID

ถัดมาในส่วนของเนื้อหา จะแสดงข้อมูลเป็นตารางโดยมีรายละเอียดดังนี้ 1.ลำดับ 2.รูปภาพ 3.ค่าความถูกต้อง 4.รหัสสายพันธุ์ของงู 5.ชื่อสายพันธุ์ 6.วันที่และเวลา 7.คอมเม้น 8.ปุ่มลบ

```vue
<v-container class="table-container">
      <v-data-table :headers="headers_Navbar" :items="filteredImages" class="elevation-1" fixed-header dense>

        <!-- เพิ่มส่วนหัวของตารางเอง -->
        <template v-slot:top>
          <thead>
            <tr>
              <th class="text-center">#</th>
              <th class="text-center">Image</th>
              <th class="text-center">Confident</th>
              <th class="text-center">SpeciesID</th>
              <th class="text-center">Speciesname</th>
              <th class="text-center">Date</th>
              <th class="text-center">Comment</th>
              <th class="text-center">Data Management</th>
            </tr>
          </thead>
        </template>


        <template v-slot:item.index="{ index }">
          {{ index + 1 }}
        </template>

        <template v-slot:item.url="{ item }">
          <v-img :src="item.url" height="100%" max-height="200px" width="150px" contain></v-img>
        </template>

        <template v-slot:item.confidence="{ item }">
          {{ item.confidence.toFixed(4) }}
        </template>

        <!-- รหัสสายพันธุ์ -->
        <template v-slot:item.speciesID="{ item }">
          {{ item.speciesID }}
        </template>

        <!-- ชื่อสายพันธุ์ -->
        <template v-slot:item.speciesName="{ item }">
          {{ item.speciesName }}
        </template>

        <!-- วันที่ -->
        <template v-slot:item.date="{ item }">
          {{ item.date }}
        </template>

        <!-- คอมเมนต์ -->
        <template v-slot:item.comment="{ item }">
          <div class="text-center">
            {{ item.comments || 'No comment available' }}
          </div>
        </template>

        <!-- คอลัมน์ปุ่ม ลบ -->
        <template v-slot:item.actions="{ item }">
          <v-btn color="red darken-2" small @click="deleteImage(item)">
            <v-icon left>mdi-delete</v-icon> Delete
          </v-btn>
        </template>
      </v-data-table>
    </v-container>
```

ภาพที่ ข.77 Code ในส่วนของตารางแสดงข้อมูล

```
image_list: [],
      headers_Navbar: [
        { text: "#", value: "index", sortable: false, width: "50px" },
        { text: "Image", value: "url", sortable: false, width: "200px" },
        { text: "Confident", value: "confidence", width: "120px" },
        { text: "SpeciesID", value: "speciesID", width: "100px" },
        { text: "Speciesname", value: "speciesName", width: "200px" },
        { text: "Date", value: "date", width: "150px" },
        { text: "Comment", value: "comment", width: "250px" },
        { text: "Data Management", value: "actions", sortable: false, width: "180px" },
      ],
```

ภาพที่ ข.78 Code กำหนดตัวแปรในตาราง

```
methods: {
    async fetchImages() {
      try {
        const response = await fetch("http://127.0.0.1:8000/images");
        this.image_list = await response.json();
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    },

    filterImages(speciesID) {
      this.selectedSpeciesID = speciesID;
      this.drawer = false; // ปิด Sidebar หลังจากเลือก
    },

    // ฟังก์ชันสำหรับการแก้ไขข้อมูล
    editImage(item) {
      console.log("Editi
```

ภาพที่ ข.79 Code ดึงรูปภาพจาก API

6.2 Backend

ในส่วนของ backend จะใช้ FastAPI โดยมีการตั้งค่าดังนี้ 1.มีการกำหนดอนุญาตการเข้าถึงจาก localhost:3000 2.มีการเชื่อมต่อกับฐานข้อมูลMongoDB โดยมีการดึงข้อมูลจาก LINE_SNAKE และ GridsFS 3.มีการสร้าง MAP เพื่อสามารถเปลี่ยนจากรหัสสายพันธุ์เป็นชื่อสายพันธุ์ 4.มีการสร้าง Endpoint ทั้งหมด 3 endpoint ได้แก่ 1.สำหรับดึงรายการ classification ทั้งหมดในฐานข้อมูล 2.สำหรับดึงรูปภาพจาก GridFS ที่อยู่ใน snake_detection 3.สำหรับลบรูปภาพจาก GridFS ที่อยู่ใน snake_detection ดังภาพที่ ข.80,ข.81 และ ข.82

```python
app = FastAPI()

# กำหนดให้อนุญาตการเข้าถึงจาก localhost:3000
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],  # อนุญาตให้เข้าถึงจาก localhost:3000
    allow_credentials=True,
    allow_methods=["*"],  # หรือใส่เฉพาะ HTTP method ที่ต้องการ เช่น GET, POST
    allow_headers=["*"],  # หรือใส่เฉพาะ headers ที่ต้องการ
)

# เชื่อมต่อ MongoDB
MONGO_URI = "mongodb+srv:/.                                                    ow=majority&appName=          "
client = MongoClient(MONGO_URI)
# กำหนดฐานข้อมูลที่ใช้เก็บข้อมูล classification
db_line_snake = client['LINE_Snake']
comment_collection = db_line_snake['comment_collection']
db_images=db_line_snake['images']


# กำหนดฐานข้อมูลที่ใช้เก็บไฟล์รูปภาพ (GridFS)
db_snake_detection = client['snake_detection']
fs = gridfs.GridFS(db_snake_detection)

#  * Dictionary สำหรับแมป species ID -> ชื่อสายพันธุ์
SPECIES_MAP = {
    0: "งูสามเหลี่ยม",
    1: "งูสิงหางลาย",
    2: "งูเขียวพระอินทร์",
    3: "งูจงอาง",
    4: "งูทับสมิงคลา",
    5: "งูกะปะ",
    6: "งูเห่า",
    7: "งูเขียวหางไหม้",
    8: "งูเหลือม",
    9: "งูแมวเซา"
}

# Endpoint สำหรับดึงรายการ classification (ข้อมูลอยู่ใน LINE_Snake)
@app.get("/images")
async def list_images():
    classifications = db_line_snake.snake_species_classification.find().sort("classifiedAt", -1).limit(50)
    classification_list = list(classifications)

    image_list = []
    for classification in classification_list:
        species_id = classification.get("species")
        species_name = SPECIES_MAP.get(species_id, species_id)

        # ใช้ speciesImageId จาก classification เพื่อดึงไฟล์จาก GridFS ที่อยู่ใน snake_detection
        species_image_id = classification.get("speciesImageId")
        image_id = classification.get("imageId")  # imageId ที่ใช้เชื่อมโยงข้อมูล
        if not species_image_id or not image_id:
            continue

        # ใช้ imageId จาก classification เพื่อดึง userId จาก images
        image_data = db_images.find_one({"imageId": image_id})
        if image_data:
            user_id = image_data.get("userId")
        else:
            user_id = None

        # ใช้ userId เพื่อดึง comment จาก comment_collection
        comment = None
        if user_id:
            # หาค่า comment จาก comment_collection โดยใช้ userId
            comment_doc = comment_collection.find_one({"userId": user_id})
            if comment_doc:
                comment = comment_doc.get("comment")

        # สร้างรายการผลลัพธ์
        image_list.append({
            "id": str(classification["_id"]),
            "imageId": image_id,
            "url": f"http://127.0.0.1:8000/image/{species_image_id}",
            "confidence": classification.get("speciesConfidence"),
            "speciesID": species_id,
            "speciesName": species_name,
            "comments": comment,  # แสดง comment ที่ไม่เป็น null
            "date": str(classification.get("classifiedAt"))
        })

    return image_list
```

ภาพที่ ข.80 Code connect MongoDB and @app.get(“/images”)

```python
# Endpoint สำหรับดึงรูปภาพจาก GridFS ที่อยู่ใน snake_detection
@app.get("/image/{image_id}")
async def get_image(image_id: str):
    try:
        # ใช้ speciesImageId (ซึ่งควรเป็น ObjectId string) ในการดึงไฟล์จาก GridFS
        file = fs.get(ObjectId(image_id))
        return StreamingResponse(io.BytesIO(file.read()), media_type="image/jpeg")
    except gridfs.errors.NoFile:
        return {"error": "Image not found"}
```

ภาพที่ ข.81 Code @app.get(“/image/{image_id}”)

```python
# Endpoint สำหรับดึงรูปภาพจาก GridFS ที่อยู่ใน snake_detection
@app.delete("/delete-image/{image_id}")
async def delete_image(image_id: str):
    try:
        # เริ่มต้นการทำงานใน session และ transaction ของ MongoDB
        with client.start_session() as session:
            with session.start_transaction():
                # 🔹 ค้นหาข้อมูลจาก snake_species_classification โดยใช้ imageId
                classification = db_line_snake.snake_species_classification.find_one({"imageId": image_id}, session=session)
                if not classification:
                    raise HTTPException(status_code=404, detail="Classification not found")

                user_id = classification.get("userId")  # ดึง user_id
                species_image_id = classification.get("speciesImageId")  # ดึง speciesImageId

                # 🔹 ลบไฟล์ใน GridFS (ตรวจสอบว่า speciesImageId เป็น ObjectId ที่ถูกต้อง)
                if species_image_id and fs.exists(ObjectId(species_image_id)):
                    fs.delete(ObjectId(species_image_id), session=session)

                # 🔹 ลบข้อมูลจาก snake_species_classification
                db_line_snake.snake_species_classification.delete_one({"imageId": image_id}, session=session)

                # 🔹 ลบข้อมูลจาก snake_detection_results
                db_snake_detection.snake_detection_results.delete_one({"imageId": image_id}, session=session)

                # 🔹 ลบข้อมูลจาก images โดยใช้ imageId
                db_snake_detection.images.delete_one({"imageId": image_id}, session=session)

        return {"message": "Image and related data deleted successfully"}

    except Exception as e:
        raise HTTPException(status_code=500, detail=f"An error occurred: {str(e)}")
```

ภาพที่ ข.82 Code @app.delete(“/delete+image/{image_id}”)