JOURNAL OF COMPUTER SCIENCES INSTITUTE

JCSI 23 (2022) 105–111
Received: 11 March 2022
Accepted: 29 March 2022

Detrimental Starfish Detection on Embedded System: A Case Study of YOLOv5 Deep Learning Algorithm and TensorFlow Lite framework

Nguyen Quoc Toan*

Department of Electronic and Electrical Engineering, Hongik University, Wausan-ro 94, Mapo-gu, Seoul, South Korea

Abstract

There is a great range of spectacular coral reefs in the ocean world. Unfortunately, they are in jeopardy, due to an over-abundance of one specific starfish called the coral-eating crown-of-thorns starfish (or COTS). This article provides research to deliver innovation in COTS control. Using a deep learning model based on the You Only Look Once version 5 (YOLOv5) deep learning algorithm on an embedded device for COTS detection. It aids professionals in optimizing their time, resources, and enhances efficiency for the preservation of coral reefs worldwide. As a result, the performance over the algorithm was outstanding with Precision: 0.93 - Recall: 0.77 - $F1_{score}$: 0.84.

Keywords: deep learning; computer vision; YOLO; embedded system

*Corresponding author
Email address: **quoctoann3@gmail.com** (N. Q. Toan)

1. Introduction

COTS (crown-of-thorns starfish) is a natural coral predator. However, when populations reach epidemic levels (about 15 starfish per hectare), corals are eaten quicker by COTS than they can develop. Crown-of-thorns starfish may devour up to 90% of a reef's living coral tissue during an epidemic. This adds to the pressures already exerted on the reef by problems such as bleaching and climate change. Many environmental organizations, universities, government agencies, and members of the public have contributed a vast amount of information and resources to help understand and monitor COTS outbreaks. On the other hand, new outbreaks occur in 1–15-year cycles, making it impossible to pinpoint exact causes or even keep its numbers under control. COTS outbreaks have been related to a variety of reasons, including ocean "stressors" such as surges in ocean nutrients generated by coastal and agricultural run-off into the ocean, as well as a loss of predators due to overfishing, according to decades of studies on the Great Barrier Reef. In this study, I trained and evaluated the COTS detection model (YOLOv5 small version 6) with the You Only Live Once algorithm which is used for embedded systems and mobile devices. It has a state-of-the-art convolutional neural network (CNN) at its core with the required configuration that parameters were optimized. On top of that, I applied advanced data augmentation methods for enhancing the quality and quantity of the dataset from 'The CSIRO Crown-of-Thorn Starfish Detection Dataset' [1].

2. Methodology

Deep learning has been well-known in recent years for its capacity to learn from experience and is now being utilized to solve complicated issues. Deep convolutional neural networks (CNNs) have made significant progress in large-scale object recognition in particular [7]. Figure 2 shows the workflow of a deep CNN-based object detection system with components.

Figure 1: Crown-Of-Thorns Starfish (COTS) on coral.

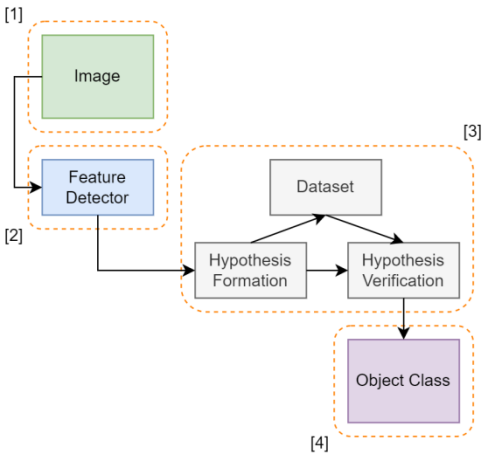

Figure 2: Deep CNN-based object detection system's workflow.

2.1. You Only Look Once algorithm (YOLO)

YOLO is a one-stage object detection algorithm that has been around for a long time. It transforms the detection issue into a regression problem. Instead of extracting RoI, it uses the regression approach to obtain the bounding box coordinates and probability of each class. It considerably enhances detection speed when compared to faster R-CNN. It doesn't employ a region proposal or a sliding window like other networks; It reframes object detection as a single regression problem. After only one

look at the input image, YOLO turns it into a grid of S x S cells. Each grid cell predicts B bounding boxes and a confidence score that shows if the predicted bounding box intersects with the ground truth bounding box and whether the predicted bounding box includes some objects:

$$Confidence = Pr(object) * IOU_{pred}^{truth} \quad (1)$$

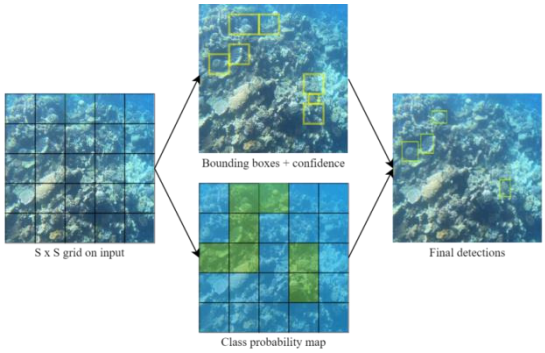

Figure 3: YOLO algorithm's anchor box.

The YOLO network predicts bounding boxes as deviations from a set of 'anchor box' dimensions to produce box predictions.

2.2. YOLOv5 Network

YOLOv5 outperforms all prior versions in terms of speed and accuracy. YOLOv5 version 6 was applied for this research. The YOLOv5 algorithm adjusts the width and depth of the backbone network using the depth-multiple and width-multiple parameters, yielding five different models: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x.

As a backbone, it uses CSPDarknet53. This backbone handles the repeating gradient information in big backbones and incorporates gradient change into feature maps, which speeds up inference, improves accuracy, and decreases the model size by lowering parameters. It boosts information flow by using a path aggregation network (PANet) as a neck. PANet uses a novel feature pyramid network (FPN) with many bottom-up and top-down layers. This enhances the model's low-level feature propagation. PANet increases the object's localization accuracy by improving localization in lower levels. Moreover, the focus structure using CSPdarknet53 as a backbone reduces the amount of CUDA memory required, increases forward propagation, and decreases back-propagation.

2.3. TensorFlow Lite

TensorFlow Lite was employed in this research for a variety of reasons:

- Light-weight: In terms of storage and compute capacity, embedded devices have minimal resources. Deep learning models consume a lot of resources, thus the models we deploy on embedded devices should be light and have reduced binary sizes.
- Low Latency: Regardless of network connectivity, deep learning models on the embedded systems should produce faster inferences irrespective.
- Pre-trained: Models can be trained on-premise or cloud for various deep learning tasks. Such as image classification, object detection, speech recognition, etc. and can be simply deployed to make inferences.

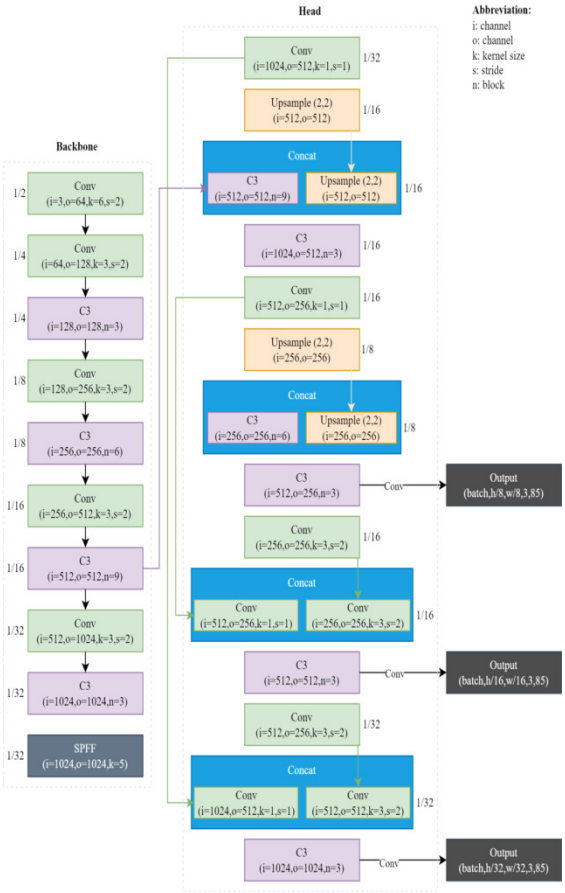

Figure 4: YOLOv5s version 6 network architecture.

It has all the capabilities needed to make inferences on the embedded systems. It is a cross-platform, open-source deep learning framework that transforms a pre-trained model into its format. It is a specific format model that is efficient in terms of performance and a lightweight version that will take up less space, these features make it ideal for use on mobile and embedded devices. Therefore, I decided to use it because it is not only easy to use but also optimized to execute.

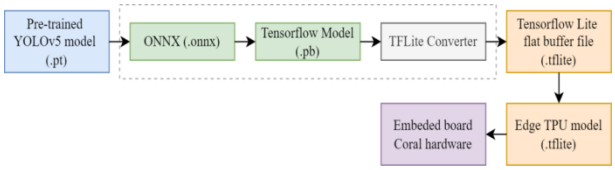

Figure 5: YOLOv5 to TensorFlow Lite weight conversion proces.

The TensorFlow Lite model can be deployed on mobile devices. Such as Android and iOS, or on embedded devices like Raspberry and Microcontrollers in general. The following are the steps to making an inference from embedded devices:

- Initialize the interpreter and load the model into it.
- Allocate the tensor and obtain the input and output tensors.

- Preprocess the images by reading them into a tensor.
- Invoke the interpreter to make the inference on the input tensor.
- Get the image's result by mapping the inference's results.

3. Experiments

The flowchart below depicts the experiment steps in my proposed research:

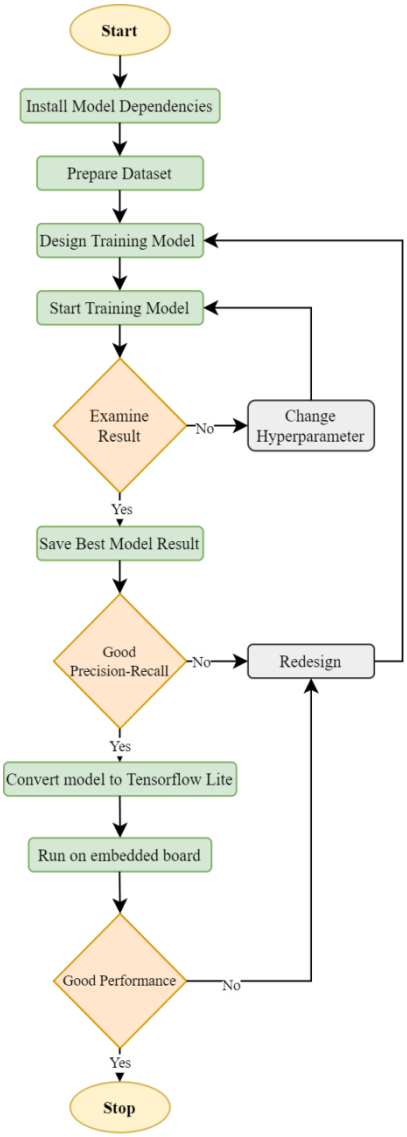

Figure 6: Flowchart of the proposed research.

3.1. Data Collection and Processing

The dataset [1] was gathered by a CSIRO team using a GoPro Hero9 camera that had been modified for use in the Manta Tow technique. The camera was specifically affixed to the bottom of the manta tow board, which the snorkeller-diver held during the surveys. As the diver explores the reef, the camera displays an oblique field of view of the reef below him, with the distance to the reef constantly changing. The distance from the bottom is usually several meters, although it might be as little as a few tens of centimeters or as much as 10 meters or more. The survey boat travels at a top speed of 5 knots and pulls the diver for two minutes, covering a distance of around 200 meters. The boat then comes to a complete halt to allow the diver to record data collected throughout the transect on a sheet of paper. Expert annotators used pre-trained COTS detection models to identify every COTS in the photos, which was followed by an AI-assisted annotation and quality assurance procedure. Using the annotation program, all of the COTS detection in a picture was marked with a box. The data was gathered on a reef in the Swain Reefs section of the GBR on a single day in October 2021. Lighting, visibility, coral habitat, depth, distance from the bottom, and viewpoint all vary.

The collection comprises underwater picture sequences recorded at five distinct locations on a reef in the GBR. There are almost 35000 photos in all, with hundreds of individual COTS visible. However, only 4888 photos, including COTS, which I utilized to augment for the dataset to train the proposed model.

The size of the original images has been normalized to 640x640 to give a detection technique that fulfills the requirement for real-time and precise operation. This boosts the detection speed without deleting any important data in the image. The normalized photos have also been divided into two groups: a test set and a training set. After including the detection model's anchor box size for clustering purposes, the pre-processing criteria would be finished. I employed certain processes for data augmentation. Such as rotation, horizontal and vertical shear, mosaic. Those approaches do not change the image's pixel values; they merely change the image's position. As a result, the learning ability of the convolutional neural network would be modified to learn high-quality features while avoiding overfitting. Table 1 depicts the description of the augmented dataset after my data processing step. Figure 7 depicts a flowchart of the data augmented implementation process.

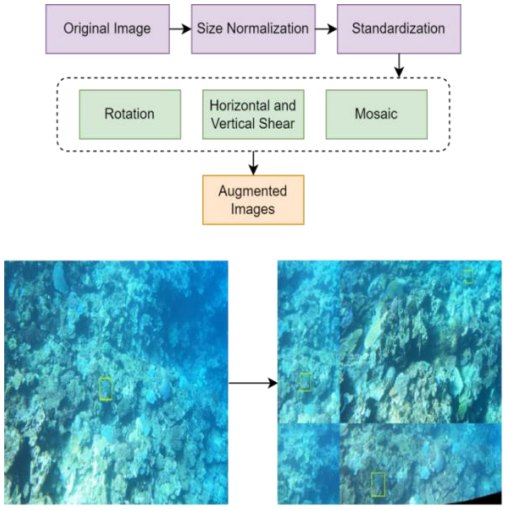

Figure 7: Data augmentation proces.

Table 1: Augmented dataset summary

Total Image	Size	Class	Labelled box
14.646	640x640	COTS	40.000

3.2. Training

Model training was carried out on 2 GPUs - NVIDIA GeForce RTX 2080 Ti (11GB-GDDR6).

Algorithm 1: Training model

Input: Algorithm (YOLOv5s) with installed necessary libraries and dependencies
Output: COTS detection model weights
Data: Dataset (Train/Val/Test)

```
start
for YOLOv5s Model do
    Train Dataset
    (img 640 –batch 24 –epochs 100 –device 0,1)
    Evaluate The model
    Compute From TensorBoard
    (Precision, Recall, F1-score, mAP)
    Display Training Performance
    Run Model inference on test images
    Visualize Inference on test images
    Save Model weights
end
```

4. Results

4.1. Evaluation metrics

To evaluate YOLOv5's algorithm, F1score and mAP were employed. The F1score is the harmonic mean of accuracy and recall. It is also the model's test accuracy. The maximum possible F1score value is 1, indicating flawless accuracy and memory, while the lowest possible score is 0, indicating that either precision or recall is zero. Furthermore, mAP is determined by averaging the average precision (AP), where q is the number of queries and AveP(q) is the average precision for that particular query. The mean of AP may then be used to determine mAP. mAP may also be thought of as a metric for calculating the accuracy of machine learning algorithms. In COTS detection, True Positive (TP) is the predicted COTS was correctly detected and the actual class was the same as the predicted one. False Positive (FP) is the model that predicted COTS and the actual class was not. False Negative (FN) is predicted class was not COTS and the actual class was COTS. True Negative (TN) is the predicted class was not COTS and the actual class was also not COTS.

$$F1_{score} = \frac{2*Precision*Recall}{Precision+Recall} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

$$mAP = \sum_{q=1}^{Q} \frac{AveP(q)}{Q} \quad (5)$$

It is worth noting that precision is computed as the ratio of true prediction to the total number of predictions. For instance, if a model generates 50 predictions and all of them are true, the precision is 100%. Precision does not take into account the actual number of true objects present in an image; whereas recall computes the ratio of true predictions to the total number of objects in an image. For example, if a model detects 75 true objects and the image contains 100 true objects, recall is determined to be 75%. The presence of simply high precision or only high recall does not imply that the model is a good one. It must strike a balance between precision and recall for an object detection algorithm to be considered outstanding. Therefore, we use the F1score to determine if a model is a good one or not.

4.2. Model's weight information

Table 2: Model's weight information

Item	Information
Model	YOLOv5s.pt
Precision	0.93
Recall	0.77
F1score	0.84
mAP(%)	83
Training time	4 hours + 53 minutes

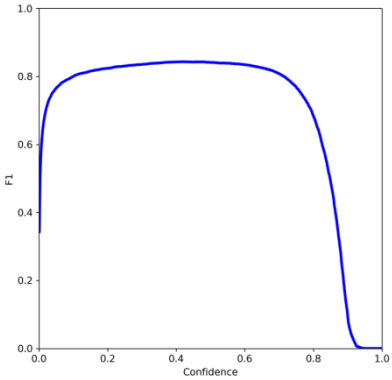

Figure 8: Model's $F1_{score}$ curve.

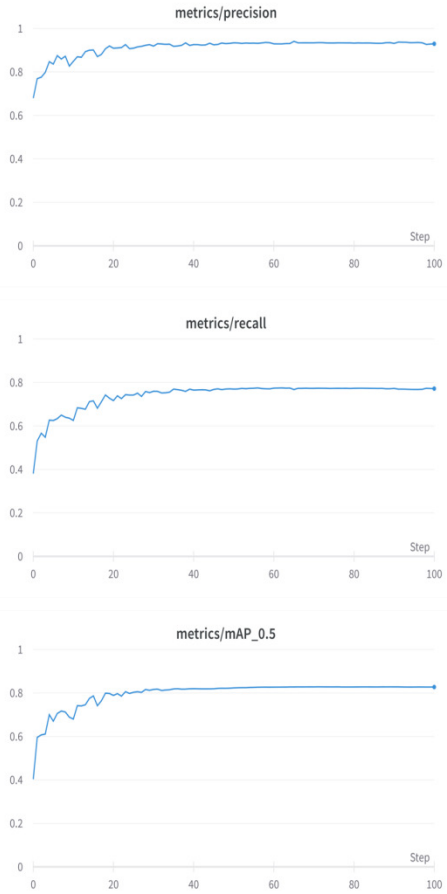

Figure 9: Model's Precision-Recall-mAP.

4.3. Model's performance on embedded system

Figure 10: Setup of the proposed embedded system: 1) Odroid XU4 - 2) Microsoft Camera C3000 - 3) Google Edge TPU coprocessor - 4) Adafruit 7-inch monitor.

Table 3: Proposed embedded system in detail

Item	Property
Microcontroller	Odroid XU4
Identifier	ARM implementer 65 architecture 7 variant 2 part 3087 revision 3
CPU	Samsung Exynos5422 Cortex™-A15 2Ghz and Cortex™-A7 Octa-core
GPU	ARM Mali-T628 6 Cores
RAM	2GB LPDDR3
TPU	Google Edge Coral
Camera	Mircosoft C3000 (720p)
Monitor	Adafruit 7 inch (720p)
OS	Ubuntu 18.04 MATE
Language	Python 3.7.3

Table 4: Performance on the proposed embedded system

Item	Property
Model	COTS.tflite
Model size	7.3 MB
Input size	640x480
FPS	30
Inference time (Single-core)	30 milliseconds (0.03 seconds)
Inference time (Multiple-cores)	8 milliseconds (0.008 seconds)
Accuracy	93%

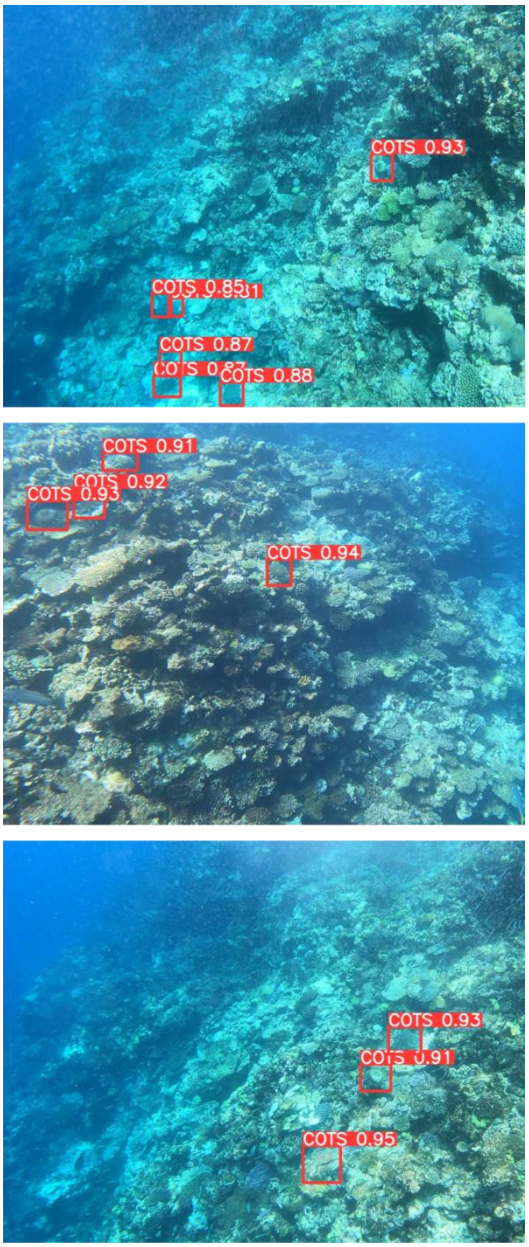

Figure 12: COTS detection inferences on the proposed system.

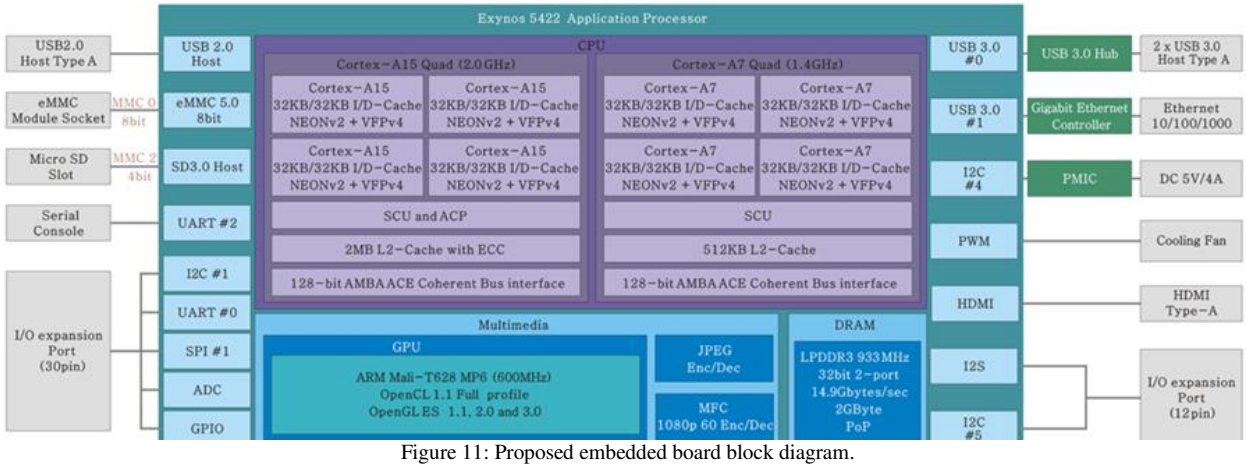

Figure 11: Proposed embedded board block diagram.

5. Conclusion

In this paper, I provided a Crown-of-thorns starfish (COTS) detection system based on the YOLOv5 algorithm. This approach can identify COTS in real-time, videos or images, and recognize whether it appears on the scene. I trained a model to provide a real-time system, but still, be able to maintain high accuracy. Furthermore, the size of my model is quite favorable; it is only 7.3 MB in size. It can undoubtedly be applied to real-world scenarios. It demonstrated that a deep CNN can indeed be applied to the task with promising results in evaluated metrics. I expect that the outcomes will be far better than those shown here with more training data. A restriction of embedded system configuration is also a concern. In the future, I will do research to develop a system that balances the needed configuration and model performance.

6. Acknowledgment

I would like to send my sincere appreciation to HAIL (Hongik University Artificial Intelligence Laboratory) which is advised by Prof. Seongwon Cho for supporting me in this research.

References

[1] L. Jiajun, K. Brano, M. Ross, D. Brendan, M. Torsten, C. Joey, S. Andy, H. Nic, V. R. Karl, T. S. Lachlan, A. A. David, A. A. Mohammad, C. Geoffrey, B. Russ, M. Peyman, S. Daniel, D. Tim, E.M. Kemal, W. Martin, M. Megha, The CSIRO Crown-of-Thorn Starfish Detection Dataset, arXiv, 2021, https://doi.org/10.48550/arXiv.2111.14311

[2] W. Junlong, K. Wei, Z. Wei, H. Fengbiao, T. Xuefeng, W. Qiong, Helmet Detection Algorithm Based on the Improved YOLOv5 and Dynamic Anchor Box Matching, Proceedings of the IEEE International Conference on Emergency Science and Information Technology (ICESIT), (2021) 79-83, http://dx.doi.org/10.1109/ICESIT53460.2021.9696525.

[3] Y. Zhong, J. Wang, J. Peng, L. Zhang, Anchor Box Optimization for Object Detection, Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), (2020) 1275-1283, http://dx.doi.org/10.1109/WACV45572.2020.9093498.

[4] T. F. Dima, M. E. Ahmed, Using YOLOv5 Algorithm to Detect and Recognize American Sign Language, Proceedings of the International Conference on Information Technology (ICIT), (2021) 603-607, http://dx.doi.org/10.1109/ICIT52682.2021.9491672.

[5] G. Verma, Y. Gupta, A. M. Malik, B. Chapman, Performance Evaluation of Deep Learning Compilers for Edge Inference, Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), (2021) 858-865, http://dx.doi.org/10.1109/IPDPSW52791.2021.00128.

[6] T. Zhi, S. Chunhua, C. Hao, H. Tong, FCOS: Fully Convolutional One-Stage Object Detection, arXiv, 2019, https://doi.org/10.48550/arXiv.1904.01355.

[7] L. Wei, A. Dragomir, E. Dumitru, S. Christian, R. Scott, F. Cheng-Yang, B. C. Alexander, SSD: Single Shot MultiBox Detector, Lecture Notes in Computer Science, 2016, https://doi.org/10.1007/978-3-319-46448-0_2.

[8] Z. Ni, J. Chen, N. Sang, C. Gao, L. Liu, Light YOLO for high-speed gesture recognition, Proceedings of The 2018 25th IEEE International Conference on Image Processing (ICIP), (2018) 3099-3103, http://dx.doi.org/10.1109/ICIP.2018.8451766.

[9] A. Aleena, S. Ayesha, J. Tauseef, U.K. Asif, Small Object Detection using Deep Learning, arXiv, 2022, https://doi.org/10.48550/arXiv.2201.03243.

[10] B. G. Han, J. G. Lee, K. T. Lim, D. H. Choi, Design of a scalable and fast YOLO for edge-computing devices, Sensors, 2020, https://doi.org/10.3390/s20236779.

[11] B. Liang, S. Wu, K. Xu, J. Hao, Butterfly detection and classification based on integrated YOLO algorithm, arXiv, 2020, https://doi.org/10.48550/arXiv.2001.00361.

[12] C. Shaobin, L. Wei, Embedded System Real-Time Vehicle Detection based on Improved YOLO Network, Proceedings of the IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), (2019) 1400-1403, http://dx.doi.org/10.1109/IMCEC46724.2019.8984055.

[13] Y. Zhu, C. Yao, X. Bai, Scene text detection and recognition: recent advances and future trends, Frontiers of Computer Science, 2015, http://dx.doi.org/10.1007/s11704-015-4488-0.

[14] Q. Lu, Y. Yuan, Improved YOLO Algorithm for Object Detection in Traffic Video, Proceedings of the International Conference in Communications, Signal Processing and Systems, 2019, http://dx.doi.org/10.1007/978-981-13-9409-6_198.

[15] R. Shaoqing, H. Kaiming, G. Ross, S. Jian, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, arXiv, 2016, https://doi.org/10.48550/arXiv.1506.01497.

[16] H. Shijie, W. Zhonghao, S. Fuming, LEDet: A Single-Shot Real-Time Object Detector Based on Low-Light Image Enhancement, The Computer Journal, 2021, https://doi.org/10.1093/comjnl/bxab055.

[17] A. A. Choudhury, R. Saha, S. Z. Shoumo, S. R. Tulon, J. Uddin, M. K. Rahman, An efficient way to represent braille using YOLO algorithm, Proceedings of the Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV), (2018) 10-19, http://dx.doi.org/10.1109/ICIEV.2018.8641038.

[18] D. Volivier, M. Saïd, A. M. Sidi, M. Pierre, L. Frédéric, A new Edge Architecture for AI-IoT services deployment, Proceedings of the 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), (2020) 10-19, https://doi.org/10.1016/j.procs.2020.07.006.

[19] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems, (2012) 1097-1105, https://doi.org/10.1145/3065386.

[20] D. S. Viraktamath, P. Navalgi, A. Neelopant, Comparison of YOLOv3 and SSD Algorithms,

Proceedings of the International Journal of Engineering Research & Technology (IJERT), (2021) 1156–1160.

[21] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016) 779–788, http://dx.doi.org/10.1109/CVPR.2016.91.

[22] A. Bochkovskiy, C. Y. Wang, H. Y. M. Liao, YOLOv4: Optimal speed and accuracy of object detection, arXiv, 2020, https://doi.org/10.48550/arXiv.2004.10934.

[23] B. Jiang, R. Luo, J. Mao, T. Xiao, Y. Jiang, Acquisition of Localization Confidence for Accurate Object Detection, In Proceedings of the European conference on computer vision (ECCV), (2018) 8-14, http://dx.doi.org/10.1007/978-3-030-01264-9_48.

[24] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, W. Zuo, Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation, arXiv, 2021, https://doi.org/10.48550/arXiv.2005.03572.

[25] K. K. Reddy, M. Shah, Recognizing 50 human action categories of web videos, Machine Vision and Applications, (2016) 971-981, https://doi.org/10.1007/s00138-012-0450-4.