Available online at http://scik.org

Commun. Math. Biol. Neurosci. 2023, 2023:136

https://doi.org/10.28919/cmbn/8338

ISSN: 2052-2541

DETECTION OF ENDANGERED SPECIES PROBOSCIS MONKEY USING COMPUTER VISION TECHNIQUE ON LOW COMPUTE DEVICE

JUANRICO ALVARO*, GEDE PUTRA KUSUMA

Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia

Abstract: The uncommon and endemic proboscis monkey, also known as Nasalis larvatus, is a unique member of Indonesia's diverse animal population that is in the list of endangered species. With the development of technology, the ability to detect this species can be proven useful to further look out this species on the dense forest, their inhabitant. This research paper presents the dataset and compares the performance and computational load to run on a low computation device from three state of the art object detection models: You Only Look Once version 7 (YOLOv7), Single Shot Multibox Detection (SSD) and Faster Region Convolutional Neural Network (Faster R-CNN). All the models are evaluated using low computing device with their respective backbone MobileNetV2 for SSD and tiny-version for YOLOv7. According to the report, Faster R-CNN give the best accuracy of 89.16 and 53.46 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ on test dataset with the slowest time among other models with FPS of 1.26 s and huge memory usage. In the other hand, SSD give the fastest detection speed with FPS of 0.31s and the lightest model to run in low computing devices with a decent average precision of 81.35 and 42.99 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ respectively. Hence, the best proposed model for real-time detection is SSD with MobileNetV2 as the backbone.

Keywords: object detection; models evaluation; deep learning; computational cost; low computing devices.

2020 AMS Subject Classification: 68T45, 92D50.

1. INTRODUCTION

Indonesia, a nation renowned for its rich biodiversity, is home to a remarkable primate species known as the proboscis monkey, scientifically identified as Nasalis larvatus. However, the

*Corresponding author

E-mail address: juanrico.alvaro@binus.ac.id

Received November 13, 2023

population of proboscis monkeys faces a significant threat [1], with estimates suggesting their numbers may have dwindled to just around 20,000 individuals [2]. This concerning decline in population, coupled with their rare and endemic status in Borneo, has led to their classification as endangered by the IUCN Red List. Efforts to protect this unique species are challenged by their scattered presence in dense forests, making it locate them a necessity for conservation and research purposes. Traditional methods of observation are costly, logistically complex, and often disruptive to the wildlife, posing risks to human observers. To overcome these challenges, ecologists are increasingly turning to automation and technology, specifically computer vision and object detection techniques, to identify and locate organisms in their natural habitats.

Research within the field of proboscis monkey conservation has been instrumental in recognizing the need for technological breakthroughs. A recent paper, honored with the 20th Miyadi Award by the Ecological Society of Japan [3], underscores the difficulties of monitoring these elusive primates in their dense and swampy forest habitats. It emphasizes the pressing need for advanced machine learning techniques capable of overcoming the harsh conditions encountered during observation.

The primary objective of this paper is to introduce a system that aids in locating and monitoring proboscis monkey populations while significantly reducing the associated costs and labor. This system leverages computer vision and object detection to activate cameras only when proboscis monkeys are within the camera's range, thereby optimizing resource allocation.

Recent research in animal identification, such as Vidal et al.'s work [4], offers valuable insights into computer vision techniques and their applications in identifying animals with distinctive visual characteristics. Despite significant progress, challenges remain, especially in the identification of unmarked and featureless animals in the wild.

This paper bridges the gap by focusing on proboscis monkey detection and aims to determine the most effective computer vision model for early detection alerts. The selected model will prioritize lightweight and fast performance, as it will operate in real-time on low computing devices. The three leading object detection models will be assessed: Single Shot Multibox Detection (SSD) with MobileNetV2 backbone, You Only Look Once version 7 (YOLOv7) on tiny-version, and Faster R-CNN.

This research aims to contribute to the effective monitoring and conservation of the proboscis monkey population, offering an early warning system to prevent their potential extinction. Our

work will primarily involve the use of a dataset of proboscis monkey photos, manually labeled using Computer Vision Annotation Tool (CVAT). The dataset will simulate challenging real-world scenarios, including complex backgrounds and visual disruptions typical of the proboscis monkey's natural habitat.

2. RELATED WORKS

The introduction of Faster R-CNN has significantly improved object detection accuracy, as demonstrated in the study titled "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" [5]. This method incorporates a Region Proposal Network (RPN) into Fast R-CNN, enhancing its performance. The RPN predicts object boundaries and scores concurrently within a fully convolutional network. The paper tested this model on the PASCAL VOC and MS COCO datasets, using the VGG-16 network alongside the Fast-R-CNN Detector. When comparing Fast R-CNN and Faster R-CNN on the MS COCO dataset with VGG-16, the results were 39.3 and 42.7 in mAP at 0.5 IoU, respectively. Further training and testing of the Faster R-CNN model using both MS COCO and PASCAL VOC 07 and 12 datasets resulted in an impressive 75.9 mAP.

Pal et al. presented a paper titled "Deep Learning in Multi-Object Detection and Tracking: State of the Art" [6], which explored deep learning techniques in the context of object detection and tracking. The study examined various detectors and trackers, such as Faster R-CNN, Mask R-CNN, YOLO [7], YOLOv2, YOLOv3, SSD, DSSD, FPN, R-FCN, and DCN. These models were evaluated on diverse datasets, including ImageNet, PASCAL VOC [8], MS COCO [9], MOT2015, and MOT2016, using multiple evaluation metrics. The evaluation metrics used included mean average precision (mAP), Multi-Object Tracking Accuracy (MOTA), Identity Switches (IDS), Multi-Object Tracking Precision (MOTP), Mostly Tracked Targets (MT), Mostly Lost Targets (ML), and Speed (frames per second, fps). The NAS-FPN method with AmoebaNet as the backbone achieved the best detection results with mAP of 48.0 on the MS COCO test-dev dataset. Additionally, using R-FCN with training data from VOC 07 + VOC 12 + MS COCO and the ResNet101 backbone resulted in mAP of 85.0 on the PASCAL VOC dataset.

The paper titled "Two Layers Machine Learning Architecture for Animal Classification Using HOG and LBP" [10] introduces a system designed to automatically identify animal types. The primary objective is to detect animals, particularly their faces and side views. The system utilizes

a two-layered architecture. In the first layer, it employs Histogram of Oriented Gradients (HOG) for feature extraction, Principal Component Analysis (PCA) for dimensionality reduction, and Support Vector Machine (SVM) for classification. The second layer uses Local Binary Patterns (LBP), a binary-based texture descriptor, and employs gradient boosting as another classifier for making final predictions. The proposed model achieves an accuracy of 88.59% in the first layer, which improves to 95.15% after processing through the second layer. The experiments involved 32,490 correctly identified animal photos using the Pascal VOC12c and KTH datasets.

In a study focusing on implementing machine learning in camera traps [11], researchers adopted an edge machine learning approach to incorporate Convolutional Neural Networks (CNNs) into their model, utilizing TensorFlow and Google Colab for training. The dataset, comprising 4,845 annotated images, was narrowed down to 4,008 images from camera traps. After resizing and cropping all images to 400 x 400 pixels, the researchers used TensorFlow's Object Detection API for training and pre-trained the model on the COCO dataset. Evaluation was carried out using PASCAL VOC, with the proposed object tracking employing a tubelet implementation. The model choices included Inception v2, MobileNet v2, quantized MobileNet v2, and lite MobileNet v2. Notably, MobileNet v2 exhibited the best performance, achieving efficient processing times (0.214s on a Raspberry Pi) and high precision (0.86), making it the ultimate model choice due to its minimal fragmentation of events, low false positives, and smaller file size compared to Inception. These qualities align with the intended platform's usage.

The authors [12] present a deep convolutional neural network (CNN) approach that provides a fully automated pipeline for face detection, tracking, and recognition of wild chimpanzees from long-term video records. The dataset yielded 10 million face images from 23 individuals over 50 hours of footage. also successfully obtained an overall accuracy of 92.5% for identity recognition and 96.2% for sex recognition. The most important thing is that the detection part using a deep CNN SSD achieved an average precision of 81%.

The next paper [13], presents a five-component detection pipeline designed for computer vision-based animal recognition. The paper proposes a unique classification system based on ILSVRC [14] classification, employing YOLO version 1 for annotation localization and a classification network architecture similar to image classification. The dataset, named WILD, comprises images from biologists, wildlife rangers, citizen scientists, and conservationists, offering rare detection scenarios not found in publicly available computer vision datasets. The paper reports positive

results, with the worst-performing species achieving a ROC area-under-the-curve (AUC) of 96.33%, while the best-performing species, the whale fluke, attains nearly perfect AUC at 99.94%. Annotation localization exhibits a mean average precision (mAP) of 90.62% and improved recall across all species. For annotation classification, an overall accuracy of 61.71% is achieved across 42 distinct categories. The Annotation of Interest (AoI) classifier effectively reduces data processing needs by over 50% while incurring only 164 missed positive AoIs.

A study [15] also focused on monitoring and analyzing wildlife in natural settings using Deep Convolutional Neural Networks (DCNN) for animal identification based on camera-trap imagery. The paper proposes utilizing a self-taught DCNN as a feature extractor for this purpose. In addition, various advanced machine learning techniques, including Support Vector Machine, K-Nearest Neighbor, and ensemble classifiers with individual variations, are employed for data classification. The study achieves a high level of accuracy, with the best results obtained using the DCNN as a detector and feature extractor in conjunction with Weighted K-Nearest Neighbor (KNN) as the classifier, reaching a maximum accuracy of 91.4%.

In the study on automation using deep learning for animal identification, counting, and behavioral description from camera trap data [16], the researchers focused on extracting up-to-date and accurate animal data. They used the Serengeti dataset, comprising 3.2 million images of animals from 48 species, with 1.4 million training images and 105,000 for testing. Various neural network architectures, including AlexNet, NiN [17], VCG [18], GoogLeNet [19], ResNet with different layers, and an ensemble model, were employed. Exceptional accuracy was achieved, with each method yielding a minimum accuracy of 95%. VCG with 22 layers performed the best, while AlexNet had the lowest accuracy at 95.8% for animal detection. For species identification, the ensemble model achieved 94.9% top-1 accuracy and 99.1% top-5 accuracy. The best single model was ResNet-152 with 93.8% top-1 and 98.8% top-5 accuracy. Regarding animal counting, the ensemble method achieved a 63.1% top-1 accuracy, with 84.7% of predictions falling within ±1 count. The best single model for counting was ResNet-152, with a 62.8% top-1 accuracy and 83.6% of predictions within ±1 count. For additional attributes like animal behavior, volunteers were used to provide labels for predictions exceeding 50% confidence, matching them with the ground truth data.

In their research on animal recognition and tracking for intrusion detection, the authors of [20] introduced an innovative system using computer vision that efficiently and reliably detects wild

animals. They employed the YOLO object detection method and utilized a dataset derived from publicly available wild animal videos. The system was trained to recognize six different species, including humans, namely elephants, zebras, giraffes, lions, and cheetahs, with images resized to 448 x 448 dimensions. The model was trained with a learning rate of 0.001, a decay rate of 0.995, and a momentum of 0.9, converging at the 135th epoch. With this configuration and dataset, the model achieved an average accuracy of 98.8% for animal detection and an impressive 99.8% for human detection.

In a study by [21], deep learning was employed for automatic analysis of ecological camera trap images. Faster R-CNN and YOLO v2.0 were proposed for animal identification, quantification, and localization, utilizing two datasets: Reconyx Camera Trap (RCT) with 946 labeled images containing bounding boxes and Gold Standard Snapshot Serengeti (GSSS) with 4,432 labeled images for classification without bounding boxes. The architecture used for both methods was ResNet-101. Evaluation was based on Intersection over Union (IoU) and accuracy, with Faster R-CNN achieving the highest accuracy and IoU for RCT (93.0% and 0.80) and GSSS (76.7% and 0.72).

A paper [22], introduces a novel annotated dataset featuring six underwater animal species with bounding boxes: big fish, small fish, starfish, shrimps, jellyfish, and crabs. YOLOv2 and YOLOv3 object detection methods were employed. The dataset includes various annotations and video occurrences for each species. YOLOv2 was fine-tuned using fish datasets from the NOAA Fisheries Strategic Initiative and ImageNet pre-training, while YOLOv3 was pre-trained using the Open Images dataset. Evaluation metrics included Average Precision (AP), mAP, AP50 (IoU threshold of 0.5), and IoU-based bounding box filtering. In the Open Images categories, fine-tuned YOLOv3 achieved the highest AP (0.3947) and AP50 (0.8458), while in the proposed dataset 'Brackish,' it delivered the highest AP (0.3893) and AP50 (0.8327).

In paper [23], the authors employed computer vision and deep learning for healthy apple detection. YOLOv3 and SSD models are proposed. The dataset contains 452 apple images, equally split between healthy and defective, and the test set has 140 images with the same split. Evaluation metrics include Intersection over Union (IoU) and average precision (AP) at IoU thresholds of 0.31, 0.51, and 0.71. YOLOv3 outperforms, achieving 0.7469 mAP at 0.31 IoU, 0.7430 mAP at 0.51 IoU, and 0.6736 mAP at 0.71 IoU.

A study comparing YOLOv3, Faster R-CNN, and SSD for real-time pill identification [24]

found that although Faster R-CNN had a high mean average precision (MAP) of 87.69%, its detection speed (7 FPS) was insufficient for real-time use. SSD performed reasonably well with an MAP of 82.41% and a speed of 32 FPS. YOLOv3, with an MAP of 80.17%, showed promise by achieving a faster speed (51 FPS). In a similar comparative research on these models for traffic sign detection [25], YOLOv2 led with an accuracy of 77.89%, followed by Faster R-CNN and SSD with 74.68% and 68.35%, respectively. YOLOv2 also had the fastest detection speed, with FPS scores of 44, followed by SSD (21 FPS), and Faster R-CNN (7 FPS).

In a 2021 study [26], the goal was to detect and classify mammals in images, addressing challenges like partial visibility, varying lighting, and unclear pictures. The approach used CNNs combined with a pre-trained SSD and MobileNet v1 architecture. The model was pre-trained on the Common Objects in Context (COCO) dataset, built for object detection and classification. The dataset included 2000 unique photos, with 1000 for each of the three categories: animals, mammals, and none. Remarkably, with only 20 epochs, the model achieved a high 98.7% detection accuracy.

Researchers conducted an animal identification study involving various animal types in [27]. They employed a unique approach by combining two state-of-the-art models, YOLOv2 and SSD. The Snapshot Serengeti dataset, comprising 1.9 million capture events with 3.2 million photos of 48 species, was used for evaluation. However, only 15,660 images featuring six species, including elephants, Grant gazelles, Thomson's gazelles, giraffes, ostriches, and zebras, were selected from the dataset. YOLOv2 and SSD models were tuned based on their performance on MS COCO, PASCAL VOC 2007, and PASCAL VOC 2012. The combined model was implemented using Dynamic Belief Fusion as proposed by [28]. The evaluation, measured using Mean Average Precision (mAP), yielded simultaneous results of 0.55, 0.67, and 0.73 for YOLOv2, SSD, and the combined model, respectively.

Following previous research, it's evident that YOLO, Faster R-CNN, and SSD are among the top-performing object detection models, delivering highly accurate predictions. YOLOv7, the latest YOLO variant, stands out as the most effective. This study aims to compare YOLOv7 with Faster R-CNN, known for its unique CNN-based approach, and SSD, which excels in real-time object detection. SSD's adaptability comes from its customizable backbone, making it an ideal choice for resource-constrained environments, utilizing MobileNetV2 as the backbone. The comparison assesses not only prediction accuracy but also model compactness, essential for deployment in resource-limited, harsh environments. Real-time detection performance, processing

speed, and computational load will be evaluated, aiming to identify the best all-around model.

3. PROPOSED METHODS

3.1. Dataset

The dataset used for this paper consisted of images containing proboscis monkeys. The data was crawled or gathered from online sources such as Google and Bing. The pictures of the proboscis monkey itself had several angles from the front side and several from the side as well. Since the application was intended for use in a forest, it was best to have several images that had some leaves or any other forest environment that might obstruct the camera. The dataset gathered was 1287 images and divided into three categories: the training dataset, validation dataset, and testing dataset, each comprising 772, 258, and 257 images respectively. All the images in the dataset were cropped in a one-to-one ratio with the size of 512x512. The example of the dataset can be seen on FIGURE 3.1.

FIGURE 3.1 Proboscis Monkey dataset example

To train the object detector, bounding box annotations were needed. Each object that had a proboscis monkey(the object that the object detector needed to detect) required a label by drawing a box around it. Many labeling tools greatly helped in creating these annotation boxes. The labeling tool that this paper proceeded with was the computer vision annotation tool (CVAT). CVAT was an open-source web-based tool for annotating digital images and videos. The goal of CVAT was to make the process of producing high-quality training data for computer vision models simpler and faster. For tasks like object identification, segmentation, classification, and tracking, it could annotate photos and movies. The application offered a variety of annotation tools and capabilities, including polygon, rectangle, and polyline annotation, 3D cuboid annotation, and more. It also enabled many users to collaborate on the same project. In addition to integrating with well-liked machine learning frameworks like TensorFlow and PyTorch, CVAT supported a wide range of file types.

3.2. Model

Each of the model were fine-tuned with the optimal tuned for MS COCO dataset and have the augmentation of random crop and random horizontal flip. Each of the model trained on batch size of 4 with the image width of 320. The reason of the image downscaling and lower batch size is because of the limited computing loads while training. The epoch of the training is 100 epoch or in another word around 20000 step after calculating the training dataset and the batch size. The training done in Google Colab with the free T4 GPU. The three models were proposed and implemented on the proboscis monkey dataset namely, SSD MobileNetv2, YOLOv7-tiny, and Faster R-CNN with ResNet50.

3.2.1. Single Shot Multibox Detector (SSD) with MobileNetV2 Backbone

This model is an optimal model used for real-time object detection problems. The reason for this is the way the network only predicts it on a single forward pass through the network, eliminating the need for complex post-processing steps. SSD [29] successfully strikes a balance between speed and accuracy. It may efficiently collect the object information and produce competitive detection performance compared to other approaches by utilizing multi-scale feature maps and anchor boxes. The model offers flexibility and a strong structural foundation make it ideal for customization to fit low-powered computing devices. MobileNetV2 [30], a backbone with a lower size that is suitable for a low computing device, serves as the foundation for the model that was selected.

3.2.2. You Only Look Once version 7 on tiny version (YOLOv7-tiny version)

The next model will be YOLOv7 [31] where the author presents Extended Efficient layer aggregation as the final layer aggregation. The next technique they presented is model scaling techniques that made the network able to identify a smaller object or a huge difference on the object scale. Re-parameterization planning has also been done on this model. The last thing they improve is the auxiliary head that able to do a coarse to fine detection. YOLOv7 has a huge improvement compared to its predecessor but it does have many more layers hence a small computational load may become the obstacle. The solution for this is to apply the Tiny version of this model to be able to be deployed on edge devices.

3.2.3. Faster Region Convolutional Neural Network (Faster R-CNN)

The last model is Faster R-CNN [5] is a popular and influential object detection framework. This model introduced Region Proposal Network which is a fully convolutional network that takes an image as input and outputs a set of object proposals along with their corresponding objectness scores. This is the reason why the model was able to reach a huge improvement on the calculating speed as well as the accuracy. The backbone will be implemented will be ResNet50 which is a classic neural network using the Residual neural networks that works by utilizing skip connections, or short-cuts to jump over some layers.

3.3. Evaluation of Detection Performance

The performance measure is done when the research is in the testing phase. Object detection itself has a unique measurement. In computer vision and object detection, average precision (AP) is a frequently used performance metric. It gauges the model's overall detection precision, or how effectively the model can precisely identify items in an image. Object detection aims to identify and categorize items in an image or video. The model creates bounding boxes around the objects in the image and labels each thing it detects with a class and a confidence score. The model's accuracy in identifying and categorizing objects is then used to calculate the average precision metric, which accounts for both the precision and recall of the detections.

The implementation of Intersection over Union (IoU) is needed in AP calculation. A measurement called IoU is frequently employed in the fields of computer vision and object detection to assess how similar two sets of data are. It stands for the product of the size of the predicted bounding box divided by the size of the ground truth bounding box this calculation can be seen in FIGURE 3.2. IoU is a statistic used to assess how well a model performs in identifying

items in an image. Since the projected bounding box and the actual bounding box overlap more, a prediction with a higher IoU score is more accurate.

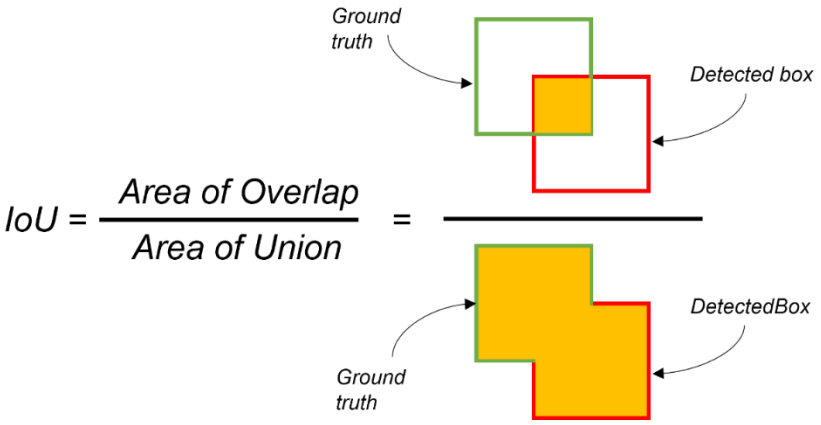

FIGURE 3.2 Intersection over union (IoU).

IoU will be served as the threshold of the AP calculation. The value of the threshold in this paper will be decided into two, AP with an IoU threshold of 0.5 and range of 0.5 to 0.95. The following metrics will be calculated based on the IoU, threshold, and class labels of the ground truth and predicted bounding boxes:

- True Positive (TP): condition when the bounding box intersects with IoU with ground truth above the threshold.
- False Positive (FP): condition when the bounding box intersects with IoU with ground truth below the threshold.
- False Negative (FN): a condition when the model failed to produce a bounding box even though the ground truth is there.
- True Negative (TN): The model was accurate in that it did not foresee a bounding box. This relates to the background, or the region devoid of bounding boxes, and is not considered when determining the final measurements.

After the three metrics are obtained, AP can then be calculated by extracting the precision and recall of the model. This precision-recall will be used to draw a Precision-Recall Curve. AP then summarizes the PR Curve to one scalar value. A high average precision score indicates that the model can accurately detect objects, with a low rate of false positives and false negatives. In contrast, a low average precision score indicates that the model is not accurately detecting objects, with a high rate of false positives or false negatives. With a low rate of false positives and false negatives, a model may accurately detect objects if it receives a high average accuracy score. A

low average precision score, on the other hand, shows that the model has a high rate of false positives or false negatives and does not reliably recognize objects. The formula can be seen in Formula (3.1), (3.2), and (3.3).

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$Average\ Precision\ (AP) = \int_{r=0}^{1} p(r)dr \tag{3.3}$$

3.4. Evaluation of Computational Loads on low computing device.

The goal of this paper was to determine which proposed model gave the highest performance in terms of detection accuracy and performance when running on low computing load devices. For the measurement of detection accuracy, Intersection over Union (IoU) was used on the average precision (AP), which was divided by IoU thresholds of 50 and the average of each 5% step from 0.5 to 0.95. Performance Measure for Object Detection.

Since detection required good performance on edge devices, other performance measurements were needed. Edge devices like smartphones had low computing load; hence, the computing load of the models was measured, along with the speed of calculation. The size of the application also posed a hindrance to the model's effectiveness, so the size of it was thoroughly compared.

The first measurement for computational load was the average detection speed. The test itself was calculated in frames per second (FPS), by dividing the total number of frames processed by the total time taken for detection, as seen in Formula (3.4). This was tested on batches of test data, and the average of the batches was calculated as stated in Formula (3.5), with FPS in the formula representing the FPS of the batches.

$$FPS = \frac{number\ of\ frames\ processed}{time\ taken} \tag{3.4}$$

$$Average\ of\ FPS = \frac{(FPS_1 + FPS_2 + \cdots + FPS_n)}{n} \tag{3.5}$$

The second measurement was the average peak of RAM and CPU usage. To proceed with testing, batches of test files were processed for RAM and CPU usage was monitored on each batch. The peak RAM and CPU usage for each batch was noted, and these peaks were used to calculate the average RAM usage as described in Formula (3.6), with RAM in the formula representing the RAM peak usage of the batches, and the average CPU usage as described in Formula (3.7), with

CPU in the formula representing the CPU peak usage of the batches.

$$Average\ of\ RAM = \frac{(RAM_1 + RAM_2 + \cdots + RAM_n)}{n} \quad (3.6)$$

$$Average\ of\ CPU = \frac{(CPU_1 + CPU_2 + \cdots + CPU_n)}{n} \quad (3.7)$$

Each of the models was trained first, and the resulting trained model was used as the detection model for the proboscis monkey dataset. The measurements, including average detection speed, average peak of random access memory (RAM) usage, and average peak of central processing unit (CPU) usage, were performed ten times for each testing as test batches, and the average of the testing from those batches was calculated. The mentioned tests were tested on several low computing devices that is listed in Table 1. Each of the model then deployed on low computational devices listed in Table 1 by using the inference method of each framework. Since Jetson Nano run on Linux, the low specification PC are also adjusted by running on WSL. Both system computational load then recorded by time command.

TABLE 1. Hardware Testing Specification.

Device	CPU	GPU	RAM size
Jetson Nano	Quad-core ARM A57 @ 1.43 GHz	128-core Maxwell	4 GB 64-bit LPDDR4
PC	Intel Core i7-4790 @ 3.60GHz (8-core)	-	8 GB of DDR3

4. RESULT AND DISCUSSION

4.1. Evaluation Result of Detection Performance.

The first model to be evaluated is YOLOv7. This model trained using PyTorch library and using the YOLOv7-tiny pre-trained weight on MsCOCO dataset as the transfer learning. The graph of training and validation of each category can be seen in FIGURE 4.1. The graph of training and validation give a visualization of how the model is already converging in 100 epoch according to the mAP graph. Since the model only trained on one class, the classification graph not showing graphical line.

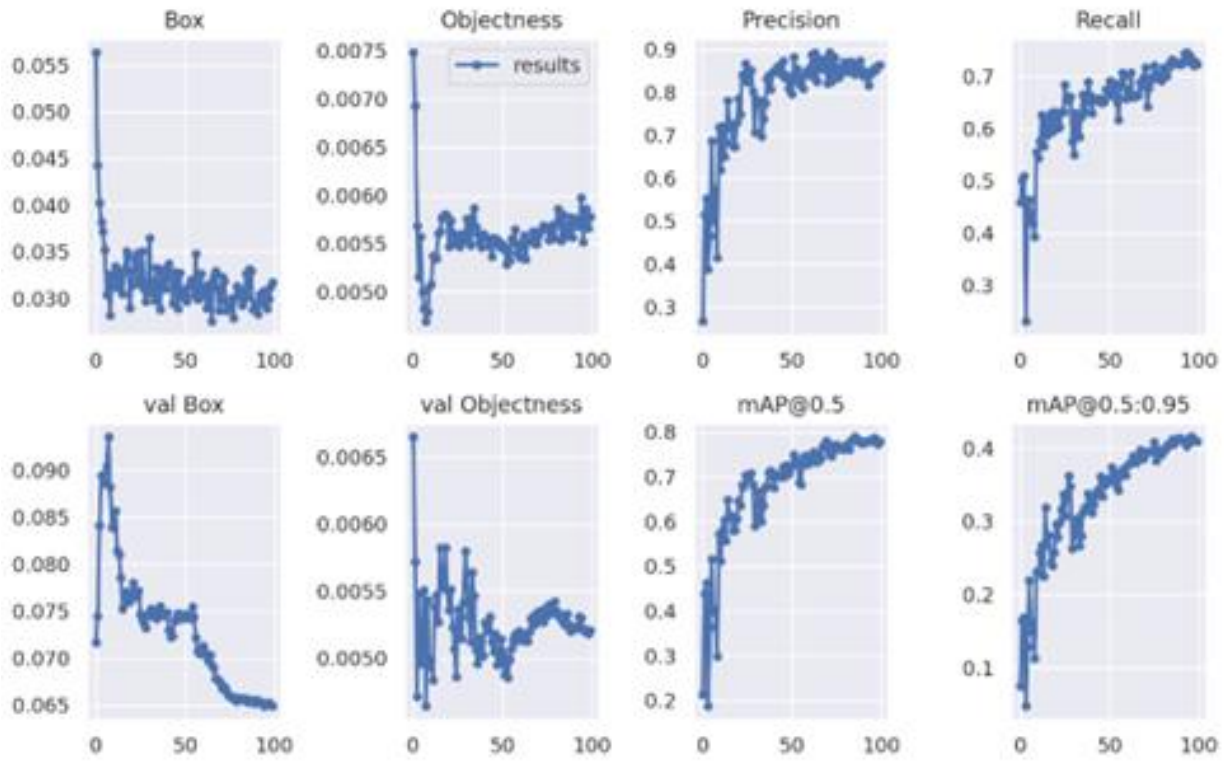

FIGURE 4.1 Train and Validation Graph of YOLOv7

The second model, SSD-MobileNetV2, run on Tensor Flow Object Detection API with the transfer learning of pre-trained on the COCO 2017 dataset. The model train and validation graph can be seen in Figure 4.2 where the blue graphical line is the validation, and the orange graphical line is the training. Figure 4.2 only show the loss of the training because the need of observe the training whether it is overfitting and underfitting. The model works well and not giving sign of either overfitting or underfitting. While FIGURE 4.3 showing mAP of the detection based on the evaluation created.

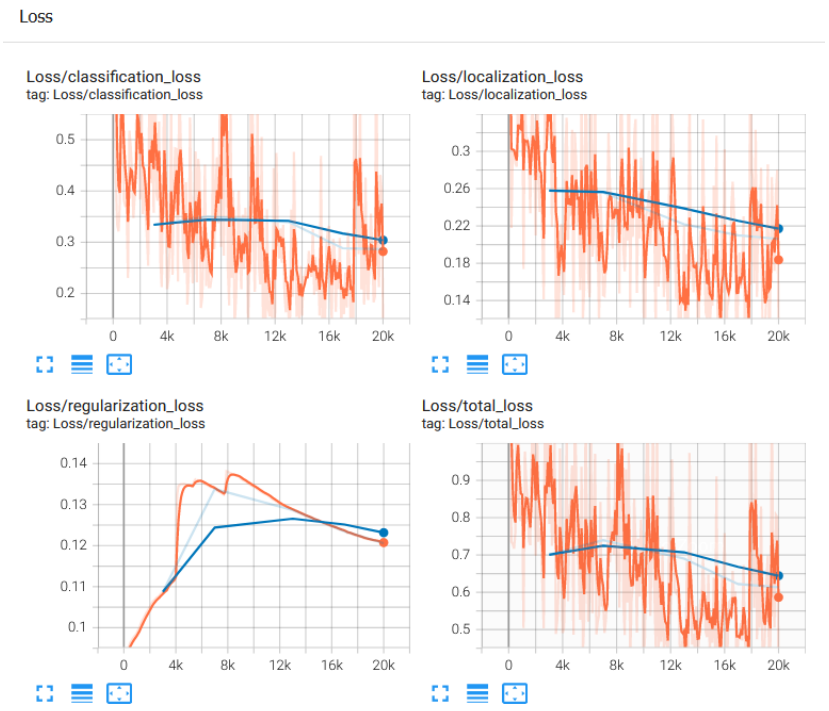

Figure 4.2 Train and Validation loss Graph of SSD-MobileNetV2

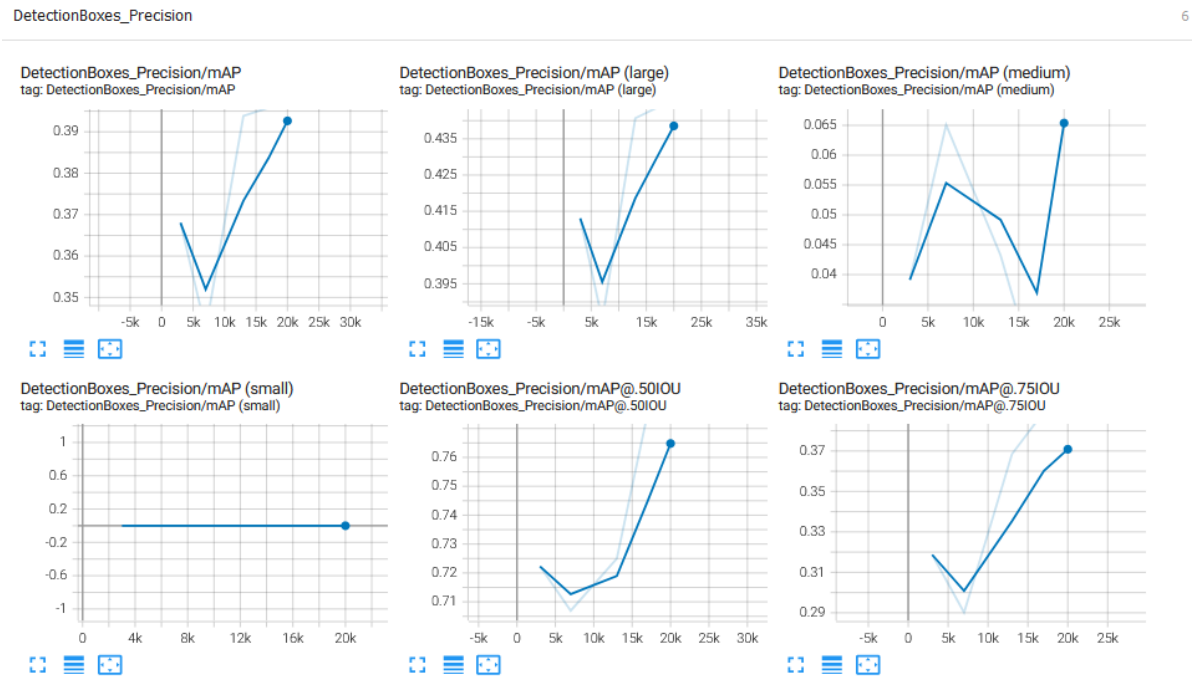

FIGURE 4.3 Validation Graph of SSD-MobileNetV2

Lastly, Faster R-CNN-ResNet50 that takes the longest time to train and takes a huge amount of memory despite having 4 batches of image only. This training anomaly happened due to its heavy

architecture and run with two detectors. The loss graph for both training and validation can be seen in where the blue graphical line is the validation and the orange graphical line is the training. Because of the two detectors, the loss graph also added, the RPNLoss. The loss graph is presented in Figure 4.4 while the mAP of the model can be seen on FIGURE 4.5.

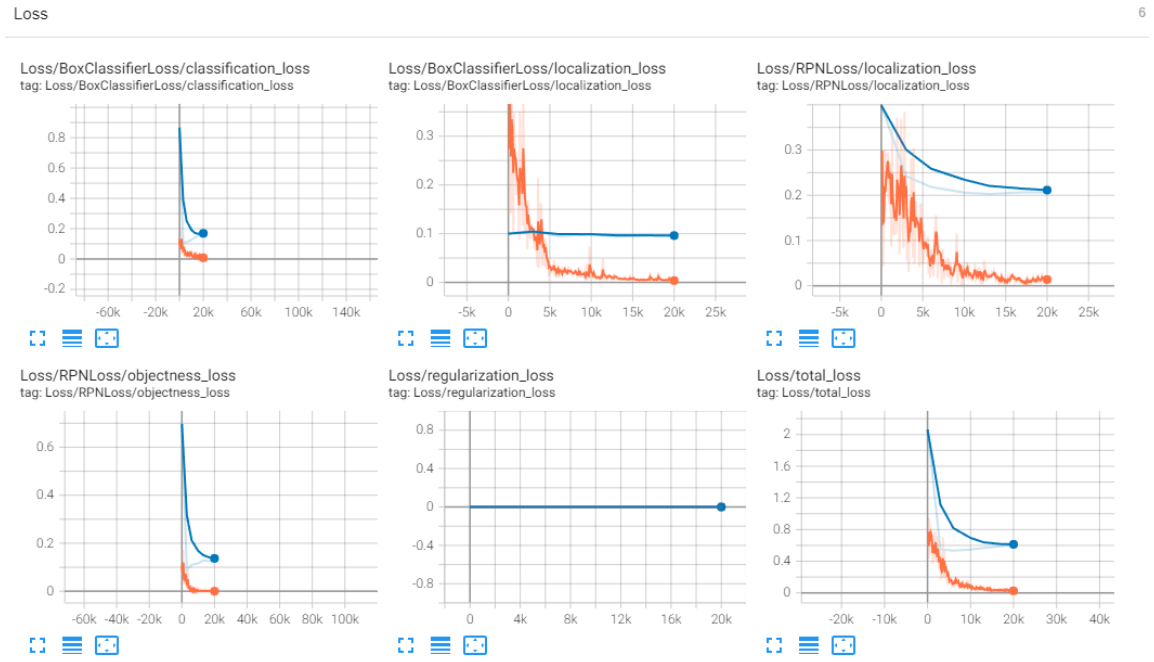

Figure 4.4 Train and Validation loss Graph of Faster R-CNN-ResNet50

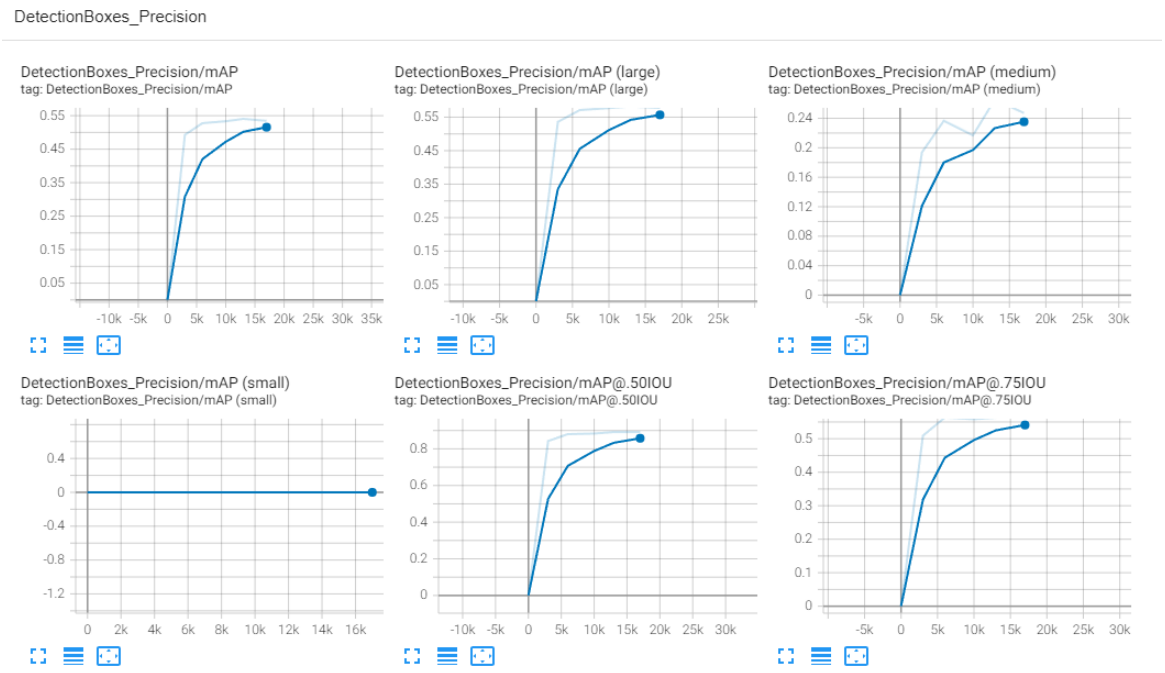

FIGURE 4.5 Validation Graph of Faster R-CNN-ResNet50

From the graph produced during training, The AP of each model can be mapped into a table on Table 2. Using the testing dataset, the result of the AP of each model can be mapped into a table on Table 3. $AP_{0.5}$ for the calculation of average precision on 50% IoU and $AP_{[0.5:0.95]}$ for the calculation of average precision on average range of 50% to 95% IoU.

TABLE 2. Model Average Precision Evaluation on Validation according to each of the IoU.

Model	$AP_{0.5}$	$AP_{[0.5:0.95].}$
YOLOv7-tiny	78.8	41.6
SSD-MobileNetV2	79.10	40.43
Faster R-CNN-ResNet50	88.07	52.61

The training evaluation reflected that Faster R-CNN able to perform better in terms of the ability of detecting with 88.07 and 52.61 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ respectively. While YOLOv7 able to produce 78.8 and 41.6 in $AP_{0.5}$, $AP_{[0.5:0.95]}$ respectively. Lastly, SSD have a performance of 79.10 and 40.43 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ respectively.

TABLE 3. Model Average Precision Evaluation on Testing according to each of the IoU.

Model	$AP_{0.5}$	$AP_{[0.5:0.95].}$
YOLOv7-tiny	79.0	41.4
SSD-MobileNetV2	81.35	42.99
Faster R-CNN-ResNet50	89.16	53.46

The testing evaluation also reflected that Faster R-CNN able to perform better in terms of the ability of detecting with 89.16 and 53.46 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ respectively. On second place, SSD have a performance of 81.35 and 42.99 in $AP_{0.5}$ and $AP_{[0.5:0.95]}$ respectively. Lastly, YOLOv7 able to produce 78.8 and 41.6 in $AP_{0.5}$, $AP_{[0.5:0.95]}$ respectively. Here, the SSD and YOLOv7 did not seem to have a very different result although SSD is better. But the difference between SSD and Faster R-CNN is big so it did give a good consideration in choosing these models.

4.2. Evaluation Result of Computational Load

TABLE 4. Evaluation of Computational Loads on Jetson Nano.

Model	Average time for test dataset (s)	FPS (s)	Average CPU Usage (%)	Average RAM Usage (% of 4 GB)
YOLOv7-tiny	169.31	0.66	81.27	25,32
SSD-MobileNetV2	78.89	0.31	22.54	11.36
Faster R-CNN-ResNet50	OOM	OOM	OOM	OOM

TABLE 5. Evaluation of Computational Loads on low specification PC.

Model	Average time for test dataset (s)	FPS (s)	Average CPU Usage (%)	Average RAM Usage (% of 8 GB)
YOLOv7-tiny	33.23	0.13	29.25	8.96
SSD-MobileNetV2	21.38	0.08	16.22	5.84
Faster R-CNN-ResNet50	325.89	1.26	80.50	22.35

Jetson Nano have the lower computational load than the low specification pc, but it cannot run Faster R-CNN caused by the memory consumed by the model is huge resulting out of memory error (OOM). On Table 4, the evaluation on Jetson Nano, SSD-MobileNetV2 give the best performance with 0.31 seconds of FPS, 22.54% of average CPU usage, and 11.36% of average RAM usage compared to YOLOv7-tiny which has 0.66 seconds of FPS, 81.27% of average CPU usage, and 25.32% of average RAM usage. In the low specification PC, showed on Table 5, SSD also dominating the benchmark with 0.08 seconds of FPS, 16.22% of average CPU usage, and 5.84% of average RAM usage with the second place nominated to YOLOv7-tiny which has 0.13 seconds of FPS, 29.25% of average CPU usage, and 8.96% of average RAM usage. The slowest, heaviest, and takes up the most memory model is Faster R-CNN-ResNet50 with 1.26 seconds of FPS, 80.50% of average CPU usage, and 22.35% of average RAM usage.

The results on devices that have been implemented show that Faster R-CNN display a huge margin on detecting potential compared to the other two model, but it comes with a huge downside on the devices such as a problem on memory usage that is encountered on the implementation on edge device like Jetson Nano. In terms of speed Faster R-CNN can not give a satisfying result also. The best model between the three proposed model, SSD give the most performance in terms of deploying on edge device with a real-time detection and a slightly better performance in detection performance compared to YOLOv7.

5. CONCLUSIONS AND FUTURE WORK

This research focused on the detecting capabilities and the resources needed to run in a low computational load device. In this research, the dataset was gathered on online sources, resized, and annotated manually. From previous research, the models that are current state of the art, Faster-RCNN, SSD and YOLOv7 were proposed on a smaller backbone that is made specifically for low computational load devices. These models trained on the gathered dataset, proboscis monkey

dataset, before tested. In terms of model detection performance Faster-RCNN give the highest score on average precision. On the other hand, in terms of model computational load on inferencing, SSD was proven to be the lightest and fastest model compared to the other two model when tested on low computational load devices, Jetson Nano and low specification PC. Hence the best model for the real-time detection of proboscis monkey is SSD with MobileNetV2 backbone.

In future works, this research can be continued by looking for the better model since in this paper we see that Faster R-CNN did have the best performance in terms of detection but not as light and fast as SSD. So, researcher can develop a new model that is lighter yet accurate and suitable to be used on real-time environment.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] Populasi Bekantan Terancam Punah. https://www.kompas.id/baca/ilmu-pengetahuan-teknologi/2017/12/07/populasi-bekantan-terancam-punah.

[2] Bekantan, Si Hidung Besar Nan Mempesona. https://indonesia.go.id/kategori/seni/941/bekantan-si-hidung-besar-nan-mempesona.

[3] I. Matsuda, Following the trail of the elusive proboscis monkey in Borneo, Ecol. Res. 37 (2022), 562-567. https://doi.org/10.1111/1440-1703.12343.

[4] M. Vidal, N. Wolf, B. Rosenberg, et al. Perspectives on individual animal identification from biology and computer vision, Integrat. Comparat. Biol. 61 (2021), 900-916. https://doi.org/10.1093/icb/icab107.

[5] S. Ren, K. He, R. Girshick, et al. Faster R-CNN: towards real-time object detection with region proposal networks, preprint, (2016). http://arxiv.org/abs/1506.01497.

[6] S.K. Pal, A. Pramanik, J. Maiti, et al. Deep learning in multi-object detection and tracking: state of the art, Appl. Intell. 51 (2021), 6400-6429. https://doi.org/10.1007/s10489-021-02293-7.

[7] J. Redmon, S. Divvala, R. Girshic, et al. You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788.

[8] J. Long, E. Shelhamer, T. Darrel, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440.

[9] T.Y. Lin, M. Maire, S. Belongie, et al. Microsoft COCO: common objects in context, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision - ECCV 2014, Springer International Publishing, Cham, 2014: pp. 740-755. https://doi.org/10.1007/978-3-319-10602-1_48.

[10] S. Rathor, S. Kumari, R. Singh, et al. Two layers machine learning architecture for animal classification using HOG and LBP, in: V. Goyal, M. Gupta, A. Trivedi, M.L. Kolhe (Eds.), Proceedings of International Conference on Communication and Artificial Intelligence, Springer Singapore, Singapore, 2021: pp. 445-453. https://doi.org/10.1007/978-981-33-6546-9_42.

[11] A. Tydén, S. Olsson, Edge machine learning for animal detection, classification, and tracking, Master of Science Thesis in Media Technology Department of Electrical Engineering, Linköping University, Sweden, 2020.

[12] D. Schofield, A. Nagrani, A. Zisserman, et al. Chimpanzee face recognition from videos in the wild using deep learning, Sci. Adv. 5 (2019), aaw0736. https://doi.org/10.1126/sciadv.aaw0736.

[13] J. Parham, C. Stewart, J. Crall, et al. An animal detection pipeline for identification, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, Lake Tahoe, NV, 2018: pp. 1075-1083. https://doi.org/10.1109/WACV.2018.00123.

[14] O. Russakovsky, J. Deng, H. Su, et al. ImageNet large scale visual recognition challenge, preprint, (2015). http://arxiv.org/abs/1409.0575.

[15] G.K. Verma, P. Gupta, Wild animal detection using deep convolutional neural network, in: B.B. Chaudhuri, M.S. Kankanhalli, B. Raman (Eds.), Proceedings of 2nd International Conference on Computer Vision & Image Processing, Springer Singapore, Singapore, 2018: pp. 327-338. https://doi.org/10.1007/978-981-10-7898-9_27.

[16] M.S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M.S. Palmer, C. Packer, J. Clune, Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning, Proc. Natl. Acad. Sci. U.S.A. 115 (2018), E5716-E5725. https://doi.org/10.1073/pnas.1719367115.

[17] M. Lin, Q. Chen, S. Yan, Network in network, preprint, (2014). http://arxiv.org/abs/1312.4400.

[18] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, preprint, (2015). http://arxiv.org/abs/1409.1556.

[19] C. Szegedy, W. Liu, Y. Jia, et al. Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9.

[20] A.V. Sayagavi, T.S.B. Sudarshan, P.C. Ravoor, Deep Learning methods for animal recognition and tracking to detect intrusions, in: T. Senjyu, P.N. Mahalle, T. Perumal, A. Joshi (Eds.), Information and Communication Technology for Intelligent Systems, Springer, Singapore, 2021: pp. 617-626. https://doi.org/10.1007/978-981-15-7062-9_62.

[21] S. Schneider, G.W. Taylor, S. Kremer, Deep learning object detection methods for ecological camera trap data, in: 2018 15th Conference on Computer and Robot Vision (CRV), IEEE, Toronto, ON, Canada, 2018: pp. 321–328. https://doi.org/10.1109/CRV.2018.00052.

[22] M. Pedersen, J.B. Haurum, R. Gade, et al. Detection of marine animals in a new underwater dataset with varying visibility, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition

(CVPR) Workshops, 2019, pp. 18-26.

[23] P. Valdez, Apple defect detection using deep learning based object detection for better post harvest handling, preprint, (2020). https://arxiv.org/abs/2005.06089.

[24] L. Tan, T. Huangfu, L. Wu et al. Comparison of YOLO v3, faster R-CNN, and SSD for real-time pill identification, preprint, (2021). https://doi.org/10.21203/rs.3.rs-668895/v1.

[25] P. Garg, D.R. Chowdhury, V.N. More, Traffic sign recognition and classification using YOLOv2, faster RCNN and SSD, in: 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, Kanpur, India, 2019: pp. 1-5. https://doi.org/10.1109/ICCCNT45670.2019.8944491.

[26] E.M.T.A. Alsaadi, N.K. El Abbadi, An automated mammals detection based on SSD-mobile net, J. Phys.: Conf. Ser. 1879 (2021), 022086. https://doi.org/10.1088/1742-6596/1879/2/022086.

[27] A. Loos, C. Weigel, M. Koehler, Towards automatic detection of animals in camera-trap images, in: 2018 26th European Signal Processing Conference (EUSIPCO), IEEE, Rome, Italy, 2018: pp. 1805-1809. https://doi.org/10.23919/EUSIPCO.2018.8553439.

[28] H. Lee, H. Kwon, R.M. Robinson, et al. Dynamic belief fusion for object detection, in: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, Lake Placid, NY, USA, 2016: pp. 1-9. https://doi.org/10.1109/WACV.2016.7477574.

[29] W. Liu, D. Anguelov, D. Erhan, et al. SSD: Single Shot MultiBox Detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016: pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

[30] M. Sandler, A. Howard, M. Zhu, et al. MobileNetV2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520.

[31] C.Y. Wang, A. Bochkovskiy, H.Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 7464-7475.