

Processing time increasement of non-rice object detection based on YOLOv3-tiny using Movidius NCS 2 on Raspberry Pi

Nova Eka Budiyanta, Catherine Olivia Sereati, Ferry Rippun Gideon Manalu

Department of Electrical Engineering, Faculty of Engineering, Universitas Katolik Indonesia Atma Jaya, Jakarta, Indonesia

Article Info

Article history:

Received Dec 11, 2021

Revised Mar 6, 2022

Accepted Mar 15, 2022

Keywords:

Embedded system

Intel Movidius NCS 2

Object detection

Raspberry Pi

YOLOv3-tiny

ABSTRACT

This study aims to increase the processing time of detecting non-rice objects based on the you only look once v3-tiny (YOLOv3-tiny) model. The system was developed on the Raspberry Pi 4 embedded system with the Movidius neural compute stick 2 (NCS 2) implementation approach. Data object in the form of gravel on a bunch of rice in the video. The video data was obtained using a webcam with a resolution of 1920 x 1080 pixels with a total of 2759 frames. From the test results, frames per second (FPS) have increased by 1.27x in the Movidius NCS 2 implementation compared to processing using the central processing unit (CPU) from the Raspberry Pi 4. The object detection processing on video data is complete at 1871.408 seconds with 1.474 FPS using the CPU from the Raspberry Pi 4 and finished at 1477.141 seconds with 1.868 FPS using Movidius NCS 2. From these differences, it can be seen that the application of Movidius NCS 2 succeeded in increasing the object detection processing in this study by 26.69% with the YOLOv3-tiny model approach on the Raspberry Pi 4 embedded system.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Nova Eka Budiyanta

Department of Electrical Engineering, Faculty of Engineering, Universitas Katolik Indonesia Atma Jaya

Jl. Raya Cisauk Lapan, Sampora, Kec. Cisauk, Kabupaten Tangerang, Banten 15345, Jakarta, Indonesia

Email: nova.eka@atmajaya.ac.id

1. INTRODUCTION

In this modern era, the application of technology in the agricultural world was increasingly crowded [1], [2]. Several approaches have been taken to support the efficiency of agriculture. Besides using the internet of things (IoT) for smarthome systems [3], [4], for plant monitoring, both in terms of soil fertility to plant health regularly and automatically [5], [6]. The more advanced the agriculture monitoring process has developed, the more researchers are researching the camera-based agricultural monitoring process [7], [8], combined with the IoT system on end computing [9], [10]. The camera is used as a sensor which then the data obtained, processed, and then can be used as a reference for concluding, and some conclusions are drawn using machine learning.

The combination of using cameras as input for machine learning is considered suitable for solving object detection problems in the monitoring process [11]–[14]. However, in the case of remote monitoring, the latency of sending and processing data is considered a problem if the detection process is carried out on the server. To overcome this, few studies have also been carried out related to the object detection process in edge computing. In question, edge computing is carried out on the client/sensor side so that the computing process is faster than computing on the server.

To support the complexity of tools on edge computing, in this study, a study was conducted regarding the application of the Raspberry Pi to the camera-based object detection task. Not only referring to computing using the central processing unit (CPU) of the Raspberry Pi but the Movidius neural compute

stick 2 (NCS 2) is also applied in this study. Furthermore, the CPU usage of the Raspberry Pi was compared with NCS 2, where NCS 2 serves to speed up the computing process on low-performance devices [15], such as its implementation on Raspberry Pi [16]–[18].

2. METHOD

2.1. Research method

This research focuses on the application of object detection on the Raspberry Pi embedded system to detect gravel in a bunch of rice. The model used for the detection process is you only look once (YOLO) v3-tiny (YOLOv3-Tiny) with the Darknet-19 network, where YOLOv3-tiny is an upgraded algorithm of YOLO [19], YOLO9000 [20], and YOLOv3 [21], which includes updated development of the convolutional neural network (CNN) [22]. YOLOv3-Tiny is a development of YOLOv3 with fewer networks to optimize computing speed [23]. Darknet-19 used in the YOLOv3-Tiny configuration can be seen in Table 1. There are several stages in this research. The research stages can be seen in Figure 1.

Table 1. Darknet-19 architecture on the YOLOv3-tiny algorithm [23]

Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3 x 3/1	416 x 416 x 3	416 x 416 x 16
1	Maxpool		2 x 2/2	416 x 416 x 16	208 x 208 x 16
2	Convolutional	32	3 x 3/1	208 x 208 x 16	208 x 208 x 32
3	Maxpool		2 x 2/2	208 x 208 x 32	104 x 104 x 32
4	Convolutional	64	3 x 3/1	104 x 104 x 32	104 x 104 x 64
5	Maxpool		2 x 2/2	104 x 104 x 64	52 x 52 x 64
6	Convolutional	128	3 x 3/1	52 x 52 x 64	52 x 52 x 128
7	Maxpool		2 x 2/2	52 x 52 x 128	26 x 2 x 128
8	Convolutional	256	3 x 3/1	26 x 2 x 128	26 x 26 x 256
9	Maxpool		2 x 2/2	26 x 26 x 256	13 x 13 x 256
10	Convolutional	512	3 x 3/1	13 x 13 x 256	13 x 13 x 512
11	Maxpool		2 x 2/2	13 x 13 x 512	13 x 13 x 512
12	Convolutional	1024	3 x 3/1	13 x 13 x 512	13 x 13 x 1024
13	Convolutional	256	1 x 1/1	13 x 13 x 1024	13 x 13 x 256
14	Convolutional	512	3 x 3/1	13 x 13 x 256	13 x 13 x 512
15	Convolutional	255	1 x 1/1	13 x 13 x 512	13 x 13 x 255
16	YOLO				
17	Route 13				
18	Convolutional	128	1 x 1/1	13 x 13 x 256	13 x 13 x 128
19	Up-sampling		2 x 2/2	13 x 13 x 128	26 x 26 x 128
20	Route 19 8				
21	Convolutional	256	3 x 3/1	13 x 13 x 384	13 x 13 x 256
22	Convolutional	255	1 x 1/1	13 x 13 x 256	13 x 13 x 256
23	YOLO				

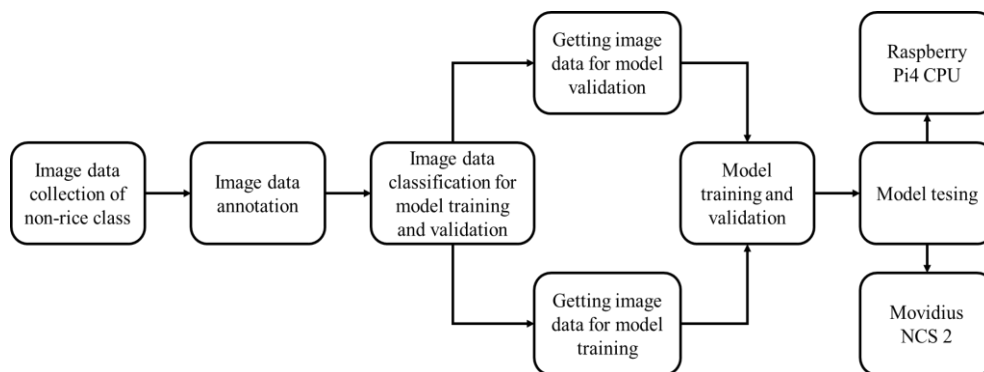


Figure 1. Research stages

At this stage, the image data used for training and model validation is annotated first. Furthermore, the data is separated between image data used as training data and image data used as validation data. After the image data is divided, the training model begins. Finally, after the trained model was obtained, the model was tested using two computational methods: the CPU from the Raspberry Pi 4 model B and the Intel Movidius NCS 2 to get computational speed data between the two methods.

As a detection model evaluation method, the mean average precision (mAP) is used in this study to support data analysis, because mAP is a popular measurement method to obtain the accuracy of the object detector [11]. The mAP value is obtained from the average value of the total average precision (AP) of each class, where the AP is determined from the precision-recall (PR) curve obtained from the association of each detection instance with overlapping ground truths. The mAP value is represented by the equation $mAP = \frac{1}{N} \sum_{i=1}^N AP_i$, where N is the total number of classes. The YOLO loss function exploits sum-squared error to calculate loss between the ground truth and the prediction, which contains localization loss, confidence loss, and classification loss [19]. The loss function is represented by (1).

$$\begin{aligned}
 Loss = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{(c \in classes)} (P_i(c) - \hat{P}_i(c))^2
 \end{aligned} \tag{1}$$

Where S represents cell grid to predict B bounding box in coordinate x, y, w, h . $\mathbb{1}_{ij}^{obj}$ value is 1 if the j^{th} bounding box in cell i is responsible to detect the object, otherwise will produce value 0, while λ_{coord} used to increase weight for loss in bounding box coordinates. C_i represents the confidence score of the box j in cell i , while λ_{noobj} denotes weight down the loss when background is detected, and also, $P_i(c)$ represents the conditional class probability for class c in cell i . The model tested by detecting the spreading rice containing gravel. For the model testing purpose, the rice was loaded by hand and spread on the table that had been set before by installing webcam above the table.

2.2. Hardware and software

To support this research, hardware and software are needed in testing the gravel detection process. The YOLOv3-tiny model was trained using a personal computer (PC) with an intel core i5-8400 2.8 GHz processor, 8 gigabytes of random access memory (RAM), and an NVIDIA GeForce GTX 1070 Ti graphics processing unit (GPU). The hardware implementation used is a Raspberry Pi 4 model B with 4 gigabyte RAM equipped with a broadcom BCM2711 processor, quad-core cortex-A72 (ARM v8) 64-bit system-on-a-chip (SOC) @ 1.5 GHz. In comparison to the processing speed test, the hardware is provided with a MOVIDIUS NCS 2 with Intel Movidius myriad X vision processing unit (VPU) with 16 SHAVE cores (128-bit very long instruction word (VLIW) vector processors). In addition to embedded systems and processing components, on the hardware aspect, the logitech C525 webcam is used in this study to record video as data to be processed with 720 p resolution. NCS 2 is used in this study because, so far, the neural compute stick is quite successful in handling CNN-based object detection processing [24], [25].

Software configuration is also applied in this study to support hardware performance. On Raspberry Pi 4 model B, Debian v11 based RaspIoT Bullseye is implemented. RaspIoT is equipped with the Python, OpenCV, and OpenVino programming languages as a framework to help test the computing speed between the CPU and NCS 2. Meanwhile, to assist the model training process on a PC, CUDA 9.0, and cuDNN 7.5 were implemented equipped with *labeling* as a means of labeling data and a Darknet framework to run the YOLOv3-Tiny algorithm training process.

3. RESULTS AND DISCUSSION

The Raspberry Pi 4 model B-based hardware in this test was assembled with and without Intel Movidius NCS 2. This was done to support 2 test methods. The first test is the operation of the Raspberry Pi 4 model B to run computing on the CPU-based object detection process. In comparison, the second test is the operation of the Raspberry P4 model B to run computing on the object detection process based on Intel Movidius NCS 2.

3.1. The results of training and model validation

The hyperparameter configuration of the YOLOv3-tiny algorithm was obtained and worked well for the training data process with decay settings of 0.0005, momentum 0.9, saturation 1.5, exposure 1.5, and Hue 0.1. The learning rate in the training process is set at 0.001 and the training model is recorded in every 1000 iterations with a maximum iteration of 15000. The training and validation process results are loss values that affect the object detection process in the image. The model training result means that the smaller the resulting

loss value and the larger the mAP value, the better the model detecting objects in the new image data. In the training process, the YOLOv3-tiny configuration was executed with 15000 iterations. The training process result can be seen in Figure 2. Figure 2 shows that the model training process minimized the loss value of 2.14 in the 1000th iteration to 0.27 in the 15000th iteration.

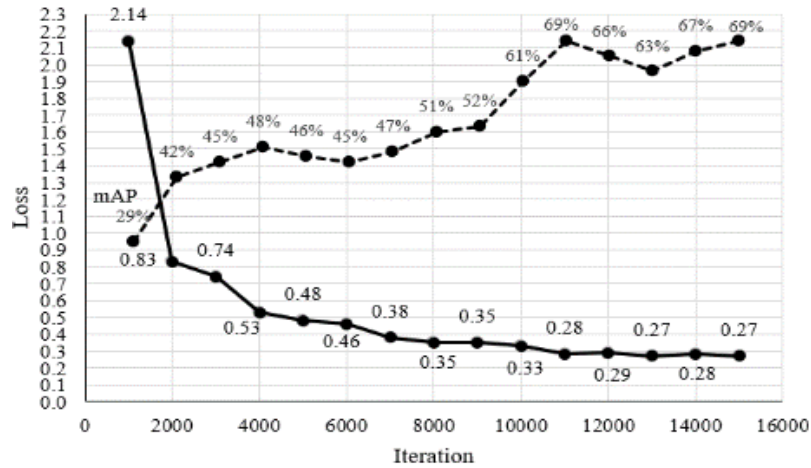


Figure 2. The results of the model training process

3.2. Model test results

The model that has been trained is tested using 2759 frames of video with a resolution of 1920 x 1080 px, which each frame represented in orange dots for Movidius NCS 2 processing and blue dots for Raspberry Pi 4 model B CPU processing. The test results using the Raspberry Pi 4 model B CPU obtained an object detection processing speed of 1.474 frames per second (FPS). On the other hand, testing using the Raspberry Pi 4 model B with Intel Movidius NCS 2, the object detection processing speed was obtained at 1.868 FPS. The graph of the test results can be seen in Figure 3.

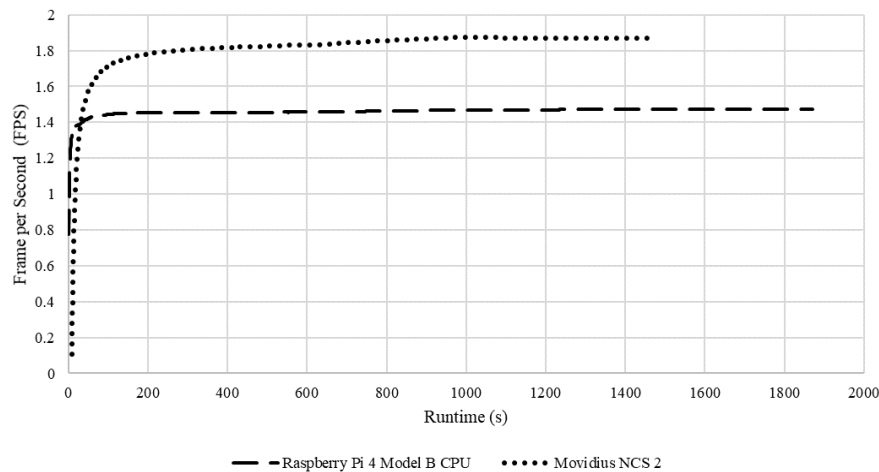


Figure 3. The comparison of object detection processing speed

3.3. System test results

After the training process was complete, the model was tested on a Python and OpenCV-based detection and classification system using image data that the model had not recognized. Model testing in this study was carried out using a video containing 2759 frames with a resolution of 1920 x 1080 px. Image data taken from output video results can be seen in Figure 4. Test results using video data on; 3 objects in Figure 4(a), 4 objects in Figure 4(b), 5 objects in Figure 4(c), 6 objects in Figure 4(d), 7 objects in Figure 4(e), and 8 objects in Figure 4(f).

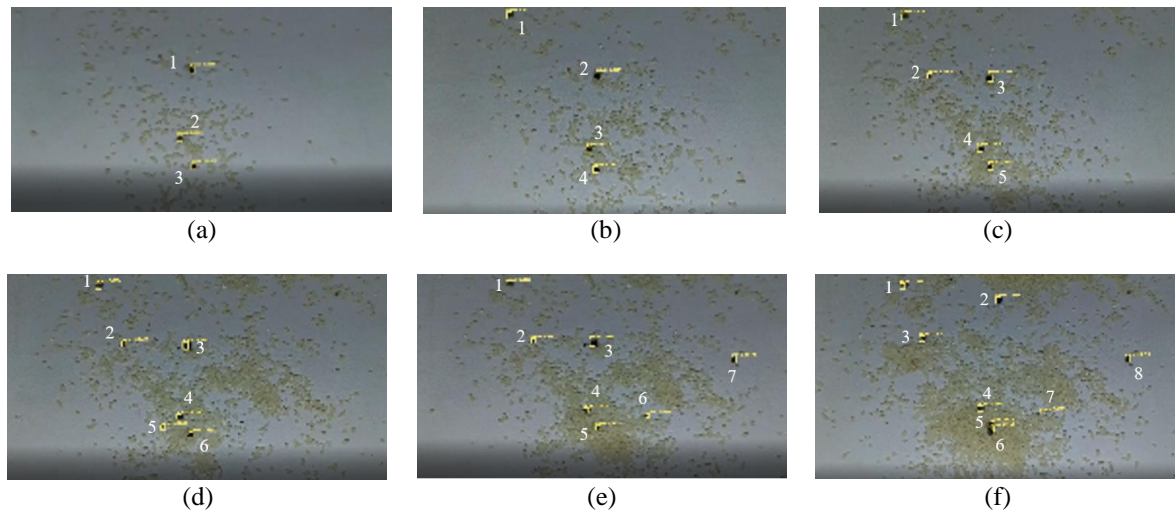


Figure 4. Test results using video data on (a) 3 objects, (b) 4 objects, (c) 5 objects, (d) 6 objects, (e) 7 objects, and (f) 8 objects

4. CONCLUSION

The test results found that the object detection processing speed increased by 1.27x on the Intel Movidius NCS 2 implementation compared to processing using the CPU from the Raspberry Pi 4. Object detection processing on video data was completed in 1871.408 seconds with 1,474 FPS using the CPU from the Raspberry Pi 4 model B and finished at 1477.141 seconds with 1.868 FPS using Movidius NCS 2. From these differences, it can be seen that the application of Intel Movidius NCS 2 succeeded in increasing object detection processing in this study by 26.69% with the tiny-YOLOv3 model approach on the Raspberry Pi 4 model B embedded system. In this study, NCS exploitation to get fps improvement in object detection task has been done successfully, and we can potentially use it with some end effector mechanisms to pick out the detected non-rice objects in further research. Furthermore, other parallel processing using more than one NCS as well as modified algorithm can be used to increase the fps for real-world applications.

ACKNOWLEDGEMENTS

The authors would like to thank the Faculty of Engineering, Universitas Katolik Indonesia Atma Jaya, and PT. Guna Elektro for fully supporting this work.




REFERENCES

- [1] K. Noinan, S. Wicha, P. Sureepong, and R. Chaisricharoen, "Requirements Analysis and Design for Thai Beef-Cattle Farm Management System," *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*, 2021, pp. 356–359, doi: 10.1109/ECTIDAMTNC51128.2021.9425779.
- [2] R. Alfred, J. H. Obit, C. P. -Y. Chin, H. Haviluddin, and Y. Lim, "Towards Paddy Rice Smart Farming: A Review on Big Data, Machine Learning, and Rice Production Tasks," in *IEEE Access*, vol. 9, pp. 50358–50380, 2021, doi: 10.1109/ACCESS.2021.3069449.
- [3] Y. -C. Lee and C. -M. Lee, "Real-Time Smart Home Surveillance System of Based on Raspberry Pi," *2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 2020, pp. 72–74, doi: 10.1109/ECICE50847.2020.9301929.
- [4] V. Simadiputra and N. Surantha, "Rasefiberry: Secure and efficient Raspberry-Pi based gateway for smarhome IoT architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 2, pp. 1035–1045, April 2021, doi: 10.11591/eei.v10i2.2741.
- [5] T. M. Bandara, W. Mudiyansele, and M. Raza, "Smart farm and monitoring system for measuring the Environmental condition using wireless sensor network-IOT Technology in farming," *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*, 2020, pp. 1–7, doi: 10.1109/CITISIA50690.2020.9371830.
- [6] W. -L. Chen, Y. -B. Lin, F. -L. Ng, C. -Y. Liu, and Y. -W. Lin, "RiceTalk: Rice Blast Detection Using Internet of Things and Artificial Intelligence Technologies," in *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1001–1010, February 2020, doi: 10.1109/JIOT.2019.2947624.
- [7] Y. Zhang, D. Xiao, and Y. Liu, "Automatic Identification Algorithm of the Rice Tiller Period Based on PCA and SVM," in *IEEE Access*, vol. 9, pp. 86843–86854, 2021, doi: 10.1109/ACCESS.2021.3089670.
- [8] S. D. Fabiyi *et al.*, "Varietal Classification of Rice Seeds Using RGB and Hyperspectral Images," in *IEEE Access*, vol. 8, pp. 22493–22505, 2020, doi: 10.1109/ACCESS.2020.2969847.
- [9] D. B. C. Lima, R. M. B. da S. Lima, D. de F. Medeiros, R. I. S. Pereira, C. P. de Souza, and O. Baiocchi, "A Performance Evaluation of Raspberry Pi Zero W Based Gateway Running MQTT Broker for IoT," *2019 IEEE 10th Annual Information*




- Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2019, pp. 0076-0081, doi: 10.1109/IEMCON.2019.8936206.
- [10] P. Ramos-Giraldo *et al.*, "Low-cost Smart Camera System for Water Stress Detection in Crops," *2020 IEEE SENSORS*, 2020, pp. 1-4, doi: 10.1109/SENSORS47125.2020.9278744.
- [11] A. Hryvachevskiy, S. Fabirovskyy, I. Prudyus, L. Lazko, and J. Matuszewski, "Method of Increasing the Object Detection Probability by the Multispectral Monitoring System," *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, 2019, pp. 77-80, doi: 10.1109/CADSM.2019.8779348.
- [12] K. R. B. Legaspi, N. W. S. Sison, and J. F. Villaverde, "Detection and Classification of Whiteflies and Fruit Flies Using YOLO," *2021 13th International Conference on Computer and Automation Engineering (ICCAE)*, 2021, pp. 1-4, doi: 10.1109/ICCAE51876.2021.9426129.
- [13] R. A. Setyawan, A. Basuki, and C. Y. Wey, "Machine Vision-Based Urban Farming Growth Monitoring System," *2020 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, 2020, pp. 183-187, doi: 10.1109/EECCIS49483.2020.9263449.
- [14] B. N. Rohith, "Computer Vision and IoT Enabled Bot for Surveillance and Monitoring of Forest and Large Farms," *2021 2nd International Conference for Emerging Technology (INCET)*, 2021, pp. 1-8, doi: 10.1109/INCET51464.2021.9456180.
- [15] O. Aleksandrova and Y. Bashkov, "Face recognition systems based on Neural Compute Stick 2, CPU, GPU comparison," *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*, 2020, pp. 104-107, doi: 10.1109/ATIT50783.2020.9349313.
- [16] A. Pester and M. Schrittmesser, "Object detection with Raspberry Pi3 and Movidius Neural Network Stick," *2019 5th Experiment International Conference (exp.at'19)*, 2019, pp. 326-330, doi: 10.1109/EXPAT.2019.8876583.
- [17] Y. Xie, L. Ding, A. Zhou, and G. Chen, "An Optimized Face Recognition for Edge Computing," *2019 IEEE 13th International Conference on ASIC (ASICON)*, 2019, pp. 1-4, doi: 10.1109/ASICON47005.2019.8983596.
- [18] D. Brown, "Mobile Attendance based on Face Detection and Recognition using OpenVINO," *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, 2021, pp. 1152-1157, doi: 10.1109/ICAIS50930.2021.9395836.
- [19] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.
- [20] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, "Proceedings of the IEEE conference on computer vision and pattern recognition", 2017, pp. 7263-7271.
- [21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, April 2018, doi: 10.48550/arXiv.1804.02767.
- [22] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging*, vol. 9, no. 4, pp. 611-629, June 2018, doi: 10.1007/s13244-018-0639-9.
- [23] W. He, Z. Huang, Z. Wei, C. Li, and B. Guo, "TF-YOLO: An Improved Incremental Network for Real-Time Object Detection," *Applied Sciences*, vol. 9, no. 16, p. 3225, doi: 10.3390/app9163225.
- [24] K. Muchtar *et al.*, "Embedded-based Tomato Septoria Leaf Detection with Intel Movidius Neural Compute Stick," *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, 2021, pp. 907-908, doi: 10.1109/GCCE53005.2021.9621829.
- [25] S. P. Kaarmukilan, A. Hazarika, A. Thomas K., S. Poddar, and H. Rahaman, "An Accelerated Prototype with Movidius Neural Compute Stick for Real-Time Object Detection," *2020 International Symposium on Devices, Circuits and Systems (ISDCS)*, 2020, pp. 1-5, doi: 10.1109/ISDCS49393.2020.9262996.

BIOGRAPHIES OF AUTHORS






Nova Eka Budiayanta    got his Bachelor Degree in Mechatronics Engineering Education from Universitas Negeri Yogyakarta, then pursuing his Master Degree in Electrical Engineering Education from Universitas Negeri Yogyakarta and Electrical Engineering Master Program from Universitas Katolik Indonesia Atma Jaya. He has experience in hardware-software programming. He is now acting as a lecturer of Department of Electrical Engineering, Universitas Katolik Indonesia Atma Jaya, concerned with image processing, robotics, and machine learning research field. He can be contacted at email: nova.eka@atmajaya.ac.id.



Catherine Olivia Sereati    is a lecturer and researcher at Universitas Katolik Indonesia Atma Jaya. Catherine's interest subject of researches are electronic instrumentation system and system on chip (SoC). She was also involved in several research projects to design cognitive instrumentation systems. Some of them are a building a software cognitive interpretation of ship movements, for Indonesian marine security purposes, and cognitive electro cardiograph (ECG) design. Currently her research project is focusing to designing the architecture of cognitive processor. She can be contacted at email: catherine.olivia@atmajaya.ac.id.



Ferry Rippun Gideon Manalu    is a lecturer and researcher at Universitas Katolik Indonesia Atma Jaya. Ferry's interest subject of research are Robotics, Autonomous Vehicle, and Control System. He was also involved in various research projects in design and implementation of autonomous vehicles such as soccer robot algorithm and autonomous vehicle for the blind. Currently, his research projects focus in the implementation path planning for multi robot. He can be contacted at email: ferry.rippun@atmajaya.ac.id.