Comparison of YOLO Power Usage on Edge-AI Device with Smartmeter

Chandra Wijaya[1,2], Anggi Andriyadi[1,3], Shih-Yen Chen[4], I-Jan Wang[1], and Chao-Tung Yang[4,5(✉)]

[1] Department of Industrial Engineering and Enterprise Information, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung City 407224, Taiwan R.O.C.
chandraw@unpar.ac.id, ijwang@thu.edu.tw

[2] Center For Data Science and Artificial Intelligence System, Informatics Department, Parahyangan Catholic University, Bandung 40141, West Java, Indonesia

[3] Information System Department, Institut Informatika dan Bisnis Darmajaya, Bandar Lampung 35141, Lampung, Indonesia
anggi.andriyadi@darmajaya.ac.id

[4] Department of Computer Science, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung City 407224, Taiwan R.O.C.
{g12350014,ctyang}@thu.edu.tw

[5] Research Center for Smart Sustainable Circular Economy, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung City 407224, Taiwan R.O.C.

Abstract. This research presents a detailed comparison of power consumption when deploying the YOLOv8 (You Only Look Once) object detection model on the NVIDIA Jetson Orin Nano, an edge AI device. The study examines two execution environments: a Docker containerized setup and a non-Docker (bare-metal) environment. Key metrics evaluated include the total wattage consumed during model operation, pre- and post-processing time, and the inference duration. A smart meter is used to capture real-time power usage data for both environments, ensuring accurate measurements. By comparing these parameters, the research sheds light on the impact of Docker containerization on power efficiency and processing performance. While Docker is widely used for its convenience and portability, it may introduce additional overheads that affect power consumption and execution speed, which are critical factors for AI applications deployed on edge devices with limited resources. The study seeks to determine if the trade-offs introduced by containerization are significant when compared to running YOLOv8 in a native environment. The results aim to provide actionable insights for developers and engineers looking to optimize power consumption and performance for AI models on edge devices.

Keywords: Power Usage Consumption · Edge Device Power Measurement · Object Recognition · Smartmeter

D.-J. Deng and J.-C. Chen (Eds.): SGIoT 2024, LNICST 633, pp. 3–10, 2026.
https://doi.org/10.1007/978-3-031-93825-2_1

1 Introduction

The recent growth of the Internet of Things has led to widespread adoption of edge computing devices across different sectors for data collection and preprocessing. However, the diverse architectures of these heterogeneous edge devices present challenges in evaluating and monitoring their performance. In this study, we will conduct an experiment to assess the power consumption of the NVIDIA Jetson Orin Nano, a compact yet powerful edge AI device. The Jetson Orin Nano is designed to support AI applications in fields such as computer vision, smart surveillance, IoT edge analytics, and autonomous robots or drones. Its combination of a small size and robust computational capabilities makes it an ideal solution for developers looking to implement AI while minimizing power usage.

YOLO (You Only Look Once) is a real-time object detection algorithm known for its speed and efficiency. Initially introduced by Joseph Redmon in 2016, it has gone through several updates to enhance its accuracy, speed, and overall performance. YOLO is commonly applied in various fields, including autonomous driving, robotics, and surveillance systems [6,7].

Docker is a platform that allows developers to package, distribute, and run applications within isolated environments known as containers. These containers are lightweight, portable, and contain everything required for the application to function, including code, libraries, and dependencies. Docker ensures a consistent environment for applications, making sure they run the same way across different systems, regardless of the deployment location.

The data gathered from the Smart Meter ECO WPM-100 shown in Fig. 1, typically includes real-time information on power consumption, such as wattage, voltage, and current, along with energy usage trends over time. This device can provide detailed insights into the electrical power consumption of various appliances or systems, allowing for precise monitoring and management of energy use. The smart meter could help assess the power usage impact when running models like YOLO, especially when comparing Docker vs. non-Docker environments. This kind of detailed data is crucial for optimizing both energy efficiency and performance.

2 Related Works

Endah et al. [1,2] have implemented iSEC, an advanced system that integrates sensors with cloud and edge computing architectures. This paper showcases an optimized model for utilizing both cloud and edge environments in deep learning tasks, specifically for image classification. The study involved training deep learning models in the cloud and performing inference at the edge. Results were compared before and after fine-tuning to evaluate enhancements in both the training and inference phases.

2024-08-02 10:00:29.366	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248700.0	0.2	0.0	0.0	114.1	0.0	0.0	0.497	0.0	0.0	114.1	0.0	0.0	59.95
2024-08-02 10:01:20.558	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248700.0	0.2	0.0	0.0	114.0	0.0	0.0	0.497	0.0	0.0	114.0	0.0	0.0	59.91
2024-08-02 10:02:25.656	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248700.0	0.2	0.0	0.0	114.1	0.0	0.0	0.0	0.0	0.0	114.1	0.0	0.0	59.98
2024-08-02 10:03:36.862	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248700.0	0.2	0.0	0.0	113.9	0.0	0.0	0.0	0.0	0.0	113.9	0.0	0.0	59.93
2024-08-02 10:04:31.883	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248700.0	0.2	0.0	0.0	113.9	0.0	0.0	0.503	0.0	0.0	113.9	0.0	0.0	60.0
2024-08-02 10:05:05.878	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	113.8	0.0	0.0	0.512	0.0	0.0	113.8	0.0	0.0	60.04
2024-08-02 10:06:16.063	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	113.8	0.0	0.0	0.499	0.0	0.0	113.8	0.0	0.0	60.0
2024-08-02 10:07:40.300	564	141	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	113.7	0.0	0.0	0.503	0.0	0.0	113.7	0.0	0.0	59.93
2024-08-02 10:08:36.856	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.479	0.0	0.0	114.0	0.0	0.0	59.98
2024-08-02 10:09:07.298	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.0	0.0	0.0	114.0	0.0	0.0	59.95
2024-08-02 10:10:22.623	564	141	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	113.8	0.0	0.0	0.0	0.0	0.0	113.8	0.0	0.0	60.02
2024-08-02 10:11:02.415	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.503	0.0	0.0	114.0	0.0	0.0	60.03
2024-08-02 10:12:37.895	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.497	0.0	0.0	114.0	0.0	0.0	59.95
2024-08-02 10:13:07.878	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.1	0.0	0.0	0.499	0.0	0.0	114.1	0.0	0.0	60.01
2024-08-02 10:14:06.103	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.498	0.0	0.0	114.0	0.0	0.0	60.07
2024-08-02 10:15:13.683	564	141	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.2	0.0	0.0	0.495	0.0	0.0	114.2	0.0	0.0	60.07
2024-08-02 10:16:10.181	564	141	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.1	0.0	0.0	0.0	0.0	0.0	114.1	0.0	0.0	59.96
2024-08-02 10:16:10.371	564	141	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.1	0.0	0.0	0.0	0.0	0.0	114.1	0.0	0.0	59.96
2024-08-02 10:17:16.089	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.0	0.0	0.0	0.0	0.0	0.0	114.0	0.0	0.0	59.92
2024-08-02 10:18:04.876	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	113.8	0.0	0.0	0.501	0.0	0.0	113.8	0.0	0.0	59.88
2024-08-02 10:19:00.738	564	144	c18edd	NULL	NULL	0.0	0.0	0.0	5470700.0	3248800.0	0.2	0.0	0.0	114.2	0.0	0.0	0.488	0.0	0.0	114.2	0.0	0.0	60.02

Fig. 1. Sample data from Smart Meter

Lien et al. [3] successfully applied the YOLO series model for firework detection in their study. Despite spending additional time on encoding and facing issues with non-optimal encoders that led to some resource wastage during transmission, the model is effective in accurately identifying various types of fires, including both large flames and small fires, as well as detecting smoke.

Endah et al. [4] have integrated edge computing with deep learning to create an innovative application for detecting smoke and flames, addressing the need for drones to be equipped with infrared heat sensors. These sensors are used to identify the source of flames, but their effectiveness is constrained by the distance of infrared rays and the added weight they impose on the drone. The study successfully employed object detection and transfer learning techniques to detect smoke and flames, deploying these methods on edge devices such as Raspberry Pi and NVIDIA Jetson Xavier NX.

Halim et al. [5] implemented Kubernetes to create a robust service monitoring environment for heterogeneous edge equipment performance using Docker containers, which offer rapid deployment capabilities. The study focuses on deploying a high-quality service monitoring plan and visualizing the results with Grafana.

3 Metholodogy

Figure 4 illustrates the various stages of the research process. All experiments in this study were conducted using the NVIDIA Jetson Orin Nano with JetPack 5.1.3, running the Ubuntu 20.04.6 LTS operating system. The object recognition model employed is YOLO version 8 [8], executed in two different environments: an environment without Docker and an environment using Docker container. Once the environments were set up, the datasets used included two image-based datasets and two video-based datasets. The image datasets consist of VisDrone 2019 [9] and Pascal Visual Object Classes [10], while the video datasets feature

clips with durations of 5 min and 30 min. The visdrone dataset consists of 6741 images in size of 1.44 gigabytes containing cars, trucks, humans and other objects taken by drones. PASCAL Visual Object Classes (VOC2012) contains 4952 images of humans, animals, and another objects in size of 414 megabytes. The 5 min and 30 min video contains trucks and cars in the highway taken from youtube. The visible example from the datasets is shown below. Figure 2a and Fig. 2b show examples of images used in the experiments, while Figure 3 shows an example capture from the video dataset.

(a) VisDrone 2019 example

(b) Pascal VOC 2012 example

Fig. 2. The examples of image used in the experiments

Fig. 3. The example of video capture used in the experiment

After completing the experiments on all datasets, performance metrics of the YOLO model and the total power consumption during execution were recorded for analysis by using smartmeter.

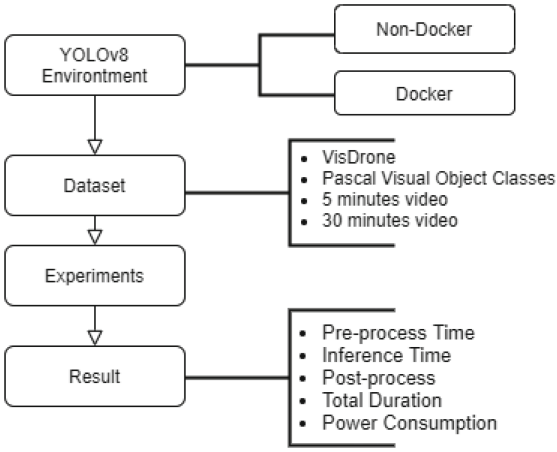

Fig. 4. Research Methodology

4 Implementation

Figure 5 depicts the experiment conducted with the NVIDIA Jetson Orin Nano. The figure demonstrates the model's attempt to recognize objects using the VisDrone dataset, which comprises 6,471 images. The object recognition result such as 1 car, 1 boat, 1 truck, 4 cell phones, etc.

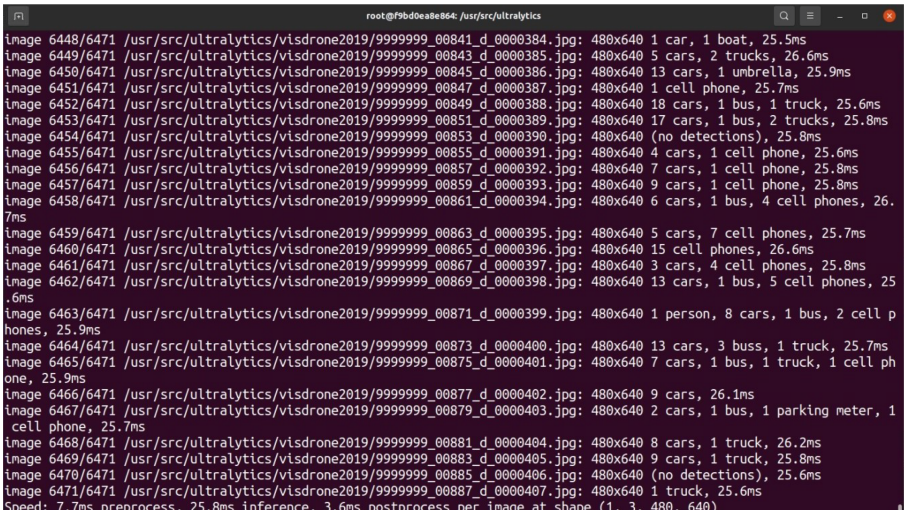

Fig. 5. Running YOLO experiment on Jetson Orin Nano

Figure 6 compare the CPU performance of the NVIDIA Jetson Orin Nano during object recognition with the VisDrone dataset. The results indicate that using Docker leads to a longer duration for object recognition compared to running without Docker. This is due to Docker's resource management, which aims to reduce energy consumption, resulting in lower average CPU and GPU usage levels compared to running YOLO in a non-Docker environment. Figure 7 illustrates the comparison of power consumption on the Pascal VOC dataset. The figure reveals that power usage is lower when Docker is utilized compared to running without Docker.

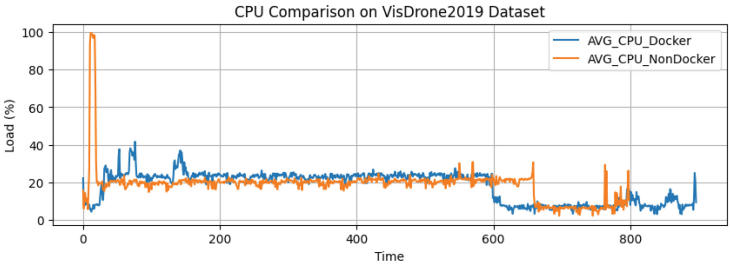

Fig. 6. CPU Comparison on VisDrone2019 Dataset

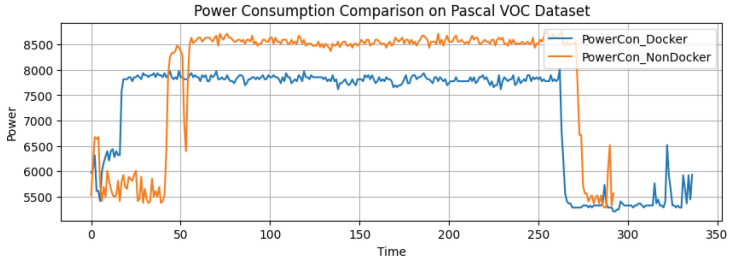

Fig. 7. Power Usage Comparison on Pascal VOC Dataset

After running several experiments on several prepared datasets, we saved the experimental data in the Table 1.

Figure 8 and Fig. 9 shows the comparison of the experiments result in a chart and the average power consumption/second.

Table 1. Performance Comparison during the experiments

	VisDrone		Pascal VOC		5 min video		30 min video	
Parameter	Non-docker	Docker	Non-docker	Docker	Non-docker	Docker	Non-docker	Docker
pre-process (ms)	6.1	5.7	8.1	7.7	4.2	3.7	4.2	3.7
inference (ms)	16.1	27.2	13.3	25.8	13.0	25.5	15.2	25.9
post-process (ms)	5.6	3.6	4.1	3.6	4.4	3.7	6.1	3.8
total duration (s)	233	263	650	698	784	969	3994	5336
power usage consumption (watt)	1258.7	1413.7	2583	2764.4	2866	3437.8	13221	15574.4

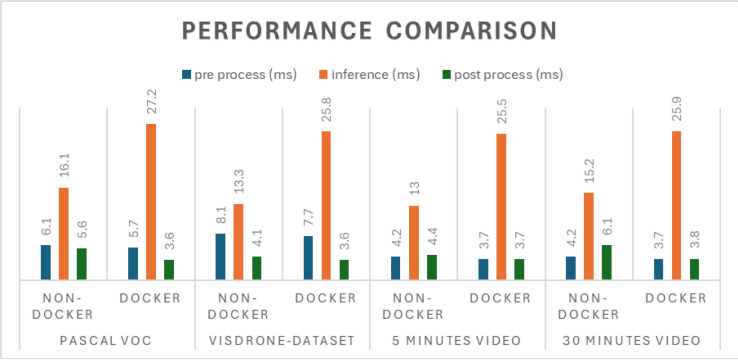

Fig. 8. Performance Comparison Chart

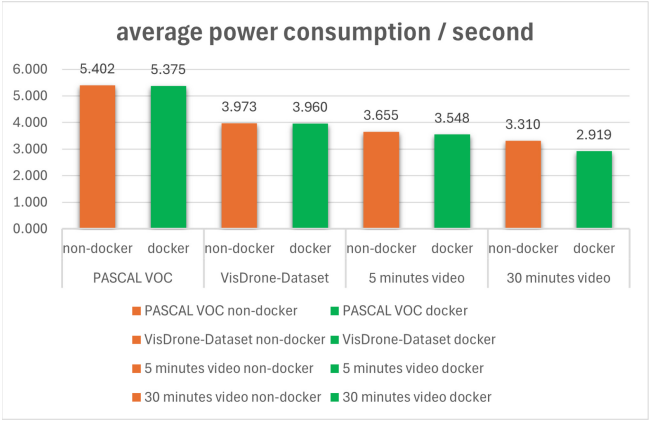

Fig. 9. Average Power Consumption/second

5 Conclusion

In conclusion, Docker offers convenience, isolation, and portability, making it a popular choice for many developers. However, bare-metal environments may deliver slightly better performance and are often preferred when direct hardware access or minimal overhead is essential. The experiments conducted led to the conclusion that power consumption for object recognition using the YOLO machine learning model is reduced when Docker is used, in comparison to running the model without Docker.

Acknowledgements. This work was sponsored by the National Science and Technology Council (NSTC), Taiwan ROC, under Grant No. 113-2622-E-029-003, and 113-2221-E-029-028-MY3.

References

1. Kristiani, E., Yang, C.T., Huang, C.Y.: iSEC: an optimized deep learning model for image classification on edge computing. IEEE Access **8**, 27267–27276 (2020)
2. Kristiani, E., Yang, C.T., Huang, C.Y., Ko, P.C., Fathoni, H.: On construction of sensors, edge, and cloud (iSEC) framework for smart system integration and applications. IEEE Internet Things J. **8**(1), 309–319 (2021)
3. Lien, C.T.: Fire and smoke detection using yolo through kafka. In: Frontier Computing on Industrial Applications vol. 4, pp. 269–275. Springer Nature Singapore (2024)
4. Kristiani, E., Chen, Y.-C., Yang, C.-T., Li, C.-H.: Flame and smoke recognition on smart edge using deep learning. J. Supercomput. **79**(5), 5552–5575 (2023)
5. Fathoni, E.: A container-based of edge device monitoring on kubernetes. In: Frontier Computing, pp. 231–237. Springer Singapore (2021)
6. Anish, A.R.S., Malini, A., Archana, T.: Enhancing surveillance systems with YOLO algorithm for real-time object detection and tracking. In: 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1254–1257 (2023)
7. Sahin, O., Ozer, S.: YOLODrone: improved YOLO architecture for object detection in drone images. In 2021 44th International Conference on Telecommunications and Signal Processing (TSP), pp. 361–365 (2021)
8. Quick Start Guide: NVIDIA Jetson with Ultralytics YOLOv8. https://docs.ultralytics.com/guides/nvidia-jetson/. Accessed 11 Sept 2024
9. VisDrone-Dataset. https://github.com/VisDrone/VisDrone-Dataset. Accessed 11 Sept 2024
10. Visual object classes challenge 2012 (VOC2012). http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html Accessed 11 Sept 2024