Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 9, No. 4, December 2023, pp. 1168-1178
ISSN: 2338-3070, DOI: 10.26555/jiteki.v9i4.27402 

# Design Human Object Detection Yolov4-Tiny Algorithm on ARM Cortex-A72 and A53

Rachmat Muwardi[1], Ahmad Faizin[1], Puput Dani Prasetyo Adi[2], Rizky Rahmatullah[3], Yanxi Wang[4], Mirna Yunita[5], Dendi Mahabror[2]
[1]Department of Electrical Engineering, Universitas Mercu Buana, Jakarta, Indonesia
[2]National Research and Innovation Agency, Indonesia
[3]School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing, China
[4] School of Electronics and Information Engineering, Beijing Institute of Technology, Beijing, China
[5]School of Computer Science and Technology, Beijing Institute of Technology, China

**ARTICLE INFO**

**Article history:**

Received October 18, 2023
Revised December 06, 2023
Published January 05, 2024

**Keywords:**

YOLOV4 Tiny;
Optimization;
Image Processing;
Computer Vision;
Object Detection;
ARM Processor

**ABSTRACT**

Currently, many object detection systems still use devices with large sizes, such as using PCs, as supporting devices, for object detection. This makes these devices challenging to use as a security system in public facilities based on human object detection. In contrast, many Mini PCs currently use ARM processors with high specifications. In this research, to detect human objects will use the Mini PC Nanopi M4V2 device that has a speed in processing with the support of CPU Dual-Core Cortex-A72 (up to 2.0 GHz) + Cortex A53 (Up to 2.0 GHz) and 4 Gb DDR4 Ram. In addition, for the human object detection system, the author uses the You Only Look Once (YOLO) method with the YoloV4-Tiny type, With these specifications and methods, the detection rate and FPS score are seen which are the feasibility values for use in detecting human objects. The simulation for human object recognition was carried out using recorded video, simulation obtained a detection rate of 0.9845 or 98% with FPS score of 3.81-5.55. These results are the best when compared with the YOLOV4 and YOLOV5 models. With these results, it can be applied in various human detection applications and of course robustness testing is needed.



**Corresponding Author**:

Rachmat Muwardi, Department of Electrical Engineering, Universitas Mercu Buana, Jakarta, Indonesia
Email: rachmat.muwardi@mercubuana.ac.id

## 1. INTRODUCTION

In the current era of globalization, technological developments are very rapid, and a dependence on technology cannot be denied. Several rapidly developing technologies include technology systems on microprocessors in various electronic devices. These security systems use IoT-based cameras, attendance tools that use facial detection and security systems that follow an object, and other technological systems. One of the security systems that is developed and is experiencing rapid development is the system that uses object recognition identification. In this system, the program recognizes objects caught by the camera with high speed and accuracy [1], [2], [3].

Object recognition systems have been developed in various fields, such as object recognition to detect the quality of crops on agricultural land, the usage in security systems, and multiple ways. This object recognition system can also determine how many objects, such as humans, are caught by a camera in one place. This can be useful for limiting the number of visitors or a monitoring system to survey how many people are visiting at a time [4]-[7].

In object recognition, a hardware system with high specifications is needed, especially regarding the microprocessor on said device. A microprocessor is an IC chip that connects the core functions of a computer's central processing unit (CPU) or the central system that performs data processing on a computer [8], [9], [10].

Microprocessors can be programmed to receive digital data as an input which will be processed according to the instructed program and output. As a result data. The microprocessor reads the program sequentially to run according to the user's or programmer's wishes. Various programming languages can be used to provide commands so that the microprocessor can run them, such as the VHDL, Verilog, and Python programming languages, etc [11]-[14].

To date, the implementation of object recognition using digital image processing using the YOLO algorithm requires hardware with high specifications, especially in the ARM Processor for architecture for computer processors. This device with high specifications and suitable algorithm aims to obtain optimal results when conducting training on the dataset that has been collected to get a high accuracy score during implementation. In addition, using an Arm Processor with high specifications can result in digital image processing on videos that do not experience a decrease in quality, such as a decrease in the fps of the video, either in real-time or inputting the video recordings manually [15]-[19].

Solve the problems using multi-thread. There will be some issues and questions regarding this multithread, such as what image will be processed, what is the purpose of image processing, how to process images, how to make a detailed and correct image processing framework, how to make the core work regularly and directed, and utilize core tools with high efficiency when using the image processing framework real-time. The following section will address and explain all the problems mentioned above. In this work, the image processing optimization methodology will be described. Every Biometric system has four main features: Human Detection, Preprocessing, Feature Extraction, and Human Recognition, which are shown in Fig. 1. Capturing an image is the first task of a human recognition system. Images are captured by video, camera, or from a database. This image is provided for the next step of the human recognition system. The computer is supported by the software, the photo by the device sampled, and the output results or data of the recognized human image. Finally, the output data can be displayed on the screen. The data output will be identified to see the accuracy of each algorithm used.

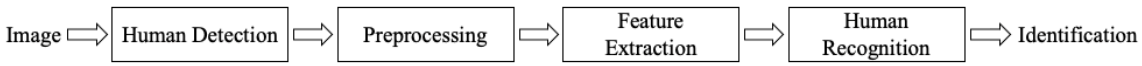



**Fig. 1.** The architecture of human recognition system

Therefore, to make a device that can be used to detect objects and find out how accurately the device detects objects with which is a low cost and portable. In this study, use a camera device and ARM Processor from the NanoPi M4V2, which has medium to high specifications and has a reasonably affordable price. In addition, the NanoPi M4V2 can also be connected to a screen that can display the results from the camera that has been processed on the microprocessor so it can display images from the object detection, accuracy score, and the number of images displayed sequentially in each second or fps (frame per second) obtained from the camera. Hence the author makes a device that can identify objects in real-time and video input.

## 2. METHODS

### 2.1. YOLOV4 TINY

Which stands for You Only Look Once, is a method to perform an image detection process with a different algorithmic approach compared to other methods. Yolo cannot only classify objects but can also determine the position of objects with accuracy in real-time. Joseph Redmo first developed Yolo, and it was created several times until there existed various versions such as VIZ, YoloV2, YoloV3, YoloV4, dan YoloV5, and in all of these versions, there is a Tiny version which is a compressed result of the Yolo version with a simpler algorithm and light to execute so that it doesn't burden the PC [20]-[23]. Based on various research that has been released shows that the YoloV4 Tiny is the most effective and light version of Yolo by far.

Reflecting on the development and usage of the previous version of Yolo, which has been successfully implemented as an object detection algorithm with a high confidence level to detect an object [25]-[28].

YOLOv4-tiny is a compressed version of YOLOv4 that makes the network structure simpler and reduces parameters making it feasible for development on mobile and embedded devices. YoloV4 Tiny has a different architecture, as seen in Fig. 2. which shows the architecture of YoloV4 Tiny consists of a backbone which is the training result from an artificial neural network on the image net as well as a head (neck) which is used to calculate classification and determine boxes on objects [29]-[33].

To be able to run on a Graphic Processing unit (GPU), the backbones that can be used include VGG, ResNet, and ResNext, while Squize net, mobile net, or shuffle net can run on a Central processing unit (CPU). For the head, which is a sub of the backbone, there are two parts: 1) Stage detector/Dense prediction (Yolo, SSD, Retina Net) and 2) two-stage detector/Spars prediction (Fast R-CNN, Faster R-CNN). For real-time

object detection, YOLOv4-tiny is better than YOLOv4 because faster inference time is more important than precision or accuracy when working with real-time object detection environments.

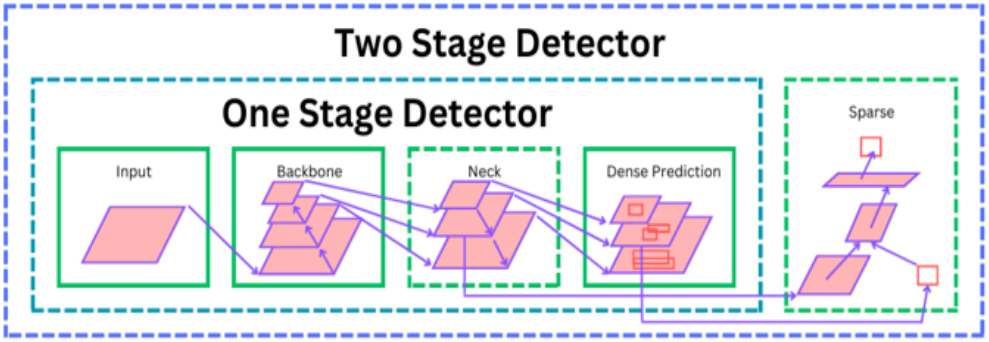



**Fig. 2.** YOLOV4 Tiny Architecture

## 2.2. Open CV

Library of programming functions mainly aimed at real-time computer vision. Advance vision research by providing open and optimized code for basic vision infrastructure. No more reinventing the wheel. Disseminate vision knowledge by providing a shared infrastructure that developers could build on, making code more readily readable and transferable. Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that does not require code to be open or accessible itself [34], [35], [36].

OpenCV is a trendy open-source real-time computer vision library initially developed by Intel and now maintained by Willow Garage. It is a collection of highly optimized libraries containing more than 500 algorithms and is designed to take advantage of Intel Integrated Performance Primitives (IPP) for better performance. This work focuses on the OpenCV1.0 version, which provides five main library components [37], [38], [39].

The structure of OpenCV is shown in Fig. 3. The basic structure of OpenCV. The CV component contains a collection of computer vision and image processing algorithms. The Machine Learning Library (MLL) contains statistical classifiers and clustering tools, HighGUI provides routines for storing and loading video and images, and CvAux contains defunct and experimental algorithms. The core support, like basic data structures and low-level algorithmic support, is provided by the CXCORE library component. Since this work is based on the NanoPi M4V2, using the IPP library for better performance is impossible. Instead, OpenCV algorithms are redesigned to take advantage of the architecture present on the NanoPi M4V2. The reference algorithm cvMatchTemplate for correlation is taken for benchmarking from the OpenCV library defined in the spatial domain [40], [41], [42].

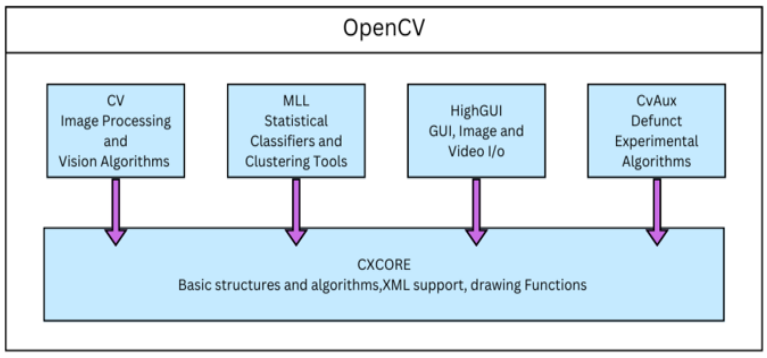



**Fig. 3.** The structure of OpenCV

## 2.3. Hardware Design and Analyst

Firstly, the program must import the library, then load the necessary dataset. We can identify whether the code is valid by running it in Python. Then we put the loop "While true" to avoid exiting by itself. Inside the loop, we put the core functionality. It is as follows: (1) capture video by frame, (2) detect multi-scale, (3) draw a rectangle around a human, (4) display the resulting frame.

The device is connected to the network, USB Camera (C922), and SSD through a GPIO pin. Then for, the native display, it relates to the LG monitor through HDMI port. The program design is as follows in Fig. 4. and Fig. 5.

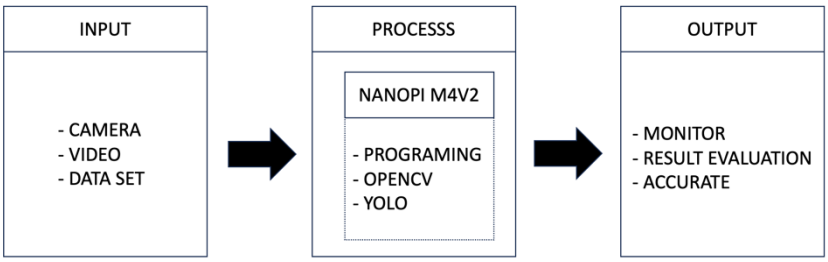



**Fig. 4.** Diagram Block

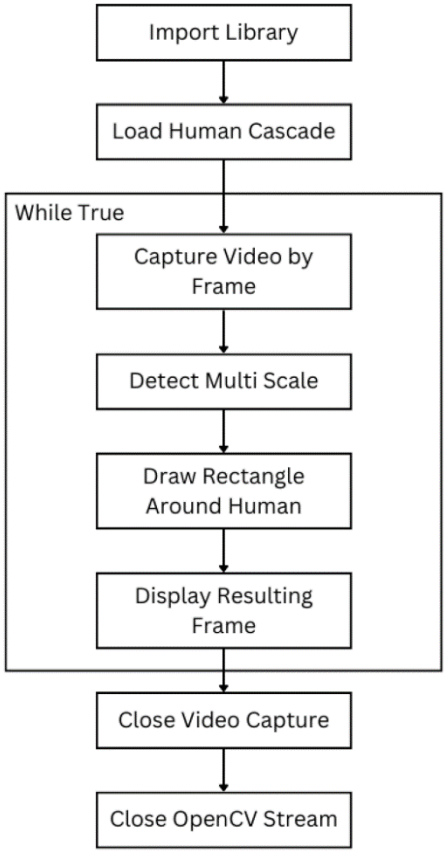



**Fig. 5.** Program Architecture

Input: in this section, there is an input device in the form of a camera that takes pictures in real-time. Aside from that, a video is also entered into the process section to be analyzed. Lastly, in the form of a datasheet, the data sheet section contains the calculation results from the object detection being investigated and used as the Real score of the video, which will later be compared with the reading results from the process section. Process: in this section, use a laptop containing software to process Python, OpenCV, and YoloV4-Tiny programs. The program analyzes input in video or in real-time from the camera. Output: In this section, a monitor is used to display the object detection results with an algorithm embedded in the laptop to process object detection. The reading results are shown as an image on the monitor and data on the datasheet. The datasheet output contains the score obtained from the processing results in the form of Recall, Precision, Accurate, the number of detected objects, and the FPS of the processed video.

Initial setting handler. This function is necessary for creating the initial setting or applying the saved set. This function will also handle various system calls needed for the framework's initial start. Optionally, this will also contain the necessary process for analysis as a time recorder: processor core and hardware identification.

The framework needs to identify the specification of the processor. Then this function will create the hardware configuration file that the framework will utilize.

The process will also prepare the other entity's necessary system call and hardware utilization function—image retrieval by a camera. Image data will be retrieved by this function inside the framework. This function will analyze and settle the hardware communication and system call necessary for the framework. Core utilization analysis. This function will monitor the processor's utilization data and determine the available core. We can set the policy for each core's utilization. For example, the core with high utilization will not be utilized for the task. This function also provides data for the workload assigner function. Workload division mapping and assigner for each core. This function will receive data from the core utilization analysis function. This will determine which task that will be executed by which core. This function will manage the assignment on the OS layer and process utilization, so it can pinpoint to which core the given task will run. Execute image processing task. The image processing function will run at a given core.

Retrieve and arrange the raw result and data of the task. Because of the parallelization, the development of each core's process will be retrieved, and those will be combined into necessary data for further processing. Further result processing. This function will handle the result finalization and data shaping as intended. This function will check the finalized data and running time of one cycle of the framework. This function will also handle functions necessary for analysis purposes.

This framework aims to make human recognition and utilize multiprocessors with different architectures so that analysis can be done by running it head-to-head with the default configuration. The framework design allows the entity to work independently by dividing each entity into separate executable functions or in asynchronous order before specified instructions. The algorithm shown in Fig. 6. is as follows.

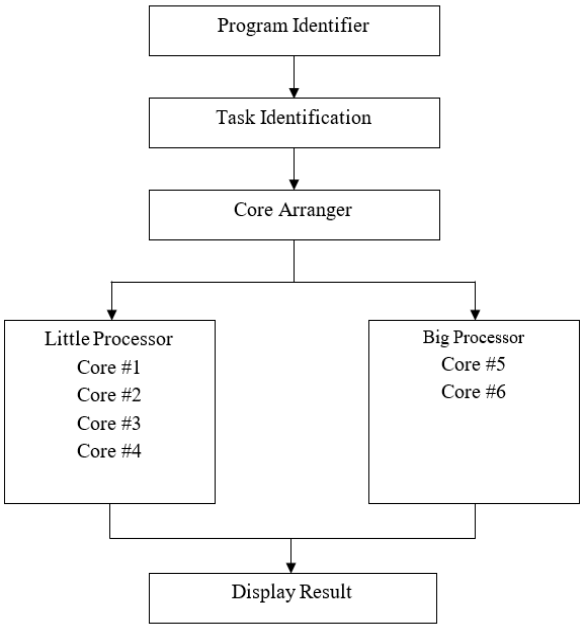



**Fig. 6.** Workflow the process

### 2.4. Algorithm Programming

The YOLO algorithm applies the function to train images through 4 main stages: (1) determining YOLO features, (2) making integral images, (3) Ad boost training, and (4) linking with human classifiers. This article will share a short tutorial using the YOLO algorithm to discuss which areas contain human recognition in images using OpenCV.

The neural networks give a nonlinear solution to the human recognition problem. The basic idea is to consider a net with a neuron for every pixel in the image. The advantage of neural networks in classification over linear ones is that they can reduce misclassifications among the neighborhood classes. The Neural Network contains neurons and can carry the data from the input to the output. The data is processed and calculated as the weighted sum of the information applied as a non-linear function.

(1) Initial setting handler;
(2) Processor core and hardware identification;
(3) Image retrieval by camera;

(4) Core utilization analysis;
(5) Workload division mapping and assigner for each core;
(6) Execute image processing task;
(7) Retrieve and arrange the raw result and data of the task;
(8) Further result processing;
(9) Result analysis.

## 3. RESULTS AND DISCUSSION

This section will compare three existing algorithms: YOLOV4, YOLOV4 TINY, and YOLOV5. The experiment will be tested using Nanopi M4V2. The parameters used for each algorithm use an image scale of 640×640 with a total of 480 frames using a video of the same duration and the same image.

Testing this level of accuracy is used to determine the level of accuracy of the tool that has been made so that it can be said that the device can be used properly with a perfect level of accuracy. From the existing calculations, the results will be obtained: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). If you get it, you can consider the existing Recall (R) and Precision (P) to determine an algorithm's accuracy level. The definition is as follows.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Mean Average Precision (mAP) is used to evaluate the model. The purpose of mAP is as follows:

$$mAP = \frac{1}{N}\sum_{t=1}^{N} APi \tag{3}$$

Discusses the video that was run for 20 seconds with 24 FPS (Total of 480 FPS). Can be detected a human image object using a video. This can be done using a real-time camera without reducing the accuracy level. the YOLOV4 Tiny algorithm obtains satisfactory results because it obtains improvisation from the big core and little core so that the processing is more focused and improves the processor performance much better so that it can beat YOLOV5 that is only implemented without any improvisation process in that ARM Processor.

Fig. 7 explains the evaluation of the human object model, which is run with several algorithms that are used. It can be seen that YOLOV4-Tiny is still superior to the others because there is processor improvisation, so the image process is much better and more remarkable compared to the YOLOV5 version, which is superior to YOLOV4-Tiny.

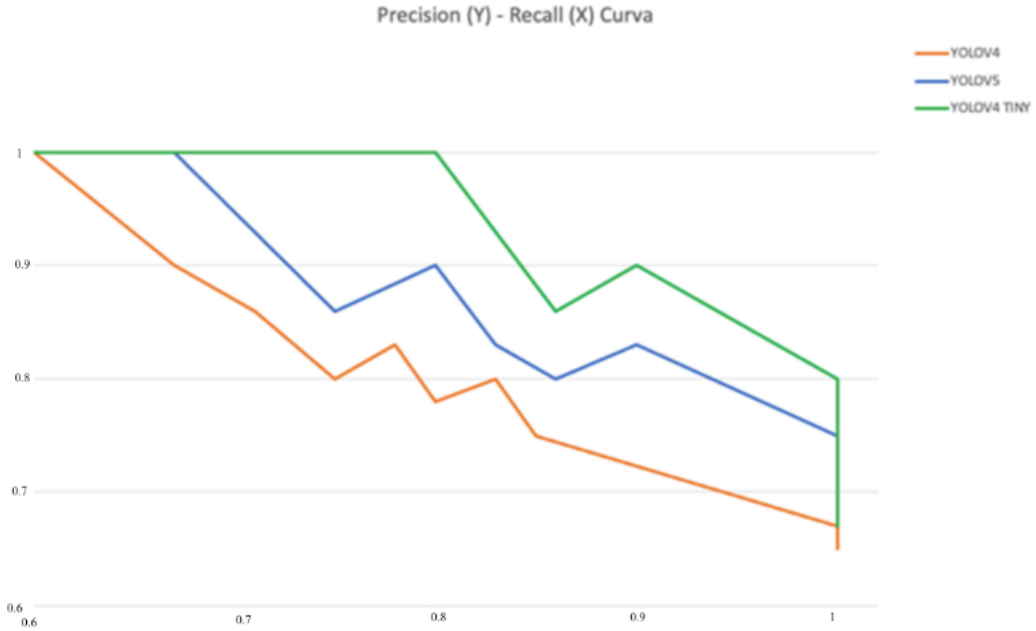
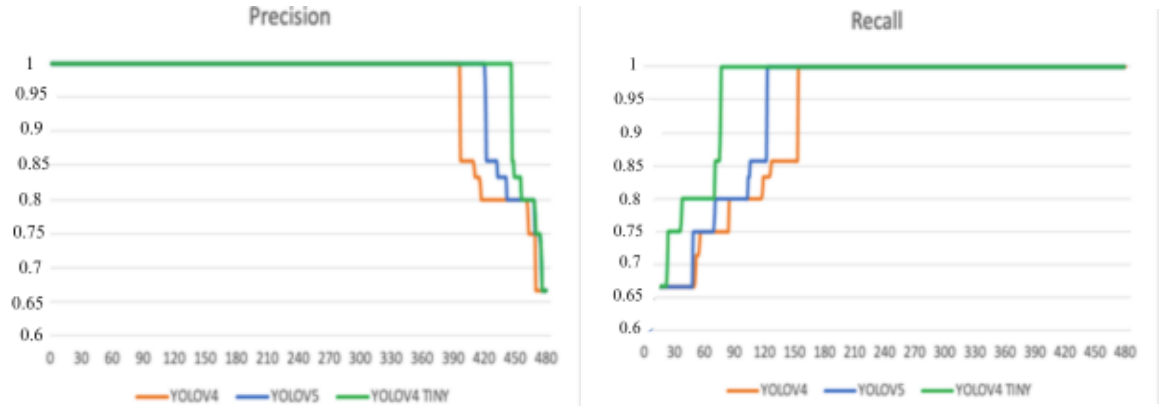



**Fig. 7.** Precision VS Recall Curva

Evaluation of the human object model that was run with several algorithms is precise that YOLOV4-Tiny is still relatively superior to the others because there is an improvised processor, so the image process is much better and superior to YOLOV5.

Evaluation will also explain that each existing FPS has recall and precision. in Fig. 8. explains each recall and precision to explain the testing results of each calculation that appears in each video frame with a duration of 16 seconds (1 Second / 30 FPS).




(a) (b)

**Fig. 8.** (a) Precision for 480 FPS (b) Recall for 480 FPS

The process image is in Fig. 9. The video will be processed by the program, and the processing results will be displayed in the form of the video's value, the accuracy level, and the detected object's name.

Table 1 describes the mAP of an algorithm used and the average time needed to read images with a total of 480 frames (24 FPS) contained in a video with a duration of 20 seconds.

The author will discuss the performance of RAM (Random Access Memory) and CPU (Central Processor Unit) when the digital image processing is executed in Fig. 10.

Explaining the image data obtained, the CPU usage on the device is not too high or not up to 100%, and the RAM that is used is only 25% of 100% or 1.02Gb of the available 4Gb RAM aside from that, the temperature on the device also does not overheat. So overall, the program and device run well, and there are no problems with their usage.


**Fig. 9.** Human Detection

**Table 1.** Evaluation Result

| Model | YOLOV4 | YOLOV5 | YOLOV4 TINY |
|---|---|---|---|
| mAP | 0.9632 | 0.9751 | 0.9845 |
| *Average Time* | 55 ms/frame | 50 ms/frame | 47 ms/frame |

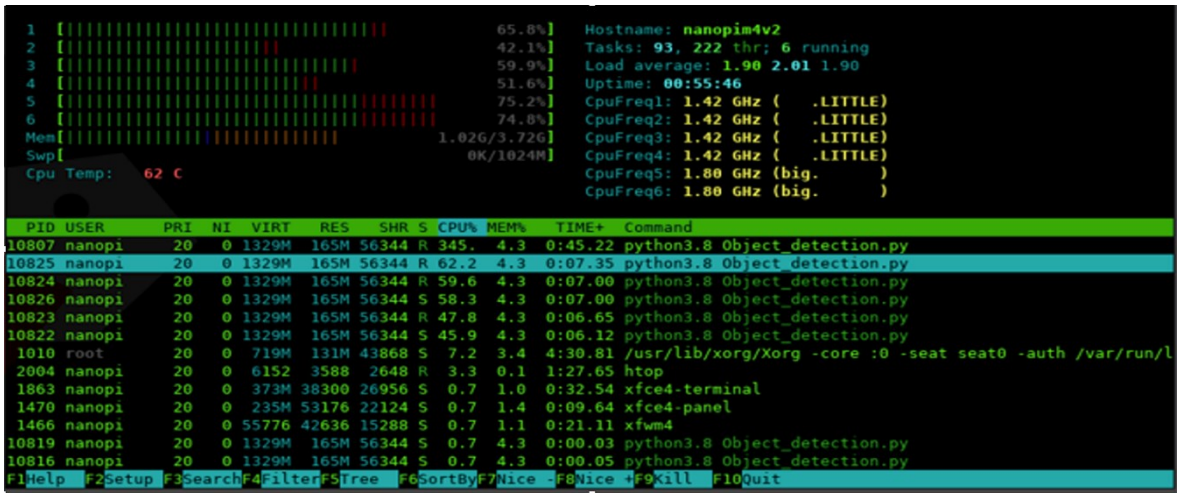



**Fig. 10.** YOLOV4 Tiny Performance

## 4. CONCLUSION

The results of the experiments that have been carried out, it can be concluded that the device and system that have been made can work well to detect human objects. The NanoPi M4V2 device has an FPS (Frame Per Second) rate reasonable for digital image processing in video format with the YOLOV4 Tiny algorithm. The accuracy level in the YOLOV4 Tiny algorithm using the NanoPi M4V2 device has a high level of accuracy with video input with brighter lighting. When the device is run, there is no problem with the CPU usage on the NanoPi M4V2 using the YOLOV4-Tiny algorithm to detect human objects.

## Acknowledgments

The first special thanks go to Universitas Mercu Buana for supporting foreign collaborative research, the second to the National Research and Innovation Agency Indonesia. Third to the Beijing Institute of Technology, Mirna Yunita, and Galatia Erica Yehezkiel for her help and cooperation during this research. There will always be a paper collaboration with Beijing Institute of Technology and National Research and Innovation Agency in further research.

## REFERENCES

[1] F. S. Leira, H. H. Helgesen, T. A. Johansen, T. I. Fossen, "Object detection, recognition, and tracking from UAVs using a thermal camera," *Journal of Field Robotics*, vol. 38, no. 2, pp. 242-267, 2021, https://doi.org/10.1002/rob.21985.

[2] F. Becattini, F. Palai and A. D. Bimbo, "Understanding Human Reactions Looking at Facial Microexpressions With an Event Camera," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9112 - 9121, 2022, https://doi.org/10.1109/TII.2022.3195063.

[3] R. Muwardi, H. Gao, H. U. Ghifarsyam, M. Yunita, A. Arrizki and J. Andika, "Network Security Monitoring System Via Notification Alert," *Journal of Integrated and Advanced Engineering (JIAE)*, vol. 1, no. 2, 2021, https://doi.org/10.51662/jiae.v1i2.22.

[4] R. Muwardi, H. Qin, H. Gao, H. U. Ghifarsyam, M. H. I. Hajar and M. Yunita, "Research and Design of Fast Special Human Face Recognition System Publisher: IEEE Cite This," *in 2nd International Conference on Broadband Communications, Wireless Sensors and Powering (BCWSP)*, pp. 68-73, 2020, https://doi.org/10.1109/BCWSP50066.2020.9249452.

[5] R. Muwardi, M. Yunita, H. U. Ghifarsyam and H. Juliyanto, "Optimize Image Processing Algorithm on ARM Cortex-A72 and A53," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika,* vol. 8, no. 3, pp. 399-409, 2022, https://doi.org/10.26555/jiteki.v8i3.24457.

[6] R. Muwardi, J. M. R. Permana, H. Gao and M. Yunita, "Human Object Detection for Real-Time Camera using Mobilenet-SSD," *Journal of Integrated and Advanced Engineering (JIAE),* vol. 3, no. 2, pp. 141-150, 2023, https://doi.org/10.51662/jiae.v3i2.108.

[7] K. Bhujbal and S. Barahate, "Custom Object detection Based on Regional Convolutional Neural Network & YOLOv3 With DJI Tello Programmable Drone," *in 7th International Conference on Innovation & Research in Technology & Engineering (ICIRTE)*, 2022, https://doi.org/10.2139/ssrn.4101029.

[8] P. Bombelli, A. Savanth, A. Scarampi, S. J. L. Rowden, D. H. Green, A. Erbe, E. Årstøl, I. Jevremovic, M. F. Hohmann-Marriott, S. P. Trasatti, E. Ozer and C. J. Howe, "Powering a microprocessor by photosynthesis," *Energy & Environmental Science*, vol. 15, no. 6, pp. 2529-2536, 2022, https://doi.org/10.1039/D2EE00233G.

[9] A. Y. Montoya, M. J. -Aparicio, J. H. -Alvidrez and M. J. Reno, "A Fast Microprocessor-Based Traveling Wave Fault Detection System for Electrical Power Networks," *in IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1-5, 2023, https://doi.org/10.1109/ISGT51731.2023.10066370.
[10] D. Naraharisetti, R. Karne, J. Weymouth and A. Wijesinha, "Obsolescence in Operating Systems and Microprocessors," in *IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 110-115, 2023, https://doi.org/10.1109/SERA57763.2023.10197809.
[11] A. S. a. N. A. V.R.S. Mani a, "Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using ZYNQ FPGA node," *Microelectronics Journal*, vol. 119, p. 105319, 2022, https://doi.org/10.1016/j.mejo.2021.105319.
[12] A. Ravikumar and H. Sriraman, "Acceleration of Image Processing and Computer Vision Algorithms," in *Handbook of Research on Computer Vision and Image Processing in the Deep Learning Era*, pp. 1-18, 2023, https://doi.org/10.4018/978-1-7998-8892-5.ch001.
[13] D. G. Bailey. *Design for embedded image processing on FPGAs*. John Wiley & Sons. 2023, https://doi.org/10.1002/9781119819820.
[14] V. K. Tiwari and M. Meribout, "A Hybrid FPGA-GPU-based Hardware Accelerator for High Throughput 2D Electrical Impedance Tomography," *Authorea Preprints,* 2023, https://doi.org/10.36227/techrxiv.22262332.v1.
[15] A. Mauri, R. Khemmar, B. Decoux, N. Ragot, R. Rossi, R. Trabelsi, R. Boutteau, J.-Y. Ertaud and X. Savatier, "Deep Learning for Real-Time 3D Multi-Object Detection, Localisation, and Tracking: Application to Smart Mobility," *Sensors*, vol. 20, no. 2, p. 532, 2020, https://doi.org/10.3390/s20020532.
[16] T. Diwan, G. Anirudh and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, pp. 9243–9275, 2023, https://doi.org/10.1007/s11042-022-13644-y.
[17] P. Jiang, D. Ergu, F. Liu, Y. Cai and B. Ma, "A Review of Yolo Algorithm Developments," *Procedia Computer Science*, vol. 199, pp. 1066-1073, 2022, https://doi.org/10.1016/j.procs.2022.01.135.
[18] M. Majumder and C. Wilmot, "Automated Vehicle Counting from Pre-Recorded Video Using You Only Look Once (YOLO) Object Detection Model," *Journal of Imaging,* vol. 9, no. 7, p. 131, 2023, https://doi.org/10.3390/jimaging9070131.
[19] Y. Chen, X. Yuan, R. Wu, J. Wang, Q. Hou and M.-M. Cheng, "YOLO-MS: Rethinking Multi-Scale Representation Learning for Real-time Object Detection," *arXiv preprint arXiv:2308.05480,* 2023, https://doi.org/10.48550/arXiv.2308.05480.
[20] Z. Jiang, L. Zhao, S. Li and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny," *Computer Vision and Pattern Recognition*, vol. 7, no. 1, 2020, https://doi.org/10.48550/arXiv.2011.04244.
[21] C. Guo, X.-l. Lv, Y. Zhang and M.-l. Zhang, "Improved YOLOv4-tiny network for real-time electronic component detection," Scientific Reports, vol. 11, p. 22744, 2021, https://doi.org/10.1038/s41598-021-02225-y.
[22] S.-J. Ji, Q.-H. Ling and F. Han, "An improved algorithm for small object detection based on YOLO v4 and multi-scale contextual information," *Computers and Electrical Engineering,* vol. 105, p. 108490, 2023, https://doi.org/10.1016/j.compeleceng.2022.108490.
[23] H. Zhu, W. Xie, J. Li, J. Shi, M. Fu, X. Qian, H. Zhang, K. Wang and G. Chen, "Advanced Computer Vision-Based Subsea Gas Leaks Monitoring: A Comparison of Two Approaches," *Sensors,* vol. 23, no. 5, p. 2566, 2023, https://doi.org/10.3390/s23052566.
[24] F. Ma'arif, Z. Gao, F. Li and H. U. Ghifarsyam, "The New Analysis of Discrete Element Method Using ARM Processor," *in IOP Conference Series: Earth and Environmental Science,* vol. 832, no. 1, p. 012016, 2020, https://doi.org/10.1088/1755-1315/832/1/012016.
[25] W. Fang, L. Wang and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," *IEEE Access*, vol. 8, pp. 1935 - 1944, 2019, https://doi.org/10.1109/ACCESS.2019.2961959.
[26] J. Ruan and Z. Wang, "An Improved Algorithm for Dense Object Detection Based on YOLO," in *The 8th International Conference on Computer Engineering and Networks (CENet)*, pp. 592-599, 2020, https://doi.org/10.1007/978-3-030-14680-1_65.
[27] X. Zhao, Y. Ni and H. Jia, "Modified Object Detection Method Based on YOLO," in *CCF Chinese Conference on Computer Vision*, pp. 233-244, 2017, https://doi.org/10.1007/978-981-10-7305-2_21.
[28] S. Zhao, J. Zheng and S. S. a. L. Zhang, "An Improved YOLO Algorithm for Fast and Accurate Underwater Object Detection," *Symmetry,* vol. 14, no. 8, p. 1669, 2022, https://doi.org/10.3390/sym14081669.
[29] J. Du, "Understanding of object detection based on CNN family and YOLO," *in Journal of Physics: Conference Series, 2nd International Conference on Machine Vision and Information Technology (CMVIT)*, vol. 104, p. 012029, 2018, https://doi.org/10.1088/1742-6596/1004/1/012029.
[30] P. Bharati and A. Pramanik, "Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey," *in Computational Intelligence in Pattern Recognition: Proceedings of CIPR*, pp. 657-668, 2020, https://doi.org/10.1007/978-981-13-9042-5_56.
[31] H. Li, C. Li, G. Li and L. Chen, "A real-time table grape detection method based on improved YOLOv4-tiny network in complex background," *Biosystems Engineering*, vol. 212, pp. 347-359, 2021, https://doi.org/10.1016/j.biosystemseng.2021.11.011.
[32] Y. Tang, H. Zhou, H. Wang and Y. Zhang, "Fruit detection and positioning technology for a Camellia oleifera C. Abel orchard based on improved YOLOv4-tiny model and binocular stereo vision," *Expert Systems with Applications,* vol. 211, p. 118573, 2023, https://doi.org/10.1016/j.eswa.2022.118573.

[33] M.-L. Huang and Y.-S. Wu, "GCS-YOLOV4-Tiny: A lightweight group convolution network for multi-stage fruit detection," *Mathematical Biosciences and Engineering,* vol. 20, no. 1, pp. 241–268, 2022, https://doi.org/10.3934/mbe.2023011.
[34] J. Li, X. Liang, S. Shen, T. Xu, J. Feng and S. Yan, "Scale-Aware Fast R-CNN for Pedestrian Detection," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 985-996, 2018, https://doi.org/10.1109/TMM.2017.2759508.
[35] M. Khan, S. Chakraborty, R. Astya and S. Khepra, "Face Detection and Recognition Using OpenCV," in 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2019, https://doi.org/10.1109/ICCCIS48478.2019.8974493.
[36] S. Gollapudi, "OpenCV with Python," Learn Computer Vision Using OpenCV, p. 31–50, 2019, https://doi.org/10.1007/978-1-4842-4261-2_2.
[37] G. Bradski. *The OpenCV Library. Dr. Dobb's Journal of Software Tools*. 2000, https://www.scirp.org/reference/ReferencesPapers?ReferenceID=1692176.
[38] H. Zhang, C. Li, L. Li, S. Cheng, P. Wang, L. Sun, J. Huang and X. Zhang, "Uncovering the optimal structural characteristics of flocs for microalgae flotation using Python-OpenCV," *Journal of Cleaner Production,* vol. 385, p. 135748, 2023, https://doi.org/10.1016/j.jclepro.2022.135748.
[39] D. D. K. A. A. Levin, A. A. Nechunaev, L. S. Prokhorenko, D. S. Mishchenkov, A. G. Nosova, D. A. Astakhov, Y. V. Poduraev and D. N. Panchenkov, "Assessment of experimental OpenCV tracking algorithms for ultrasound videos," *Scientific Reports volume,* vol. 13, no. 1, p. 6765 , 2023, https://doi.org/10.1038/s41598-023-30930-3.
[40] J. C. Carver and T. Epperly, "Software Engineering for Computational Science and Engineering [Guest editors' introduction]," *in Computing in Science & Engineering*, vol. 16, no. 3, pp. 6-9, 2014, https://doi.org/10.1109/MCSE.2014.72.
[41] K. Pulli, A. Baksheev, K. Kornyakov and V. Eruhimov, "Real-time computer vision with OpenCV," *Communications of the ACM*, vol. 55, no. 6, p. 61–69, 2012, https://doi.org/10.1145/2184319.2184337.
[42] K. Yu, T. Tao, H. Xie, Z. Lin, T. Liang, B. Wang, P. Chen, D. Hao, Y. Wang and X. Liang, "Benchmarking the Robustness of LiDAR-Camera Fusion for 3D Object Detection," *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3187-3197, 2023, https://doi.org/10.1109/CVPRW59228.2023.00321.

## BIOGRAPHY OF AUTHORS

**Rachmat Muwardi,** is currently a Lecturer in the Department of Electrical Engineering, Universitas Mercu Buana, Jakarta, Indonesia. He graduated from the Beijing Institute of Technology in 2020 with a Master's in Electronic Science and Technology. Currently, He declared as a recipient of a China Scholarship Council (CSC) to continue his doctoral program at the Beijing Institute of Technology in September 2022, majoring in Optical Engineering. During his undergraduate, he received a double degree scholarship from Universitas Mercu Buana and Beijing Institute of Technology in Electrical Engineering and Computer Science. His research interest is Object Detection, Target Detection, and Embedded System. He can be contacted at email: rachmat.muwardi@mercubuana.ac.id.

**Ahmad Faizin,** is currently an engineer in a medical equipment company. He graduated from Polytechnic Health Ministry in 2014 with Diploma in Electro Medical Engineering, after a few years he continued his studies in Mercu Buana University and graduated with Bachelor's in Electrical Engineering in 2022. He is currently studying for a Master's degree at Mercu Buana University, majoring in Electrical Engineering. He can be contacted at email: Faizin657@gmail.com.

**Rizky Rahmatullah,** he is currently a master's degree at the School of Electronics and Information Engineering, Beijing Institute of Technology (BIT), Beijing, China and works as a researcher at the Indonesian National Research and Innovation Agency. He is interested in research on the Internet of Things, Wireless Communication and Antennas. He can be contacted at email: rizk032@brin.go.id / 3820231111@bit.edu.cn.

**Yanxi Wang,** is currently a Master's student at the School of Electronics and Information Engineering, Beijing Institute of Technology (BIT), Beijing, China. He graduated from Hainan University in 2022 and received a Bachelor's degree in Communication Engineering. He can be contacted at email: shaluoyan523@163.com.

**Mirna Yunita,** received a Master's in Computer Science and Technology from Beijing Institute of Technology, Beijing, China. Currently, as a Ph.D. student at the School of Computer Science and Technology, Beijing Institute of Technology, China. She is interested in related topics in Machine Learning, Web Development, Data Mining, and Bioinformatics. She was a Frontend and Mobile Application Developer in a Logistics & Supply Chain company in Jakarta, Indonesia. She can be contacted at email: mirnayunita@bit.edu.cn.