โมบายแอปพลิเคชันสำหรับจำแนกสายพันธุ์นกด้วยวิธีการเรียนรู้เชิงลึก

Mobile Application for Breeding Bird Classification using Deep Learning Technique

*ภูมิภักดิ์ พรมรังกา, สหรัฐ แหวนทอง และ ชูพันธุ์ รัตนโภคา**

*Phummiphak Promrangka, Saharat Wanthong, and Choopan Rattanapoka**

วิทยาลัยเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

College of Industrial Technology, King Mongkut's University of Technology North Bangkok

ABSTRACT – Currently, several bird parks have walk-in aviaries where visitors can see various bird species up close. However, visitors are occasionally unable to identify the bird species they are watching. Because the bird species found in walk-in aviaries are not typically seen in everyday life, and there are no zoo signs inside the walk-in aviaries. Therefore, this article proposes a design and development of a mobile application using the Flutter framework. The application can detect and classify 10 bird species using a deep learning model named EfficientDet Lite, which is created by the TFLite model maker library. The users can use a smartphone camera to examine the birds. Then, when birds are found, the application will provide users the information of the bird. From the experiments, we found that the EfficientDet Lite 0, which is the smallest version of EfficientDet Lite, gave the most suitable results for the application. The model took 62.3 ms for the inference time with the precision, recall, accuracy and F1-score of 0.94, 0.94, 0.99, and 0.94, respectively.

KEY WORDS: Mobile application, Flutter, EfficientDet model, TensorFlow Lite, TFLite model maker library

บทคัดย่อ -- ปัจจุบันมีสวนนกหลายแห่งที่มีกรงนกเปิด ซึ่งผู้ชมสามารถเข้าไปชมนกสายพันธุ์ต่าง ๆ ได้อย่างใกล้ชิด แต่อย่างไรก็ตามผู้เข้าชมอาจจะไม่สามารถทราบสายพันธุ์ของนกที่กำลังชมอยู่ได้ เนื่องจากเป็นนกสายพันธุ์ที่ไม่ได้พบทั่วไปในชีวิตประจำวัน และการไม่มีป้ายระบุข้อมูลกำกับ ดังนั้นบทความนี้เสนอการออกแบบและพัฒนาโมบายแอปพลิเคชันด้วย Flutter Framework สำหรับตรวจสอบและจำแนกสายพันธุ์นกทั้งหมด 10 สายพันธุ์ ด้วยการใช้แบบจำลองการเรียนรู้เชิงลึก EfficientDet Lite ซึ่งถูกสร้างผ่านไลบรารี TFLite Model Maker ผู้ใช้งานสามารถใช้กล้องของสมาร์ทโฟนส่องไปที่นก เมื่อเจอนกแล้วแอปพลิเคชันจะแสดงข้อมูลของนกตัวนั้นให้ผู้ใช้งาน จากการทดลองพบว่าแบบจำลอง EfficientDet Lite 0 ซึ่งเป็นรุ่นของแบบจำลอง EfficientDet Lite ที่มีขนาดเล็กที่สุด ให้ผลลัพธ์ที่เหมาะสมกับการใช้งานมากที่สุด โดยใช้เวลาแปรผลภาพที่ 62.3 ms และมีค่า Precision, Recall, Accuracy และ F1-Score เท่ากับ 0.94, 0.94, 0.99 และ 0.94 ตามลำดับ

คำสำคัญ: โมบายแอปพลิเคชัน, Flutter, แบบจำลอง EfficientDet, TensorFlow Lite, ไลบรารี TFLite model maker

**Corresponding Author: choopan.r@cit.kmutnb.ac.th*

1. บทนำ

มนุษย์มีความคุ้นเคยและใกล้ชิดกับนก เนื่องจากนกเป็นทั้งสัตว์เลี้ยงและสัตว์ตามธรรมชาติ อีกทั้งยังมีส่วนช่วยกำจัดศัตรูพืชและขยายพันธุ์พืช [1] ในปัจจุบันมีสวนสัตว์เปิดที่มีนกอยู่หลายสายพันธุ์อาศัยอยู่ด้วยกัน นักท่องเที่ยวบางคนที่ไปเยี่ยมชมไม่สามารถจำแนกชนิดของนกได้ เพราะว่านกแต่ละสายพันธุ์มีลักษณะเฉพาะที่แตกต่างกัน ซึ่งยากต่อการจดจำ ทั้งในเรื่องของลักษณะของรูปร่าง สีขน ลักษณะเฉพาะของขน จะงอยปาก และเป็นนกที่ไม่ค่อยได้พบเจอทั่วไปในเมืองหรือหมู่บ้านที่คนอยู่อาศัย ถ้าหากมีการนำเทคโนโลยีที่น่าสนใจและแปลกใหม่มาช่วยในการจำแนกสายพันธุ์ของนก นอกจากจะทำให้ผู้คนรู้จักนกมากขึ้นแล้ว จะยังสามารถดึงดูดให้ผู้ใช้ที่เป็นเด็กในช่วงปฐมวัยไปจนถึงวัยประถมศึกษาให้มาสนใจในนก เพราะเด็กในช่วงอายุนี้มักจะให้ความสนใจในเทคโนโลยีที่แปลกใหม่ [2]

โทรศัพท์มือถือแบบสมาร์ทโฟนได้มีบทบาทมากขึ้นในชีวิตประจำวัน ในปัจจุบันมีแพลตฟอร์มที่ยืดหยุ่นอย่าง Flutter ที่สามารถพัฒนาแอปพลิเคชันบนมือถือได้สะดวกขึ้น และเป็น Cross-Platform Framework ที่ได้รับความนิยมมากที่สุดในปี 2021 คิดเป็น 42% ของทั้งหมด [3] รวมถึงโทรศัพท์มือถือนั้นมีประสิทธิภาพมากพอที่จะประมวลผลได้เทียบเท่ากับคอมพิวเตอร์ขนาดเล็ก ในขณะเดียวกันเทคโนโลยีการเรียนรู้ของเครื่อง (Machine Learning) ก็มีบทบาทสำคัญมากในการเข้ามาจำแนกชนิดหรือประเภทของวัตถุ หรือจำแนกชนิดหรือสายพันธุ์ของสิ่งมีชีวิตต่างๆ [4],[5] โดยการใช้เทคนิคโครงข่ายประสาทคอนโวลูชัน (Convolutional Neural Network: CNN) ในการจำแนกรูปภาพ [6] อีกทั้งโทรศัพท์ยังมีกล้องถ่ายรูปในตัวและสามารถนำไปใช้งานร่วมกับเทคโนโลยีที่กล่าวไว้ข้างต้นอีกด้วย

ดังนั้นบทความนี้จึงนำเสนอการพัฒนาโมบายแอปพลิเคชันที่ช่วยในการตรวจสอบสายพันธุ์ของนก เพื่อให้ผู้ใช้สามารถศึกษา เรียนรู้ และสามารถจำแนกนกชนิดต่าง ๆ ได้ โดยมีการใช้งานร่วมกับแบบจำลองการเรียนรู้เชิงลึกที่ใช้เทคโนโลยี CNN ช่วยในการจดจำรูปภาพ แอปพลิเคชันนี้จะพัฒนาด้วยภาษา Dart บน Flutter Framework ซึ่งจะช่วยผู้ใช้ในการตรวจสอบ และดูข้อมูลรายละเอียดเกี่ยวกับนกในสายพันธุ์ต่างๆ ได้อย่างสะดวก และยังช่วยเพิ่มความน่าสนใจในการศึกษาเกี่ยวกับนก

2. ทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้อง

2.1 Flutter

Flutter [7] คือ Framework ที่เป็นแบบโอเพนซอร์ส พัฒนาโดย Google ใช้สำหรับการสร้างแอปพลิเคชันที่มีความสวยงาม และสามารถทำงานได้บนหลายแพลตฟอร์ม ได้แก่ อุปกรณ์พกพา เว็บ เดสก์ทอป และอุปกรณ์ฝังตัว ด้วยโค้ดเดียวกัน คุณสมบัติหลักของ Flutter Framework คือ

- ความเร็ว : โค้ดของ Flutter จะถูกคอมไพล์เป็นชุดคำสั่งแบบ ARM หรือ Intel จึงทำให้มีประสิทธิภาพ และทำงานได้อย่างรวดเร็วบนอุปกรณ์ทุกชนิด

- ความง่ายในการพัฒนาโปรแกรม : ใน Flutter มีการทำงานของ Hot Reload ซึ่งทำให้ผู้พัฒนาโปรแกรมสามารถ Build และ Run ซ้ำอย่างรวดเร็ว อีกทั้งดูการเปลี่ยนแปลงของโปรแกรมได้เกือบจะในทันทีโดยไม่สูญเสียสถานะของโปรแกรมที่ทำงานอยู่

- ความยืดหยุ่นในการแสดงผล : ผู้พัฒนาโปรแกรมสามารถควบคุมตำแหน่งต่าง ๆ บนหน้าจอได้อย่างละเอียด จึงทำให้มีความหลากหลายและยืดหยุ่นในการออกแบบ

- สามารถใช้งานได้หลายแพลตฟอร์ม : โค้ดของ Flutter นั้นสามารถนำไปใช้กับอุปกรณ์หลายชนิด เช่น อุปกรณ์เคลื่อนที่ เว็บ เดสก์ท็อป และอุปกรณ์ Embeded โดยไม่จำเป็นต้องแก้ไขโค้ด

2.2 TensorFlow

TensorFlow [8] คือ ไลบรารีสำหรับสร้างแบบจำลองการเรียนรู้เชิงลึก ซึ่งมีการนำไปประยุกต์เพื่อเพิ่มประสิทธิภาพกับผลิตภัณฑ์หลากหลายประเภท เช่น เครื่องมือค้นหา (Search Engine), การแปลภาษา (Translation), คำบรรยายภาพ (Image Captioning) และ การแนะนำ (Recommendations)

TensorFlow ถูกพัฒนาโดย Google เพื่อให้นักวิจัยและนักพัฒนาทำงานกับแบบจำลอง AI ได้ เมื่อพัฒนาและปรับปรุงชักระยะหนึ่ง TensorFlow ก็ถูกปล่อยออกมาให้คนทั่วไปใช้งานได้ โดยเปิดโอเพนซอร์สในปี 2015 และปล่อยตัวสมบูรณ์ออกมาในปี 2017 พร้อมลิขสิทธิ์แบบ Apache Open Source ให้คนทั่วไปสามารถใช้งาน ดัดแปลง และแจกจ่ายตัวที่ถูกดัดแปลงมาแล้วโดยที่ไม่มีค่าใช้จ่าย

TensorFlow เป็นที่นิยม เนื่องจากถูกสร้างมาเพื่อให้ทุกคนเข้าถึงได้ง่าย TensorFlow ได้รวมเอา API ที่แตกต่างกัน เพื่อ

สร้างสถาปัตยกรรมแบบ Deep Learning อย่าง Convolutional Neural Network (CNN) และ Recurrent Neural Network (RNN) TensorFlow ยังมี Graph เป็นตัวคำนวณหลัก เพื่อช่วยให้นักพัฒนาเห็นภาพโครงสร้าง Neural Network รวมทั้งสามารถทำงานร่วมกับ Tensorboad เครื่องมือที่ช่วยให้นักพัฒนาหาข้อผิดพลาดของโปรแกรม และสุดท้าย TensorFlow สามารถทำงานได้ทั้งบน CPU และ GPU

2.3 TensorFlow Lite

TensorFlow Lite (TFLite) [9] คือ Tools ที่ช่วยให้นักพัฒนาสามารถรันแบบจำลองของ TensorFlow ทำ Inference บนมือถือ Mobile, Android, iOS, อุปกรณ์ Edge, IoT Device, Raspberry Pi, Jetson Nano, Arduino และ Microcontroller ได้ เนื่องจากแบบจำลองของ TFLite มีความซับซ้อนน้อยกว่าแบบจำลองของ Tensorflow ปกติ ทำให้แบบจำลองของ TFLite มีขนาดเล็กลง ทำงานได้เร็วขึ้น แต่อย่างไรก็ตามต้องยอมรับกับความแม่นยำของแบบจำลองที่ลดลง

2.4 Convolutional Neural Network

การเรียนรู้เชิงลึกมีพื้นฐานมาจาก Neural Network ซึ่งเป็นสาขาหนึ่งของเทคโนโลยี Artificial Intelligence ที่มีแนวคิดมานานตั้งแต่ปี 1943 คือ การพัฒนาเครือข่ายของอัลกอริทึมให้ทำงานแบบเดียวกับเครือข่ายระบบประสาทของมนุษย์

Convolutional Neural Network (CNN) [10] หรือ โครงข่ายประสาทแบบคอนโวลูชัน จะจำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อย ๆ และนำกลุ่มของพื้นที่ย่อย ๆ มาผสานกัน เพื่อดูว่าสิ่งที่เห็นอยู่เป็นอะไร โดยการใช้ Matrix ชนิดพิเศษ ที่เรียกว่า Convolution Matrix ซึ่งทำหน้าที่สกัดเอาส่วนต่างๆ ของภาพออกมา เช่น เส้นขอบของวัตถุต่างๆ เพื่อให้สามารถเรียนรู้ลักษณะของภาพได้อย่างมีประสิทธิภาพและแม่นยำ

2.5 EfficientDet

แบบจำลอง EfficientDet [11] นั้นถูกพัฒนาขึ้นในปี 2020 จาก Google Brain Team ซึ่งมีจุดประสงค์ในการพัฒนาแบบจำลองสำหรับตรวจจับวัตถุ (Object Detection) ที่มีความแม่นยำสูงและยืดหยุ่นตามข้อจำกัดทางด้านทรัพยากร

ภายในแบบจำลอง EfficientDet นั้นจะมี BiFPN หรือ Bi-directional Feature Pyramid Network เป็น Feature Fusion Network สำหรับการรวม Feature ที่จะถูกสกัดจากแบบจำลองต่างๆ

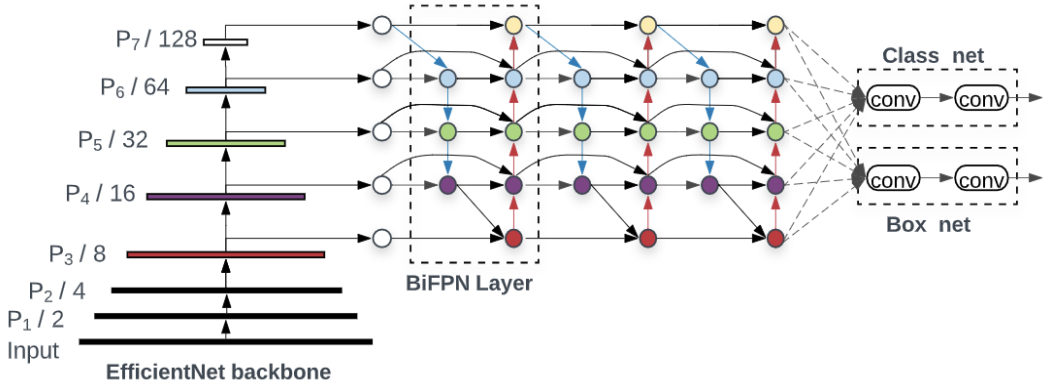
สถาปัตยกรรมของ EfficientDet แสดงดังรูปที่ 1 ประกอบด้วยแบบจำลอง EfficientNet [12] ทำหน้าที่เป็น Backbone ของแบบจำลอง ซึ่งจะสกัดคุณลักษณะของรูปภาพในขนาดต่าง ๆ ออกมา จากนั้นข้อมูลคุณลักษณะเหล่านั้นจะถูกป้อนเข้าไปใน BiFPN ที่ทำหน้าที่เป็น Feature Fusion Network และท้ายสุดข้อมูลที่สกัดได้จาก BiFPN ก็จะถูกป้อนเข้า Class/Box Prediction Network เพื่อทำนาย Object ต่าง ๆ โดยแบบจำลอง EfficientDet จะมีรุ่นย่อย 5 รุ่นคือ EfficientDet 0 จนถึง EfficientDet 4 โดยแต่ละรุ่นจะมีความซับซ้อนของโครงสร้างที่แตกต่างกัน ดังแสดงในตารางที่ 1

2.6 TFlite Model Maker

ไลบรารี TFLite Model Maker [13] เป็นไลบรารีที่ช่วยลดความยุ่งยากในการสร้าง และฝึกแบบจำลองบน TensorFlow Lite โดยใช้ชุดข้อมูลที่กำหนดเอง ใช้การถ่ายโอนการเรียนรู้เพื่อลดจำนวนข้อมูล และระยะเวลาในการฝึกแบบจำลอง โดยในปัจจุบันไลบรารี TFLite Model Maker รองรับงานการเรียนรู้ของเครื่อง ดังต่อไปนี้

- การจำแนกรูปภาพ
- การตรวจจับวัตถุ
- การจัดประเภทข้อความ
- การตอบคำถามกับ BERT
- การจำแนกประเภทเสียง
- คำแนะนำ

สำหรับการตรวจจับวัตถุ TFLite Model Maker มีคลาสคำสั่งสำหรับใช้งานในการสร้าง และฝึกแบบจำลองอยู่หลัก ๆ ทั้งหมด 3 คลาส คือ (1) DataLoader ใช้สำหรับจัดการกับชุดข้อมูลที่ใช้ในการฝึกแบบจำลอง (2) ModelSpec ใช้สำหรับการเลือกรุ่นของแบบจำลอง โดยมีแบบจำลองให้สามารถเลือกใช้ได้ตั้งแต่ EfficientDet Lite 0 ถึง EfficientDet Lite 4 ซึ่งจะเรียงความซับซ้อนของแบบจำลองไปตามลำดับ โดยแบบจำลอง EfficientDet Lite นั้นเป็นแบบจำลอง EfficientDet ที่ลดความซับซ้อนลงมาให้เหมาะกับการใช้งานในอุปกรณ์ IoT โทรศัพท์มือถือ และอุปกรณ์ชนิดอื่น ๆ ที่มีหน่วยประมวลผลขนาดเล็ก [14] และ (3) ObjectDetector ใช้สำหรับการฝึกอบรมแบบจำลอง



รูปที่ 1 สถาปัตยกรรมของแบบจำลอง EfficientDet (ที่มา: จากบทความ [12])

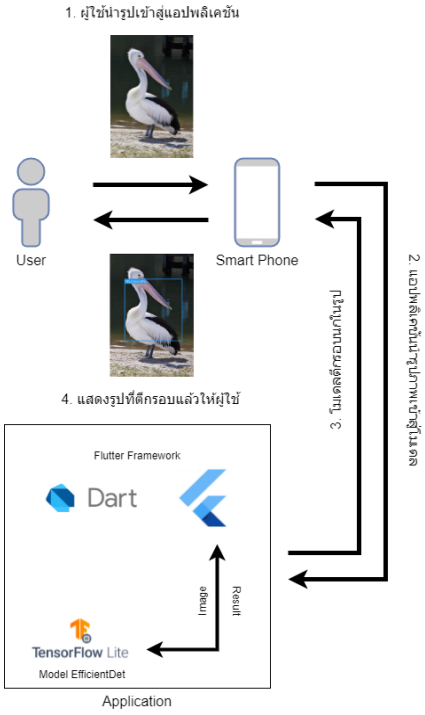
ตารางที่ 1 คุณลักษณะของแบบจำลอง EfficientDet แต่ละประเภท

แบบจำลอง	ขนาด Input	Backbone	BiFPN		จำนวน Layer
			จำนวน Channels	จำนวน Layers	ของ Box/ Class
EfficientDet 0	512	EfficentNet 0	64	3	3
EfficientDet 1	640	EfficentNet 1	88	4	3
EfficientDet 2	768	EfficentNet 2	112	5	3
EfficientDet 3	896	EfficentNet 3	160	6	4
EfficientDet 4	1024	EfficentNet 4	224	7	4

3. วิธีการวิจัย

3.1 โครงสร้างและหลักการทำงานของแอปพลิเคชัน

โมบายแอปพลิเคชันนี้ถูกพัฒนาด้วยภาษา Dart บน Flutter Framework โดยเมื่อผู้ใช้งานเรียกใช้งานแอปพลิเคชันผ่านทางสมาร์ทโฟน แอปพลิเคชันในส่วนของการจำแนกสายพันธุ์นกจะมีขั้นตอนดังแสดงในรูปที่ 2 ซึ่งเริ่มจากผู้ใช้งานสามารถเปิดกล้องเพื่อส่องไปยังนก หรือเลือกภาพนกที่อยู่ในคลังภาพของสมาร์ทโฟน จากนั้นภาพจะถูกส่งไปยังแบบจำลองการเรียนรู้เชิงลึก EffificentDet Lite ซึ่งเป็นแบบจำลองของ EfficientDet ที่ทำงานผ่าน TensorFlow Lite แทน TensorFlow เพื่อให้มีประสิทธิภาพการทำงานที่รวดเร็ว และเหมาะสมกับบนอุปกรณ์พกพา จากนั้นเมื่อแบบจำลองได้รับภาพนก ก็จะประมวลผลและทำนายสายพันธุ์นก พร้อมตีกรอบรอบนกที่พบในภาพกลับมาแสดงให้กับผู้ใช้งาน หลังจากนั้นถ้าผู้ใช้งานต้องการทราบข้อมูลรายละเอียดเกี่ยวกับนก ก็สามารถใช้นิ้วกดบริเวณภายในกรอบที่แอปพลิเคชันได้ตีไว้รอบนกได้



รูปที่ 2 ขั้นตอนการตรวจจับนกของแอปพลิเคชัน

3.2 การเลือกสายพันธุ์นกสำหรับสร้างแบบจำลอง

สำหรับการพัฒนาแบบจำลองการเรียนรู้เชิงลึกในบทความนี้ได้กำหนดให้แบบจำลองสามารถรู้จำสายพันธุ์นกได้ทั้งหมด 10 สายพันธุ์ที่พบทั่วไปในกรงนกเปิด และเพื่อเพิ่มความถูกต้องในการตรวจจับมากขึ้น จึงกำหนดกลุ่มของข้อมูลอีกประเภทหนึ่ง คือ ไม่ใช่นก ดังแสดงในตารางที่ 2

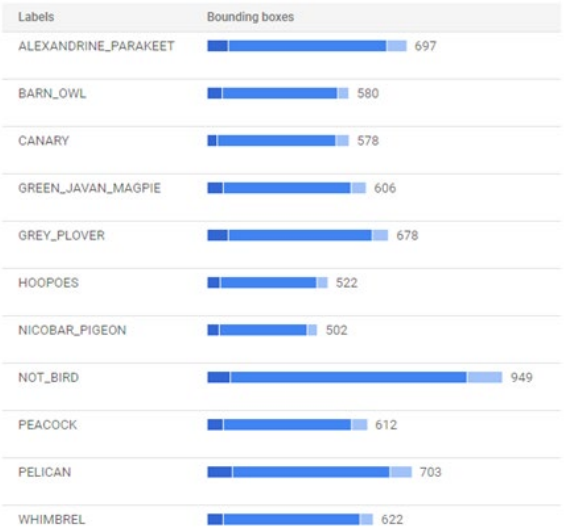
ตารางที่ 2 *รายชื่อสายพันธุ์นกในการพัฒนาแบบจำลอง*

ลำดับที่	ชื่อนกภาษาไทย	ชื่อนกภาษาอังกฤษ
1	นกสาลิกาเขียว	Green Javan Magpie (GRJ)
2	นกขมิ้น	Canary (CAN)
3	นกหัวโต	Grey Plover (GRP)
4	นกกะดง	Nicobar Pigeon (NIC)
5	นกยูง	Peacock (PEA)
6	นกกะรางหัวขวาน	Hoopoes (HOP)
7	นกอีก๋อย	Whimbrel (WHI)
8	นกกระทุง	Pelican (PEL)
9	นกแขกเต้า	Alexandrine Parakeet (ALE)
10	นกแสก	Barn Owl (BAO)
11	ไม่ใช่นก	Not Bird (Not)

3.3 การสร้างและฝึกแบบจำลอง

ในการพัฒนาแบบจำลองการเรียนรู้เชิงลึกในบทความนี้ ผู้แต่งได้เลือกใช้งานไลบรารี TFLite Model Maker ซึ่งเป็นไลบรารีสำหรับการสร้างแบบจำลองที่ทำงานบน TensorFlow Lite มากไปกว่านั้นในไลบรารีนี้มีแบบจำลอง EfficientDet Lite มาให้เรียบร้อยและพร้อมใช้งานแล้ว

สำหรับการฝึกแบบจำลอง เนื่องจากแบบจำลองนี้ได้ถูกออกแบบไว้เพื่อให้สามารถจดจำสายพันธุ์นกได้ 10 สายพันธุ์ และภาพไม่ใช่นก ดังนั้นแบบจำลองนี้จะสามารถทำนายผลได้ทั้งหมด 11 ประเภท จึงแบ่งการฝึกแบบจำลองออกเป็น 11 คลาส โดยแต่ละคลาสใช้ภาพในการฝึกประมาณ 500 – 1000 ภาพ และแบ่งภาพที่รวบรวมและเตรียมไว้ออกเป็นภาพสำหรับการฝึกแบบจำลอง 80% และภาพสำหรับการตรวจสอบการทำงานของแบบจำลอง 20% ดังรูปที่ 3



รูปที่ 3 จำนวนภาพที่ใช้ในการฝึกแบบจำลอง

ในไลบรารี TFLite Model Maker มี Method ที่ใช้สำหรับฝึกแบบจำลองอยู่แล้ว โดยใช้คำสั่ง Object_detector.create และมีพารามิเตอร์ได้แก่ ไฟล์ข้อมูลการฝึก, ชื่อแบบจำลอง, จำนวนรอบการสอน, ขนาดของ Batch, ทำ Fine Tuning หรือไม่ และไฟล์ข้อมูลการตรวจสอบความถูกต้อง

โดย “ไฟล์ข้อมูลการฝึก” และ “ไฟล์ข้อมูลการตรวจสอบความถูกต้อง” เป็นไฟล์รูปภาพแต่ละคลาสที่จัดเก็บในรูปแบบ TFRecord ซึ่งในบทความนี้ได้ใช้คำสั่ง DataLoader ที่มีอยู่ในไลบรารี TFLite Model Maker โหลดรูปภาพจาก Google Cloud Storage ที่ผู้แต่งได้จัดเตรียมไว้ นำมาจะเก็บเป็นรูปแบบ TFRecord เพื่อนำไปใช้ในการฝึก ตรวจสอบความถูกต้อง และทดสอบแบบจำลอง

“ชื่อแบบจำลอง” จะเป็นการเลือกแบบจำลอง แต่ละแบบจำลองจะมีโครงสร้างที่แตกต่างกัน รวมถึงความแม่นยำและ Latency ที่แตกต่างกัน โดยในบทความนี้ได้เลือกใช้แบบจำลอง EfficientDet Lite ในการรู้จำสายพันธุ์ของนก เพราะมีความแม่นยำที่สูง และรวดเร็ว เหมาะสำหรับตรวจจับวัตถุที่มีการเคลื่อนไหว [15]

“จำนวนรอบการสอน” (Epoch) คือ จำนวนรอบทั้งหมดที่นำข้อมูลภาพทั้งหมดป้อนเข้าไปสอนแบบจำลอง จนครบ โดยในบทความนี้ได้กำหนดไว้ที่ 85 รอบ

“ขนาดของ Batch” คือจำนวนรายการข้อมูลที่จะให้ Optimiser คำนวณในหนึ่งครั้ง โดยบทความนี้ได้กำหนดขนาดไว้เท่ากับ 16

3.4 ข้อมูล และรายละเอียดของนก

ข้อมูลของนกชนิดต่าง ๆ ทางผู้จัดทำได้ทำการสืบค้นข้อมูลมาจากวิกิพีเดีย และสารานุกรมสัตว์ออนไลน์ขององค์การสวนสัตว์แห่งประเทศไทยในพระบรมราชูปถัมภ์ [16] และนำมาเรียบเรียงลงในแอปพลิเคชัน โดยมีรายละเอียดข้อมูลของนกแต่ละสายพันธุ์ดังนี้

- ชื่อนกภาษาไทย
- ชื่อนกภาษาอังกฤษ
- ลักษณะทั่วไปของนก
- พฤติกรรมของนก
- อาหารที่กิน
- ถิ่นที่อยู่อาศัย

3.5 การประเมินประสิทธิภาพของแบบจำลอง

ในบทความนี้ได้สร้าง และฝึกแบบจำลองทั้งหมด 3 แบบจำลอง คือ EfficientDet Lite 0, EfficientDet Lite 2 และ EfficientDet Lite 4 จากนั้นจะทำการเลือกแบบจำลองที่ดีที่สุดเพื่อนำไปใช้งานกับแอปพลิเคชัน จึงจำเป็นที่จะต้องมีการประเมินประสิทธิภาพของแบบจำลอง ซึ่งแบ่งออกเป็น 2 หมวดหมู่ คือ ความเร็วและความแม่นยำของแบบจำลอง

3.5.1 การประเมินประสิทธิความเร็วของแบบจำลอง

แบบจำลองแต่ละรุ่นมีความเร็วในการทำงานที่แตกต่างกันโดยจะวัดประสิทธิภาพอยู่ 2 ค่าคือ ระยะเวลาความเร็วในการแปรผลของแบบจำลอง (Inference Time) และ FPS ของกล้องวิดีโอเมื่อนำแบบจำลองไปใช้งานจริงบนสมาร์ทโฟน

3.5.2 การประเมินประสิทธิความแม่นยำของแบบจำลอง

การประเมินประสิทธิภาพความแม่นยำของแบบจำลองสำหรับการจัดหมวดหมู่นั้น ตามปกติมีค่าที่พิจารณาอยู่ 4 ค่า ได้แก่ (1) True Positive (TP) จำนวนที่ทำนายตรงกับข้อมูลจริงในคลาสที่กำลังพิจารณา (2) True Negative (TN) จำนวนที่ทำนายตรงกับข้อมูลจริงในคลาสที่ไม่ได้กำลังพิจารณา (3) False Positive (FP) จำนวนที่ทำนายผิดในคลาสที่กำลังพิจารณา และ (4) False Negative (FN) จำนวนที่ทำนายผิดในคลาสที่ไม่ได้กำลังพิจารณา โดยจะประเมินประสิทธิภาพของแบบจำลอง ด้วยกันทั้งหมด 4 ค่า ได้แก่ Precision, Recall, Accuracy และ F1-score โดยคำนวณได้ดังสมการที่ (1) ถึง สมการที่ (4)

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

สำหรับแบบจำลองแบบหลายคลาส (Multi-Class) นั้น จำเป็นต้องใช้ Confusion Matrix เพื่อมาช่วยในการคำนวณประสิทธิภาพความแม่นยำของแต่ละคลาส โดยแต่ละคลาสสามารถคำนวณค่า Precision ได้จากการนำค่าที่ทำนายตรงกับข้อมูลจริงในคลาสที่กำลังพิจารณา (TP) นำไปหารกับผลรวมของค่าทั้งหมดในแถวของคลาส ที่กำลังพิจารณา (TP + FP) และคำนวณค่า Recall ได้จากการนำค่าที่ทำนายตรงกับข้อมูลจริงในคลาสที่กำลังพิจารณา (TP) นำไปหารกับผลรวมของค่าทั้งหมดในคอลัมน์ของคลาส ที่กำลังพิจารณา (TP + FN) ยกตัวอย่างเช่น ข้อมูลของ Confusion Matrix ดังภาพที่ 4 สามารถคำนวณ Precision ของคลาส A, B และ C ได้ดังสมการที่ (5) ถึง (7) ตามลำดับ

Actual Values

Predicted Values	A	B	C	Precision
A	20	0	0	1.00
B	0	18	2	0.90
C	0	1	19	0.95
Recall	1.00	0.95	0.90	

รูปที่ 4 ตัวอย่าง Confusion Matrix

$$Precision_A = \frac{20}{20 + 0 + 0} = 1.00 \quad (5)$$

$$Precision_B = \frac{18}{0 + 18 + 2} = 0.90 \quad (6)$$

$$Precision_C = \frac{19}{0 + 1 + 19} = 0.95 \quad (7)$$

และสามารถคำนวณ Recall ของคลาส A, B และ C ดังสมการที่ (8) ถึง (10) ตามลำดับ

$$Recall_A = \frac{20}{20 + 0 + 0} = 1.00 \quad (8)$$

$$Recall_B = \frac{18}{0 + 18 + 1} = 0.95 \quad (9)$$

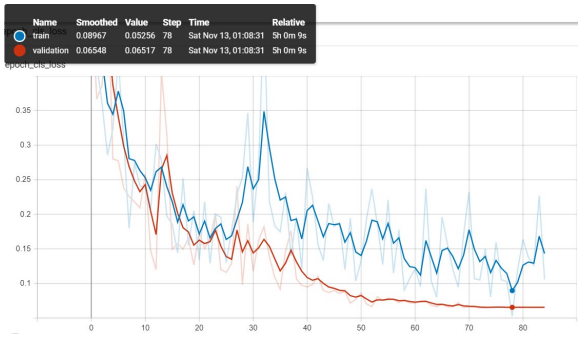
$$Recall_C = \frac{19}{0 + 2 + 19} = 0.90 \quad (10)$$

4. ผลการทดลอง

4.1 ผลการฝึกแบบจำลอง

การฝึกแบบจำลองได้ใช้บริการของ Google Colab ในการฝึก โดยมีหน่วยประมวลผลกลางคือ Intel(R) Xeon(R) หน่วยความจำขนาด 24 GB และมีหน่วยประมวลผลภาพคือ Nvidia K80 / T4/ P100

จากรูปที่ 5 แสดงกราฟค่าการสูญเสียระหว่างการฝึกแบบจำลอง EfficientDet Lite 0 จะเห็นว่าค่าการสูญเสียลงไปต่ำสุดในรอบที่ 78 โดยมีค่าการสูญเสียเท่ากับ 0.05256 จากนั้นจะเห็นว่าหลังจากรอบที่ 78 เป็นไปค่าการสูญเสียจะแกว่งสูงและมีแนวโน้มที่จะเกิด การ Overfit



รูปที่ 5 ค่าความสูญเสียระหว่างการฝึกของแบบจำลอง EfficientDet Lite 0

4.2 ประสิทธิภาพของแบบจำลอง

สำหรับการทดสอบแบบจำลอง ได้นำแบบจำลองที่ทำการฝึกเสร็จเรียบร้อยแล้วมาใช้งานกับแอปพลิเคชัน บนสมาร์ทโฟนรุ่น Oneplus 7 Pro ที่มีหน่วยประมวลผลกลาง Snapdragon 855 หน่วยความจำขนาด 8 GB โดยมีหน่วยประมวลผลภาพ Adreno 640 ทำการตรวจจับนกในภาพทั้งหมด 11 คลาส (Class) คลาสละ 20 ภาพ

จากนั้นนำผลลัพธ์ที่ได้จากการทำนายผลของแบบจำลองมาสร้าง Confusion Matrix และคำนวณประสิทธิภาพของแบบจำลองได้แก่ค่า Precision และ Recall โดยมีผลลัพธ์ประสิทธิภาพของแบบจำลองดังรูปที่ 6

ขั้นตอนต่อไปคือเลือกรูปภาพจากขั้นตอนก่อนหน้านี้ 10 ภาพ เพื่อหาระยะเวลาในการแปรผล (Inference Time) เฉลี่ย และใช้ Flutter Performance Profiling Mode ในการแสดงค่า FPS เฉลี่ยตอนเปิดหน้ากล้อง ได้ผลลัพธ์ดังตารางที่ 3

จากตารางจะเห็นได้ว่าทั้ง 3 แบบจำลอง จำนวน FPS ของกล้องมีค่าเท่ากันคือ 23 FPS เนื่องจากตัวกล้องและแบบจำลองนั้นทำงานคู่ขนานกัน ทำให้ถึงแบบจำลองจะมีขนาดซับซ้อนมากเท่าไหร่ก็จะไม่ส่งผลต่อค่า FPS ของกล้อง แต่เวลาในการแปรผลของแบบจำลอง (Inference Time) นั้นต่างกันมาก ยิ่งเวลาในการแปรผลมาก ก็จะทำให้แบบจำลองทำการตรวจจับหรือตีกรอบตัวนกไม่ทันหรือเกิดการหน่วง (Delay) ได้

Actual Values

Predicted Values: Det 0	PEL	ALE	GRJ	BAO	CAN	NIC	HOP	PEA	GRP	WHI	Not	Precision
PEL	20	0	0	0	0	0	0	0	0	0	0	1.00
ALE	0	18	0	0	0	0	0	0	0	0	1	0.94
GRJ	0	2	19	0	0	0	0	0	0	0	0	0.90
BAO	0	0	1	20	0	0	0	0	0	0	0	0.95
CAN	0	0	0	0	20	0	0	0	0	0	0	1.00
NIC	0	0	0	0	0	20	0	2	0	0	1	0.86
HOP	0	0	0	0	0	0	20	0	0	0	1	0.95
PEA	0	0	0	0	0	0	0	18	0	0	1	0.94
GRP	0	0	0	0	0	0	0	0	17	2	0	0.89
WHI	0	0	0	0	0	0	0	0	3	18	0	0.85
Not	0	0	0	0	0	0	0	0	0	0	16	1.00
Recall	1.00	0.90	0.95	1.00	1.00	1.00	1.00	0.90	0.85	0.90	0.80	
Precision Average						0.94						
Recall Average						0.94						

รูปที่ 6 Confusion Matrix และค่าประสิทธิภาพจากการทดสอบแบบจำลอง EfficientDet Lite 0

ตารางที่ 3 ประสิทธิภาพของแบบจำลอง

รุ่นของแบบจำลอง	ความเร็วของแบบจำลอง		ความแม่นยำของแบบจำลอง			
	Inferencce Time (ms)	FPS ของกล้อง	Precision	Recall	Accuracy	F1-score
EfficientDet Lite 0	62.3	23	0.94	0.94	0.99	0.94
EfficientDet Lite 2	153.7	23	0.94	0.94	0.99	0.94
EfficientDet Lite 4	574.3	23	0.95	0.95	0.99	0.95

ในขณะที่แบบจำลองกำลังทำการแปรผลอยู่นั้นแบบจำลองจะทำการข้ามเฟรม (Skip Frame) ในระหว่างช่วงเวลานั้นทิ้งไปแล้วจะนำเฟรมหลังจากที่แบบจำลองแปรผลเสร็จ เข้าไปแปรผลในแบบจำลองต่อ ทำให้จำนวน FPS ที่แท้จริงของแบบจำลองหรือ FPS ของกรอบที่ขึ้นมาจริงนั้น ไม่เท่ากับกล้องวิดีโอ โดย FPS ที่ได้จากการประมวลผลตีกรอบของแต่ละรุ่นแบบจำลอง แสดงดังสมการที่ (11) ถึงสมการที่ (13)

$$FPS_{EfficientDet\ Lite\ 0} = \frac{1}{62.3} \times 1000 = 16.05 \quad (11)$$

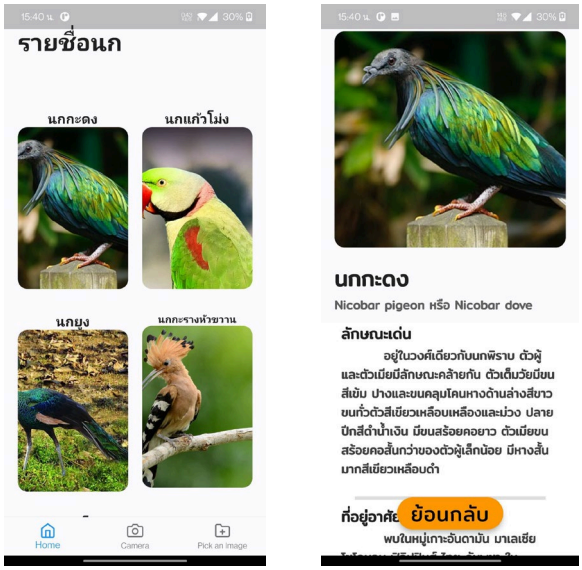
$$FPS_{EfficientDet\ Lite\ 2} = \frac{1}{152.7} \times 1000 = 6.54 \quad (12)$$

$$FPS_{EfficientDet\ Lite\ 4} = \frac{1}{573.6} \times 1000 = 1.74 \quad (13)$$

เมื่อเทียบ Inference Time ของ EfficientDet Lite 0 กับ EfficientDet Lite 2 และ EfficientDet Lite 4 พบว่า EfficientDet Lite 0 นั้นเร็วกว่าถึง 3 และ 10 เท่าตามลำดับ เนื่องจาก EfficientDet Lite 0 มีจำนวนของชั้นและจำนวนพารามิเตอร์น้อยที่สุดในแบบจำลอง EfficientDet Lite ด้วยกัน ในขณะที่ความแม่นยำนั้นมีค่าใกล้เคียงกันมาก ดังนั้น EfficientDet Lite 0 จึงเหมาะสมมากที่สุด เพราะมีความเร็วและความแม่นยำที่ค่อนข้างสูงเพียงพอ

4.3 การใช้งานแอปพลิเคชัน

เมื่อเปิดใช้งานแอปพลิเคชันครั้งแรกจะมีหน้าฝึกสอนการใช้งานให้ผู้ใช้ได้ศึกษา เพื่อเพิ่มความคล่องแคล่วในการใช้งาน และจากนั้นจะเข้าสู่หน้ารายชื่อนก ผู้ใช้จะสามารถดูสายพันธุ์นกทั้งหมดได้ ดังรูปที่ 7(ก) โดยจากหน้านี้ผู้ใช้สามารถแตะที่รูปของนกแต่ละสายพันธุ์ เพื่อดูรายละเอียดข้อมูลของนกได้ ดังแสดงในรูปที่ 7(ข)



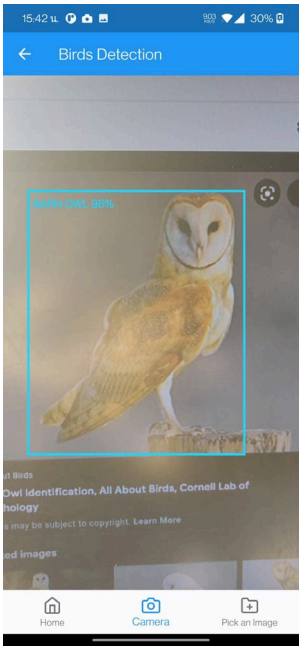
(ก) (ข)

รูปที่ 7 หน้าแอปพลิเคชัน (ก) หน้ารายชื่อนก และ (ข) หน้ารายละเอียดข้อมูลนก

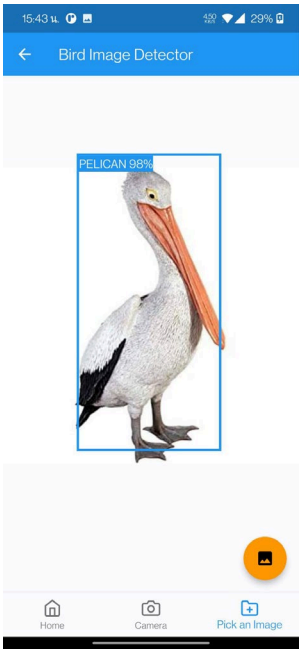
รูปที่ 8 แถบนำทางของแอปพลิเคชัน

ทางด้านล่างของแอปพลิเคชันจะมีแถบนำทาง (Navigation Bar) ดังรูปที่ 8 เพื่อนำทางไปยังหน้าต่าง ๆ ปุ่มแรกไปยังหน้ารายชื่อนก (Home) ปุ่มที่ 2 นำทางไปยังหน้ากล้อง (Camera) และปุ่มที่ 3 นำทางไปยังหน้าเลือกรูปจากคลัง (Pick an Image)

หน้ากล้องเป็นหน้าที่ใช้สำหรับตรวจจับและตรวจสอบสายพันธุ์ของนก เมื่อผู้ใช้นำกล้องไปส่องนก หากแอปพลิเคชันเจอตัวนก ก็จะทำการตีกรอบและระบุชื่อสายพันธุ์ของนกที่เจอ ดังรูปที่ 9 เมื่อผู้ใช้งานสัมผัสหรือกดไปที่กรอบจะเข้าสู่หน้าข้อมูลของนกตัวนั้น



รูปที่ 9 หน้ากล้อง

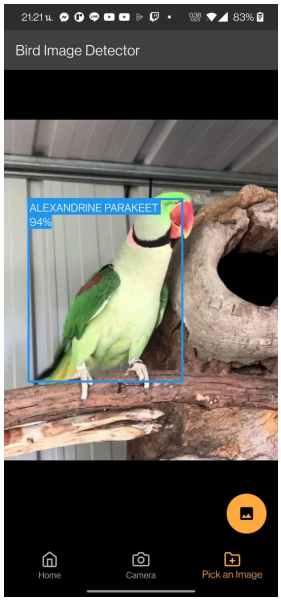


รูปที่ 10 หน้าเลือกจากคลังรูปภาพ

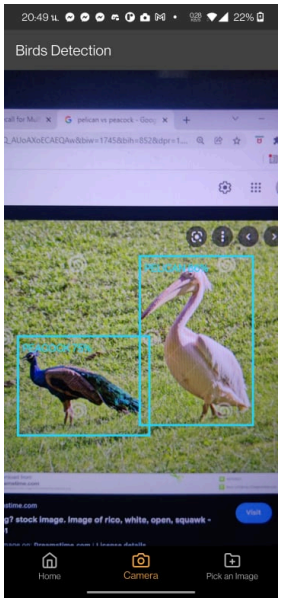
หน้าเลือกจากคลังรูปภาพ เมื่อเข้าสู่หน้านี้ถ้าต้องการจะเลือกรูปภาพต้องกดปุ่มที่มุมล่างขวา เพื่อเลือกรูปจากคลังรูปภาพ เมื่อแอปพลิเคชันเจอนกในรูปภาพก็จะทำการตีกรอบและระบุชื่อพันธ์ของนกที่เจอ ดังรูปที่ 10 และผู้ใช้สามารถสัมผัสหรือกดไปที่กรอบเพื่อเข้าสู่หน้าข้อมูลของนกตัวนั้น

4.4 ตัวอย่างผลลัพธ์จากการจับภาพ

ตัวอย่างผลลัพธ์การตรวจจับภาพนกผ่านหน้ากล้อง และเลือกรูปจากคลัง แสดงผลลัพธ์ดังรูปที่ 11

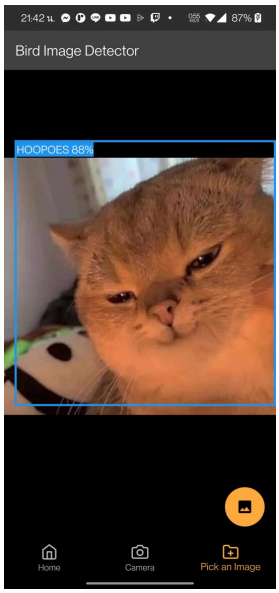


(ก)



(ข)

(ค)



(ง)

รูปที่ 11 ตัวอย่างผลลัพธ์การตรวจจับนกผ่านกล้อง
(ก) ผลลัพธ์ถูกต้องในกรณีมีนก 1 ตัว
(ข) ผลลัพธ์ถูกต้องในกรณีมีนกมากกว่า 1 ตัว
(ค) ผลลัพธ์ถูกต้องในการจับภาพที่ไม่ใช่นก
และ (ง) ผลลัพธ์ผิดพลาดตรวจจับแมวเป็นนก

5. สรุปผลการทดลอง

บทความนี้เสนอการออกแบบและพัฒนาโมบายแอปพลิเคชันสำหรับตรวจสอบและจำแนกสายพันธุ์นกด้วยวิธีการเรียนรู้เชิงลึก ผู้ใช้สามารถใช้งานแอปพลิเคชันเพื่อทำการตรวจจับและตรวจสอบสายพันธุ์ของนก และสามารถศึกษาข้อมูลของนกที่กำลังทำการตรวจสอบอยู่ได้ แอปพลิเคชันพัฒนาด้วยภาษา Dart บน Flutter Framework และใช้ไลบรารี TFLite Model Maker ในการสร้างแบบจำลอง EfficientDet Lite ในการค้นหาและจดจำรูปภาพนก และจากผลการทดลองจึงได้เลือกแบบจำลอง EfficientDet Lite 0 สำหรับนำไปใช้งานจริงกับแอปพลิเคชัน โดยแบบจำลองที่พัฒนาขึ้นสามารถจดจำสายพันธุ์ของนกได้ทั้งหมด 10 สายพันธุ์ และวัตถุที่ไม่ใช่นก ซึ่งรวมทั้งหมดเป็น 11 คลาส จากการสอนด้วยภาพคลาสละ 500 - 1000 ภาพ จำนวนทั้งหมด 78 รอบ แบบจำลองมีประสิทธิภาพด้านความแม่นยำ Precision, Recall, Accuracy และ F1-score เท่ากับ 0.94, 0.94, 0.99 และ 0.94 ตามลำดับ และจากการทดสอบบนโทรศัพท์มือถือจริงพบกว่าเวลาในการแปรผลของแบบจำลองอยู่ที่ 62.30 ms และไม่ได้มีผลกระทบต่อจำนวนเฟรมของวิดีโอ โดยมีค่า FPS อยู่ที่ 23 FPS

เอกสารอ้างอิง

[1] J. Law, "Why we need birds (far more than they need us)", *BirdLife International,* 2021. [Online]. Available: https://www.birdlife.org/news/2019/01/04/why-we-need-birds-far-more-than-they-need-us/. [Accessed 15 November 2021].

[2] P. Chatayapha. “Technology and early childhood in 21st century,” *Journal of Graduate Studies Valaya Alongkorn Rajabhat University*, vol. 14, no. 3, 2020.

[3] "Cross-platform mobile frameworks used by global developers 2021 | Statista", Statista, 2021. [Online]. Available: https://www.statista.com/statistics/869224/worldwide-software-developer-workinghours/?fbclid=IwAR1QrmP_JDP_yMdGu_C56ami-xPuUniTqCv-D0W1Eyw8cGJNrVK-7ZmfAaU. [Accessed 15 November 2021].

[4] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” *Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 160, no. 1, pp. 3-24, 2007.

[5] M. Tabak, et al, “Machine learning to classify animal species in camera trap images: Applications in ecology,” *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 585-590, 2018.

[6] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra and J. M. Z. Maningo, “Object detection using convolutional neural networks,” *TENCON 2018 - 2018 IEEE Region 10 Conference*, pp. 2023-2027, 2018.

[7] “Flutter - Beautiful native apps in record Time”, Flutter.dev, 2021. [Online]. Available: https://flutter.dev/. [Accessed 15 November 2021].

[8] “TensorFlow”, TensorFlow, 2015. [Online]. Available: https://www.tensorflow.org/. [Accessed 15 November 2021].

[9] “TensorFlowLite”, TensorFlow, 2018. [Online]. Available: https://www.tensorflow.org/lite/guide. [Accessed 15 November 2021].

[10] S. Albawi, T. A. Mohammed and S. Al-Zawi, “Understanding of a convolutional neural network”. *In 2017 International Conference on Engineering and Technology (ICET)*, pp. 1-6, 2017.

[11] M. Tan, R. Pang and Q. V. Le, "EfficientDet: scalable and efficient object detection", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.1-10, 2020.

[12] M. Tan and Q. V. Le, "EfficientNet: rethinking model scaling for convolutional neural networks", *International Conference on Machine Learning*, pp.1-11, 2019.

[13] "TensorFlow Lite Model Maker", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/lite/guide/model_maker. [Accessed 15 November 2021].

[14] "Object Detection with TensorFlow Lite Model Maker", TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/lite/tutorials/model_maker_object_detection. [Accessed 15 November 2021].

[15] M. X. He and P. Hao, "Robust Automatic Recognition of Chinese License Plates in Natural Scenes," in *IEEE Access*, vol. 8, pp. 173804-173814, 2020.

[16] "The Zoological Park Organization of Thailand", Zoothailand.org, 2021. [Online]. Available: http://zoothailand.org/animal_more.php. [Accessed 15 November 2021].