Real-time Object Detection Performance Analysis Using YOLOv7 on Edge Devices

Ricardo C. Câmara de M. Santos, Mateus Coelho Silva, and Ricardo A. R. Oliveira

***Abstract*—Real-time object detection in images is one of the most important areas in computer vision and finds applications in several fields, such as security systems, protection, independent vehicles, and robotics. Many of these applications need to use edge hardware platforms, and it is vital to know the performance of the object detector on these hardware platforms before developing the system. Therefore, in this work, we executed performance benchmark tests of the YOLOv7-tiny model for real-time object detection using a camera and three embedded hardware platforms: Raspberry Pi 4B, Jetson Nano, and Jetson Xavier NX. We tested and analyzed the NVIDIA platforms and their different power modes. The Raspberry Pi 4B achieved an average of 0.9 FPS. The Jetson Xavier NX achieved 30 FPS, the maximum possible FPS rate, in three power modes. In the tests, it was possible to notice that the maximum CPU clock of the Jetson Xavier NX impacts the FPS rate more than the GPU clock itself. The Jetson Nano achieved 7.4 and 5.2 FPS in its two power consumption modes.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9019

***Index Terms*—Object detection, YOLOv7, Embedded devices.**

I. Introduction

Object detection is one of the essential areas in computer vision, playing a fundamental role in various practical scenarios. Currently, various applications such as robotics [1], autonomous vehicles [2], security systems [3], and industrial processes [4] have adopted object detection algorithms in images as a crucial part. These systems use these algorithms to interpret their environment by detecting the presence or absence of specific objects by processing images of the environment. When these objects are present, in addition to object classification, i.e., providing information about the type of objects present, the position and area occupied by them in the image are also provided.

Fig. 1 shows two robots with which we apply the concept of object detection in images executed on edge devices. The robot on the left uses a Jetson Xavier NX. It was used in the proposal for a navigation method based on image detection [5] where it performed object detection to execute the perception task of the robot. The robot on the right uses a Jetson Nano. We utilized this robot in the ground truth collection methodology for odometry based on computer vision proposed in [6], and object detection was performed to execute the location task.

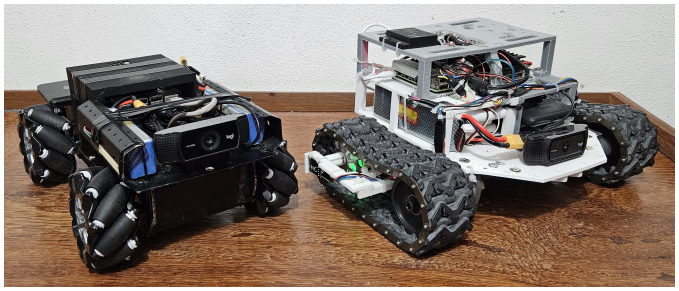


Fig. 1. Robots using NVIDIA embedded devices. The robot on the left, shown in [5], uses a Jetson Xavier NX, while the robot on the right, in [6], uses a Jetson Nano.

Object detection in real-time images is a challenging task and, consequently, is not inexpensive in terms of computation cost. Therefore, the scientific community has made great efforts to reduce the computational cost while maintaining a high accuracy rate of these algorithms. Making these algorithms computationally cheaper is even more relevant for embedded systems such as robots and autonomous vehicles, where detection must occur on edge devices with limited computational capabilities rather than cloud servers or desktop computers.

Today, object detection in images is typically performed in real-time by a deep neural network which, in most cases, depending on its architecture, cannot be executed in real-time on a Central Processing Unit (CPU), especially when it comes to single-core processors or embedded CPUs. Thus, specific hardware is required for its use. These hardware components can include, for instance, Graphics Processing Unit (GPU) [7], Neural Processing Unit (NPU) [8], and Tensor Processing Unit (TPU) [9]. Before developing real-world applications utilizing object detection algorithms, it is imperative to understand the algorithm's performance in advance when executing the system's final hardware. Therefore, conducting benchmark tests of these algorithms on different hardware configurations is essential. These tests enable measurement of the hardware's performance in executing the algorithms and evaluating whether it meets the system's requirements.

In this article, we present an analysis of the performance of the You Only Look Once (YOLO) model for real-time object detection when executed on three popular embedded

The associate editor coordinating the review of this manuscript and approving it for publication was Javier Moreno-Valenzuela (Corresponding author: Ricardo C. C. de M. Santos).

This work is supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

R. C. C. de M. Santos, M. C. Silva, and R. A. R. Oliveira are with laboratório iMobilis, Universidade Federal de Ouro Preto, Brazil (e-mails: ricardocamara03@gmail.com, mateuscoelhoccom@gmail.com, and rrabelo@gmail.com).

TABLE I
HARDWARE SPECIFICATIONS OF THE THREE EMBEDDED PLATFORMS USED IN THIS STUDY

	NVIDIA Jetson Nano	**NVIDIA Jetson Xavier NX**	**Raspberry Pi 4B**
Edge Accelerator	128-core NVIDIA Maxwell GPU	384-core NVIDIA Volta GPU (48 Tensor Cores)	-
AI Performance	0.5 TFLOPs	1.3 TFLOPs	-
CPU	Quad-core ARM Cortex-A57 MPCore processor	6-core NVIDIA Carmel ARM v8.2 64-bit CPU 6 MB L2 + 4 MB L3	Quad-core ARM Cortex-A72
Memory	4 GB 64-bit LPDDR4 25.6 GB/s	8 GB 128-bit LPDDR4x 51.2 GB/s	4 GB LPDDR4
Dimensions	69.6 mm × 45 mm	69.6 mm × 45 mm	85 mm × 56 mm
Nominal Power Envelope	5W–10W	10W–20W	3W–6.25W

development platforms. YOLO is a one-stage object detection in images algorithm created by Redmon et al. [10]. The platforms used in this work are the Raspberry Pi 4B, Jetson Nano, and Jetson Xavier NX. We carried out this analysis by comparing the processing speed of the neural network, using the Frames per Second (FPS) rate on each of these platforms and the use of hardware resources on both NVIDIA platforms. The FPS rate is the crucial index for object detection models and one of the main factors for deploying AI applications on intelligent edges. The hardware resources used in the comparison are the usage rate of Random Access Memory (RAM), CPU, and GPU. The tests presented here consider the different power consumption modes of the two NVIDIA platforms.

II. RELATED WORKS

A. Applications of Object Detection in Edge Devices

Edge devices are end devices closest to the user, such as mobile phones, cyber-physical systems (CPSs), wearables, the Internet of Things (IoT), embedded and autonomous systems, and intelligent sensors [11]. These devices perform data processing near the origin or location where the data is generated instead of sending it to servers for processing. These devices are a fundamental part of edge computing.

Object detection in edge device images proves beneficial in a wide range of applications. In [12], the authors compare YOLOv3 and Faster R-CNN for citrus fruit detection in embedded applications. Meanwhile, in [13], the authors proposed a fire detection system using YOLOv7 on edge devices. The automotive industry extensively utilizes object detection in edge devices ranging from fatigue detection applications [14] to more advanced ADAS systems [15]. Another area frequently using object detection is robotics [16]–[18]. In [5], we demonstrate the usage of real-time object detection for sensing and local navigation in autonomous mobile robots.

B. Performance Analysis of Object Detection in Edge Devices

In [19], performance tests are presented by executing YOLOv3 and PPYOLO [20] models on the Jetson Nano and Jetson Xavier NX. PPYOLO is a modified version of YOLO. They used the tiny model for the tests on Jetson Nano. They tested the models using two input image sizes, 320x320 and 608x608. The PPYOLO model proved faster than YOLOv3 on both hardware. They showed that the input image size significantly affects the inference speed, with a smaller size resulting in faster inference speed. However, using a smaller input image size compromises the model's accuracy. In addition to inference time tests, the study analyzes the usage of CPU, GPU, and RAM when using the models on different hardware. For the tests, they used 500 images from the COCO dataset [21] rather than real-time collected camera images.

In [22], the authors conducted tests with four models of object detectors in images: YOLOv3, YOLOv3-tiny, YOLOv4, and YOLOv4-tiny on NVIDIA Jetson Nano, NVIDIA Jetson Xavier NX, and Raspberry Pi 4B (RPi) with Intel Neural Compute Stick2 (NCS2). Again, they made a comparison between the tiny models and the standard models of each YOLO version. The FPS rate of the RPi + NCS2 set running YOLOv3-tiny was 7.6 times higher than that of YOLOv3. Jetson Nano achieved a rate of 15 FPS when running YOLOv4-tiny. Jetson Xavier NX reached a rate of 41 FPS when running YOLOv3-tiny. They tested the models with four input image sizes: 320, 416, and 608. The authors executed the tests without using a camera through the processing of previously recorded videos.

Thus far, we have not encountered any literature that analyzes the performance of YOLOv7-tiny on Jetson Nano and Jetson Xavier NX devices, nor do we find studies that examine the FPS rate and hardware resource consumption across their various power consumption modes. This rationale underpins the experiments carried out in this study.

TABLE II
JETSON XAVIER NX POWER MODES

Mode ID	Power Mode (watts)	Online CPU	CPU Max. Freq. (MHz)	GPU Max. Freq. (MHz)
0	15	2	1907	1109
1	15	4	1420	1109
2	15	6	1420	1109
3	10	2	1497	803
4	10	4	1190	803
5	10	4	1907	510
6	20	2	1907	1109
7	20	4	1420	1109
8	20	6	1420	1109

TABLE III
JETSON NANO POWER MODES

Mode ID	Power Mode (watts)	Online CPU	CPU Max. Freq. (MHz)	GPU Max. Freq. (MHz)
0	10	4	1479	921.6
1	5	2	918	640

III. BACKGROUND AND METHODOLOGY

A. Object Detection and YOLOv7

Object detection involves classifying objects in images or videos while simultaneously returning their location. After processing the image, the detector returns a pair of coordinates X and Y, height and width values, and the object type for each detected object. The coordinates and the height and width values are typically given as bounding boxes.

Object detection has started using deep learning models since the early 2000s [23]. These models are generally classified as one-stage or two-stage detectors. Before understanding these two approaches, it is crucial to know the classification and region proposal concepts. Classification involves categorizing objects into specific classes, while region proposal is an algorithm that identifies areas where objects may be located. A one-stage detection model performs both classification and region proposal in a unified manner, aiming to increase speed.

YOLO is a single-stage object detection algorithm [10]. It was named this way because it is a one-stage detector. Its popularity is due to the speed of processing and accuracy it offers. Since its publication, YOLO has undergone various improvements over the years, reaching its seventh version recently [24].

YOLOv7 [25] brings several improvements compared to its predecessors. One of these improvements is anchor boxes, a set of pre-defined boxes with different aspect ratios that detect objects of different shapes. The network uses nine anchor boxes, facilitating the detection of a broader range of shapes and sizes of objects compared to previous versions. This also contributes to reducing false positives. Another improvement is the use of a new loss function called focal loss. Previous versions used the cross-entropy loss function, which is less effective in detecting small objects than focal loss. Focal loss addresses this issue by reducing the weight of the loss for well-classified examples and focusing on the most difficult objects to detect.

B. Hardware Platforms

In this work, we used two devices from NVIDIA Corporation's Jetson series and a Raspberry Pi 4B. The Jetson series is characterized by small size and powerful performance. Jetson Nano is the entry-level platform in the series, with the lowest price and computational power. Jetson Xavier NX has higher performance while maintaining a compact size and can also be used as edge devices and, depending on the scenario, even as edge servers. These two devices have heterogeneous CPU-GPU architecture, which is highly compatible with various edge AI applications. In addition to the two NVIDIA devices, we also conducted performance tests with the Raspberry Pi 4B. The main difference between the Raspberry Pi and the Jetsons is that the Raspberry Pi 4 does not have an AI accelerator. Table I shows the specifications of the three devices used in the benchmark of this work.

The NVIDIA hardware platforms offer a variety of power modes, each with its unique characteristics. These power modes have unique identifiers and configurable power consumption, CPU usage, and maximum CPU/GPU frequency, providing flexibility and control to the developers. Table II details the power modes of the Jetson Xavier NX, while Table III outlines the power modes of the Jetson Nano. The Jetson Nano, for instance, offers two power modes, one at 10 watts and the other at 5 watts. On the other hand, the Jetson Xavier NX offers nine standard power modes, ranging between 10, 15, and 20 watts. This detailed information allows users to optimize the performance of these devices for their specific needs.

C. Methodology

We used the YOLOv7-Tiny model trained to detect traffic cones to carry out the tests performed in this work. YOLOv7-tiny is a small model of YOLOv7 that is better suited for edge hardware platforms. To train this model, we used 584 images collected from the internet, each containing one or more cones. During training, we used 537 images for training and 47 for validation. This model was the same one used in [5]. The model input size is 640x640. YOLOv7 uses the PyTorch framework, which offers support for NVIDIA GPUs. We used Python3 and the OpenCV library to develop the application used in our experiments. The code can be found here: https://github.com/ricardocamara03/fps_yolov7

We perform FPS and hardware utilization measurements when running YOLOv7-tiny. In order to carry out tests similar to real applications, pre-recorded videos were not used, but rather a Logitech C920 camera with a resolution of 640x480. We executed the tests of FPS and hardware utilization measurements while running the model in each power mode for 120 seconds. To measure hardware utilization, we used the NVIDIA tegrastats tool, configured with a delay between measurements of 20 ms. Hardware utilization at idle for each power mode was also measured. Thus enabling a comparison of hardware utilization at idle and when running the model. Before carrying out the tests, we certified that using the camera without running the model achieves an average rate of 30 FPS, so the maximum FPS rate possible in our case is 30 FPS. We

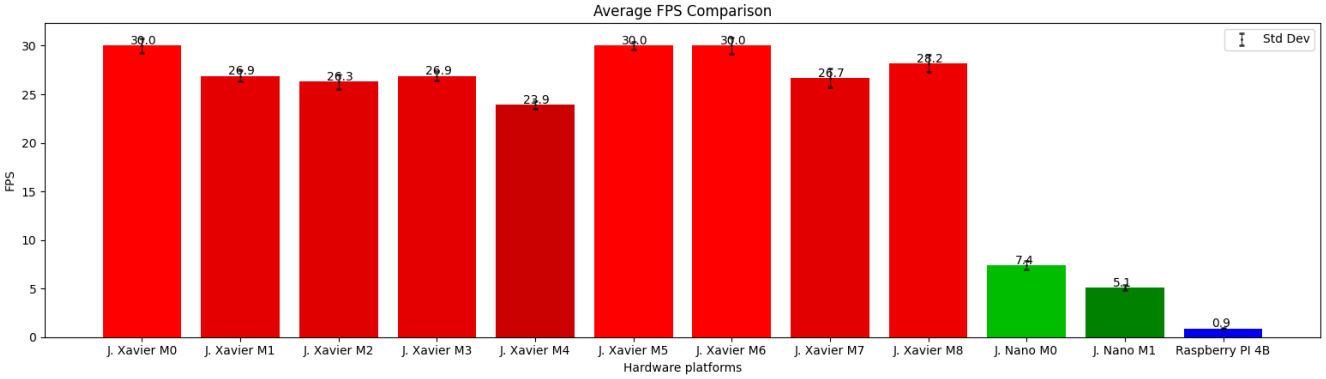


Fig. 2. Average FPS comparison.

used a 30 FPS camera because it is the most common FPS rate in the market. Therefore, our results can benefit more people.

IV. Results

Fig. 2 shows the results of measuring the average FPS rate for each device in each power mode. The Jetson Xavier NX reaches a rate of 30 FPS for power modes 0, 5, and 6. Therefore, we can see that on the Jetson Xavier NX, the CPU clock influenced the FPS rate more than the GPU clock itself since these three power modes are the only ones that utilized the highest maximum CPU clocks. Our hypothesis is that the application's bottleneck lies in collecting the images until they are submitted to the neural network, which demands only the CPU rather than the object detection itself. Regarding the GPU, mode 5 reached 30 FPS despite having the lowest maximum clock value among the modes. There was no evidence that the power options and number of CPUs online considerably influenced the FPS rate since modes 0 and 6 only use two CPUs, and mode 5 uses 10 watts of power and still reaches the maximum FPS rate. Another fact that indicates that the maximum CPU clock is the factor that most influences the FPS rate is that mode 4, with the lowest maximum CPU clock, resulted in the lowest FPS rate among all Jetson Xavier NX modes, 23.9 FPS. The Jetson Nano achieved a rate of 7.4 FPS for mode 0 and 5.1 FPS for mode 1.

The Raspberry PI 4B achieved the lowest FPS rate among all hardware, with 0.9 FPS. This is because it is the only platform among all three hardware that does not have an AI accelerator to execute the model. Therefore, the hardware utilization measurements below were only carried out on the two NVIDIA platforms.

Fig. 3 shows the hardware utilization in percentage for CPU, GPU, and RAM related to all energy consumption modes of the Jetson Xavier NX at idle without running the object detection software. This work's CPU usage percentage calculation does not consider offline CPU. The calculation was done by using the average consumption of online CPUs. In idle mode, RAM usage remains fixed at 26%, and GPU consumption is around 1%. CPU consumption reaches a maximum consumption of 13% for modes 0, 3, and 6 as they are the only modes with only two online CPUs. CPU consumption is inversely proportional to the number of CPUs online when the system is idle, as the computational load remains the same and the number of available CPUs varies.

Fig. 4 shows the Jetson Xavier NX's hardware utilization for all power consumption modes when running the YOLOv7-tiny model. RAM consumption remains between 52% and 56% considering all power modes. CPU consumption maintains the same behavior as when the system was idle, with a decreasing consumption for 2, 4, and 6 CPUs online in that order. For modes with only two CPUs online, consumption is between 55% and 59%; for four CPUs online, 30% and 26%; and for those with six CPUs online, consumption is between 15% and 20%. So, it shows that Jetson Xavier NX higher FPS rates are more related to the maximum CPU clock rate than the number of available CPUs. GPU consumption varies between 29% and 61%, reaching the highest FPS rate in power mode 5, which has the lowest maximum GPU frequency. We notice that GPU consumption has the most significant standard deviation among all hardware utilization measurements.

Fig. 5 shows the hardware utilization of CPU, GPU, and RAM for the Jetson Nano at idle. It presents 31% for RAM in both power modes. In contrast to the CPU, it presents 4% for mode 0 and 9% for mode 1. About the GPU, the image presents practically zero consumption for both modes, being 0.02 for mode 0 and 0.01% for mode 1.

Fig. 6 shows the hardware utilization for Jetson Nano when running the YOLOv7-tiny model. Average RAM consumption is between 71% and 78%. The average CPU consumption was 27% for mode 0 and 63% for mode 1. The average GPU consumption was 61% for mode 0 and 49% for mode 1.

V. Conclusions

Real-time object detection in images is one of the most important areas of computer vision. It plays a fundamental role in several practical applications, such as security systems, industrial applications, autonomous cars, and robotics. Many of these applications have an embedded computing platform as their leading hardware. Therefore, before developing a system of this nature, it is crucial to know the performance of the object detector on the target hardware.

This article presented an analysis of the performance of the YOLOv7-tiny model for real-time object detection. We conducted this analysis by running the model while processing

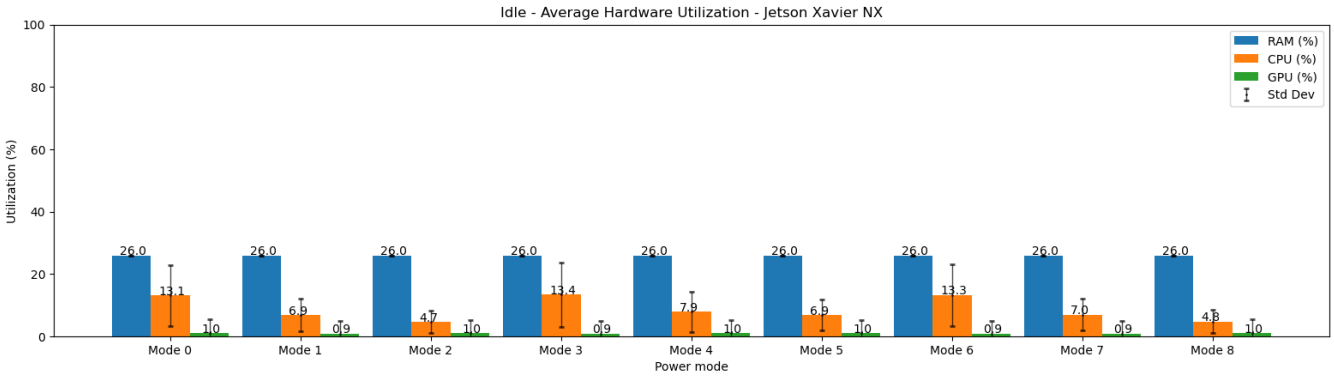


Fig. 3. Average hardware utilization for Jetson Xavier NX in idle.

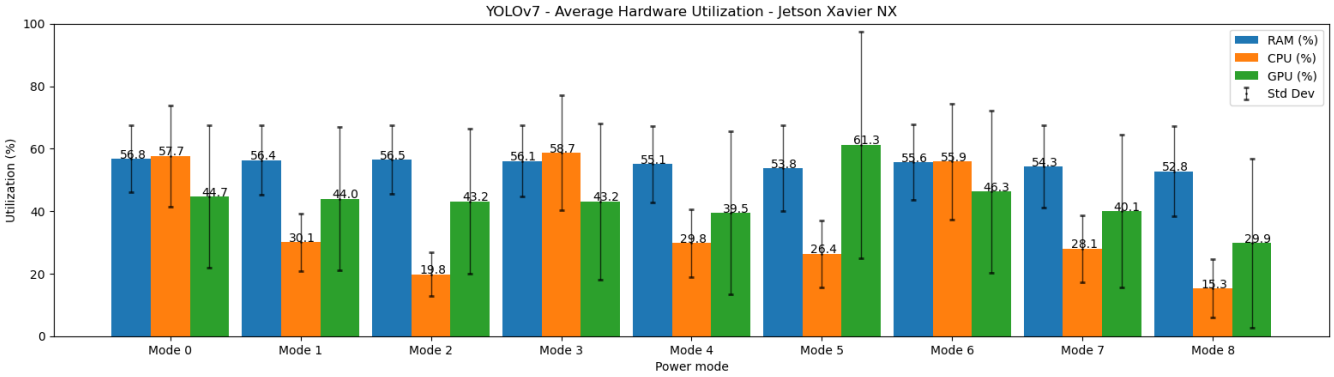


Fig. 4. Average hardware utilization for Jetson Xavier NX executing YOLOv7.

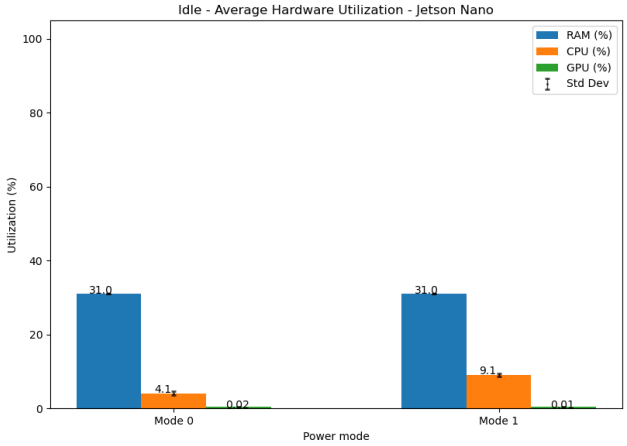


Fig. 5. Average hardware utilization for Jetson Nano in idle.

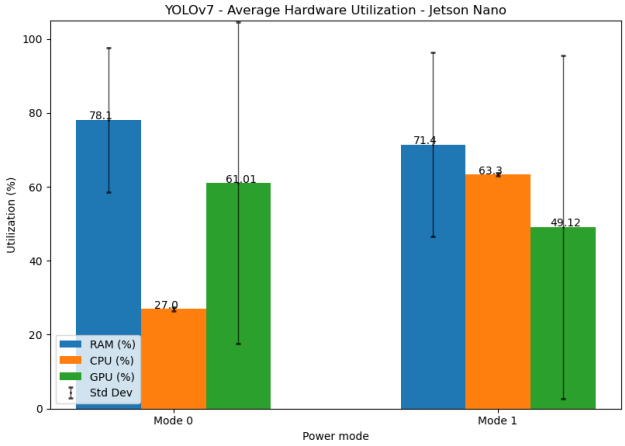


Fig. 6. Average hardware utilization for Jetson Nano executing YOLOv7.

images captured from a 30 FPS camera on three embedded hardware platforms: Raspberry Pi 4B, Jetson Nano, and Jetson Xavier NX. Regarding the FPS rate, the Raspberry Pi 4B proved unfeasible for executing the model in real-time, reaching an average FPS rate of 0.9. We covered nine different power modes in the Jetson Xavier NX experiments. These modes have different settings, including the number of CPUs online, maximum CPU/GPU frequency, and power. The Jetson Xavier NX reached maximum FPS in three different power modes and showed a significant relationship between FPS rate and maximum CPU clock. The maximum CPU clock has been shown to have a more significant impact on the FPS rate than the GPU clock and the power of the modes. The Jetson Nano achieved an FPS rate of 7.4 and 5.2 in its two power

consumption modes.

In future work, we intend to compare the FPS rate and hardware utilization when processing images from previously recorded videos and images captured from cameras in real-time, as shown here. This allows an analysis of the impact of camera use on hardware utilization. As the Jetson Xavier NX proves to be the best platform, we intend to keep it as the main hardware platform to execute real-time object detection for our mobile robots' ongoing research.

ACKNOWLEDGEMENTS

The authors would like to thank MICHELIN Connected Fleet, NVIDIA, UFOP, CAPES and CNPq for supporting this work. This work was partially financed by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001, and by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - Finance code 308219/2020-1.

REFERENCES

[1] E. Maiettini, G. Pasquale, L. Rosasco, and L. Natale, "On-line object detection: a robotics challenge," *Autonomous Robots*, vol. 44, no. 5, pp. 739–757, 2020. doi: https://doi.org/10.3390/pharmaceutics13081318. [Online]. Available: https://www.mdpi.com/1999-4923/13/8/1318

[2] D. Feng, A. Harakeh, S. L. Waslander, and K. Dietmayer, "A review and comparative study on probabilistic object detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 9961–9980, 2021. doi: https://doi.org/10.1109/TITS.2021.3096854. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9525313

[3] J. Wu, X. Xu, and J. Yang, "Object detection and x-ray security imaging: A survey," *IEEE Access*, 2023. doi: https://doi.org/10.1109/ACCESS.2023.3273736. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10120944

[4] X. Zhang, C. Wang, Y. Tang, Z. Zhou, and X. Lu, "A survey of few-shot learning and its application in industrial object detection tasks," in *International Workshop of Advanced Manufacturing and Automation*. Springer, 2021. doi: https://doi.org/10.1007/978-981-19-0572-8-81 pp. 637–647. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-19-0572-8_81

[5] R. C. C. d. M. Santos, M. C. Silva, R. L. Santos, E. Klippel, and R. A. Oliveira, "Towards autonomous mobile inspection robots using edge ai." in *ICEIS (1)*, 2023. doi: https://www.scitepress.org/Link.aspx?doi=10.5220/0011972200003467 pp. 555–562. [Online]. Available: https://sol.sbc.org.br/index.php/semish/article/view/25073/24894

[6] R. C. C. d. M. Santos, M. C. Silva, and R. A. Oliveira, "A computer vision-based method for collecting ground truth for mobile robot odometry," *Proceedings of the 26th International Conference on Enterprise Information Systems - (Volume 1)*, pp. 116–127, 2024. doi: http://dx.doi.org/10.5220/0012622900003690. [Online]. Available: https://www.scitepress.org/publishedPapers/2024/126229/pdf/index.html

[7] T. Fukagai, K. Maeda, S. Tanabe, K. Shirahata, Y. Tomita, A. Ike, and A. Nakagawa, "Speed-up of object detection neural network with gpu," in *2018 25th IEEE International conference on image processing (ICIP)*. IEEE, 2018. doi: https://doi.org/10.1109/ICIP.2018.8451814 pp. 301–305. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8451814

[8] L. Chen, J. Hu, X. Li, F. Quan, and H. Chen, "Onboard real-time object detection for uav with embedded npu," in *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2021. doi: https://doi.org/10.1109/CYBER53097.2021.9588193 pp. 192–197. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9588193

[9] B. Kovács, A. D. Henriksen, J. D. Stets, and L. Nalpantidis, "Object detection on tpu accelerated embedded devices," in *Computer Vision Systems: 13th International Conference, ICVS 2021, Virtual Event, September 22-24, 2021, Proceedings 13*. Springer, 2021. doi: https://doi.org/10.1007/978-3-030-87156-7-7 pp. 82–92. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-87156-7_7

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. doi: https://doi.org/10.1109/CVPR.2016.91 pp. 779–788. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780460

[11] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, 2023. doi: 10.1109/JPROC.2022.3226481

[12] J. C. da Silva, M. C. Silva, E. J. Luz, S. Delabrida, and R. A. Oliveira, "Using mobile edge ai to detect and map diseases in citrus orchards," *Sensors*, vol. 23, no. 4, p. 2165, 2023. doi: https://doi.org/10.3390/s23042165. [Online]. Available: https://www.mdpi.com/1424-8220/23/4/2165

[13] L. Wenzheng and W. Jie, "A yolov7 forest fire detection system with edge computing," in *2023 IEEE 13th International Conference on Electronics Information and Emergency Communication (ICEIEC)*. IEEE, 2023. doi: https://doi.org/10.1109/ICEIEC58029.2023.10200044 pp. 223–227. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10200044

[14] R. C. C. d. M. Santos, M. C. Silva, and R. A. R. Oliveira, "Evaluating the effect of audio feedback on the behavior of automotive fatigue and distraction detection system users," in *2019 IX Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, 2019. doi: https://doi.org/10.1109/SBESC49506.2019.9046047 pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9046047

[15] F. L. M. de Sousa, M. J. da Silva, R. C. C. de Meira Santos, M. C. Silva, and R. A. R. Oliveira, "Deep-learning-based embedded adas system," in *2021 XI Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, 2021. doi: https://doi.org/10.1109/SBESC53686.2021.9628316 pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9628316

[16] G. Liu, Y. Hu, Z. Chen, J. Guo, and P. Ni, "Lightweight object detection algorithm for robots with improved yolov5," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106217, 2023. doi: https://doi.org/10.1016/j.engappai.2023.106217. [Online]. Available: https://dl.acm.org/doi/abs/10.1016/j.engappai.2023.106217

[17] Z. Li, B. Xu, D. Wu, K. Zhao, S. Chen, M. Lu, and J. Cong, "A yolo-ggcnn based grasping framework for mobile robots in unknown environments," *Expert Systems with Applications*, vol. 225, p. 119993, 2023. doi: https://doi.org/10.1016/j.eswa.2023.119993. [Online]. Available: https://dl.acm.org/doi/abs/10.1016/j.eswa.2023.119993

[18] G. Xu, A. S. Khan, A. J. Moshayedi, X. Zhang, and Y. Shuxin, "The object detection, perspective and obstacles in robotic: a review," *EAI Endorsed Transactions on AI and Robotics*, vol. 1, no. 1, 2022. doi: http://dx.doi.org/10.4108/airo.v1i1.2709. [Online]. Available: https://eudl.eu/doi/10.4108/airo.v1i1.2709

[19] J. Zhu, H. Feng, S. Zhong, and T. Yuan, "Performance analysis of real-time object detection on jetson device," in *2022 IEEE/ACIS 22nd International Conference on Computer and Information Science (ICIS)*, 2022. doi: 10.1109/ICIS54925.2022.9882480 pp. 156–161. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9882480

[20] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding *et al.*, "Pp-yolo: An effective and efficient implementation of object detector," *arXiv preprint arXiv:2007.12099*, 2020. doi: https://ui.adsabs.harvard.edu/link-gateway/2020arXiv200712099L/doi:10.48550/arXiv.2007.12099. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2020arXiv200712099L/abstract

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014. doi: https://doi.org/10.1007/978-3-319-10602-1-48 pp. 740–755. [Online]. Available: https://home.ttic.edu/~mmaire/papers/pdf/coco_eccv2014.pdf

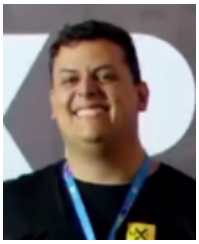
[22] H. Feng, G. Mu, S. Zhong, P. Zhang, and T. Yuan, "Benchmark analysis of yolo performance on edge intelligence devices," *Cryptography*, vol. 6, no. 2, p. 16, 2022. doi: https://doi.org/10.3390/cryptography6020016. [Online]. Available: https://www.mdpi.com/2410-387X/6/2/16

[23] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023. doi: https://doi.org/10.1109/JPROC.2023.3238524. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10028728

[24] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, 2023. doi: https://doi.org/10.3390/make5040083. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83

[25] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023. doi: https://doi.org/10.1109/CVPR52729.2023.00721 pp. 7464–7475. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10204762

Ricardo C. Câmara de M. Santos is MSc in Computer Science from UFOP in 2020. Bachelor of Computer Science from UFOP in 2014. He has experience in the field of Computer Science, with emphasis on embedded computing, embedded systems, and computer vision. He has research experience, conducting scientific initiation from 2012 to 2014 at the Imobilis laboratory at UFOP, in addition to the master's degree completed in the same environment from 2017 to 2020. As an IT professional, he has experience in developing vehicular systems based on computer vision, working in this area from 2015 to the present at Michelin Connected Fleet. Today he's a PhD candidate at Universidade Federal de Ouro Preto. Contact him at ricardocamara03@gmail.com.



Mateus Coelho Silva is currently a Postdoctoral Researcher in Robotics at the Vale Technological Institute - Federal University of Ouro Preto. He obtained his M.Sc. and Ph.D. in Computer Sciences at the Federal University of Ouro Preto. His current research interests include Machine and Deep Learning, Cyber-Physical Systems, IoT, Wearable Devices, and Robotics. Contact him at mateuscoelho.ccom@gmail.com.

Ricardo A. R. Oliveira received his Ph.D. degree in Computer Science from the Federal University of Minas Gerais in 2008. Nowadays he is an Associate Professor in the Computing Department at the Federal University of Ouro Preto. Has experience in Computer Science, acting on the following subjects: Wavelets, Neural Networks, 5G, VANT, and Wearables. Contact him at rrabelo@gmail.com.