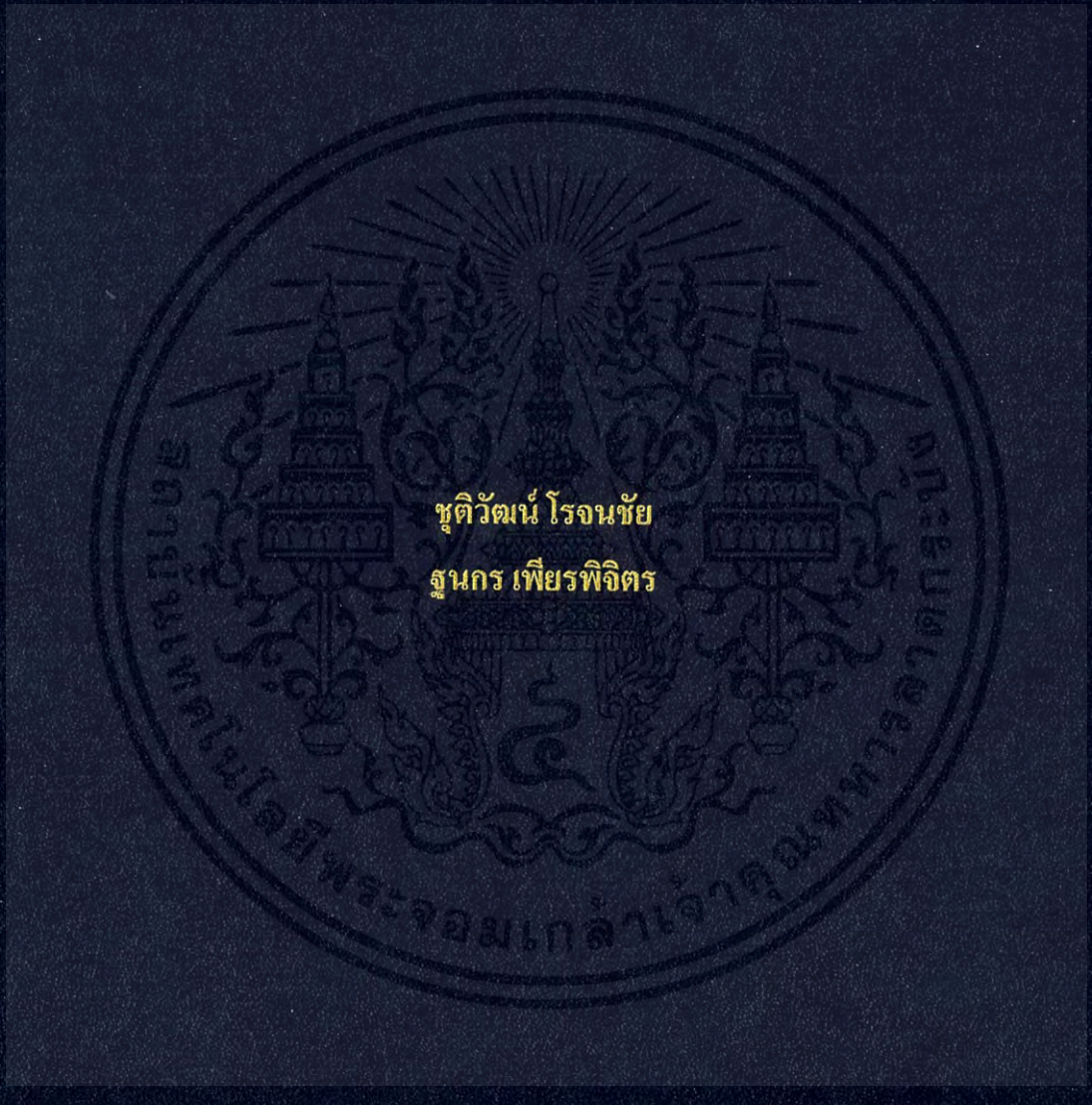
การตรวจจับและจำแนกวัตถุเพื่อนับจำนวนสุนัขจากวิดีโอ

OBJECT DETECTION AND CLASSIFICATION TO DETERMINE DOG POPULATION FROM VIDEO



ชุติวัฒน์ โรจนชัย

ฐนกร เพียรพิจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

การตรวจจับและจำแนกวัตถุเพื่อนับจำนวนสุนัขจากวิดีโอ

OBJECT DETECTION AND CLASSIFICATION TO DETERMINE DOG POPULATION FROM VIDEO

ชุติวัฒน์ โรจนชัย
ฐนกร เพียรพิจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การตรวจจับและจำแนกวัตถุเพื่อนับจำนวนสุนัขจากวิดีโอ

OBJECT DETECTION AND CLASSIFICATION TO DETERMINE DOG POPULATION FROM VIDEO

ผู้จัดทำ

1. นายชุติวัฒน์ โรจนชัย รหัสนักศึกษา 58010300
2. นายฐนกร เพียรพิจิตร รหัสนักศึกษา 58010316

ชุติเมษฎ์ ศรีนิลทา อาจารย์ที่ปรึกษา

(ผศ. ดร. ชุติเมษฎ์ ศรีนิลทา)

การตรวจจับและจำแนกวัตถุเพื่อนับจำนวนสุนัขจากวิดีโอ

นายชุติวัฒน์	โรจนชัย	58010300
นายฐนกร	เพียรพิจิตร	58010316
ผศ. ดร. ชุติเมษฏ์	ศรีนิลทา	อาจารย์ที่ปรึกษา

ปีการศึกษา 2561

บทคัดย่อ

การทราบจำนวนสุนัขในชุมชนนั้นเป็นเรื่องสำคัญทั้งในเรื่องของการวางแผนป้องกันโรคพิษสุนัขบ้าและการจัดหาที่พักอาศัยให้สุนัขจรจัดเหล่านี้ เทคนิคการสำรวจสำมะโนประชากรทั่วไปรวมถึงการขึ้นทะเบียนสุนัขนั้นเป็นวิธีการที่ใช้ได้กับสุนัขที่มีเจ้าของ แต่สำหรับสุนัขจรจัดการนับดูจะเป็นทางเลือกเดียวซึ่งต้องใช้เวลาและความพยายามในการประเมินอย่างมาก

เป้าหมายของโครงการนี้คือการออกแบบและพัฒนาระบบที่นับจำนวนสุนัขจากวิดีโอโดยอัตโนมัติ มีการตรวจจับการเคลื่อนไหวของสุนัขเพื่อไม่ให้เกิดการนับสุนัขที่ซ้ำซ้อนกัน

OBJECT DETECTION AND CLASSIFICATION TO DETERMINE DOG POPULATION FROM VIDEO

Mr.	Chutiwat	Rojanachai	58010300
Mr.	Tanakorn	Peanpijit	58010316
Asst.Prof.Dr.	Chutimet	Srinilta	Advisor

Academic Year 2018

ABSTRACT

It is important to know number to know number of dogs in a community Rabies and shelter control programs should be planned according to number of dogs in the community. Common census techniques include registering dogs, conducting survey and counting. For stray dogs, Counting seem to be the only choice. It takes lots of time and effort to get a close estimation from counting.

The goal of this project is to design and develop a system that automatically counts number of dog from video clip. Dog movements are tracked so that any given dog will not be counted twice.

กิตติกรรมประกาศ

โครงงาน การตรวจจับและจำแนกวัตถุเพื่อนับประชากรสุนัขจากวิดีโอ สามารถสำเร็จลุล่วงไปได้ด้วยดี ต้องขอขอบคุณ อาจารย์ ผศ. ดร. ชุติเมษฎ์ ศรีนิลทา ผู้คอยให้คำปรึกษาในด้านปัญหา และการตัดสินใจในแนวทางในการทำโครงงานฉบับนี้

ขอขอบคุณอาจารย์ทุกท่านที่สั่งสอนสะสมองค์ความรู้ให้ผู้จัดทำมาตั้งแต่อดีตจนถึงปัจจุบัน จนมีความสามารถที่จะทำหลาย ๆ สิ่งเพื่อให้ปริญญานิพนธ์เล่มนี้เสร็จสมบูรณ์ได้ ซึ่งผู้จัดทำได้ซาบซึ้งพระคุณของอาจารย์ทุก ๆ ท่านเป็นอย่างยิ่ง

ขอขอบคุณคณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่เอื้อเฟื้อทรัพยากรต่าง ๆ ในการทำปริญญานิพนธ์ฉบับนี้

ขอขอบคุณห้องวิจัย SAIG ที่เอื้อเฟื้อสถานที่ความรู้ในด้านต่าง ๆ ประกอบงานวิจัย

ขอขอบคุณครอบครัว เพื่อนและบุคคลทุก ๆ ท่าน ที่คอยเป็นกำลังใจ และ สนับสนุนทุกสิ่งที่ส่งผลให้โครงงานนี้สำเร็จลุล่วงไปด้วยดี

ชุติวัฒน์ โรจนชัย

ฐนกร เพียรพิจิตร

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูป หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ตั้งแต่อดีตจนถึงปัจจุบัน ปัญหาสุนัขจรจัดตามถนนสาธารณะ หรือ ตามวัด และหลาย ๆ สถานที่อื่น ๆ เป็นปัญหาที่สะสมมานานซึ่งเทศบาลท้องถิ่นจะเป็นหน่วยงานแรกในการจัดการปัญหาสุนัขเหล่านี้ ไม่ว่าจะเป็น การจับสุนัขจรจัด การทำหมัน การให้วัคซีนป้องกันโรคพิษสุนัขบ้า และอื่น ๆ อีกมาก ซึ่งในการทำงานแต่ละครั้งมักจะพบว่าไม่สามารถทำได้อย่างทั่วถึง สาเหตุหนึ่งเนื่องจาก การเตรียมอุปกรณ์ วัคซีน หรือสัตวแพทย์ที่ไม่เพียงพอกับจำนวนสุนัขในพื้นที่ ทำให้ปัญหาสุนัขจรจัดเป็นปัญหาที่แก้ไขได้ยาก ซึ่งระบบในการนับประชากรสุนัขที่ใช้กันในปัจจุบันมีอยู่หลายวิธี ไม่ว่าจะเป็นการทำแบบสอบถามคนในท้องที่ การนับอินดิเคเตอร์ลักษณะสมดุล อาทิเช่น ตัวผู้ ตัวเมีย ลูกสุนัข จะมีอัตราส่วนใกล้เคียงกัน จึงสามารถนับแค่อินดิเคเตอร์เดียวก็สามารถนับประชากรสุนัขได้โดยประมาณ

ในปัจจุบันมีเทคโนโลยีต่าง ๆ ให้ความสนใจทางด้านการประมวลผลภาพและจำแนกได้ว่าวัตถุในแต่ละภาพนั้นมีอะไรบ้าง กลุ่มของข้าพเจ้าจึงมีแนวความคิดที่ต้องการพัฒนาระบบใช้การจำแนกภาพ (Image Classification) เพื่อแยกภาพสุนัขออกจากวัตถุอื่นวิดีโอแล้วนำภาพในแต่ละเฟรมมาตรวจสอบว่าเป็นสุนัขตัวเดิมหรือไม่เพื่อเพิ่มความแม่นยำในการเก็บจำนวนประชากรสุนัข

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อเป็นต้นแบบระบบการนับประชากรสุนัขด้วยวิดีโอที่ช่วยอำนวยความสะดวกในการเก็บข้อมูลสถิติ
2) เพื่อศึกษาเทคโนโลยีและขอบเขตของระบบการจำแนกภาพว่าจะสามารถแยกได้แม่นยำ และละเอียดมากน้อยเพียงใด
3) เป็นการศึกษาการทำงานร่วมกันของเทคโนโลยีต่าง ๆ ในระบบคอมพิวเตอร์ อาทิ Machine Learning ที่นำมาใช้ในการประมวลผลภาพ และอื่น ๆ

1.3 ขอบเขตของโครงงาน

1) ใช้เทคโนโลยีการประมวลผลภาพของ Mask R-CNN ที่ทำงานผ่าน TensorFlow สร้างโปรแกรมนับจำนวนสุนัขจากวิดีโอผ่าน Environment ของกูเกิ้ลคลาวด์แพลทฟอร์ม
2) ระบบสามารถวิเคราะห์แลกจำแนกสุนัขจากวิดีโอได้อย่างถูกต้องและแม่นยำให้ได้มากที่สุด
3) ระบบสามารถบอกจำนวนสุนัขในวิดีโอนั้น ๆ ว่ามีจำนวนสุนัขอยู่ในวิดีโอเป็นจำนวนเท่าใด
4) ระบบจะให้ผู้ใช้งานอัพโหลดวิดีโอเข้ามาผ่านทางเว็บแอพพลิเคชั่นและทำการประมวลผลผ่าน Google Cloud และแสดงผลให้ผู้ใช้งานทางเว็บแอพพลิเคชั่น

1.4 ขอบเขตของโครงงาน

1) สามารถติดตามประชากรสุนัขในพื้นที่ได้อย่างแม่นยำ
2) สามารถนำโครงงานนี้ไปใช้งานในงานแขนงอื่นได้
3) พัฒนาระบบการประเมินจำนวนประชากรจากทฤษฎีที่มีอยู่เดิม

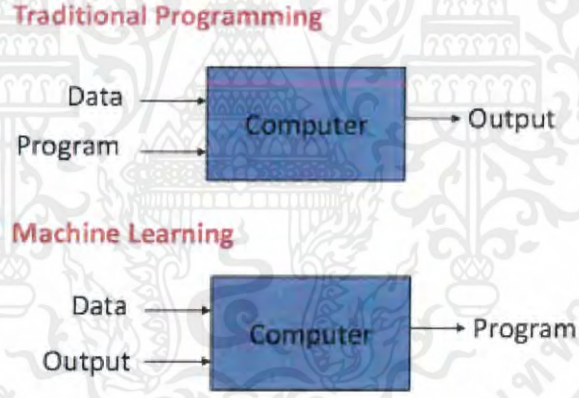
1.5 ข้อจำกัดของโครงงาน

1) วิดีโอที่ผู้ใช้งานอัพโหลดเข้ามาต้องไม่มีฟิลเตอร์พิเศษในการแก้สีให้มีความผิดเพี้ยนอาทิเช่น การกรองแสงภาพกลางคืน (Night Vision) เป็นต้น
2) วิดีโอที่ผู้ใช้งานอัพโหลดเข้ามาต้องเห็นภาพสุนัขอย่างชัดเจนโดยสายตามนุษย์

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้ของเครื่อง (Machine Learning)

การเรียนรู้ของเครื่อง (Machine Learning) เป็นหนึ่งในสาขาของวิชาปัญญาประดิษฐ์ เกี่ยวกับการศึกษาการสร้างแนวคิดอัลกอริทึมที่สามารถเรียนรู้ และ ทำนายข้อมูลได้ซึ่งการเรียนรู้ของเครื่องนั้นจะแตกต่างกับการเขียนโปรแกรมโดยทั่วไปเนื่องจาก การเขียนโปรแกรมทั่วไปนั้นผู้ใช้งานจะต้องป้อนข้อมูล และ วิธีการจัดการกับข้อมูลเหล่านั้นและให้คอมพิวเตอร์คำนวณผลลัพธ์ออกมาให้ผู้ใช้งาน แต่การเรียนรู้ของเครื่องนั้นจะแตกต่างออกไปโดยผู้ใช้งานจะใส่ข้อมูลและผลลัพธ์ที่ผู้ใช้งานต้องการเข้าไปและให้คอมพิวเตอร์คำนวณหาความสัมพันธ์ของข้อมูลและผลลัพธ์ออกมาให้มีความตรงกันให้มากที่สุด



รูป 2.1 ข้อเปรียบเทียบการเขียนโปรแกรมปกติกับการเรียนรู้ของเครื่อง

โดยการเรียนรู้ของเครื่องจะสามารถแบ่งได้เป็น 2 ประเภทหลัก ๆ โดยจำแนกตามหน้าที่ขอบเขตของงานได้ดังต่อไปนี้

2.1.1 การเรียนรู้แบบมีผู้สอน (Supervised Learning)

การเรียนรู้แบบมีผู้สอนนั้นนักพัฒนาจะต้องคอยจำแนกข้อมูลให้เครื่องว่าถ้าข้อมูลที่ป้อนเข้ามามีลักษณะแบบไหนแล้วจะได้ผลลัพธ์ลักษณะไหนซึ่งการสอนเครื่องนั้นมีได้หลายรูปแบบ อาทิ การใส่ข้อมูลใส่ลักษณะตาราง การกำหนดกรอบรูปบริเวณที่สนใจในภาพ และ อื่น ๆ เครื่องจะทำ

การค้นหาความสัมพันธ์ของข้อมูลที่เราได้ป้อนเข้าไปจากนั้นจะบันทึกความสัมพันธ์นี้ไว้เป็นโมเดลของข้อมูล ผู้ใช้งานก็จะนำโมเดลข้อมูลนี้ทดลองกับข้อมูลใหม่ที่ไม่ได้ถูกนำมาสอนและให้คอมพิวเตอร์ทำนายผลลัพธ์ออกมา การเรียนรู้แบบมีผู้สอนนี้มักใช้ในการ คำนวณราคาต่าง ๆ , การวิเคราะห์รูปภาพ , การวิเคราะห์ช่วงเวลาการเข้าซื้อขายหุ้น และอื่น ๆ

2.1.2 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning)

การเรียนรู้แบบไม่มีผู้สอนนั้นจะมีลักษณะคล้ายคลึงกับการเรียนรู้แบบมีผู้สอน เพียงแต่ชุดข้อมูลที่ส่งไปให้จำแนกชนิดและหาความสัมพันธ์นั้นคอมพิวเตอร์จะไม่ทราบว่าข้อมูลที่ได้รับมาคือสิ่งใด บริเวณที่สนใจอยู่ส่วนไหน การเรียนรู้แบบไม่มีผู้สอนนั้นจะเป็นการจับข้อมูลออกเป็นกลุ่ม ๆ ตามที่คอมพิวเตอร์นั้นทำการคัดแยกและคาดเดาแบบสุ่ม ๆ ได้คอมพิวเตอร์นั้นมักจะแยกข้อมูลออกมาเป็นลักษณะกลุ่ม ตามลักษณะข้อมูลที่คอมพิวเตอร์รับรู้ว่ามีลักษณะคล้ายกัน โดยที่ข้อมูลที่คล้ายกันจะนำมาอยู่กลุ่มเดียวกันแล้วให้นักพัฒนากำหนดชื่อของข้อมูลแต่ละกลุ่มต่อไป

2.1.3 ขั้นตอนการสร้างโมเดล

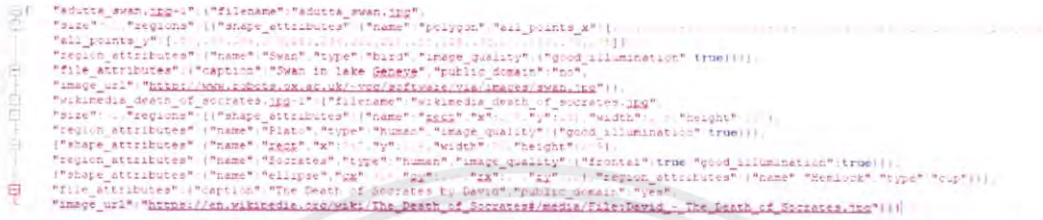
ขั้นตอนในการสร้างโมเดลนั้นมีขั้นตอนในการทำงานอยู่ 3 ขั้นตอนดังนี้

2.1.3.1 การเตรียมข้อมูลสร้างโมเดล

การเตรียมข้อมูลสร้างโมเดลนั้นได้จากการศึกษาข้อมูลว่าอยู่ในลักษณะอย่างไร โดยจะทำการจัดเตรียมข้อมูลนั้นในรูปแบบที่เราต้องการ อาทิเช่น รูปภาพ ตัวหนังสือ และอื่น ๆ และทำการสร้างโครงข้อมูลบอกจุดที่เราสนใจเพื่อทำการสอนคอมพิวเตอร์ว่าสิ่งที่เราสนใจอยู่บริเวณนี้

รูป 2.2 การเตรียมข้อมูลสร้างโมเดลโดยใช้รูปภาพ

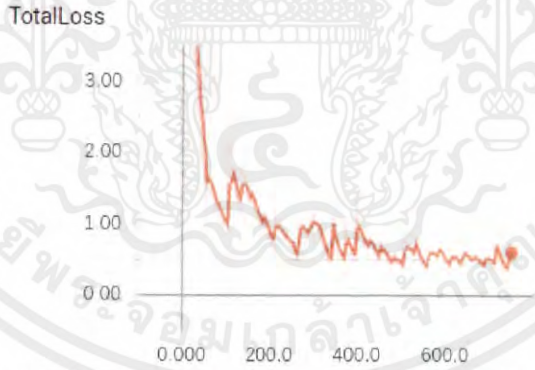
ในรูปจะแสดงให้เห็นว่ามีภาพอยู่หนึ่งภาพเป็นภาพหงส์ซึ่งมนุษย์ได้ทำการลากเส้นรอบตัวหงส์เพื่อระบุว่าภาพส่วนนี้คือส่วนที่มนุษย์สนใจโดยสามารถทำได้โดยการเก็บเป็นไฟล์ตัวหนังสือที่ระบุ Position ในแต่ละจุดลากบรรจบกันและเก็บข้อมูลในลักษณะ JSON ไฟล์



รูป 2.3 ตัวอย่างไฟล์ JSON บอกลักษณะรูปร่างที่ผู้ใช้งานเตรียมไว้

2.1.3.2 ฝึกสอนโมเดล

ขั้นตอนนี้คือการสอนให้เครื่อง ทำการหาความสัมพันธ์ของไฟล์โมเดลที่เราสร้างขึ้นมาโดยผ่านอัลกอริทึมที่เราสร้างเพื่อทำการฝึกสอน โดยคอมพิวเตอร์จะทำการสร้างชุดคำสั่งที่พยายามสร้างผลลัพธ์ให้ตรงกับชุดข้อมูลที่เราสร้างขึ้นให้ได้ใกล้เคียงที่สุด และเลือกโมเดลที่ใกล้เคียงที่สุดเอาไว้เก็บไว้เป็นไฟล์โมเดลที่ใช้งานจริง



รูป 2.4 กราฟทดสอบความแม่นยำของโมเดลอิงจากข้อมูลที่เตรียมมา

กราฟที่ได้นี้เป็นกราฟที่ได้จากการนำโมเดลข้อมูลที่ได้มาทำการเปรียบเทียบกับชุดข้อมูลที่เราได้นำไปสอนว่ามีค่าความแตกต่างกันมากน้อยเพียงใดในแต่ละรอบโดยกราฟรูป 2.3 เป็นตัวอย่างแสดงถึงค่าความผิดพลาดของผลลัพธ์ที่การเรียนรู้ของเครื่องนำมาใช้ เทียบกับชุดข้อมูลที่ผู้ใช้งานสอนให้คอมพิวเตอร์เรียนรู้ว่าในแต่ละรอบแตกต่างกันมากน้อยเพียงใด

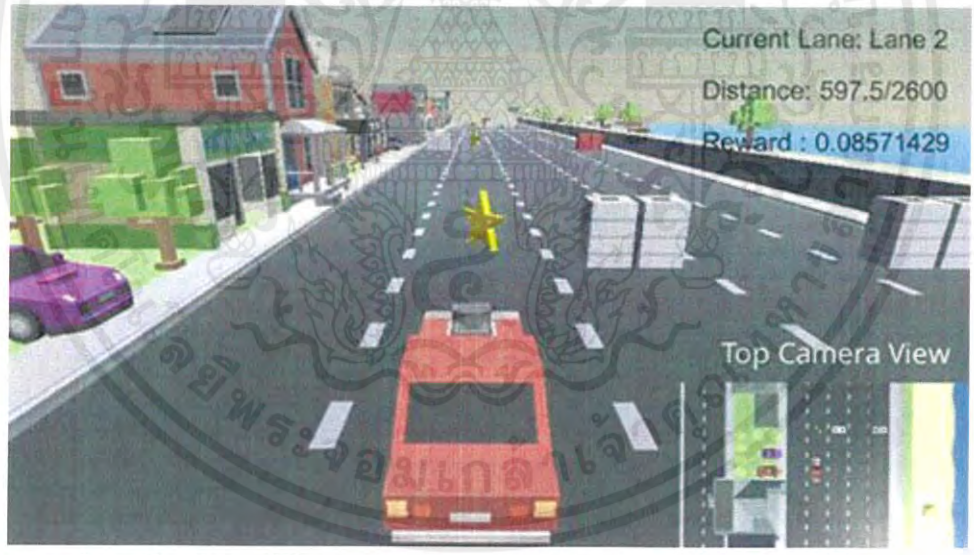
2.1.3.3 การทดสอบโมเดล

ขั้นตอนการทดสอบโมเดลจะทำการวัดผลว่าโมเดลที่เราได้ทำการฝึกมานั้น มีความแม่นยำมากแค่ไหน โดยจะนำข้อมูลทดสอบมาทดสอบกับโมเดลข้อมูล ค่าความแม่นยำ และความถูกต้อง โดยค่าความแม่นยำสามารถคำนวนได้จากสมการนี้

$$\text{ความแม่นยำ} = \frac{\text{จำนวนภาพที่ทายถูกต้อง}}{\text{จำนวนภาพทั้งหมด}}$$

2.1.4 การเรียนรู้แบบเสริมกำลัง

การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) เป็นวิธีการหนึ่งของการทำการเรียนรู้โดยเครื่อง โดยการใช้วิธีลองผิดลองถูกตัวอย่างเช่นการให้คอมพิวเตอร์ทำการประคองเลนบนถนนโดยการกำหนดว่า หากเข้าใกล้เส้นขอบทางด้านซ้ายมากเกินไปคะแนนที่คอมพิวเตอร์จะได้ก็จะลดลง และ หากเข้าใกล้เส้นขอบทางด้านขวาเกินไปคะแนนที่คอมพิวเตอร์จะได้ก็จะลดลงเช่นกัน



รูป 2.5 ตัวอย่างการใช้ Reinforcement Learning กับการประคองเลนบนถนน

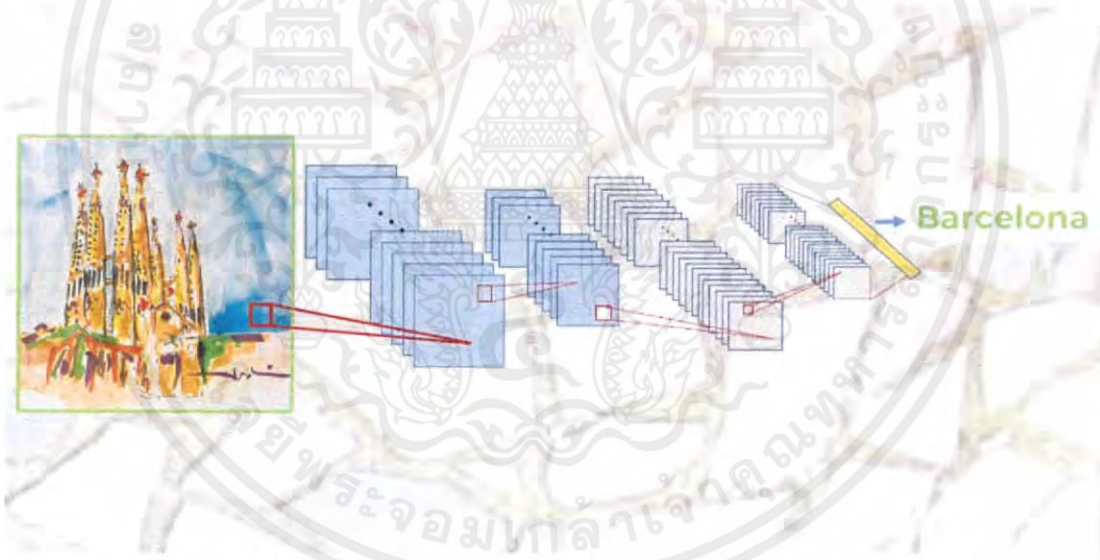
(ที่มา : https://unity3d.com/machine-learning, 2019)

2.2 โครงข่ายประสาทเทียม

โครงข่ายประสาทเทียม (Neural Network) เป็นเทคนิคหนึ่งของการเรียนรู้ของเครื่อง โดยมีแนวทางมาจากการพยายามจำลองระบบประสาทของมนุษย์จากสมองที่มีการทำงานซับซ้อนสูงเป็นอย่างมาก ซึ่งในสมองนั้นมีส่วนย่อยในสมองที่ถูกเรียกว่า นิวรอน ซึ่งนิวรอนเป็นแรงบันดาลใจในการเรียกชื่อโครงข่ายประสาทเทียมนี้ว่า Neural Network

2.3 โครงข่ายประสาทเทียมคอนโวลูชัน

โครงข่ายประสาทเทียมคอนโวลูชัน (Convolutional Neural Network) เป็นโครงข่ายประสาทเทียมที่ได้ต้นแบบมาจากการมองเห็นของมนุษย์ โดยมีหลักการทำงานคือ มองภาพเป็นพื้นที่ย่อย ๆ และจะทำการแยกคุณลักษณะ (Feature Extraction) ของพื้นที่นั้นว่ามีเส้นลักษณะใดและทำการมองพื้นที่รอบ ๆ พื้นที่สนใจไปเรื่อย ๆ

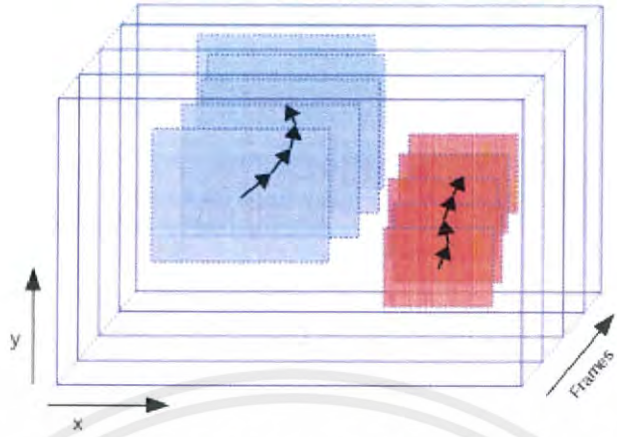


รูป 2.6 การดึงคุณลักษณะของภาพโดยใช้ Convolutional Neural Network

(ที่มา : https://towardsdatascience.com, 2018)

2.4 การตรวจจับการเคลื่อนไหววัตถุ (Object Tracking)

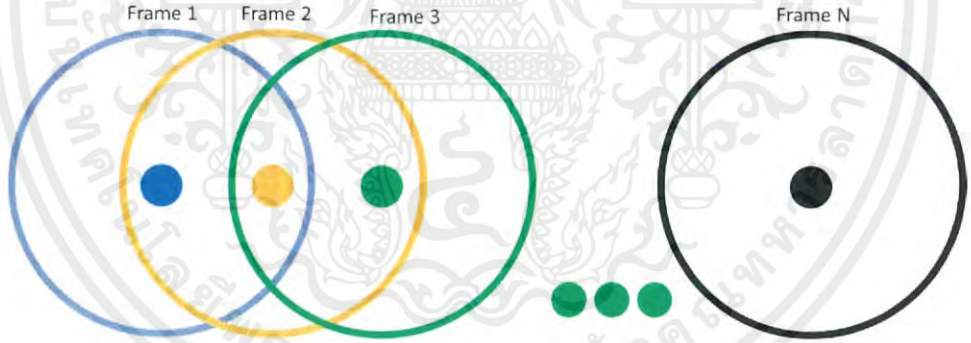
การตรวจจับการเคลื่อนไหววัตถุเป็นวิธีการหนึ่งของการประมวลผลภาพ (Image Processing) ที่มีวัตถุประสงค์ในการเรียนรู้การเคลื่อนไหวของวัตถุต่าง ๆ ในวิดีโอซึ่งปัจจุบันมีอัลกอริทึมในการนับวัตถุในวิดีโอหลายรูปแบบให้เลือกใช้งาน



รูป 2.7 การทำงานของระบบตรวจจับการเคลื่อนไหวของวัตถุ

2.4.1 การตรวจจับจุดศูนย์กลางวัตถุ (Centroid Tracking)

การตรวจจับจุดศูนย์กลางวัตถุเป็นอัลกอริทึมพื้นฐานในการตรวจจับการเคลื่อนไหวของวัตถุมีหลักการคือตรวจจับวัตถุในแต่ละภาพและตรวจสอบว่าจุดศูนย์กลางของวัตถุที่สนใจนั้นยังคงอยู่ในกรอบภาพที่แล้วหรือไม่

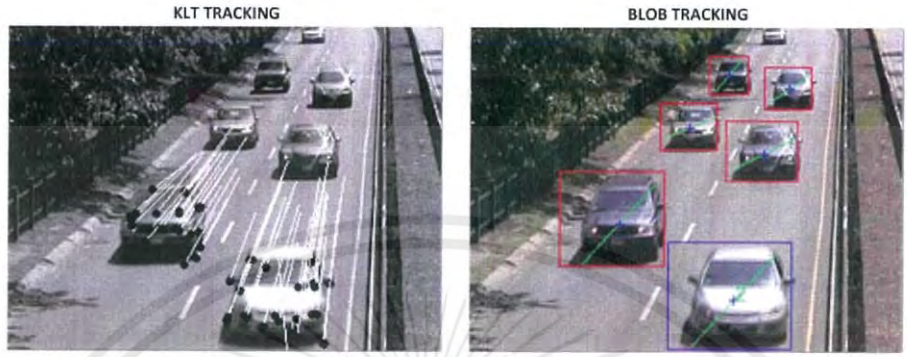


รูป 2.8 การทำงานของการตรวจจับจุดศูนย์กลางวัตถุ

2.4.2 OpenCV Object Tracking

OpenCV เป็นLibrary ในการทำ Computer Vision ที่ถูกพัฒนาจากบริษัท Intel ซึ่ง OpenCV นั้นมีฟังก์ชั่นเกี่ยวกับการทำงานของรูปภาพหลายฟังก์ชั่น อาทิ การตรวจจับใบหน้าคน การ

นับวัตถุ การจดจำท่าทาง และอื่น ๆ อีกทั้งเป็นซอฟแวร์ฟรีรองรับได้หลายภาษา อาทิ Java, Python, C++ และอื่น ๆ



รูป 2.9 ตัวอย่างการทำงานของฟังก์ชันการตรวจจับวัตถุของ OpenCV

(ที่มา : https://towardsdatascience.com, 2015)

ในที่นี้เราได้ใช้ KLT Tracking และ BLOB Tracking ที่มีอยู่ใน OpenCV ในการทดลอง โดย KLT Tracking จะใช้หลักการคือนำ Bounding Box ที่สนใจนั้นมาลากเส้น Vector ตามที่ Pixel นั้นเลื่อนไป และทำการใช้ Linear Regression Algorithm ในการจัดกลุ่มของแต่ละ Bounding Box ว่ามี Vector ที่ใกล้เคียงกันก็จะนับว่าเป็นวัตถุเดียวกัน

รูป 2.10 ตัวอย่างการทำงานของฟังก์ชันการตรวจจับวัตถุของ OpenCV โดยใช้ KLT Algorithm

(ที่มา : https://towardsdatascience.com, 2015)

BLOB Tracking จะเป็นจะใช้ค่าความสว่างในพื้นที่ที่สนใจและทำการตรวจสอบว่าพื้นที่สว่างนั้นมีการแพร่ค่าแสงไปในทิศทางรูปแบบใด

รูป 2.11 ตัวอย่างการทำงานของฟังก์ชันการตรวจจับวัตถุของ OpenCV โดยใช้ BLOB Algorithm (ที่มา : https://github.com/andrewssobral/simple_vehicle_counting, 2015)

2.5 เครื่องมือที่ใช้ในการติดต่อกับเซิร์ฟเวอร์

2.5.1 เทนเซอร์โฟลว์ (TensorFlow)

เทนเซอร์โฟลว์เป็น Library ที่ช่วยเหลือการทำ กระบวนการเรียนรู้ของเครื่องจักร (Machine Learning) ถูกพัฒนาด้วยทีม Google Brain และเป็น OpenSource และยังรองรับกระบวนการประมวลผล Deep Learning รองรับภาษา Python 2.0 , 3.0 รวมไปถึงภาษา C++

ข้อดีของเทนเซอร์โฟลว์คือ ไม่จำเป็นต้องพัฒนาโครงสร้างระบบเครือข่ายประสาท (Neural Network) และ เทนเซอร์โฟลว์ยังรองรับการประมวลผลโดยใช้หน่วยประมวลผลกราฟฟิกร่วม (GPU Driven Programming) ทำให้ทำงานได้อย่างรวดเร็วแม้โค้ดที่พัฒนาจะไม่ใช่ Parallel Programming ก็ตาม

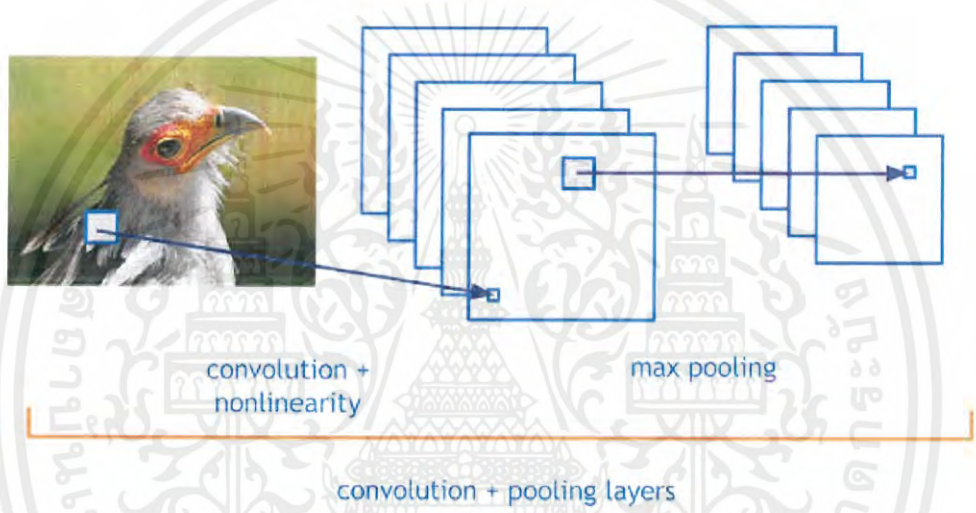
2.5.2 มาสก์อาร์ซีเอ็นเอ็น (Mask R-CNN)

มาสก์อาร์ซีเอ็นเอ็น (Mask R-CNN) ย่อมาจาก Mask Region with Convolutional Neural Network เป็นระบบที่ใช้งานในการจำแนกวัตถุในภาพโดยพัฒนาโดย Facebook IR Team เป็น Library ที่รองรับการ Segmentation โดยใช้ Dataset ของภายนอก หรือ Dataset ของตนเอง หรือนำ Dataset ของคนอื่นมาพัฒนาต่อก็ได้ รองรับภาษา Python

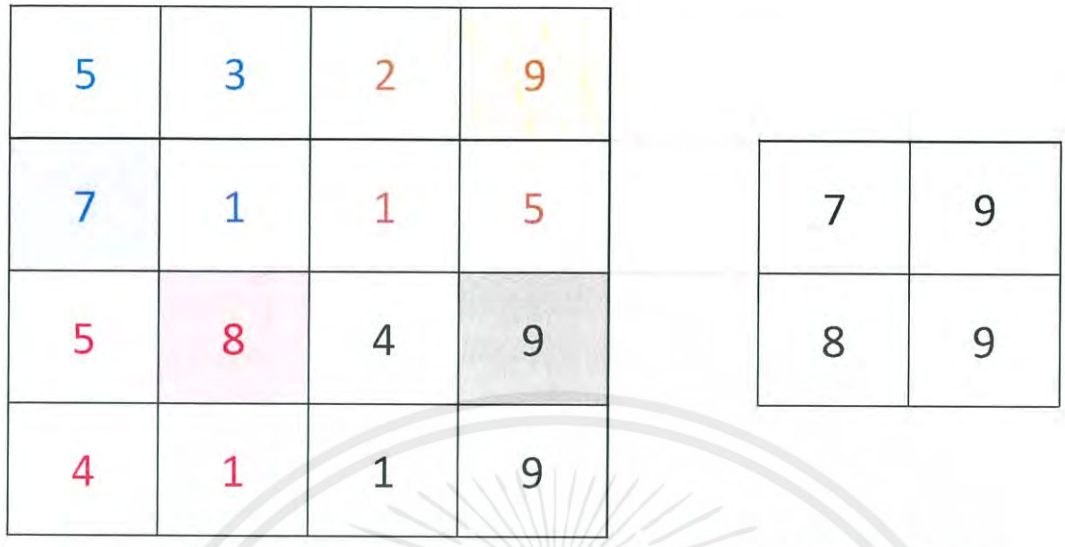
Mask R-CNN แบ่งปัญหาออกเป็น 2 ส่วน

2.5.2.1 การตรวจจับวัตถุ (Object Detection)

เป็นกระบวนการในการนำภาพแต่ละภาพมาประมวลผลว่ามีจำนวน วัตถุในภาพจำนวนเท่าใดโดยจะสร้างกรอบครอบวัตถุที่สนใจเอาไว้ (Bounding Box) โดยการให้คอมพิวเตอร์คำนวณภาพในตำแหน่งต่าง ๆ จากนั้นลดทอนลายละเอียดโดยการใช้ทฤษฎี Max Pooling Layer และทำซ้ำ ๆ ไปเรื่อย ๆ จนได้ ภาพขนาด 28*28 Pixel ซึ่งเป็นค่ามาตรฐานของ Mask R-CNN

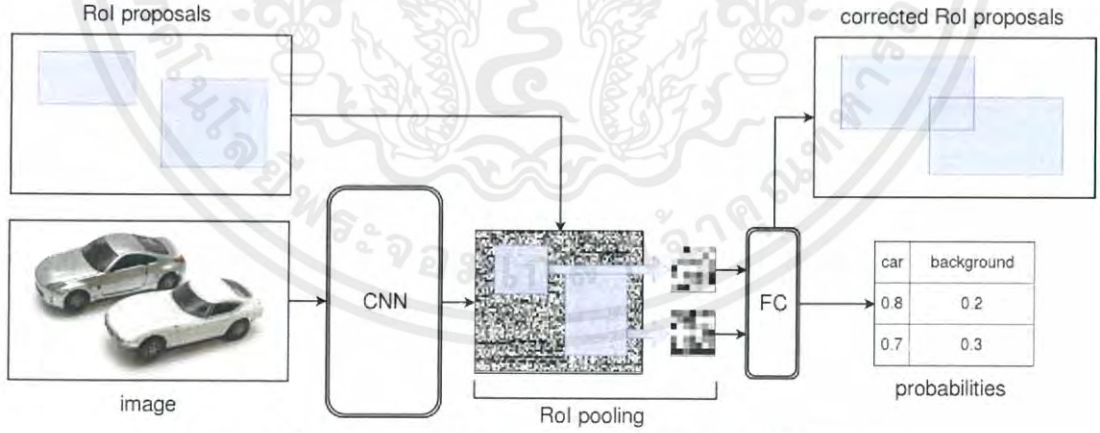


รูป 2.12 กระบวนการสร้าง Bounding Box บน Mask R-CNN



รูป 2.13 กระบวนการลดทอนรายละเอียดภาพของ Max Pooling Layer

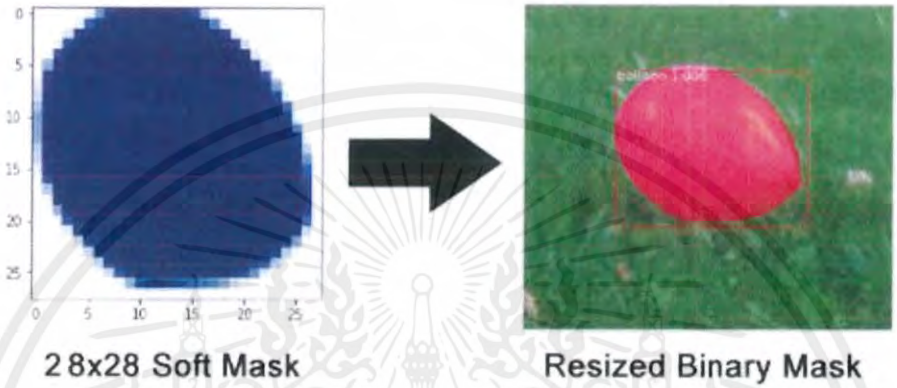
Mask R-CNN จะทำการสุ่มพื้นที่บนภาพเป็นสี่เหลี่ยมผืนผ้า (Region of Interest) จากนั้นก็จะนำกรอบภาพเหล่านั้นเข้าสู่กระบวนการ Convolutional Neural Network เพื่อหา Feature ที่สนใจ และทำการ Max Pooling Layer ให้ภาพมีขนาดเล็กลงแต่ยังคงเก็บคุณสมบัติต่าง ๆ ไว้ให้ได้มากที่สุดแล้วจึงนำไปตรวจสอบกับข้อมูล Fully Connected Layer ว่ามีความน่าจะเป็นว่าเป็น Object Class ใดกี่เปอร์เซ็นต์



รูป 2.14 การใช้ ROI Pooling เปรียบเทียบ ROI แต่ละส่วนกับ Dataset

2.5.2.2 การอธิบายความหมายของภาพ (Semantic Segmentation)

การอธิบายความหมายของภาพเป็นการนำ Bounding Box ที่หลังจากคำนวณเสร็จแล้วนำมาขยายจาก Mask ขนาด 28*28 Pixel เป็นภาพที่ใหญ่ขึ้นเท่าภาพจริงโดยใช้ Edge Detection ขยาย Mask ให้ใกล้เคียงภาพจริงที่สุด

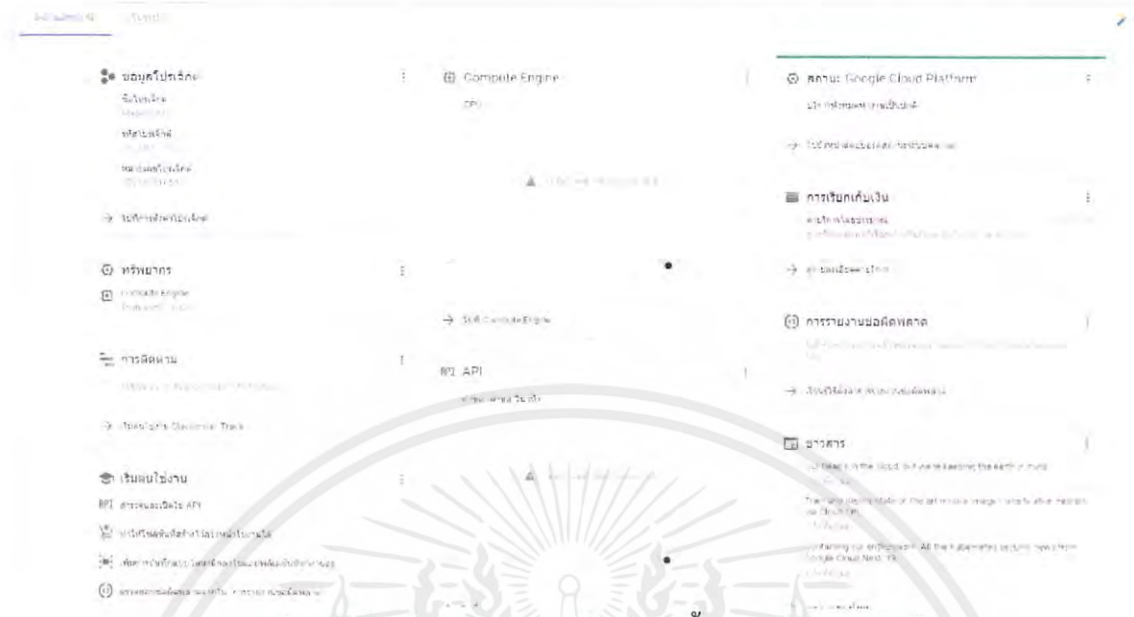


รูป 2.15 การนำ Mask มาขยายบนภาพจริงให้ถึงขอบของวัตถุ

รูป 2.16 กระบวนการของ Mask R-CNN

2.5.3 กูเกิ้ลคลาวด์แพลทฟอร์ม (Google Cloud Platform)

กูเกิ้ลคลาวด์แพลทฟอร์ม เป็นบริการเช่าฮาร์ดแวร์พลังสูงจาก กูเกิ้ล ที่มีบริการมากมาย อาทิ เช่าเซิร์ฟเวอร์กำลังสูงในการรัน API ต่าง ๆ เป็นระบบที่นับค่าใช้จ่ายตามการใช้งานของเซิร์ฟเวอร์ สามารถกำหนดระบบปฏิบัติการความต้องการของฮาร์ดแวร์ความเร็วของระบบเครือข่ายได้ตามที่ต้องการและจ่ายเป็นรายชั่วโมง

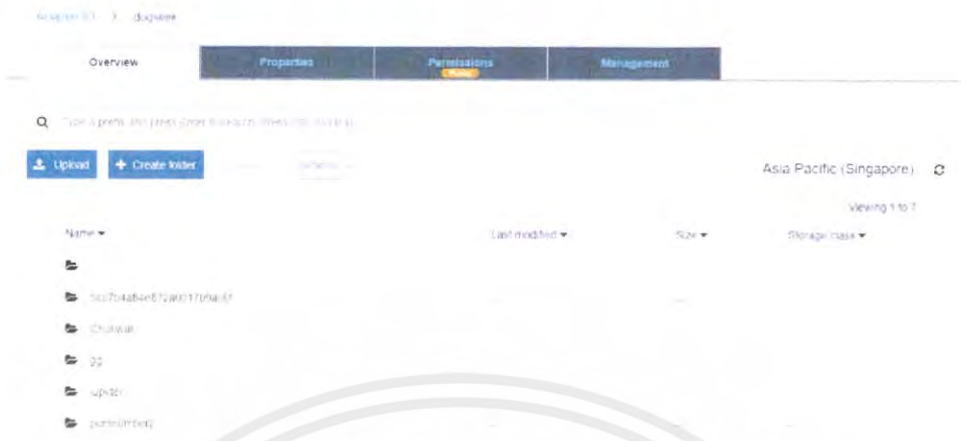


รูป 2.17 ตัวอย่างหน้าแดชบอร์ดของกูเกิ้ลคลาวด์แพลทฟอร์ม

ข้อดีของ กูเกิ้ลคลาวด์แพลทฟอร์ม คือสามารถเข้าถึงบริการอื่น ๆ ได้ง่าย เช่น AWS S3 Bucket , DynamoDB ได้เนื่องจาก กูเกิ้ลคลาวด์แพลทฟอร์ม เป็นระบบที่เป็นมิตรไม่ผูกค่ายและยังคงมีการพัฒนาอยู่ในปัจจุบัน มีสังคมคอมมูนิตี้ในการสอบถามได้มากมาย เช่น Quora , Stack Overflow และอื่น ๆ อีกมากมาย

2.5.4 อเมซอนเว็บเซอร์วิส เอสทรี บั๊กเก็ต (AWS S3 Bucket)

อเมซอนเว็บเซอร์วิส เอสทรี บั๊กเก็ต เป็นบริการเช่าพื้นที่บนระบบคลาวด์ของบริษัทอเมซอน เป็นบริการเก็บข้อมูลแบบเชิงวัตถุ (Object Cloud Storage) สามารถกำหนดสิทธิ์เข้าถึงไฟล์งานในลักษณะบั๊กเก็ตได้ เช่น บั๊กเก็ตนี้สามารถเข้าได้แต่แอดมินเท่านั้น เป็นต้น ทำให้เป็นระบบคลาวด์ที่สามารถแยกระบบเสมือนโดยไม่เกี่ยวข้องกันได้ง่าย และ สามารถใช้งานกับระบบอเมซอนเว็บเซอร์วิสอื่น ๆ ได้เช่น AWS:RDS , DynamoDB สามารถอัพโหลดไฟล์ภาษาคอมพิวเตอร์ได้หลายภาษา อาทิ Python , NodeJS , Java และอื่น ๆ



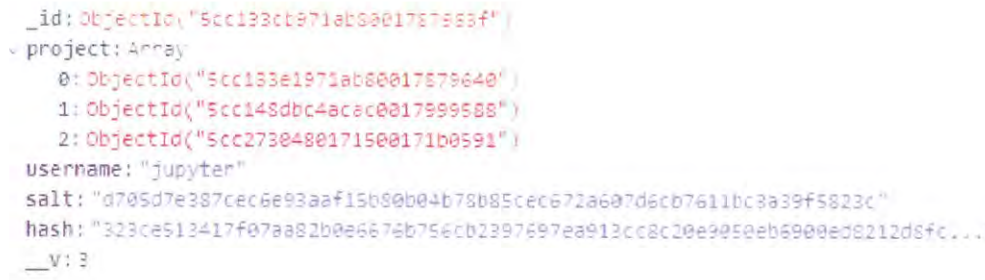
รูป 2.18 หน้าแดชบอร์ดของ AWS S3 Bucket

```
// upload image to S3
app.post("/api/upload", function (req, res) {
  const folder = (req.user.username + "/");
  const file = (req.body.imageUpload);
  const params = {
    Bucket: bucketName,
    Key: (folder + file),
    ACL: 'public-read',
    Body: file
  };
  console.log("Folder name: " + folder);
  console.log("File: " + file);

  s3.putObject(params, function (err, data) {
    if (err) {
      console.log("Error: ", err);
    } else {
      console.log(data);
    }
  });
  res.redirect("/feed");
});
```

รูป 2.19 ตัวอย่างการอัพโหลดไฟล์เข้าสู่ S3 โดยใช้ภาษา JavaScript

2.5.5 MongoDB

MongoDB เป็นระบบจัดการฐานข้อมูลที่ไม่อิงกับภาษา SQL หรือเรียกว่า NOSQL (Not Only SQL) เป็นภาษาที่เก็บข้อมูลในระบบ JSON ไม่อิงกับตารางทำให้มีความยืดหยุ่นสูง ถือเป็นภาษา NOSQL ที่มีความนิยมสูงมากภาษาหนึ่งของโลก



รูป 2.20 ตัวอย่าง Schema ของ MongoDB

จากรูปเป็น Schema ของระบบยืนยันตัวตนซึ่ง MongoDB นั้นจะทำการสร้างข้อมูล _id ไว้ให้อัตโนมัติซึ่ง _id มีไว้เพื่อระบุความสัมพันธ์ของข้อมูลต่าง ๆ ที่ได้เห็นใน project ซึ่งเชื่อมต่อข้อมูลของ ObjectID ไว้สามชุดข้อมูลและ MongoDB ได้สร้างข้อมูล __V ไว้เพื่อเป็นตัวระบุว่า Object นี้มีความสัมพันธ์กับObject อื่น ๆ อีกกี่ Object ซึ่ง MongoDB สร้างไว้ให้อัติโนมัติในส่วนของผู้ใช้งาน name จะเก็บข้อมูลเป็นอักษรแต่จะเก็บพาสเวิร์ดจะเก็บในรูปแบบ hash ที่ผ่าน private key ในชุดข้อมูล salt เพื่อทำการเข้ารหัส

2.5.6 NodeJS

NodeJS เป็นภาษา JavaScript ที่ทำงานบนเซิร์ฟเวอร์ไว้ใช้ในการรับส่งข้อมูลจาก Google Cloud , S3 , MongoDB และอื่น ๆ ข้อดีของ NodeJS คือมี Plugin มากมายให้เลือกใช้และเซิร์ฟเวอร์หลาย ๆ ที่รองรับภาษา NodeJS อยู่แล้วทำให้ NodeJS เป็นที่นิยมอย่างมาก

2.6 เครื่องมือที่ใช้ในการติดต่อกับผู้ใช้

2.6.1 Embedded JavaScript

เป็นภาษาที่ถูกปรับปรุงพัฒนามาจากภาษา HTML ซึ่งเพิ่มความสามารถจาก HTML เดิม โดยสามารถดึงข้อมูลจาก Backend ผ่านภาษา JavaScript ได้ทำให้ไม่ต้องใช้ DOM หรือ jQuery ในการปรับเปลี่ยน Element ของ HTML ซึ่ง Embedded JavaScript เป็น Template ใช้งานง่ายและรองรับ PHP อีกด้วยโดยการใช้งานภาษา JavaScript นั้นจะใช้สัญลักษณ์ <% ตามด้วยโค้ดภาษา JavaScript แล้วปิดท้ายด้วยสัญลักษณ์ %> ดังตัวอย่างในรูป 2.20

```
        <tr>
            <td>First name</td>
            <td><%= fname %></td>
        </tr>

        <tr>
            <td>Last name</td>
            <td><%= lname %></td>
        </tr>

        <tr>
            <td>Username</td>
            <td><%= userName %></td>
        </tr>

        <tr>
            <td>Domain name</td>
            <td><%= domainName %></td>
        </tr>

    </tbody>

</table>
```

รูป 2.21 ตัวอย่างการทำงานภาษา EJS

2.6.2 BOTO3

BOTO3 เป็น Plugin ที่ช่วยในการ Upload ไฟล์ผ่านภาษา Python และ JavaScript สามารถสร้าง Folder ใน S3 ได้โดยใช้ URL เท่านั้นจึงเป็นที่นิยมในการใช้งานอย่างมาก

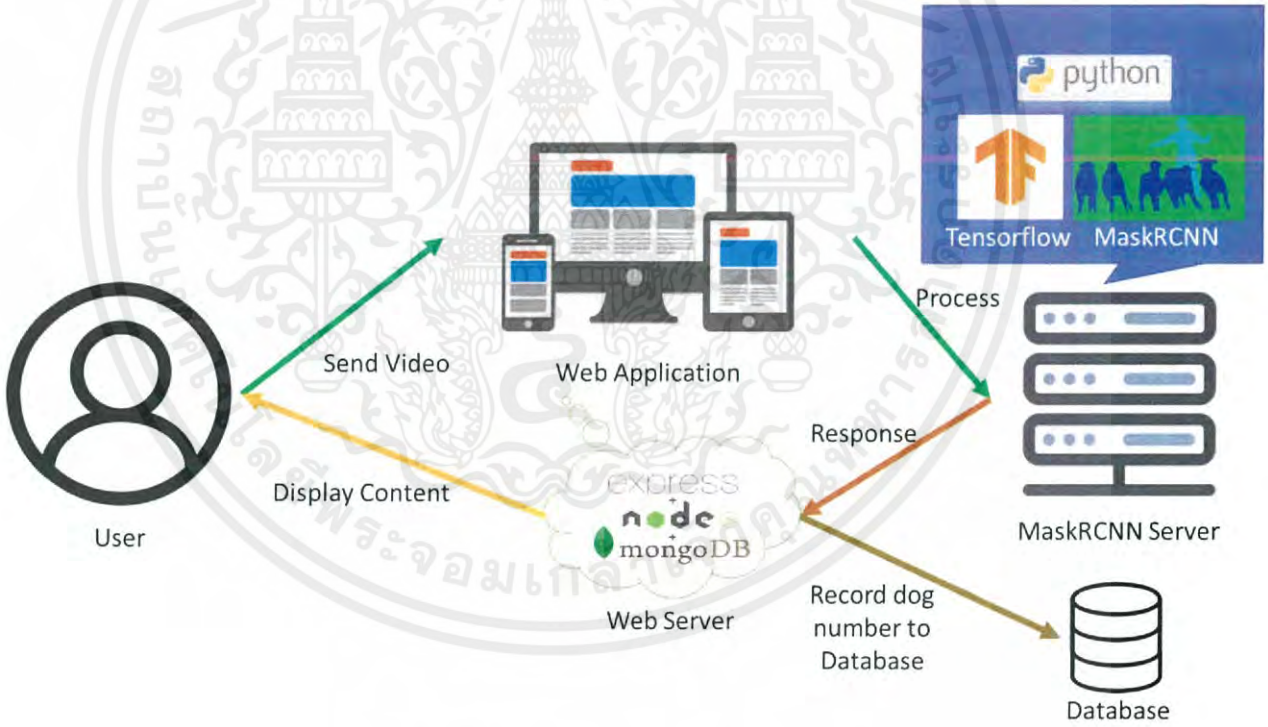
```
from __future__ import print_function
import os, sys, re, uuid, boto3
import json, logging
import PIL.Image
import flask
from flask import request, Response
from logging import FileHandler, WARNING
from PIL import Image


# Create and configure the Flask app
application = flask.Flask(__name__)
file_handler = FileHandler('logfile.log')
file_handler.setLevel(WARNING)
application.logger.addHandler(file_handler)
application.config.from_object('default_config')
application.debug = application.config['FLASK_DEBUG'] in ['true', 'True']

s3_client = boto3.client('s3', region_name='eu-west-1')

@application.route('/worker', methods = ['POST'])

def snap_worker():
  response = None

  if request.json is None:
    response = Response("", status = 415)
  else :
    message = dict()
    try:
      message = request.json
      ori_photo = message['ori_photo']
      bucket_src = message['s3_bucket_ori']
```

รูป 2.22 ตัวอย่างการเขียนโปรแกรมอัพโหลดไฟล์ไปยัง S3 Bucket

บทที่ 3
การออกแบบและพัฒนาซอฟต์แวร์

3.1 ภาพรวมของระบบ

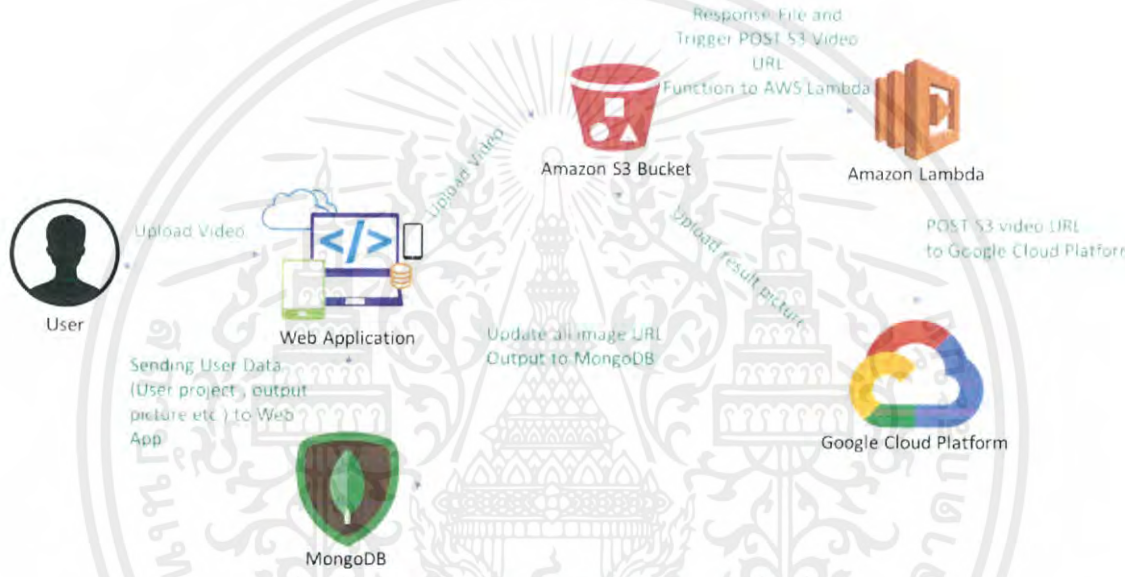
ระบบนับประชากรสุนัขเป็นการนำวิดีโอมาคำนวณจำนวนสุนัขในวิดีโอ โดยการตัดภาพแต่ละเฟรมและส่งภาพให้ Mask R-CNN ประมวลผลโดยเมื่อพบสุนัขแล้วก็จะเก็บกรอบภาพไว้แล้วทำการติดตามการเคลื่อนไหวของวัตถุโดยหากวัตถุยังอยู่ในกรอบเดิมก็จะอนุมานได้ว่าสุนัขยังคงเป็นตัวเดิมและนำตัวเลขจำนวนสุนัขที่ได้ทั้งหมดแสดงผลผ่านเว็บแอพพลิเคชั่นและเพื่อความแม่นยำสูงสุดผู้ใช้สามารถดูผลลัพธ์ภาพทั้งหมดที่ Mask R-CNN ประมวลผลแล้วได้และยังสามารถเลือกลบภาพที่ Mask R-CNN ทำงานผิดพลาดหรือสุนัขที่ประมวลผลได้ซ้ำได้อีกด้วย



รูป 3.1 ภาพรวมของระบบ

ภาพรวมของระบบจะใช้เว็บแอพพลิเคชั่นในการติดต่อสื่อสารกับผู้ใช้งานที่รันอยู่บนเว็บเซิร์ฟเวอร์ด้วยภาษา NodeJS ที่มี MongoDB คอยจัดการในเรื่องการเก็บไฟล์รูปภาพข้อมูลผู้ใช้งานและ

อื่น ๆ และเมื่อผู้ใช้งานทำการอัพโหลดวิดีโอระบบจะทำการส่งต่อไฟล์เข้าสู่ S3 เพื่อทำการเก็บไฟล์วิดีโอจากนั้น S3 จะไป Trigger ระบบ AWS Lambda ให้ทำการ POST HTTP Code ที่มี URL S3 ข้ามระบบไปหากูเกิ้ลคลาวด์แพลทฟอร์มทำการประมวลผล Mask R-CNN และเมื่อเซิร์ฟเวอร์ที่ทำงานบนกูเกิ้ลคลาวด์แพลทฟอร์มทำงานเสร็จแล้วก็จะทำการอัพโหลดไฟล์รูปภาพผลลัพธ์ทั้งหมดกลับไปที่ S3 อีกครั้งโดยใช้ Key ที่ได้จาก ผู้ใช้งานทำการอัพโหลดวิดีโอมาและนำ URL รูปผลลัพธ์ทั้งหมดไปอัพเดทฐานข้อมูล MongoDB เพื่อแสดงผลภาพและแสดงผลจำนวนสุนัขในวิดีโอ



รูป 3.2 ขั้นตอนการทำงานของระบบ

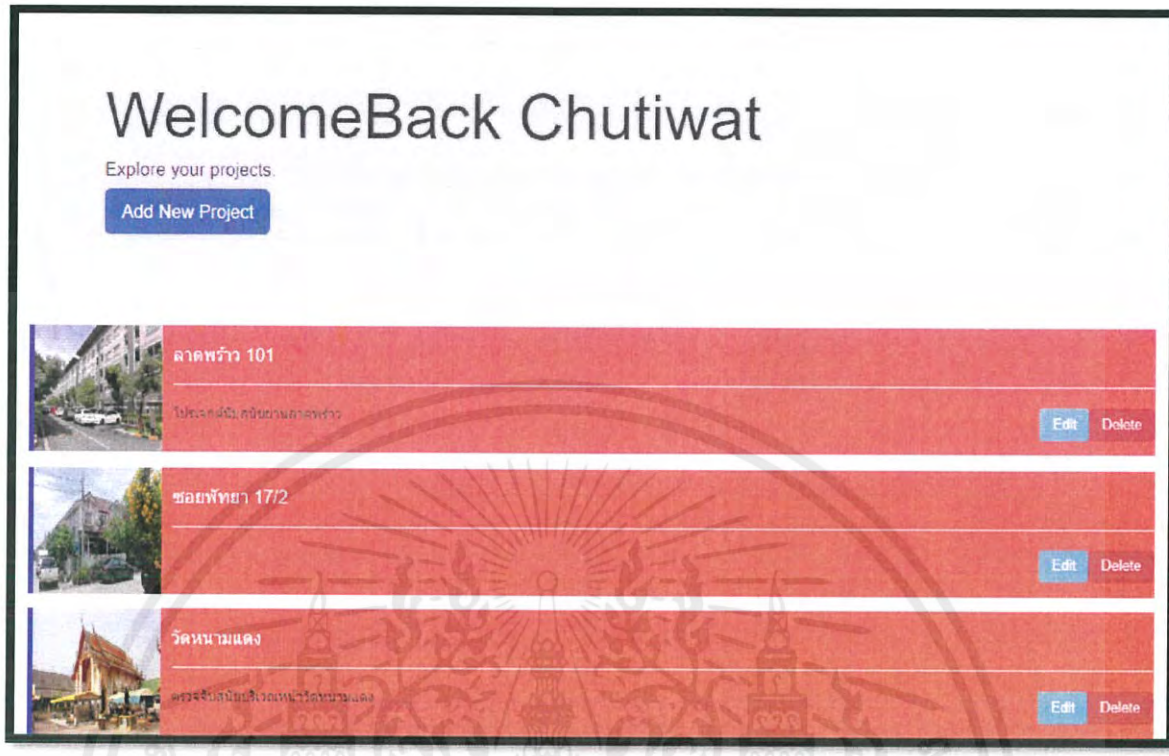
รูป 3.3 Use Case Diagram การทำงานของระบบ

ซึ่งการทำงานในส่วนของเว็บแอพพลิเคชั่นที่เป็นส่วนการทำงานติดต่อกับผู้ใช้งานนั้นจะมีฟังก์ชันหลัก ๆ ดังต่อไปนี้

3.1.1 เว็บแอพพลิเคชั่นสำหรับติดต่อผู้ใช้งาน

โดยผู้ใช้บริการจะส่งคำร้องผ่านทางเว็บโดยผ่าน PC หรือ Mobile เพื่อเตรียมอัพโหลดวิดีโอไปประมวลผลระบบนับสุนัขต่อไป

รูป 3.4 หน้าแรกของเว็บแอพพลิเคชั่น



รูป 3.5 หน้าต่างแสดงโปรเจกต์ที่ผู้ใช้งานได้สร้าง

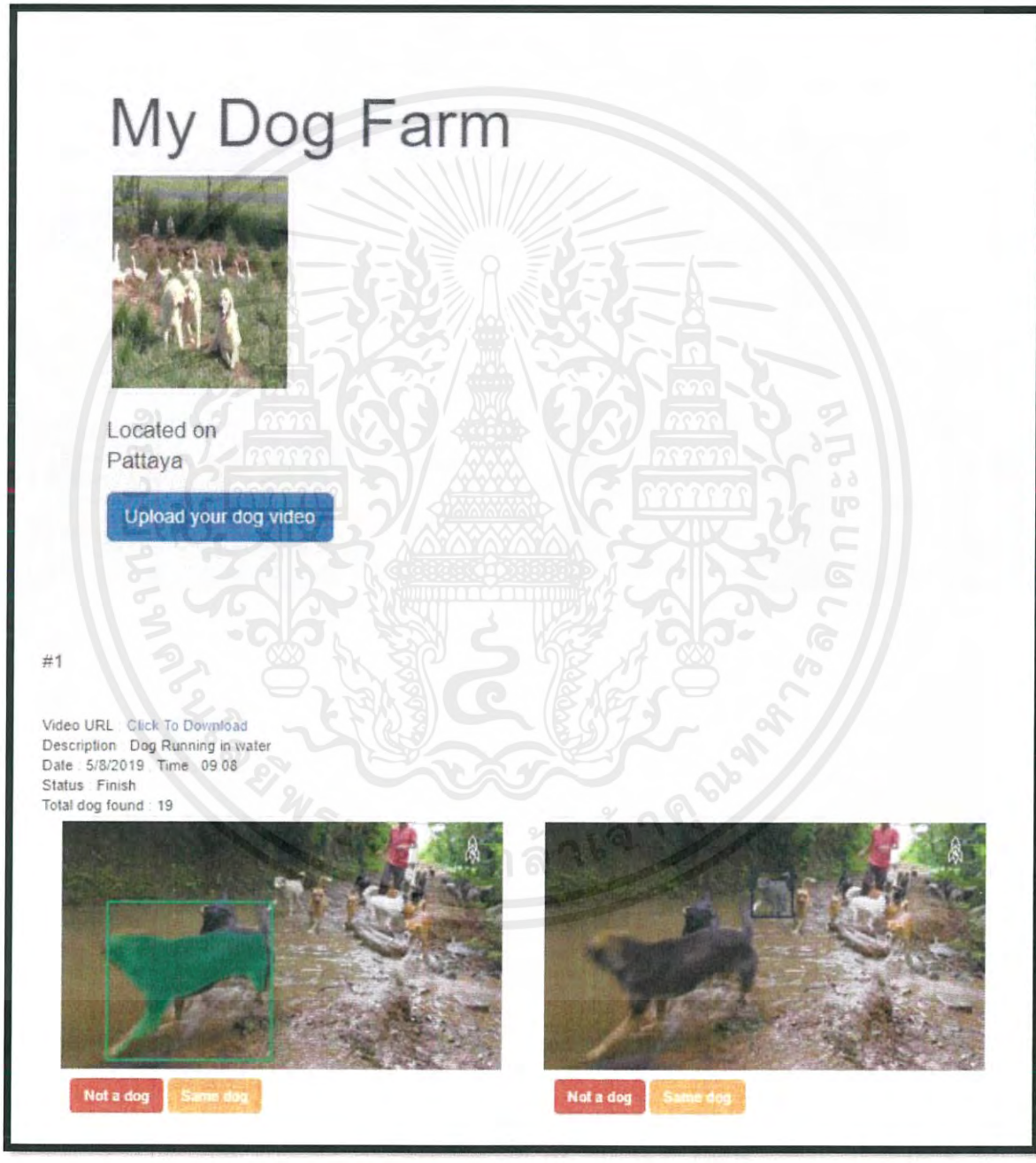
รูป 3.6 แบบฟอร์มการสร้างโปรเจกต์ใหม่

จากแบบฟอร์มที่ใช้อัพโหลดวิดีโอผู้ใช้ต้องกรอกข้อมูลดังนี้

projectname (ชื่อโปรเจกต์)

Image (รูปภาพที่นำมาเป็นปกของโปรเจกต์)

Description (รายละเอียดเพิ่มเติมของโปรเจกต์)



รูป 3.7 หน้าต่าง Dashboard ของผู้ใช้งาน

จากรูป My Dog Farm จะเป็นชื่อโปรเจกต์ซึ่งจะสามารถอัพโหลดวิดีโอได้ผ่านปุ่ม Upload your dog video และส่วนของผลลัพธ์การประมวลผลจะอยู่ด้านล่างเรียงไปตามช่วงเวลาที่อัพโหลดแต่ละวิดีโอ และแสดงรูปภาพผลลัพธ์ที่ Mask R-CNN ประมวลผลได้ซึ่งจะมีปุ่ม Not a dog และ Same dog ไว้ให้ผู้ใช้งานคอยตรวจสอบกรณีที่ Mask R-CNN ทำงานไม่แม่นยำ

รูป 3.8 ตัวอย่างแบบฟอร์มการอัพโหลดวิดีโอเข้าไปใน Mask R-CNN

จากแบบฟอร์มที่ใช้อัพโหลดวิดีโอผู้ใช้ต้องกรอกข้อมูลดังนี้

Video	(ให้ผู้ใช้งานเลือกไฟล์วิดีโอจากเครื่อง)
Description	(รายละเอียดของวิดีโอ)
Date	(วันที่ถ่ายวิดีโอนี้โดยประมาณ)
Estimated Time Taken	(เวลาที่ถ่ายวิดีโอนี้โดยประมาณ)

รูป 3.9 ตัวอย่างการแสดงผลลัพธ์ใน Mask R-CNN

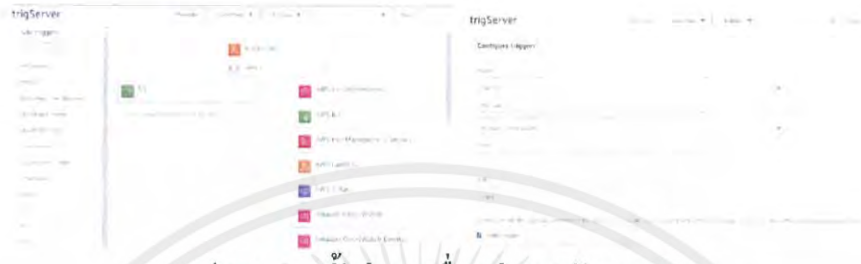
3.1.2 Amazon S3 Bucket Storage

เนื่องจาก AWS Lambda ไม่สามารถเก็บไฟล์ไว้ใน Lambda ได้ดังนั้น S3 Bucket จึงเป็นที่เก็บไฟล์ให้ AWS Lambda ประมวลผลต่อไป

รูป 3.10 ตัวอย่างไฟล์ที่ถูกอัพโหลดบน S3 Bucket

3.1.3 AWS Lambda Environment

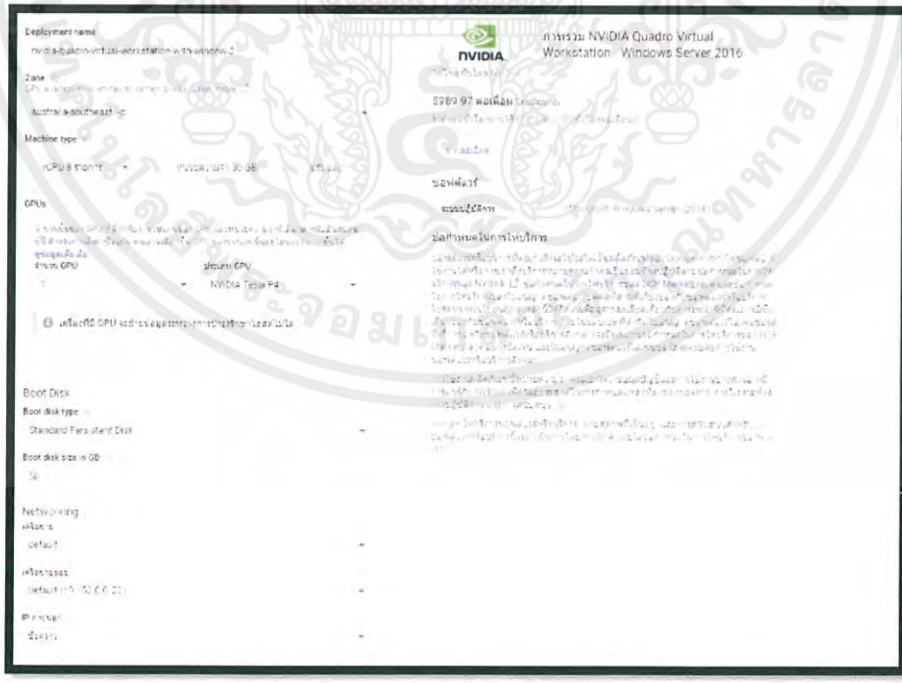
ใช้ AWS Lambda ส่ง Request ไปที่ Google Cloud เมื่อมีการ Trigger จาก S3 ตามที่ตั้งค่าไว้ในรูป 3.11 เพื่อประมวลผล Mask R-CNN ต่อไป



รูป 3.11 การตั้งค่าการเชื่อมต่อ S3 กับ Lambda

3.1.4 Mask R-CNN เซิร์ฟเวอร์

เป็นเซิร์ฟเวอร์ที่ทำงานบนกูเกิ้ลคลาวด์แพลทฟอร์มใช้ในการประมวลผล Mask R-CNN คอยรับวิดีโอจากผู้ใช้งานและทำการประมวลผลเพื่อค้นหาสุนัขในวิดีโอและนับจำนวนผ่านการติดตามการเคลื่อนไหวของสุนัขในวิดีโอนั้น ๆ หลังจากที่ประมวลผลเสร็จเซิร์ฟเวอร์จะอัพเดทผลลัพธ์การประมวลผลและรูปภาพที่ตรวจพบเข้าสู่ MongoDB



รูป 3.12 การกำหนดค่าฮาร์ดแวร์ที่ใช้งานกับกูเกิ้ลคลาวด์แพลทฟอร์ม

กลุ่มของข้าพเจ้าได้ใช้ฮาร์ดแวร์บนกูเกิ้ลคลาวด์แพลทฟอร์มเป็น CPU จำนวน 8 คอร์ หน่วยความจำ 30 กิกะไบท์ หน่วยประมวลผลกราฟฟิค NVIDIA Tesla P4 ซึ่งมีหน่วยความจำ 14 กิกะไบท์ทำงานอยู่บนระบบปฏิบัติการ Linux Ubuntu 18.04

3.2 การออกแบบส่วนการติดต่อกับผู้ใช้

ใช้สถาปัตยกรรม MVC (Model-View-Controller) ซึ่งเป็นรูปแบบที่เอาไว้ติดต่อกับผู้ใช้งานแบ่งออกได้เป็น 3 ส่วนได้แก่

3.2.1 Model

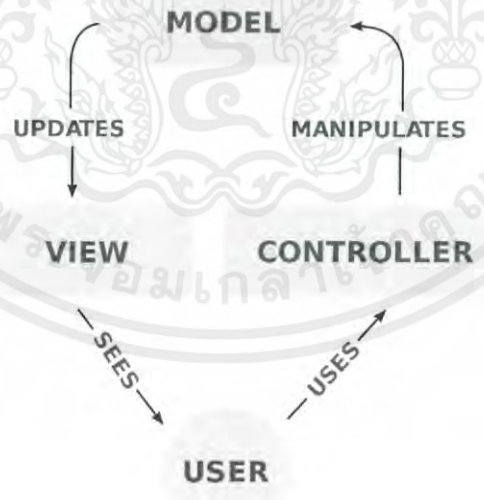
เป็นส่วนที่รับคำสั่งจาก Controller และเมื่อประมวลผลเสร็จก็สั่งการให้ Update View เพื่อแสดงผลส่งต่อไปให้ ผู้ใช้งาน

3.2.1 View

เป็นส่วนที่รับคำสั่งจาก Model ตอบสนองหลังจากผู้ใช้งานใช้งานผ่านการรับและแสดงผลทางหน้าจอ

3.2.1 Controller

เป็นส่วนที่รับคำสั่งโดยตรงจากผู้ใช้งานแล้วส่งให้ Model ประมวลผล



รูป 3.13 ไดอะแกรมการทำงานของสถาปัตยกรรม MVC

3.3 การออกแบบในส่วนการติดต่อกับเซิร์ฟเวอร์

ใช้ Mask RCNN ทำงานบนกูเกิ้ลคลาวด์แพลทฟอร์มเพื่อตัดวิดีโอเป็นภาพหลาย ๆ ภาพ จากนั้นนำไปจำแนกวัตถุเพื่อคัดกรองนำมาเพียงสุนัข จากนั้นทำการเก็บค่าตำแหน่งของวัตถุที่สนใจที่ประมวลผลได้และแสดงผลออกทางเว็บแอพพลิเคชั่น

รูป 3.14 ขั้นตอนการทำงานของระบบ Backend

3.3.1 การออกแบบฐานข้อมูล MongoDB

ฐานข้อมูลของ MongoDB ในโปรเจกต์นี้มีอยู่ 3 Schema หลัก ๆ คือ Schema ของผู้ใช้งาน , Schema ของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้น และ Schema ของไฟล์วีดีโอแต่ละวิดีโอที่ได้ทำการอัพโหลดเข้ามาซึ่งแต่ละ Schema จะเก็บข้อมูลดังนี้

3.3.1.1 การเก็บข้อมูล Schema ของผู้ใช้งาน

การเก็บข้อมูลของผู้ใช้งานจะเก็บข้อมูลดังนี้

1) _id เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติไว้ใช้อ้างอิงความสัมพันธ์ของแต่ละ Object
2) username เก็บชื่อผู้ใช้งาน
3) salt เป็นข้อมูล private key ของระบบยืนยันตัวตนที่ระบบสร้างขึ้นอัตโนมัติ
4) hash เป็นพาสเวิร์ดที่ได้จากผู้ใช้งานเป็นคนป้อน แต่ถูกเข้ารหัสด้วย master key ที่ถูกเก็บไว้ที่เซิร์ฟเวอร์เป็นความลับ
5) __v เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติเพื่อใช้ในการบ่งบอกว่า object นี้เชื่อมอยู่กับ object อื่น ๆ เป็นจำนวนเท่าใด
6) project เป็น Array ที่ใช้ในการเก็บเลข Object ID ของ Schema ของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้น

รูป 3.15 ตัวอย่าง MongoDB Schema ของผู้ใช้งาน

3.3.1.2 การเก็บข้อมูล Schema ของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้น

การเก็บข้อมูลของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้นจะเก็บข้อมูลดังนี้

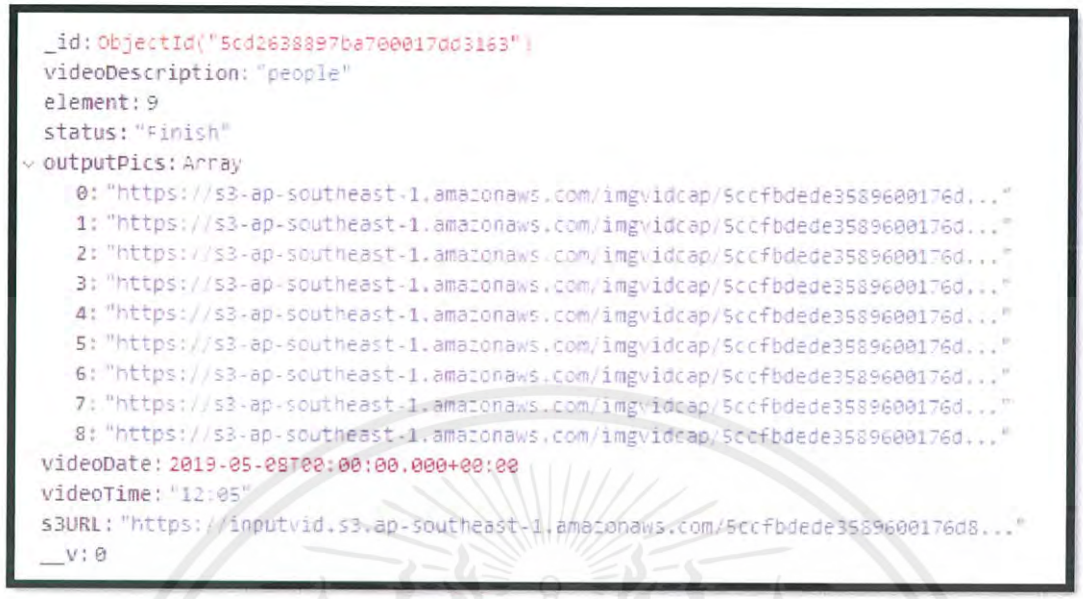
1) _id เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติไว้ใช้อ้างอิงความสัมพันธ์ของแต่ละ Object
2) name เก็บชื่อโปรเจกต์
3) description เก็บคำอธิบายที่ผู้ใช้งานป้อนเข้ามา
4) date เป็นวันที่ที่ระบบบันทึกเวลาให้อัตโนมัติว่าทำการเพิ่มข้อมูลมาในช่วงเวลาใด
5) __v เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติเพื่อใช้ในการบ่งบอกว่า object นี้เชื่อมอยู่กับ object อื่น ๆ เป็นจำนวนเท่าใด
6) picture เป็น Array ที่ใช้ในการเก็บ URL ที่ผู้ใช้งานอัพโหลดรูปภาพหน้าปกขึ้นมา
7) vidURL เป็น Array ที่ใช้ในการเก็บเลข Object ID ของ Schema ของไฟล์วิดีโอแต่ละวิดีโอที่ได้ทำการอัพโหลดเข้ามา

รูป 3.16 ตัวอย่าง MongoDB Schema ของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้น

3.3.1.3 การเก็บข้อมูล Schema ของไฟล์วีดีโอแต่ละวิดีโอที่ได้ทำการอัพโหลดเข้ามา

การเก็บข้อมูลของโปรเจกต์ที่ผู้ใช้งานสร้างขึ้นจะเก็บข้อมูลดังนี้

1) _id เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติไว้ใช้อ้างอิงความสัมพันธ์ของแต่ละ Object
2) videoDescription เก็บคำอธิบายของวิดีโอที่ User ป้อนเข้ามา
3) element เก็บเลขจำนวนของสุนัขที่ Mask R-CNN เซิร์ฟเวอร์ประมวลผลได้
4) status เป็นข้อความที่บอกกับผู้ใช้งานว่า Mask R-CNN เซิร์ฟเวอร์ประมวลผลเสร็จสิ้นแล้วหรือไม่
5) videoDate เก็บวันที่ผู้ใช้งานถ่ายวิดีโอนี้จากการป้อนเข้ามาของผู้ใช้งาน
6) videoTime เก็บเวลาที่ผู้ใช้งานถ่ายวิดีโอนี้จากการป้อนเข้ามาของผู้ใช้งาน
7) s3URL เก็บ URL ของวิดีโอหลังจากที่เว็บเซิร์ฟเวอร์อัพโหลดไฟล์วิดีโอไปยัง AWS S3 เสร็จแล้ว
8) __v เป็นเลขที่ระบบสร้างขึ้นอัตโนมัติเพื่อใช้ในการบ่งบอกว่า object นี้เชื่อมอยู่กับ object อื่น ๆ เป็นจำนวนเท่าใด
9) outputPics เป็น Array ที่ใช้ในการเก็บ URL รูปภาพที่ Mask R-CNN เซิร์ฟเวอร์ประมวลผลและอัพโหลดไปยัง S3 เสร็จแล้วเพื่อเก็บไว้แสดงผล

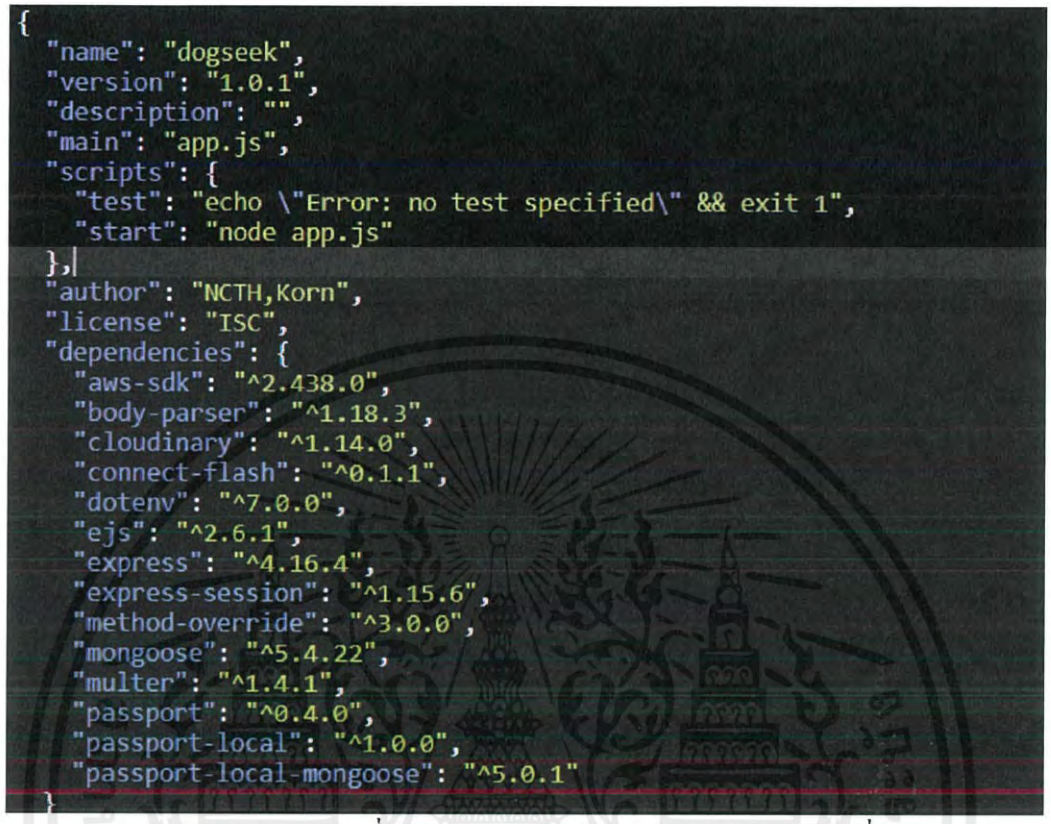


รูป 3.17 Schema ของไฟล์วีดีโอแต่ละวิดีโอที่ได้ทำการอัพโหลดเข้ามา

3.3.2 การกำหนดค่าต่าง ๆ บนเว็บแอพพลิเคชั่นเซิร์ฟเวอร์

เว็บแอพพลิเคชั่นเซิร์ฟเวอร์นั้นจะใช้ภาษา NodeJS ในการทำงานอยู่บนระบบปฏิบัติการ Linux Ubuntu 18.04 ซึ่งรองรับการทำงานของภาษา NodeJS ไว้อยู่แล้ว และยังรองรับ npm ซึ่งเป็นโปรแกรมในการจัดการส่วนเสริมของภาษา NodeJS ก็ได้ติดตั้งไว้อยู่แล้วเช่นกัน ซึ่งเราจะใช้ส่วนเสริมในการทำงานเว็บแอพพลิเคชั่นดังต่อไปนี้

1) aws-sdk ใช้ในการติดต่อกับ AWS S3
2) dotenv ใช้ในการเก็บซ่อนตัวแปรต่าง ๆ ในเซิร์ฟเวอร์เพื่อความปลอดภัย
3) express ใช้ในการจัดการกับ Routing หน้าเว็บต่าง ๆ บนเว็บไซต์
4) mongoose ใช้ในการติดต่อกับฐานข้อมูล MongoDB
5) Passport-local-mongoose , express-session , multer ใช้ในการจัดการระบบยืนยันตัวตน (User Authentication)
6) ejs ใช้ในการแปลงโค้ดภาษา ejs เป็น HTML5



```
{
  "name": "dogseek",
  "version": "1.0.1",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node app.js"
  },
  "author": "NCTH,Korn",
  "license": "ISC",
  "dependencies": {
    "aws-sdk": "^2.438.0",
    "body-parser": "^1.18.3",
    "cloudinary": "^1.14.0",
    "connect-flash": "^0.1.1",
    "dotenv": "^7.0.0",
    "ejs": "^2.6.1",
    "express": "^4.16.4",
    "express-session": "^1.15.6",
    "method-override": "^3.0.0",
    "mongoose": "^5.4.22",
    "multer": "^1.4.1",
    "passport": "^0.4.0",
    "passport-local": "^1.0.0",
    "passport-local-mongoose": "^5.0.1"
}
```

รูป 3.18 ส่วนเสริมที่ใช้ใน NodeJS เซิร์ฟเวอร์บนเว็บแอพพลิเคชั่น

จากรูปภาพที่ 3.17 สังเกตได้ว่าส่วนเสริมที่ทำการติดตั้งนั้นมีมากกว่าที่ได้ระบุไว้ซึ่งส่วนเสริมที่เพิ่มขึ้นมานั้นเป็นการเรียกใช้เพิ่มเติมของส่วนเสริมอื่น ๆ ที่ได้ติดตั้งเพิ่มเข้ามา อาทิ ส่วนเสริม express ได้ทำการติดตั้ง method-override , connect-flash เข้ามาเพิ่มด้วยตัวเอง เป็นต้น

3.4 ขั้นตอนในการทำงานของระบบ

ขั้นตอนการทำงานของโปรแกรมนั้นได้แบ่งออกเป็น 8 ขั้นตอนดังต่อไปนี้

1) Login เข้าสู่เว็บแอพพลิเคชั่น
2) สร้าง Project ใหม่
3) เลือก Project ที่ต้องการทำงาน
4) อัพโหลดวิดีโอลงบนเว็บแอพพลิเคชั่น
5) เว็บแอพพลิเคชั่นส่งไฟล์ไปที่ S3
6) S3 ส่งสัญญาณให้ Lambda ทำงาน
7) Lambda ส่ง Post Requests เรียกใช้งาน Mask R-CNN บนกูเกิ้ลคลาวด์แพลทฟอร์ม
8) กูเกิ้ลคลาวด์แพลทฟอร์มส่งผลลัพธ์ให้เว็บแอพพลิเคชั่นแสดงผล

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลองการใช้งาน Mask RCNN

การทดลองมีวัตถุประสงค์เพื่อทดสอบความสามารถในการทำงานของการทำงานของ Mask R-CNN ที่ทำงานโดยใช้โครงประสาทเทียมแบบคอนโวลูชันเพื่อทดสอบความแม่นยำในการทำงานของ Mask R-CNN ว่ามีความสามารถในการ Detect วัตถุที่สนใจได้มากน้อยเพียงใด

ในการทดลองเราจะใช้ Dataset ที่ได้จาก COCO Dataset (Common Object in Context) เป็นองค์กรที่ได้รับการสนับสนุนในการทำ Dataset จากหลายบริษัท อาทิเช่น Facebook , Microsoft , CVDF Corporation ซึ่งมี Class 80 Class และมีภาพที่ถูกเทรนมากกว่า 300,000 ภาพ

4.1.1 รายละเอียดการทดลอง

1) ใช้ Dataset ของ COCO Dataset ในการใช้งาน Mask R-CNN เบื้องต้นกับภาพสุนัข 100 ภาพ
2) ทดลองกับภาพสุนัขหลักๆ สามรูปแบบ คือ หน้าตรงระยะใกล้ , ระยะกลาง , ระยะไกลและท่าที่ผิดวิสัยสุนัข

4.1.2 วิธีการทดลอง

1) นำภาพตัวอย่างการทดลองเข้าไปทดสอบใน Mask R-CNN แล้ว Display ออกมาเป็นลักษณะ Mask ว่าครอบคลุมแค่ไหน
2) รูปแบบภาพที่แสดงออกมาจะเป็นภาพเอาต์พุตกรอบภาพที่บ่งบอกว่าเป็น Object ใด และ ค่าความมั่นใจ Confident ว่าเป็นกี่เปอร์เซ็นต์โดยแสดงออกมาเป็นค่า ระหว่าง 0.00 – 1.00
3) ทดลองการนำภาพไปประมวลผลว่าใช้เวลาในการรัน Mask R-CNN ต่อภาพ 1 เฟรม ว่าใช้เวลาเท่าใด

4.1.3 ผลการทดลอง

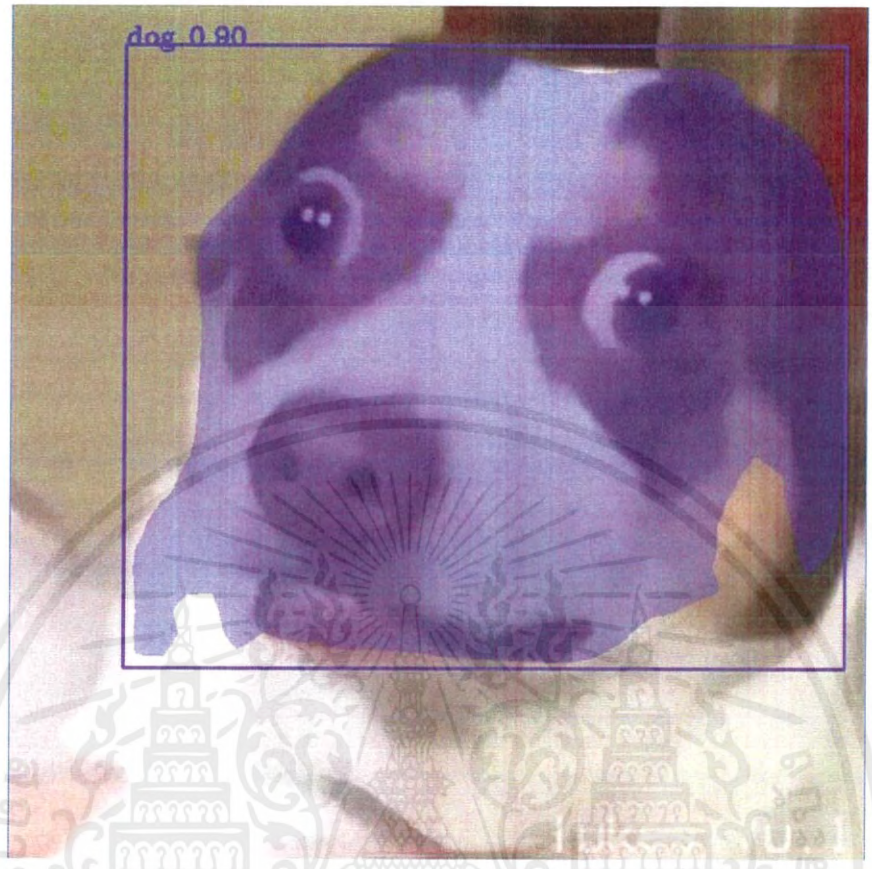
ตาราง 4.1 ตารางเปรียบเทียบจำนวนภาพที่ Mask R-CNN ทำนายได้เทียบกับความมั่นใจกับภาพสุนัข 100 ภาพ

ค่าความมั่นใจ	จำนวนรูปภาพ
>90%	81
80 – 89%	10
70 – 79%	4
60 – 69%	2
ไม่สามารถตรวจจับสุนัขได้	2

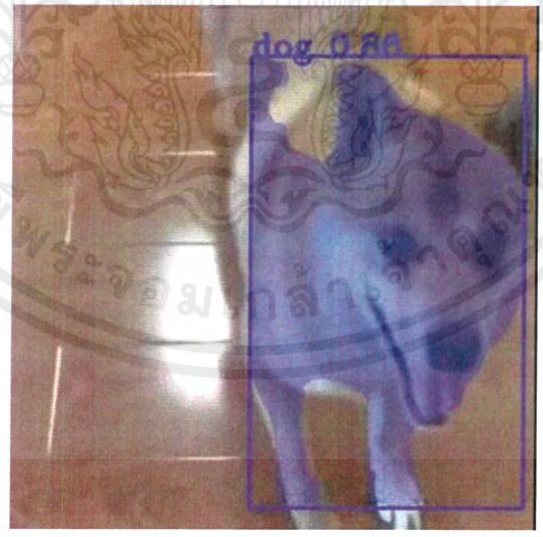
ตาราง 4.2 ตารางเปรียบเทียบเวลาในการประมวลผลโดยเฉลี่ยของ Mask R-CNN 100 ภาพ เทียบกับ Faster R-CNN 100 ภาพ

เทคโนโลยี R-CNN	เวลาในการประมวลผล
Mask R-CNN	107 วินาที
Faster R-CNN	65 วินาที

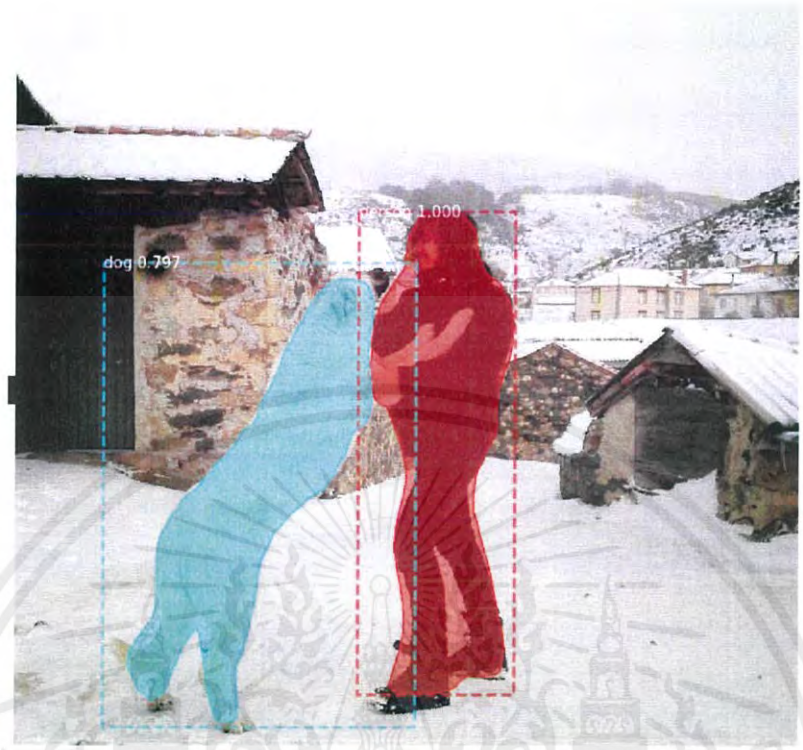
จากการทดลองการทดสอบ Mask R-CNN กับภาพสุนัขทั้งหมด 100 ภาพ นั้นจะพบว่า Mask R-CNN นั้นสามารถถูกตรวจจับสุนัขได้จากใบหน้าของสุนัขโดยมีความมั่นใจสูงระดับ 90% ขึ้นไปถ้าอยู่ใกล้และเห็นหน้าสุนัข แต่สุนัขในอิริยาบถแปลก ที่ผิดวิสัยสุนัขนั้นปรากฏว่า Mask R-CNN นั้นค่าความมั่นใจของ Computer ที่ออกมานั้นจะลดลงอย่างมีนัยยะสำคัญ ส่วนในด้านระยะเวลานั้น Mask R-CNN ใช้เวลาในการประมวลผลค่อนข้างมากเทียบกับ Object Detection จำพวก Faster R-CNN หรือ YOLO3 ซึ่งกลุ่มของข้าพเจ้าได้ใช้คอมพิวเตอร์ในการทดสอบโดยมีหน่วยประมวลผลกลาง Intel Core I7 3770 ความเร็ว 3.4 กิกะเฮิรตซ์ หน่วยความจำ 8 กิกะไบต์ และหน่วยประมวลผลกราฟฟิก ASUS Nvidia GTX 1050 TI ซึ่งมีหน่วยความจำกราฟฟิก 4 กิกะไบต์ ซึ่งผลปรากฏว่า 1 ภาพ ใช้เวลาเฉลี่ยในการประมวลผลประมาณ 1 วินาที ต่อภาพหมายความว่า หากวิดีโอที่ผู้ใช้งานอัพโหลดมีขนาด 1 วินาที จะใช้เวลาประมาณ 30 – 35 วินาทีในการประมวลผล Mask R-CNN ทุกเฟรม



รูป 4.1 ผลทดสอบ Mask R-CNN กับสุนัขระยะใกล้



รูป 4.2 ผลทดสอบ Mask R-CNN กับสุนัขระยะกลาง



รูป 4.3 ผลทดสอบ Mask R-CNN กับสุนัขระยะไกล

```
Frame 6 times( 0:00:01.100029 sec) ==> From 1 : dog = 1
Frame 7 times( 0:00:01.093772 sec) ==> From 1 : dog = 0
Frame 8 times( 0:00:01.116501 sec) ==> From 1 : dog = 1
Frame 9 times( 0:00:01.109912 sec) ==> From 1 : dog = 1
Frame 10 times( 0:00:01.100209 sec) ==> From 1 : dog = 1
Frame 11 times( 0:00:01.090036 sec) ==> From 1 : dog = 1
Frame 12 times( 0:00:01.119623 sec) ==> From 2 : dog = 1
Frame 13 times( 0:00:01.090492 sec) ==> From 1 : dog = 1
Frame 14 times( 0:00:01.128931 sec) ==> From 1 : dog = 1
Frame 15 times( 0:00:01.110521 sec) ==> From 2 : dog = 1
Frame 16 times( 0:00:01.100014 sec) ==> From 1 : dog = 1
Frame 17 times( 0:00:01.109930 sec) ==> From 3 : dog = 1
Frame 18 times( 0:00:01.110194 sec) ==> From 3 : dog = 1
Frame 19 times( 0:00:01.103878 sec) ==> From 2 : dog = 1
```

รูป 4.4 เวลาในการประมวลผล Mask R-CNN ของแต่ละรูปภาพ

เวลาในการประมวลผลนี้ อิงจากวิดีโอขนาด 1280*720 Pixel สกุลไฟล์ MP4 ที่ผ่านการ Preprocessing Bit Depth และ Bounding MSAA เพื่อลดขนาดความคมของเส้นเพื่อลดขนาดไฟล์ผ่าน CV2

4.1.4 สรุปผลการทดลอง

การทดลองนี้ได้แสดงให้เห็นว่า Mask R-CNN นั้นใช้เวลาในการประมวลผลค่อนข้างมากแต่มีความยืดหยุ่นเนื่องจาก Mask R-CNN สามารถใช้ Dataset ของภายนอกหรือสามารถกำหนด Dataset ของตัวเองขึ้นมาได้ ในส่วนของการใช้งานของ COCO Dataset นั้นบ่งบอกได้ว่า COCO Dataset นั้นจะจับใบหน้าของสุนัขเป็นหลักจึงจะมีความแม่นยำสูงขึ้นอย่างมีนัยยะสำคัญ หากแต่ภาพสุนัขในท่าทางที่แปลกออกไป COCO Dataset จะสามารถรองรับได้น้อยลงจากรูปภาพที่ 3 ที่มีค่าความมั่นใจ (Confident) เพียงแค่ 70 – 80 % เท่านั้น

4.2 การทดลองการสร้างโมเดลข้อมูลสุนัขของตัวเอง

การทดลองนี้มีวัตถุประสงค์เพื่อนำ Mask R-CNN มาใช้งานให้สามารถตรวจจับสุนัขได้หลายมุมหลายท่าทางเนื่องจาก Mask R-CNN นั้นเปิดโอกาสให้นักพัฒนาสามารถสร้าง ชุดข้อมูลโมเดลขึ้นมาเองได้ เพื่อนำมาเพิ่มความแม่นยำในคลาสที่ต้องการโดยการทดลองนี้ได้นำภาพสุนัขในหลาย ๆ ท่าทางมาสร้าง เส้นรูปร่างในการบ่งบอกบริเวณวัตถุที่สนใจในลักษณะไฟล์ JSON แล้วทำไปเทรน Machine Learning บน TensorFlow เพื่อทำการคัดเลือกชุด Dataset ที่ดีที่สุดแล้วนำมา Evaluate ว่าค่า Training Accuracy และ Testing Accuracy มีค่าเท่าใดเพื่อนำไปเปรียบเทียบกับค่า Precision และ Recall ต่อไปเพื่อไปทำบทสรุปว่า Mask R-CNN ที่เราสร้างชุดข้อมูลโมเดลที่เราได้สร้างขึ้นนั้นมีข้อดีข้อเสียอย่างไร

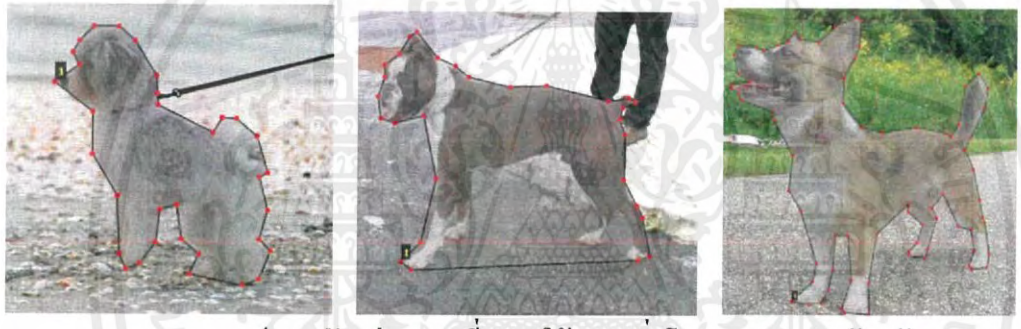
4.2.1 รายละเอียดในการทดลอง

1) ภาพที่ใช้นั้นแบ่งออกเป็น 4 ชุด คือ ภาพด้านหน้า , ด้านข้าง , ด้านหลัง , และ สุนัขท่าทางผิดวิสัยสุนัขอย่างละ 200 ภาพ เป็นภาพเอาไว้เทรน และภาพเอาไว้ทดสอบ 40 ภาพ เป็นจำนวน 1 ชุด
2) ภาพเป็นไฟล์นามสกุล JPEG ไม่กำหนดขนาด
3) ภาพทั้ง 840 ภาพต้องทำ JSON Annotator เพื่อทำการบ่งบอกบริเวณที่สนใจให้ Machine Learning เป็นภาพลักษณะ Polygon Vector

4.2.2 ขั้นตอนในการทดลอง

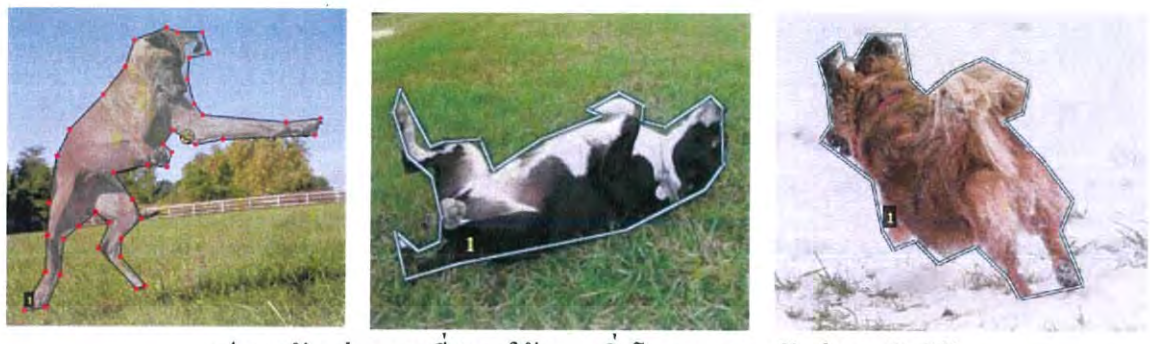
1) ในการทดลองได้นำภาพทั้ง 840 ภาพมาทำ JSON Annotator เป็น Polygon Vector แล้ว Import ออกมาเป็นไฟล์ via_region.json

รูป 4.5 ตัวอย่างภาพที่นำมาใช้เทรนนิ่งโมเดลภาพสุนัขหน้าตรง



รูป 4.6 ตัวอย่างภาพที่นำมาใช้เทรนนิ่งโมเดลภาพสุนัขด้านข้าง

รูป 4.7 ตัวอย่างภาพที่นำมาใช้เทรนนิ่งโมเดลภาพสุนัขด้านหลัง



รูป 4.8 ตัวอย่างภาพที่นำมาใช้เทรนนิ่งโมเดลภาพสุนัขท่าทางผิดวิสัย

{"16.jpg171203":{"fileref":"","size":171203,"filename":"16.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":{"name":"polygon","all_points_x":
[61,98,172,213,242,223,243,258,239,284,285,267,267,267,284,348,402,489,503,497,460,466,451,482,481,494,501,493,515,525,539,559,568,534,546,572,589,511,422,371,2
1],"all_points_y":
[947,875,809,702,707,537,455,428,387,341,217,163,136,81,77,120,128,85,91,151,217,320,358,400,444,502,597,633,706,802,847,860,888,899,944,991,994,1053,1049,1049,
016,947]},"region_attributes":{}}}},"17.jpg92711":{"fileref":"","size":92711,"filename":"17.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":
{"name":"polygon","all_points_x":
[149,171,185,160,177,165,171,154,169,177,201,201,190,188,267,303,319,339,365,375,428,442,438,407,400,429,446,451,415,426,452,468,449,460,451,501,522,536,563,567
3,712,725,732,659,421,305,309,219,169,149],"all_points_y":
[975,928,854,695,670,572,526,515,484,437,435,372,247,120,194,196,157,146,168,202,185,211,297,326,337,378,487,524,513,544,553,589,645,698,766,796,855,959,966,103
0,1123,1135,1157,1179,1172,1154,1132,1085,1063,1007,975]},"region_attributes":{}}}},"18.jpg208123":
{"fileref":"","size":208123,"filename":"18.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":{"name":"polygon","all_points_x":
[64,200,375,330,292,311,328,326,315,317,345,345,365,359,368,366,375,399,426,459,487,565,583,578,550,521,515,541,583,596,608,621,657,666,665,615,551,433,362,225,
oints_y":
[1028,929,949,929,930,888,887,649,621,533,376,315,280,230,209,187,116,111,155,163,192,180,187,212,229,276,300,454,550,587,674,725,796,867,953,1034,1042,1050,101
6,1028]},"region_attributes":{}},"1":{"shape_attributes":{"name":"polygon","all_points_x":
[599,607,633,611,628,621,642,629,645,666,707,750,759,792,842,858,836,809,803,824,830,865,876,879,866,866,917,928,959,963,1046,1132,1141,982,719,676,692,675,665,
1,599],"all_points_y":
[522,422,396,337,295,253,228,178,100,101,159,166,186,179,192,221,253,293,367,451,521,569,597,653,674,715,748,828,869,895,838,933,974,1009,1016,1021,969,938,780,
6,522]},"region_attributes":{}}}},"19.jpg435002":{"fileref":"","size":435002,"filename":"19.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":
{"name":"polygon","all_points_x":
[124,150,152,154,163,178,220,248,259,300,314,347,355,330,328,337,348,455,498,491,374,339,309,297,308,302,277,234,258,269,244,243,275,263,215,229,211,190,171,167
all_points_y":
[420,371,324,282,222,188,143,107,73,57,63,63,90,96,123,155,169,178,173,194,200,207,189,207,217,236,233,243,278,302,328,351,388,392,340,307,294,312,329,375,416,42
_attributes":{}}}},"20.jpg11375":{"fileref":"","size":11375,"filename":"20.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":
{"name":"polygon","all_points_x":[59,162,211,222,221,221,214,186,171,172,157,137,146,109,80,72,49,41,22,18,27,42,48,47,34,19,26,59],"all_points_y":
[164,142,116,100,74,63,56,76,72,62,54,65,69,84,104,84,86,89,52,67,95,108,134,150,156,143,174,164]},"region_attributes":{}}}},"21.jpg52582":
{"fileref":"","size":52582,"filename":"21.jpg","base64_img_data":"","file_attributes":{},"regions":{"0":{"shape_attributes":{"name":"polygon","all_points_x":
[295,285,236,246,242,234,256,259,268,281,314,321,335,359,388,430,424,407,444,471,460,467,481,475,422,411,381,347,326,305,295],"all_points_y":
[374,327,291,276,240,216,200,215,215,193,193,234,247,227,214,236,265,272,310,345,373,390,423,425,436,417,388,404,383,391,374]},"region_attributes":{}}}}}

รูป 4.9 ตัวอย่างไฟล์ JSON ที่ระบุตำแหน่งของวัตถุที่สนใจในรูปภาพ

2) หลังจากที่ได้ไฟล์ทั้ง 840 ภาพแล้วทำมา Training โดยใช้ TensorFlow และภาษา Python โดยกำหนดค่าดังนี้

```
class DataConfig(Config):

    # Give the configuration a recognizable name
    NAME = "dog"

    # Adjust down if you use a smaller GPU With 12 GB GPURam.
    IMAGES_PER_GPU = 1

    # Number of classes (including background)
    NUM_CLASSES = 1 + 1  # Background + dog

    # Number of training steps per epoch
    STEPS_PER_EPOCH = 100

    # Skip detections with < 90% confidence
    DETECTION_MIN_CONFIDENCE = 0.9
```

รูป 4.10 ค่าพารามิเตอร์ที่นำไปปรับแต่งการเทรนข้อมูลใน TensorFlow

1) **NAME** เป็นชื่อคลาสที่ต้องการเทรน
2) IMAGE_PER_GPU เป็นการระบุจำนวนภาพต่อการนำไปประมวลผลในแต่ละครั้งโดย Mask R-CNN แนะนำ 1 ภาพใช้หน่วยความจำหน่วยประมวลผลกราฟฟิค 6 กิกะไบท์
3) NUM_CLASSES จำนวนคลาสที่ต้องการเทรน
4) STEP_PER_EPOCH จำนวนในการเทรนว่าต้องการเทรนชุดข้อมูลกี่รอบเพื่อค้นหาค่าที่ดีที่สุด
5) DETECTION_MIN_CONFIDENCE เป็นค่าที่ให้ระบุว่าถ้าพบภาพที่การเรียนรู้ของเครื่องแสดงผลออกมาต่ำกว่ากี่เปอร์เซ็นให้ทำการข้ามไปและไม่ต้องเรียนรู้หรือเก็บข้อมูลในส่วนนั้น

หลังจากที่ทำการกำหนดค่าพารามิเตอร์ของ TensorFlow เสร็จแล้วก็ทำการ Load Mask ที่ได้ทำไว้บน JSON ไฟล์และโหลดรูปภาพทั้งหมดเพื่อเตรียมเข้าสู่ฟังก์ชันเทรนโมเดลข้อมูล

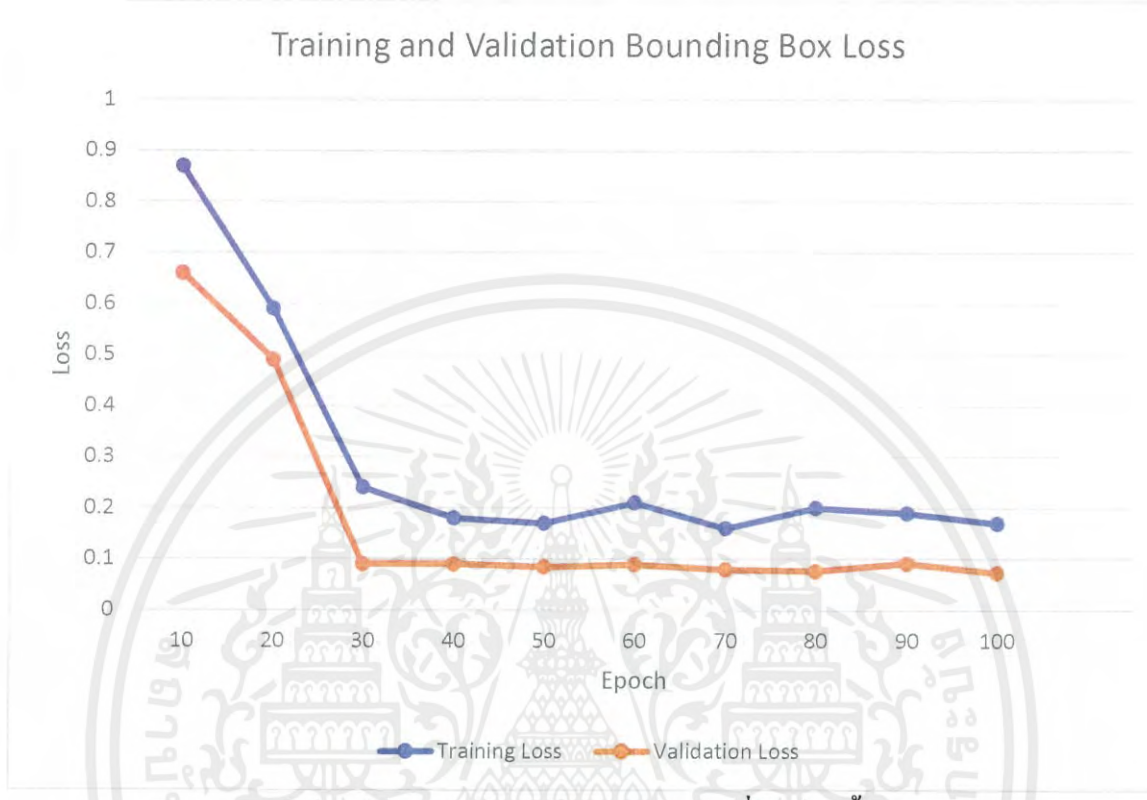
```
def load_mask(self, image_id):
    """Generate instance masks for an image.
   Returns:
    masks: A bool array of shape [height, width, instance count] with
        one mask per instance.
    class_ids: a 1D array of class IDs of the instance masks.
    """
    # If not a dog dataset image, delegate to parent class.
    image_info = self.image_info[image_id]
    if image_info["source"] != "dog":
        return super(self.__class__, self).load_mask(image_id)

    # Convert polygons to a bitmap mask of shape
    # [height, width, instance_count]
    info = self.image_info[image_id]
    mask = np.zeros([info["height"], info["width"], len(info["polygons"])],
                    dtype=np.uint8)
    for i, p in enumerate(info["polygons"]):
        # Get indexes of pixels inside the polygon and set them to 1
        rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
        mask[rr, cc, i] = 1

    # Return mask, and array of class IDs of each instance. Since we have
    # one class ID only, we return an array of 1s
    return mask.astype(np.bool), np.ones([mask.shape[-1]], dtype=np.int32)

def image_reference(self, image_id):
    """Return the path of the image."""
    info = self.image_info[image_id]
    if info["source"] == "dog":
        return info["path"]
    else:
        super(self.__class__, self).image_reference(image_id)
```

รูป 4.11 ฟังก์ชันการอ่านข้อมูล JSON และรูปภาพเข้าสู่การ Train

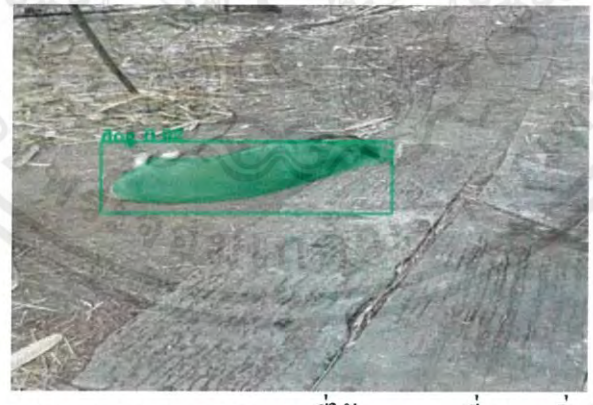
```
def train(self, train_dataset, val_dataset, learning_rate, epochs, layers,
          augmentation=None, custom_callbacks=None, no_augmentation_sources=None):
    """Train the model.
    assert self.mode == "training", "Create model in training mode"

    # Pre-defined layer regular expressions
    layer_regex = {
    }
    if layers in layer_regex.keys():
        layers = layer_regex[layers]

    # Data generators
    train_generator = data_generator(train_dataset, self.config, shuffle=True,
                                     augmentation=augmentation,
                                     batch_size=self.config.BATCH_SIZE,
                                     no_augmentation_sources=no_augmentation_sources)
    val_generator = data_generator(val_dataset, self.config, shuffle=True,
                                   batch_size=self.config.BATCH_SIZE)

    # Create log_dir if it does not exist
    if not os.path.exists(self.log_dir):
        os.makedirs(self.log_dir)

    # Callbacks
    callbacks = [
    ]

    # Add custom callbacks to the list
    if custom_callbacks:
```

รูป 4.12 ฟังก์ชันการเทรนข้อมูลโดยใช้ TensorFlow Keras บน Mask R-CNN

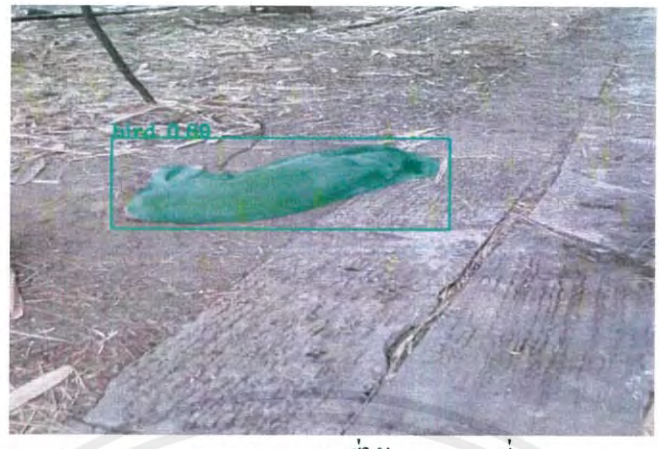
4.2.3 ผลการทดลองเทรนโมเดลข้อมูล



รูป 4.13 อัตราความผิดพลาดของ Dataset ที่ได้เทรนขึ้นมา



รูป 4.14 การ Prediction ของ Mask R-CNN ที่ใช้ Dataset ที่เทรนเพิ่มขึ้น (Dog = 0.92)



รูป 4.15 การ Prediction ของ Mask R-CNN ที่ใช้ Dataset ที่ของ COCO (Bird = 0.88)

รูป 4.16 การ Prediction ที่ผิดพลาดของ Mask R-CNN ที่ใช้ Dataset ที่เทรนขึ้นมา (Dog = 0.81)

รูป 4.17 การ Prediction ที่ผิดพลาดของ Mask R-CNN ที่ใช้ Dataset ที่เทรนขึ้นมา (Dog = 0.85)

4.2.4 สรุปผลการทดลอง

หลังจากที่ได้ทำการทดลองเทรน Data ที่เพิ่มขึ้นจบว่าจากรูป 4.13 พบว่ากราฟเริ่มมีการทำนายที่ผิดพลาดลดลงทั้งการ Training และ Validation ในรอบการเทรนที่ 1-40 และค่อนข้างคงที่หลังรอบการเทรนที่ 60 เป็นต้นไปจึงหยุดการเทรนและนำชุดโมเดลข้อมูลไปใช้งานกับรูปภาพต่าง ๆ พบว่าชุดโมเดลที่ได้เทรนขึ้นมาใหม่นั้นสามารถตรวจจับสุนัขในหลากหลายมุม หลายหลายอริยาบถมากขึ้น แต่ว่าก็ทำให้เกิด False Positive ซึ่งก็คือผลลัพธ์มีการตรวจจับสุนัขได้ทั้งที่รูปภาพไม่ได้เป็นสุนัขเป็นครั้งคราว (มักจะเกิด กับรูปอาคาร , รถยนต์) สันนิษฐานว่าชุดข้อมูลที่ได้สร้างขึ้นอาจจะมีรูปภาพที่นำไปเทรนในระบบไม่เพียงพอหรือไฟล์ JSON ที่ระบุตำแหน่งที่สนใจไม่ละเอียดพอ สาเหตุจาก Mask ของอาคารไม่ได้แพร่ทั่วทั้งอาคาร แต่แพร่เพียงบางส่วนอาจจะเกิดจากการที่ ข้อมูลในการเทรนที่ไม่เพียงพอหรือ ชุดข้อมูลที่มีลักษณะเด่นไม่เพียงพอทำให้เกิด Fault Positive ในบางครั้งกับวัตถุที่ไม่น่าจะเป็นสุนัขได้ซึ่งแนวทางการแก้ไขในอนาคตทำได้โดยการเพิ่มข้อมูลที่จะเทรนเข้าไปให้มากขึ้นกว่าเดิม หรือกำหนดคุณลักษณะให้ชัดเจน (สุนัขหน้าตรง เท่านั้น) เพื่อป้องกันความกำกวมของโมเดลข้อมูล

4.3 การทดลองประสิทธิภาพการทำงานของการติดตามวัตถุ

การทดลองนี้มีวัตถุประสงค์ในการทดสอบประสิทธิภาพการทำงานของการติดตามวัตถุว่าสามารถทำงานได้แม่นยำมากน้อยเพียงใดและใช้เวลาในการประมวลมากน้อยเท่าใด โดยการทดลองนี้ได้ใช้การตรวจจับวัตถุแบบ Centroid Tracking และ OpenCV มาเปรียบเทียบกันเพื่อทำการเลือกใช้วิธีการในการติดตามการเคลื่อนไหวของสุนัขในวิดีโอ

4.3.1 รายละเอียดในการทดลอง

1) ใช้วิดีโอตัวอย่างในการทดสอบประสิทธิภาพของการติดตามวัตถุสามชนิดได้แก่ Centroid Tracking , OpenCV KLT Tracking และ OpenCV BLOB Tracking
2) ใช้ Mask R-CNN โดยใช้ Dataset ของตัวเองในการคัดกรองสุนัขที่ตรวจพบและใช้การติดตามวัตถุในการตรวจจับการเคลื่อนไหวของสุนัข

4.3.2 วิธีการทดลอง

1) วิดีโอที่ 1 เป็นวิดีโอที่มีสุนัขจำนวน 6 ตัวโดยนับสายตาจากวิดีโอในเว็บไซต์นี้ https://youtu.be/FYlwHENzxn4

รูป 4.18 ภาพตัวอย่างวิดีโอที่ 1

2) วิดีโอที่ 2 เป็นวิดีโอที่มีสุนัขจำนวน 14 ตัวโดยนับจากสายตาจากวิดีโอในเว็บไซต์นี้ https://youtu.be/X7H5zyZ_Na8

รูป 4.19 ภาพตัวอย่างวิดีโอที่ 2

3) นำวิดีโอตัวอย่างสองวิดีโอนี้เข้าไปคำนวณใน Mask R-CNN และ Object Tracker ผลลัพธ์ที่ออกมาจะเป็นจำนวนสุนัขที่ระบบคำนวณได้ และ เวลาที่ใช้ในการประมวลผล

4.3.3 ผลการทดลอง

ตาราง 4.3 ตารางเปรียบเทียบประสิทธิภาพของการติดตามวัตถุของ Centroid Tracking , OpenCV KLT Tracking และ OpenCV BLOB Tracking วิดีโอที่ 1

วิธีที่ใช้	เวลาในการประมวลผล	จำนวนสุนัขที่พบ
Centroid Tracking	27 วินาที	9 ตัว
CV2 Object Tracker KLT Tracking	3 นาที 19 วินาที	8 ตัว
CV2 Object Tracker BLOB	7 นาที 1 วินาที	7 ตัว

ตาราง 4.4 ตารางเปรียบเทียบประสิทธิภาพของการติดตามวัตถุของ Centroid Tracking , OpenCV KLT Tracking และ OpenCV BLOB Tracking วิดีโอที่ 2

วิธีที่ใช้	เวลาในการประมวลผล	จำนวนสุนัขที่พบ
Centroid Tracking	10 วินาที	19 ตัว
CV2 Object Tracker KLT Tracking	1 นาที 16 วินาที	19 ตัว
CV2 Object Tracker BLOB	2 นาที 39 วินาที	16 ตัว

โดยคอมพิวเตอร์ที่ใช้ในการประมวลผลได้ใช้ Intel CORE i7 3770 3.4 กิกะเฮิร์ตซ์ หน่วยความจำ 8 กิกะไบต์ หน่วยประมวลผลกราฟฟิค Nvidia GTX 1050TI

4.2.4 สรุปผลการทดลอง

การทดลองครั้งนี้สรุปว่าการประมวลผลของ Object Tracker นั้น Centroid Tracking อัลกอริทึมใช้เวลาในการประมวลผลน้อยที่สุด แต่มี Error มากที่สุดเช่นกัน แต่การใช้ OpenCV ทั้งสองฟังก์ชั่นนั้นมีความสามารถในการตรวจจับการเคลื่อนไหวได้มากขึ้น แต่ใช้เวลาในการประมวลผลมากขึ้น ดังนั้นกลุ่มของข้าพเจ้าจึงได้เลือกใช้ Centroid Tracking ในการตรวจสอบการเคลื่อนไหวของสุนัขเนื่องจากต้องการความเร็วในการประมวลผลให้เร็วขึ้นแม้ว่าจะเสียความแม่นยำไปแต่เวลาที่ใช้ในการประมวลผลจะน้อยลงมาก

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 ผลลัพธ์จากการทำโครงงาน

จากการทดลองต่าง ๆ ในการนับจำนวนประชากรสุนัขด้วยวิดีโอนั้น พบว่า การใช้การเรียนรู้ของเครื่องในการนับประชากรสุนัขนั้นยังไม่มีความแม่นยำมากเท่าที่ควรจะเป็น เนื่องจากการใช้ Mask R-CNN ที่เป็นการเรียนรู้ของเครื่องแบบ Supervised Learning นั้นมีความจำเป็นต้องกำหนดข้อมูลที่ให้คอมพิวเตอร์นั้นเรียนรู้ด้วยตัวมนุษย์ ซึ่งในเวลาที่จำกัด กลุ่มของข้าพเจ้านั้นได้รวบรวมข้อมูลที่นำมาเรียนรู้ได้ไม่มากพอหรือไม่มีประสิทธิภาพมากพอทำให้ผลลัพธ์ที่ได้นั้นแม่นยำมากขึ้นในการตรวจจับสุนัขในหลาย ๆ มุม แต่ก็ทำให้เกิดปัญหาที่เครื่องนั้น มองภาพวัตถุบางชิ้น เป็นสุนัขไปด้วยและ การนำการเรียนรู้ของเครื่องมาใช้ในการตรวจจับภาพที่มีความอิสระค่อนข้างสูงอย่างวิดีโอนั้นพบว่า เกิดข้อจำกัดมากมายอาทิเช่น ชุดข้อมูลสำเร็จรูปที่ใช้แล้วไม่มีความแม่นยำมากพอจะจับสุนัขได้ การจับได้เฉพาะใบหน้า การจับสุนัขตัวเดียวกันในแต่ละเฟรมของวิดีโอ และ อื่น ๆ ทำให้การนับจำนวนประชากรสุนัขด้วยวิดีโอนั้นประสบความสำเร็จไม่ได้ตามที่ระบุไว้ในวัตถุประสงค์

5.2 ปัญหา อุปสรรคที่พบ และแนวทางแก้ไข

1) เกิดปัญหาในด้านการประมวลผลภาพจากกล้องวงจรปิด มักมีปัญหาที่ Mask R-CNN ไม่สามารถ Detect ได้ว่าภาพใน Frame นั้นคือสุนัข ซึ่งแก้ไขโดยขยายขนาดภาพให้ใหญ่ขึ้น
2) เกิดปัญหา AWS Lambda Framework ไม่สามารถทำงานกับไฟล์ที่ใหญ่กว่า 250 เมกกะไบท์ได้ จึงไม่สามารถใช้งาน Mask R-CNN บน AWS Lambda Framework ได้เนื่องจากไฟล์ทั้งหมดมีขนาดประมาณ 700 เมกกะไบท์แก้ปัญหาได้โดยการสร้าง เซิร์ฟเวอร์ ที่ทำงานตลอดเวลาแทน
3) เกิดปัญหา False Positive ของ Dataset ที่สร้างขึ้นมาส่งผลให้เกิดการ Detect ภาพอื่น ๆ ที่ไม่ใช่สุนัขทำให้เกิดความคลาดเคลื่อน ซึ่งคาดเดาว่าจะสามารถแก้ปัญหาโดยการสร้างข้อกำหนดให้ชัดเจนอาทิ สร้างชุดข้อมูลโมเดลที่มีเฉพาะใบหน้าของสุนัขแบบเดียวกันกับ COCO Dataset หรือ เทรนข้อมูลให้มากกว่าเดิมโดยการเพิ่มภาพและการระบุตำแหน่งของวัตถุที่สนใจให้ละเอียดมากขึ้น

4) ใช้ runtime บนเซิร์ฟเวอร์ Backend สูงมากในการประมวลผล Mask R-CNN ทำให้ไม่เหมาะที่จะใช้คอมพิวเตอร์ทั่วไปในการรัน Service Mask R-CNN แก้ได้โดยการเช่าบริการ Cloud Instance ต่าง ๆ ในที่นี้กลุ่มของข้าพเจ้าใช้กูเกิ้ลคลาวด์แพลทฟอร์มแทน
5) ปัญหาการนับสุนัขที่จำนวนมักเกินจากการนับสุนัขในวิดีโอด้วยสายตา เนื่องจาก Dataset มีการเกิด Fault Positive ขึ้น และการจับตำแหน่งสุนัขตัวเดิมที่เคลื่อนที่ในวิดีโอที่ไม่ได้แม่นยำมากนัก แนวทางการแก้ไขนั้นสามารถใช้ Motion Detect ในอนาคตหรือให้เครื่องนั้นจดจำคุณลักษณะของสุนัขทุกตัวที่ผ่านเข้ามาแทน

5.3 แนวทางการพัฒนาต่อ

1) เพิ่มส่วนของการเรียนรู้โดยเครื่อง (Machine Learning) พัฒนาชุดข้อมูล (Dataset) ต่อจากเดิมทุกครั้งที่มีการส่งภาพเข้ามาเพื่อให้ Dataset มีความแม่นยำที่สูงขึ้น หรือหาวิธีในการใช้ Unsupervised Learning จำพวก Data Clustering หรือ Faster RCNN แทน
2) พัฒนาในด้าน Backend เพิ่มเติมโดยเพิ่มกระบวนการพัฒนาวิธีในการรับมือวิธีการนับสุนัขตัวเดียวกันในวิดีโอเป็นการเก็บคุณลักษณะสุนัขแทน หรือวิธีอื่น ๆ
3) นำการวิจัยนี้ไปทำการดัดแปลงและสร้างชุดโมเดลข้อมูลของตัวเองได้ เพื่อนำไปใช้ประโยชน์ในแต่ละปัญหา
4) พัฒนาในด้านประสิทธิภาพในการทำงานให้เร็วขึ้นในด้านการประมวลผลของ Mask R-CNN เนื่องจากปัจจุบันยังคงทำได้ไม่ Real Time เนื่องจากต้องวนภาพเก็บข้อมูลอย่างมาก

บรรณานุกรม

Matterport, Inc. 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. [Online].Available : https://github.com/matterport/Mask_RCNN

Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. 2017. Mask R-CNN Computer Vision and Pattern Recognition. [Online].Available : https://arxiv.org/pdf/1703.06870

Priya Dwivedi . 2018. Building a Mask R-CNN Model for Detecting Car Damage. [Online].Available : https://www.analyticsvidhya.com/blog/2018/07/building-mask-r-cnn-model-detecting-damage-cars-python/

COCO,Inc. 2017. Common Object in Context. [Online].Available : http://cocodataset.org/#overview

Heng2j. 2018. Mask R-CNN for Object Detection and Segmentation. [Online].Available : https://github.com/heng2j/Mask_RCNN

agurchand. 2017. jQuery Ajax File upload with Percentage Progress bar. [Online].Available : http://theonlytutorials.com/jquery-ajax-file-upload-with-percentage-progress-bar/

Nvidia Developer. 2019 . NVIDIA cuDNN . [Online].Available : https://developer.nvidia.com/cudnn

LearnBoost. 2010 . Mongoose a MongoDB object modeling tool designed to work in an asynchronous Environment. [Online].Available : https://mongoosejs.com/docs/guide.html

Node.js Foundation,Inc. 2015 . NodeJS Documentation. [Online].Available : https://nodejs.org/en/docs/

Nuttavut Thongjo. 2016 . [Machine Learning#1] Machine Learning คืออะไร? รู้จักประเภทต่าง ๆ ของ Machine Learning. [Online].Available : https://www.babelcoder.com/blog/posts/machine-learning-introduction