Performance Evaluation of YOLO Across Diverse Computing Architectures

Kyle Acosta, Rafael Baltazar, Cuauhtemoc Lona, Peter Tran,

Department of Electrical and Computer Engineering, College of Engineering,

California State Polytechnic University, Pomona
3801 W Temple Ave, Pomona, CA 91768

kaacosta@cpp.edu

rbaltazar@cpp.edu

cxlona@cpp.edu

petertran1@cpp.edu

***Abstract*— This project focuses on the performance evaluation of the YOLO (You Only Look Once) algorithm, comparing its efficiency on different computing architectures. Seven distinct computers, each featuring unique CPU and GPU configurations, were tested to benchmark YOLO's performance metrics, including inference time, frame rate, and resource utilization. By systematically analyzing the computational efficiency of the algorithm on CPUs and GPUs, the study highlights the impact of hardware architecture on YOLO's performance. This work provides actionable insights for optimizing YOLO deployments, enabling practitioners to select the most effective platforms for real-time object detection tasks.**

I. Introduction

The YOLO (You Only Look Once) algorithm has gained significant attention in computer vision due to its ability to perform real-time object detection with high accuracy. Despite its widespread adoption, efficiently implementing YOLO across diverse computing architectures remains a challenge, as hardware capabilities can significantly influence its performance. Evaluating these performance differences is essential for identifying optimal hardware configurations tailored to specific application needs.

This study centers on the performance evaluation of YOLO, with a focus on systematically benchmarking its execution across a variety of CPUs and GPUs. Using seven distinct computer systems, the research investigates critical performance metrics, including inference time, frame rate, and resource utilization, to quantify how hardware architecture impacts YOLO's efficiency.

By prioritizing performance evaluation, this work aims to bridge the gap between algorithmic demands and hardware capabilities, providing a comprehensive understanding of YOLO's computational characteristics. The results of this study will guide practitioners in making informed decisions when deploying YOLO for real-time applications.

The remainder of this paper is organized as follows: Section II describes the system, Section III discusses the performance evaluation, Section IV reviews related work, and Section V concludes with future research directions.

II. System

A. System Setup

Seven distinct computer systems were utilized, each equipped with unique combinations of CPUs and GPUs to evaluate YOLO's performance. The software environment was standardized across all systems, using Python 3.10.5, YOLO version 8.3.39, and ONNX (Open Neural Network Exchange) runtime with DirectML for GPU acceleration. The YOLO 11m model was exported to the ONNX format for consistency. Performance was benchmarked using the first 512 images of the COCO Val 2017 dataset with a batch size of 16 to ensure compatibility across varying hardware configurations. Two of the systems operated on Windows 11, while the remaining five used Windows 10. These two systems were kept in

the evaluations, however it is noted that this would cause an unfair benchmark.

B. Evaluation Metrics

To comprehensively assess performance, three key metrics were measured:

1. Inference Time: The time required to process images, indicating algorithmic efficiency.
2. Frame Rate: The number of frames processed per second, reflecting real-time capability.
3. Resource Utilization: CPU and GPU usage during inference, highlighting hardware efficiency under load.

C. *Metric Collection*

Performance metrics were gathered using HWiNFO, a robust hardware monitoring software. This tool provided detailed, real-time insights into CPU and GPU usage, memory allocation, and power consumption. By standardizing data collection across all systems, HWiNFO ensured accurate and reliable measurements, minimizing variability in results.

C. *Testing Protocol*

To ensure accurate and repeatable results, all tests adhered to rigorous protocols. Background processes were minimized, and identical workload conditions were maintained across systems. Multiple trials were conducted, with results averaged to mitigate variability. This approach ensured reliable and consistent comparisons across hardware setups.

III. Performance Evaluation

The evaluation results underscore the significant impact of hardware architecture on YOLO's performance.

A. *GPUs vs. CPUs*

As shown in Fig. 1 (Median Throughput for GPUs) and Fig. 2 (Median Throughput for CPUs), GPUs consistently achieved higher frame rates than CPUs, demonstrating their superior parallel processing capabilities. For example, high-end GPUs such as the AMD Radeon RX 7900 GRE achieved median throughput far exceeding that of mid-range GPUs like the NVIDIA GTX 1070. This trend is further corroborated by Fig. 3 and Fig. 4, where GPUs maintained consistently lower inference times compared to CPUs, highlighting their efficiency in handling YOLO's computational demands.

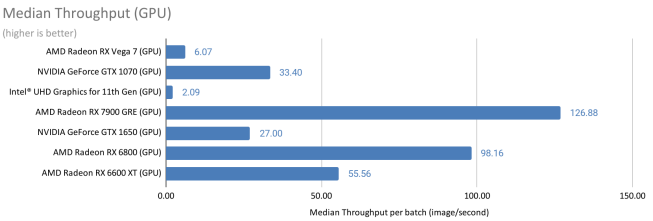

Fig. 1 Median Throughput (GPU)

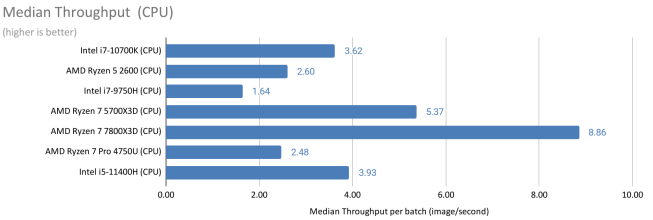

Fig. 2 Median Throughput (CPU)

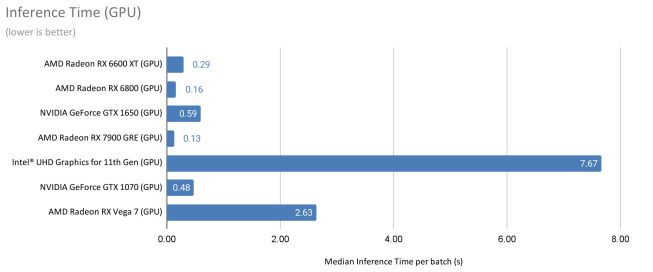

Fig. 3 Inference Time (GPU)

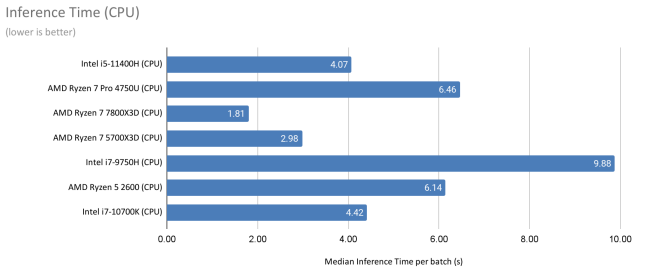

Fig. 4 Inference Time (CPU)

B. *Architecture Impact*

Performance variations across hardware configurations were substantial. Fig. 5 (Power Draw for GPUs) and Fig. 6 (Power Draw for CPUs) indicate that high-end GPUs outperformed CPUs not only in speed but also exhibited higher power consumption. However, when normalized for efficiency, as shown in Fig. 7 (Power Efficiency for GPUs) and Fig. 8 (Power Efficiency for CPUs), GPUs maintained an advantage, making them the preferred choice for real-time applications where efficiency is paramount.

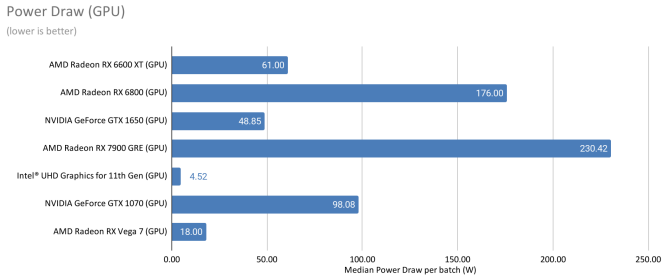

Fig. 5 Power Draw (GPU)

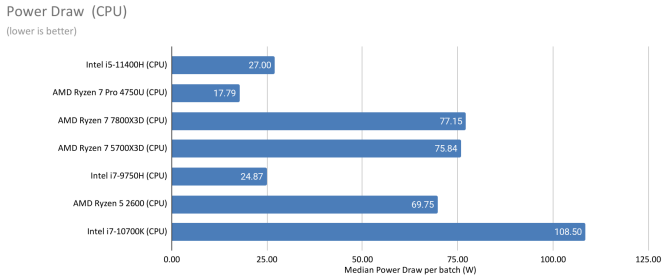

Fig. 6 Power Draw (CPU)

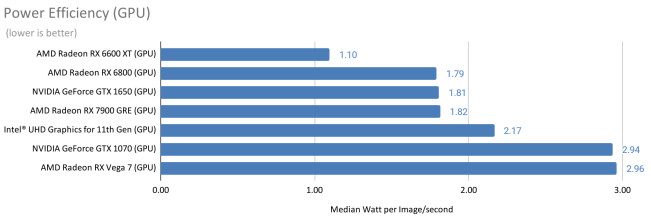

Fig. 7 Power Efficiency (GPU)

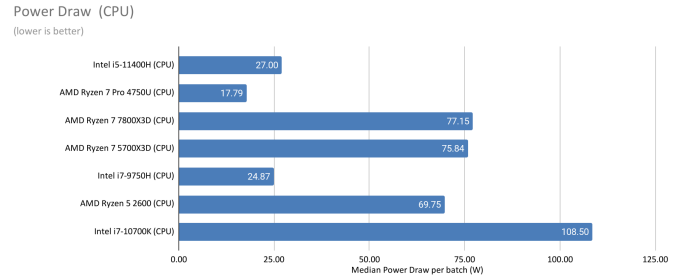

Fig. 8 Power Efficiency (CPU)

C. *Utilization Trends*

Resource utilization trends further illustrate the disparity in hardware capabilities. Fig. 9 (Memory Usage for GPUs) reveals consistently high GPU utilization during inference, demonstrating effective resource allocation. In contrast, Fig. 10 (Memory Usage for CPUs) shows frequent bottlenecks in CPU performance under similar workloads. These differences are summarized in Fig. 11 (Performance Scores for GPUs) and Fig. 12 (Performance Scores for CPUs), where GPUs achieved significantly higher scores, further validating their dominance in YOLO tasks.

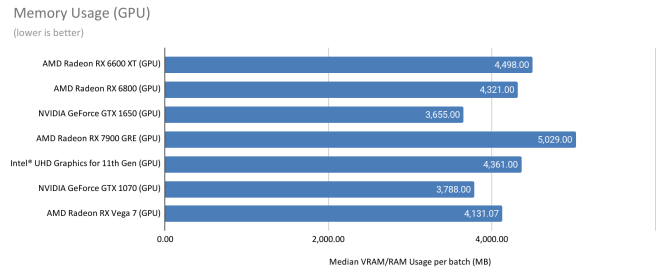

Fig. 9 Memory Usage (GPU)

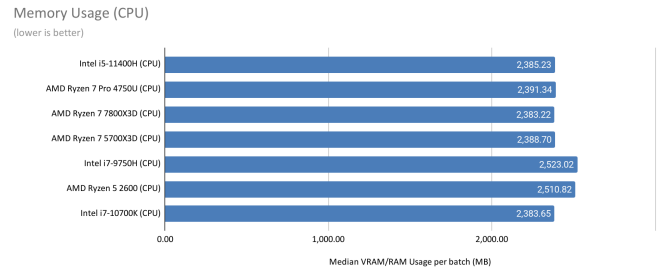

Fig. 10 Memory Usage (CPU)

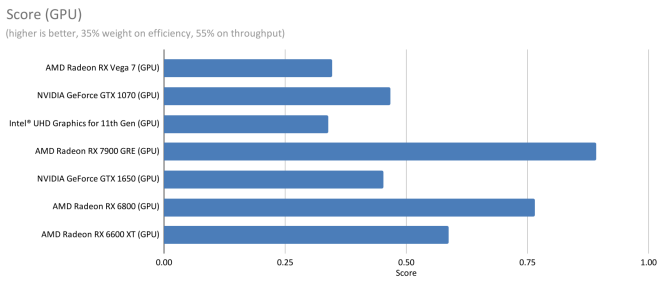

Fig. 11 Score (GPU)

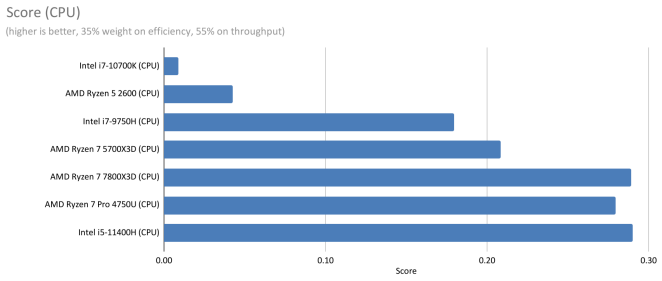

Fig. 12 Score (CPU)

By correlating metrics with graphical data, this section highlights the decisive role of hardware architecture in optimizing YOLO for diverse applications. These insights provide a foundational understanding for deploying YOLO in real-world scenarios, emphasizing the advantages of leveraging high-performance GPUs.

IV. Related Work

YOLO has been widely utilized in real-world applications due to its efficiency in object detection. For example, traffic sign detection systems employ YOLO to enhance driver assistance systems, reducing car accidents by providing accurate real-time information [1]. Similarly, underwater object detection for marine exploration leverages YOLO to identify and track objects in challenging environments [2]. In the realm of aerial imaging, YOLO has been adapted to detect small objects, addressing challenges in analyzing large-scale images [3]. Despite these advances, there remains a need for a detailed performance evaluation across diverse hardware setups to optimize deployment.

V. Conclusion and Future Work

This study highlights the importance of hardware architecture in optimizing YOLO performance. GPUs demonstrated superior efficiency and processing power compared to CPUs, making them more suitable for real-time object detection tasks. However, the results also underscore the variability in performance based on specific hardware configurations.

Future evaluations should:

1. Standardize operating systems and configurations across all setups to further eliminate variability.
2. Incorporate NVIDIA CUDA and AMD ROCm for GPU benchmarking to compare against DirectML results.
3. Develop more rigorous data collection protocols with synchronized timestamps to improve measurement precision.

This study provides a foundational understanding of YOLO's performance across diverse hardware, paving the way for more informed decision-making in deploying object detection systems.

References

[1] Flores-Calero, Marco, et al. “Traffic sign detection and recognition using Yolo Object Detection Algorithm: A systematic review.” Mathematics, vol. 12, no. 2, 17 Jan. 2024, p. 297, https://doi.org/10.3390/math12020297.

[2] Yang, Yuyi, et al. “UGC-Yolo: Underwater Environment Object Detection based on Yolo with a global context block.” Journal of Ocean University of China, vol. 22, no. 3, 13 May 2023, pp. 665–674, https://doi.org/10.1007/s11802-023-5296-z.

[3] Hu, Mengzi, et al. “Efficient-lightweight YOLO: Improving small object detection in Yolo for aerial images.” Sensors, vol. 23, no. 14, 15 July 2023, p. 6423, https://doi.org/10.3390/s23146423.