Yolov8 Benchmarking

Ruben Banzon
Electrical and Computer Engineering
California State Polytechnic University, Pomona
Pomona, USA
rsbanzon@cpp.edu

Nina L. Fajardo
Electrical and Computer Engineering
California State Polytechnic University, Pomona
Pomona, USA
hlfajardo@cpp.edu

Jeff Magbitang
Electrical and Computer Engineering
California State Polytechnic University, Pomona
Pomona, USA
jdmagbitang@cpp.edu

Alexander Ov
Electrical and Computer Engineering
California State Polytechnic University, Pomona
Pomona, USA
aov@cpp.edu

Mohamed El-Hadedy
Electrical and Computer Engineering
California State Polytechnic University, Pomona
Pomona, USA
mealy@cpp.edu

***Abstract*—Computer vision stands as a widely integrated technology within people's daily lives with object detection being the primary variation that could be further specified into numerous subcategories. Currently, an average user can experience moderately high requirements for training smaller-sized databases; however, as the implementation of object detection increases - specifically in the fields of Healthcare, Defense, Agriculture, and Capitalism - the question arises if engineers can train high-powered models on low-cost equipment. This report describes a project testing a large dataset over a high number of epochs and comparing its performance across four devices with different yet average specifications. At the moment, the possibility of training high-powered models on low-cost equipment exists, yet the time of computation and the resources required to train high-powered models do not present as viable. One would recommend that the specification of an average user's device needs to be increased before training high-powered models on low-cost equipment becomes more viable and accessible to an average person.**

***Index Terms*—yolov8, computervision, benchmarking, computer architecture**

I. Introduction

In this work, section II covers YOLOv8 Modeling and its current usage, Section III covers all the models and dataset results. Section IV covers the performance analysis of Computers 1 and 2. Section V will give a recommendation and evaluation based on the results of the experiment.

II. YOLO Modeling

YOLO is an algorithm introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. The name stands for "You Only Look Once" as YOLO's algorithm allows for real-time objection detection by interpreting object detection as a regression problem. Thus, YOLO objection detection works by utilizing four steps: residual blocks, bounding box regression, intersection over unions (otherwise known as 'IOU'), and non-maximum suppression.

Identify applicable funding agency here. If none, delete this.

Residual blocks are the NxN grid cells that are created by dividing the original image into equal parts. In this grid, each cell is responsible for detecting the class of the object and its probability of prediction or confidence value within its boundaries. Bounding box regression is used to determine which cells have objects within them by utilizing the following format, representing a final vector representation for each individual cell.

$$Y = [pc, bx, by, bh, bw, c1] \tag{1}$$

"pc" corresponds to each cell's probability of prediction or confidence value. "bx" and "by" are the x and y coordinates with respect to the center of each individual cell as its bounding box/measurement. "c1" correspond to object classes. However, there can be additional classes according to how many objects are within that individual cell in the grid. A bounding box is simply created once it detects an object and the probability score is greater than 0. The distinction between a bounding box and an individual cell is that a cell is a residual box from the NxN grid. A bounding box is a box that is bounded around the objects detected, and a comparison is made between adjacent cells to determine the size of that bounding box.

Intersection Over Unions - or IOU - is a form of filtration to only keep grid boxes that are relevant and reach a certain threshold. IoU is a measurement of how accurately the program detects an object by comparing its prediction to the real object. A high IoU means higher accuracy on how well the program detects an object. The selection threshold is predetermined by the user, but it is typically set above 0.5 or

0.75 for better accuracy. The formula for calculating IOU is as shown below:

$$IOU = [\frac{IntersectionArea}{UnionArea}] \quad (2)$$

The final step of the YOLO Modeling process is Non-Max Suppression, or NMS, where boxes with the highest probability score of detection are kept instead of keeping all the boxes with sound that meet the threshold. NMS filters out redundant or overlapping bounding box predictions for potential objects in the image by utilizing the IoU measurement. Each bounding box has its respective IoU measurement and NMS continuously repeats the process of eliminating redundant boxes until the box with the highest IoU remains.

A. *YOLOv8 Modeling*

While all YOLO modeling follows the same algorithm as above, there are multiple iterations of YOLO - all with different purposes and creators. For this particular benchmarking test, YOLOv8 was chosen. YOLOv8 was introduced by Ultralytics - who also created YOLOv5 - in 2022. YOLOv5 is the predecessor of YOLOv8, however, YOLOv8 was changed to have a new anchor-free detection system, changes to the convolutional blocks used, and mosaic augmentation applied during training. This makes YOLOv8 faster and more accurate compared to its predecessor.

YOLOv8's code is open source and is licensed under a GPL license.

B. *Current Usage of YOLOv8 Modeling*

YOLO object detection model can be used in a wide range of applications across many types of domains and its versatility makes it an excellent model to use for real-time detection tasks. In autonomous vehicles, YOLO has rapid capabilities of detecting and tracking pedestrians, vehicles, and road obstacles which ensures the safety and precision of autonomous driving systems. YOLO also aids in medical imaging and diagnosis of diseases by detecting abnormalities in radiological images.

Another notable use of YOLO involves military applications. A great example is the military aircraft dataset used for this paper. By training YOLO on images featuring military aircraft, the model effectively identifies these aircraft in various scenarios. This YOLO capability is valuable for increasing military security and reconnaissance efforts. It can contribute to the development of advanced surveillance systems, support military operations, and improve defense capabilities.

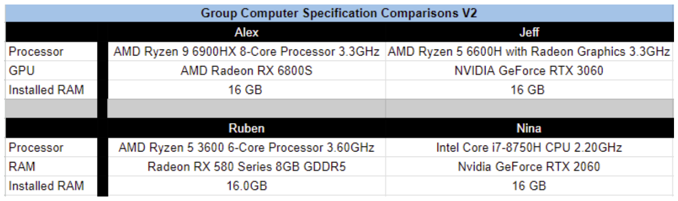
YOLO modeling extends beyond these examples as it is highly versatile and customizable to specific object detection applications. Its speed and accuracy make it a go-to model for real-time computer vision tasks in a multitude of industries.

III. Models and Dataset Results

A. *Tested Environments*

The dataset used in this report is an open-source dataset created by "university of Pretoria" called "Airplane Detection"[1]. It includes 3877 images of aircraft that have already been equipped with their corresponding aircraft type. All four devices tested in this report have all trained the same dataset using the same command template:"yolo task=detect mode=train epochs=100 data=data.yaml model=yolov8m.pt imgsz=640". The only possible difference in this command may be in the number of epochs due to time constraints or a file name may be changed depending on how it was named for each individual.

Figure 1 shows the specifications of each computer tested in this project. Computer 1 can otherwise be referred to as Alex's Computer. Computer 2 can be referred to as Jeff's Computer. Computer 3 can be referred to as Ruben's Computer, while Computer 4 can be referred to as Nina's Computer.

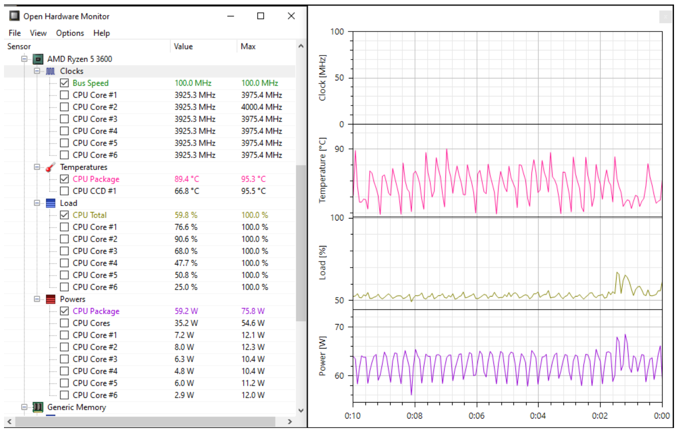


Group Computer Specification Comparisons V2		
	Alex	Jeff
Processor	AMD Ryzen 9 6900HX 8-Core Processor 3.3GHz	AMD Ryzen 5 6600H with Radeon Graphics 3.3GHz
GPU	AMD Radeon RX 6800S	NVIDIA GeForce RTX 3060
Installed RAM	16 GB	16 GB
	Ruben	Nina
Processor	AMD Ryzen 5 3600 6-Core Processor 3.60GHz	Intel Core i7-8750H CPU 2.20GHz
RAM	Radeon RX 580 Series 8GB GDDR5	Nvidia GeForce RTX 2060
Installed RAM	16.0GB	16 GB

Fig. 1. Computer specifications of tested devices.

From Figure 1, it is noted that all computers had 16 GB Installed RAM. The only predominant differences were the processors and their GHz and the RAM on each device. In addition, only one device (Computer 4) had an Intel processor. The operating systems were all the same running Windows 11 Home, Version 22H2 with the Windows Feature Experience Pack 1000.22677.1000.0. During the experiment, all computers were used only to train the models in their Best Performance modes. All additional applications that are unnecessary to boot a computer device were turned off and halted - besides our benchmarking software.

The benchmarking software used for this experiment was OpenHardwareMonitor in logging increments of 1 minute - as shown in Figure 2.



```
Sensor                     Value         Max
AMD Ryzen 5 3600
  Clocks
    ✓ Bus Speed            100.0 MHz     100.0 MHz
      CPU Core #1          3925.3 MHz    3975.4 MHz
      CPU Core #2          3925.3 MHz    4000.4 MHz
      CPU Core #3          3925.3 MHz    3975.4 MHz
      CPU Core #4          3925.3 MHz    3975.4 MHz
      CPU Core #5          3925.3 MHz    3975.4 MHz
      CPU Core #6          3925.3 MHz    3975.4 MHz
  Temperatures
    ✓ CPU Package          89.4 °C       95.3 °C
      CPU CCD #1           66.8 °C       95.5 °C
  Load
    ✓ CPU Total            59.8 %        100.0 %
      CPU Core #1          76.6 %        100.0 %
      CPU Core #2          90.6 %        100.0 %
      CPU Core #3          68.0 %        100.0 %
      CPU Core #4          47.7 %        100.0 %
      CPU Core #5          50.8 %        100.0 %
      CPU Core #6          25.0 %        100.0 %
  Powers
    ✓ CPU Package          59.2 W        75.8 W
      CPU Cores            35.2 W        54.6 W
      CPU Core #1          7.2 W         12.1 W
      CPU Core #2          8.0 W         12.3 W
      CPU Core #3          6.3 W         10.4 W
      CPU Core #4          4.8 W         10.4 W
      CPU Core #5          6.0 W         11.2 W
      CPU Core #6          2.9 W         12.0 W
Generic Memory
```

Fig. 2. OpenHardwareMonitor application user interface.

B. Trained Dataset Examples

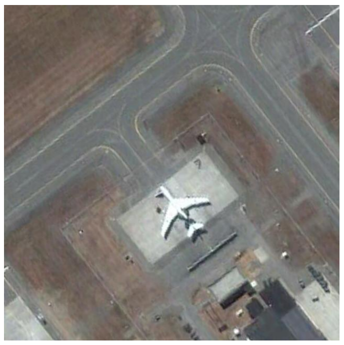


Fig. 3. Original photo from dataset before training.

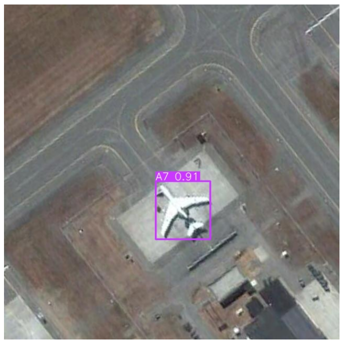


Fig. 4. Trained photo from dataset showing an A7 class airplane.

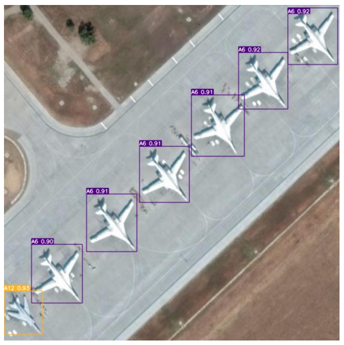


Fig. 5. Another trained photo from dataset showing two classes of airplanes.

C. Mathemathical Models

Due to the significant similarities in performance in the OpenHardwareMonitor benchmarking files and in Figures 2-4, this report from this point on will only focus on the comparison between Computer 1 and Computer 2.

IV. Performance Analysis

As shown in Fig.7, Fig.8, and Fig. 9, YOLOv8 has consistent training instances across different computers. For Fig.6, the training for Computer 1 is corrupted, thus it only shows training instances for classes A1, A10, and A11.

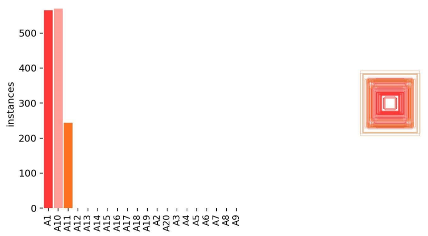


Fig. 6. Computer 1's training instances chart developed through YOLOv8 training.

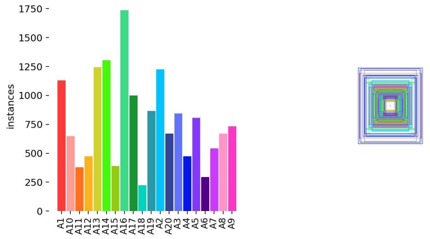


Fig. 7. Computer 2's training instances chart developed through YOLOv8 training.

The precision of the YOLOv8 model could be examined across different confidence thresholds for each class by analyzing the Precision-Confidence Curve. Again, Computer 1's training was corrupted and it can only show curves for classes A1, A10, and A11. Despite that, it can be seen that YOLOv8 is still precise across different confidence level thresholds. Similarly, as shown in Fig. 12 and Fig. 13, Computer 2 and Computer 3 show a consistent Precision-Confidence Curve with some classes having lower precision especially as the confidence level threshold gets closer to 1.0.

V. Conclusion

In conclusion, the accuracy of modeling using low-performance components such as Laptop CPUs and GPUs does not affect the accuracy of the computer vision model. However, using low-performance components significantly increases the time it takes to train the model comparing the training between the GPU and CPU there was a 700% increase in training time. In general, GPUs consume up to 300 watts of power for machine learning while CPUs consume 65 watts of power indicating that GPUs consume 461% more power than the CPU. Overall, for small-scale applications, it is feasible and reasonable to use low-performance CPUs over GPUs when using machine learning models such as YOLOv8 modeling.

References

[1] "YOLO Object Detection Explained: A Beginner's Guide," www.datacamp.com. https://www.datacamp.com/blog/yolo-object-detection-explained

[2] "YOLOv8 vs. YOLOv5: Choosing the Best Object Detection Model," www.augmentedstartups.com. https://www.augmentedstartups.com/blog/yolov8-vs-yolov5-choosing-the-best-object-detection-model

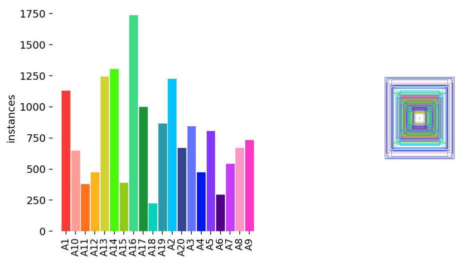


Fig. 8. Computer 3's training instances chart developed through YOLOv8 training.

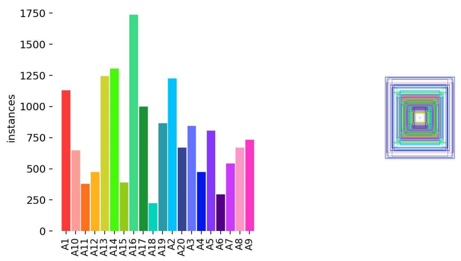
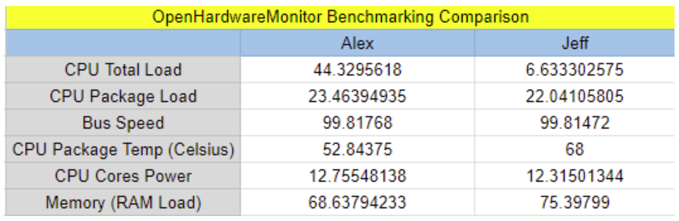


Fig. 9. Computer 4's training instances chart developed through YOLOv8 training.

[3] Bader Aldughayfiq, F. Ashfaq, N. Zaman, and M. Humayun, "YOLO-Based Deep Learning Model for Pressure Ulcer Detection and Classification," vol. 11, no. 9, pp. 1222–1222, Apr. 2023, doi: https://doi.org/10.3390/healthcare11091222.

[4] A. Sarda, S. Dixit, and A. Bhan, "Object Detection for Autonomous Driving using YOLO algorithm," 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), Apr. 2021, doi: https://doi.org/10.1109/iciem51511.2021.9445365.

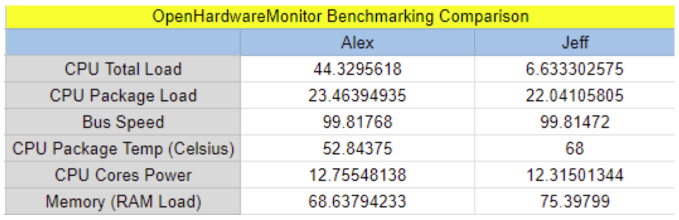
[5] "Airplane Detection Object Detection Dataset and Pre-Trained Model by university of Pretoria," Roboflow. https://universe.roboflow.com/university-of-pretoria-2deim/airplane-detection-nzg5n (accessed Dec. 06, 2023).



OpenHardwareMonitor Benchmarking Comparison

	Alex	Jeff
CPU Total Load	44.3295618	6.633302575
CPU Package Load	23.46394935	22.04105805
Bus Speed	99.81768	99.81472
CPU Package Temp (Celsius)	52.84375	68
CPU Cores Power	12.75548138	12.31501344
Memory (RAM Load)	68.63794233	75.39799

Fig. 10. OpenHardwareMonitor benchmarking averages comparison between Computers 1 and 2



OpenHardwareMonitor Benchmarking Comparison

	Alex	Jeff
CPU Total Load	44.3295618	6.633302575
CPU Package Load	23.46394935	22.04105805
Bus Speed	99.81768	99.81472
CPU Package Temp (Celsius)	52.84375	68
CPU Cores Power	12.75548138	12.31501344
Memory (RAM Load)	68.63794233	75.39799

Fig. 11. OpenHardwareMonitor benchmarking averages comparison between Computers 1 and 2