# Creating YOLO Models to Detect Aircraft Carriers

Virgil Nhieu [#1], Angie Sy [#2], Ethan Le [#3], Lyndon Lam [#4]

*Electrical and Computer Engineering Department,*
*California Polytechnic State University, Pomona*
*Pomona, USA*

[1] nhieu@cpp.edu
[2] absy@cpp.edu
[3] enle@cpp.edu
[4] llam@cpp.edu

*Abstract*— This paper delves into the benchmarking of computer architecture using YOLOv5, an object detection software. Our project specifically utilizes this by training several different central processing units (CPUs) over several learning cycles known as epochs. The main goal is to determine which CPU performs the best by training to recognize aircraft carriers through pictures. The software that we will utilize are HWinfo (Windows), turbostat (Linux), and inxi (Linux) for hardware monitoring and Yolo, which will be discussed later.

We hope to compare the specifications of each CPU compared to their training time to find the best CPU through their training time, architecture, clock speeds, and package power. We will later then compare the significance of having a higher number of epochs to train with. The significance of our project lies closely with that of benchmarking and showing the performance of the model, which will aid in informing decisions about choosing systems tailored to specific tasks or needs.

## I. Introduction

Benchmarking computer architecture is a technique for analysing and comparing various computer systems to determine how well they accomplish specific tasks. It entails performing standardized tests on components such as processors, memory, and graphics to assess their performance, efficiency, and overall capabilities/performance. The performance of YOLO is proportional to the power of the hardware the program is running on. To train a custom model for our purpose, it requires a powerful computer to complete the training process. Different architectures from different companies can have an impact on the performance of training a custom model.

The significance of benchmarking comes from its ability to assist us in making efficient decisions about the systems we use in our desired tasks. For instance, if we wanted a system that was good for gaming, we would benchmark set systems to see the gaming performance of each system. In some cases, benchmarking may be done already, and buyers can decide through overview of the benchmarks to make their own decisions.

## II. Introduction to YOLO: Real-time object detection

YOLO, short for You Only Look Once, is a neural network object detection algorithm that provides the user with real time object detection. Designed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, it gives the user a fast and accurate algorithm for Object Detection. There are certain versions of YOLO that improve further from the previous editions, with the latest version YOLOv8.

YOLO offers a pre-trained model of some popular objects in everyday life such as a cell phone, bicycles, persons, cars, and so on. If the pre-trained model does not fit the requirements for the project, it also allows you to create your own custom model as well. For this example, we will be using aircraft carriers as the object YOLO will find in images.

In this paper, we explore the speed of training a custom model designed to find aircraft carriers on Intel and AMD CPUs and then benchmarking the results.

## III. Benchmarking Process

The dataset used for benchmarking the training process was picked up from Roboflow and includes over 500 images of aircraft carriers provided by the user, new-workspace-xugd11. For YOLO, we used V7.0 of YoloV5 cloning it from GitHub onto our computers used for our benchmarking. Python version 3.11.6 and version 2.1.1 of Pytorch will be used to interact with the YOLO algorithm. For our GPUs, we will be using the Cuda 12.1 version of Pytorch. To measure our CPU statistics while it is training, on Windows, we will use the latest version of HWInfo64 as of November 29, 2023, version 7.66. On Linux, Turbostat version 6.5 and inxi version 3.3.13 will be used instead to measure the same statistics compared to Windows.

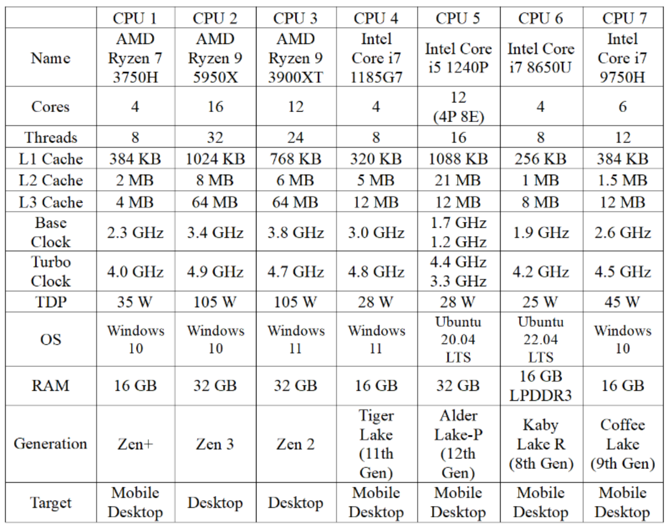
We then used a terminal (Command Prompt on Windows or the Terminal on Linux) to start the training process. The size of the image being trained is 640 and the number of epochs being benchmarked is 10. The size of the YOLO version being used is YOLOv5s. When training, the times measured per epoch was measured provided by YOLO's python program. The statistics of each CPU was taken during the training process, starting from when the first epoch started training to the final epoch being completed. From there the average clock speeds and wattage were noted.

To test the number of epochs, we used a GPU to reduce the time it takes for training. We used the same command compared to benchmarking with our CPUs, but changed the epoch amount from 20 to 100, 160 and finally 300, to compare the difference in epoch levels.

## IV. Benchmarking Materials

In Table 1, it shows the CPUs used to run and benchmark using our process stated above. The CPUs used include some desktop chips and mobile chips from AMD and primarily mobile chips from Intel. It also includes a hybrid CPU design from Intel that houses both performance cores and efficiency cores. The table includes the number of cores, threads, amount of cache for all levels, clock speeds, thermal design power, the operating system the CPU is running in, amount of RAM, the generation architecture, and the target platform for the CPU.

From AMD, we have the laptop Ryzen 7 3750H and their desktop CPUs, Ryzen 9 5950X and lastly the Ryzen 9 3900XT. From Intel the CPUs tested were all from their laptop line, including the Core i7 1185G7, Core i5 1240P, Core i7 8650U, and finally the Core i7 9750H.



| | CPU 1 | CPU 2 | CPU 3 | CPU 4 | CPU 5 | CPU 6 | CPU 7 |
|---|---|---|---|---|---|---|---|
| Name | AMD Ryzen 7 3750H | AMD Ryzen 9 5950X | AMD Ryzen 9 3900XT | Intel Core i7 1185G7 | Intel Core i5 1240P | Intel Core i7 8650U | Intel Core i7 9750H |
| Cores | 4 | 16 | 12 | 4 | 12 (4P 8E) | 4 | 6 |
| Threads | 8 | 32 | 24 | 8 | 16 | 8 | 12 |
| L1 Cache | 384 KB | 1024 KB | 768 KB | 320 KB | 1088 KB | 256 KB | 384 KB |
| L2 Cache | 2 MB | 8 MB | 6 MB | 5 MB | 21 MB | 1 MB | 1.5 MB |
| L3 Cache | 4 MB | 64 MB | 64 MB | 12 MB | 12 MB | 8 MB | 12 MB |
| Base Clock | 2.3 GHz | 3.4 GHz | 3.8 GHz | 3.0 GHz | 1.7 GHz 1.2 GHz | 1.9 GHz | 2.6 GHz |
| Turbo Clock | 4.0 GHz | 4.9 GHz | 4.7 GHz | 4.8 GHz | 4.4 GHz 3.3 GHz | 4.2 GHz | 4.5 GHz |
| TDP | 35 W | 105 W | 105 W | 28 W | 28 W | 25 W | 45 W |
| OS | Windows 10 | Windows 10 | Windows 11 | Windows 11 | Ubuntu 20.04 LTS | Ubuntu 22.04 LTS | Windows 10 |
| RAM | 16 GB | 32 GB | 32 GB | 16 GB | 32 GB | 16 GB LPDDR3 | 16 GB |
| Generation | Zen+ | Zen 3 | Zen 2 | Tiger Lake (11th Gen) | Alder Lake-P (12th Gen) | Kaby Lake R (8th Gen) | Coffee Lake (9th Gen) |
| Target | Mobile Desktop | Desktop | Desktop | Mobile Desktop | Mobile Desktop | Mobile Desktop | Mobile Desktop |

TABLE 1 – CPU INFORMATION TABLE

We have also opted to compare the performance of our CPUs with two GPUs (Graphics Processing Units). We had available an EVGA FTW3 Ultra Nvidia RTX 3090 and EVGA XC Ultra Nvidia RTX 2070 Super that were used to compare the performance of GPUs on training versus CPUs. The specifications of each graphics card are shown below in Table 2. In comparison to the CPUs in table 1, the two GPUs consume a significantly higher amount of energy.

| | GPU 1 | GPU 2 |
|---|---|---|
| Name | EVGA RTX 3090 | EVGA RTX 2070 Super |
| Cores[1] | 10496 | 2560 |
| Threads | - | - |
| L1 Cache | 10496 KB[2] | 2560 KB[3] |
| L2 Cache | 6MB | 4 MB |
| L3 Cache | - | - |
| Base Clock | 1395 MHz | 1605 MHz |
| Turbo Clock | 1800 MHz | 1800 MHz |
| TDP | 350 W | 215 W |
| OS | Windows 10 | Windows 11 |
| RAM | 24G VRAM GDDR6X | 8GVRAM GDDR6 |
| Generation | Ampere (3000 series) | Turing (2000 Series) |
| Target | Graphics Card | Graphics Card |

TABLE 2 – GPU INFORMATION TABLE

## V. Benchmarking Results

After obtaining data for all the hardware used for the benchmark, Figure 1 shows the average time taken for each epoch trained by each computer. The slowest by far is the Ryzen 7 3750H at just a little over 567 seconds. It is followed by the Intel Core i7 1185G7 then the Core i7 9750H. Rounding out the best of the rest is the Intel Core i7 8650U. The final three CPUs were very closely tied for the fastest CPU, but ultimately the fastest was the Ryzen 9 5950X, followed by the Ryzen 9 3900XT, and finally the Intel Core i5 1240P.

Compared to the CPUs tested, GPUs are massively quicker at training, tens or sometimes hundreds of times. The RTX 3090 completed training an epoch at a measly 3 seconds, and the RTX 2070 Super finishing in 5.4 seconds.

[1] CUDA Cores

[2] 128 KB across 82 Streaming Multiprocessors

[3] 64 KB across 40 Streaming Multiprocessors

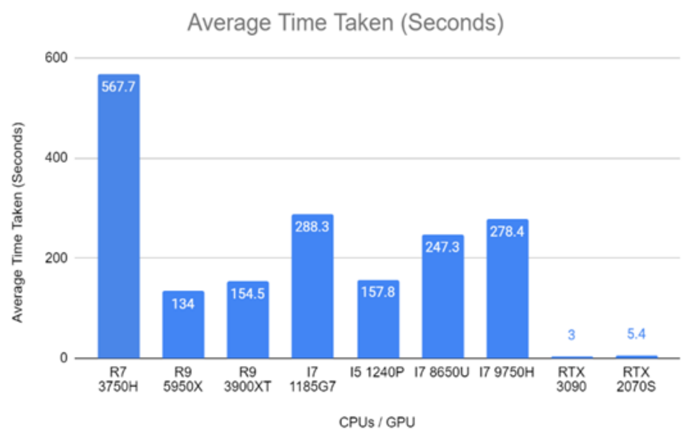


FIGURE 1 – AVERAGE TIME TAKEN

In Figure 2, the average wattage of the CPU package power is measured. The most power-hungry CPUs are the desktop CPUs, the Ryzen 9 5950X and the Ryzen 9 3900XT. The next power-hungry CPU is the i7 9750H, followed by the 1185G7, i5 1240P, and finally the Ryzen 7 3750H. The GPUs on the other hand consumed much more power, the RTX 3090 consumed 315 watts and the RTX 2070 Super consumed a little over 150 watts.

From both figures, we can tell that the Ryzen 9 5950X consumed the most power but also finished training an epoch the fastest. The 3900XT is slightly behind the Ryzen 9 5950X, but also consumed about half the power. A surprise is the Intel Core i5 1240P, which is equally as fast as the 3900XT, but consumed less than half of the power of the 3900XT at a mere 30.8 watts.

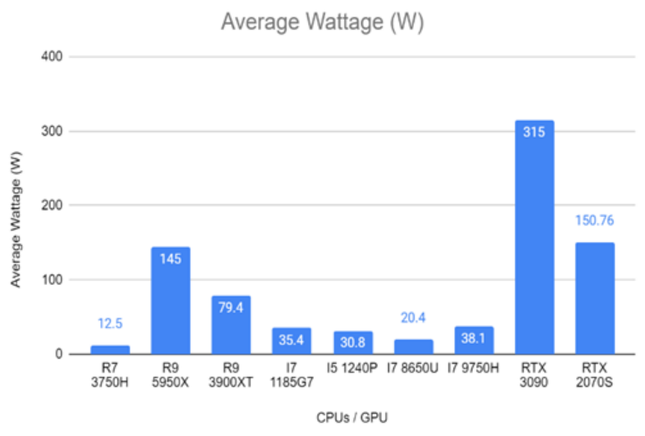
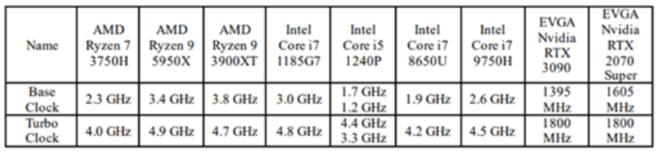


FIGURE 2 – CPU/GPU PACKAGE POWER

In Figure 3, the average clock speed of the CPUs were measured. The desktop processors ran faster overall, the Ryzen 9 5950X at 4.46 GHz and 3900XT at 4.1 GHz respectively. Surprisingly the Intel Core i5 1240P ran the slowest out of the CPUs at 2.5 GHz. It may come down to how the program reads core clocks as the 1240P is a hybrid CPU, combining both performance and efficiency cores.

From the data, the CPUs tested could not reach their rated turbo clock specifications. This is due to how CPU manufacturers rate their CPU speeds. For example, the Ryzen 9 5950X has a rated turbo clock of 4.9 GHz. When tested however, it only achieved 4.46 GHz. On AMD's specification website the boost clock of 4.9 GHz is only achievable for a singular core for a short amount of time. They call this short burst of performance "Precision Boost" or "Turbo Boost" for Intel processors. This can lead to mislead customers thinking that their processor can run at the higher speed that their CPU can.

Graphics cards on the other hand, run faster than their rated turbo clock. For the RTX 3090, the rated turbo clock is 1800 MHz, but after testing, the GPU was running at 1935 MHz. Comparatively, the RTX 2070 Super ran at 1941 MHz when the rated Turbo Clock was the same as the 3090, at 1800 MHz. This is due to how the GPU can read its limits. The GPU reads many factors; voltage, temperature, and power supplied to it, and determines if it can boost past the rated turbo clock of the GPU.



| Name | AMD Ryzen 7 3750H | AMD Ryzen 9 5950X | AMD Ryzen 9 3900XT | Intel Core i7 1185G7 | Intel Core i5 1240P | Intel Core i7 8650U | Intel Core i7 9750H | EVGA Nvidia RTX 3090 | EVGA Nvidia RTX 2070 Super |
|---|---|---|---|---|---|---|---|---|---|
| Base Clock | 2.3 GHz | 3.4 GHz | 3.8 GHz | 3.0 GHz | 1.7 GHz 1.2 GHz | 1.9 GHz | 2.6 GHz | 1395 MHz | 1605 MHz |
| Turbo Clock | 4.0 GHz | 4.9 GHz | 4.7 GHz | 4.8 GHz | 4.4 GHz 3.3 GHz | 4.2 GHz | 4.5 GHz | 1800 MHz | 1800 MHz |

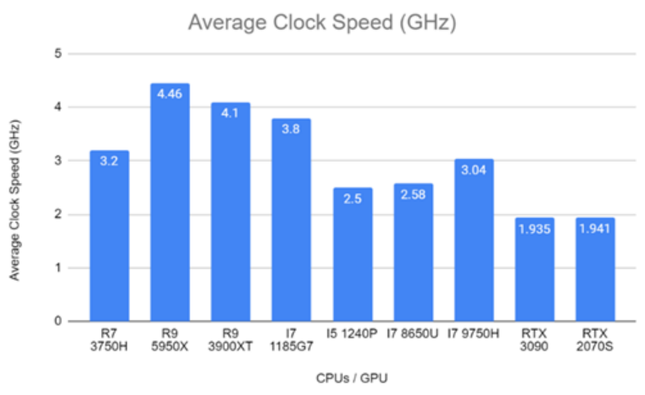


FIGURE 3 – CLOCK SPEED INFORMATION

## VI. OBJECT DETECTION

We trained our YOLO model for 300 epochs on the GPU to reduce amount of time taken on the CPU. We tested the model on four images containing aircraft carriers.

1. Japanese Helicopter Destroyer: Hyuga
2. New Aircraft Carrier: USS Gerald R. Ford
3. Old Aircraft Carrier: USS Enterprise (CV-6)
4. Aircraft Carrier with Ski-Jump Takeoff Ramp: Admiral Kuznetsov

For legal reasons, Japan does not have aircraft carriers. Arguably, their helicopter destroyers can be considered as aircraft carriers. We used our YOLO model to test this and to see what our model thinks. We tested our model on a modern aircraft carrier and on a World-War-Two era aircraft carrier. Finally, we tested our YOLO model on an aircraft carrier with a ski-jump take-off ramp.

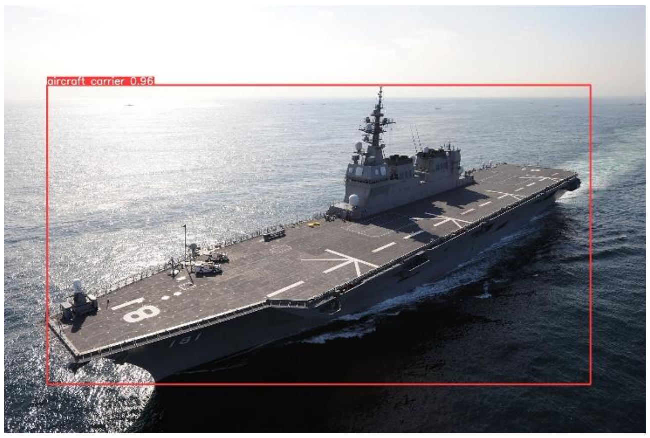


Figure 4 – Helicopter Destroyer Carrier

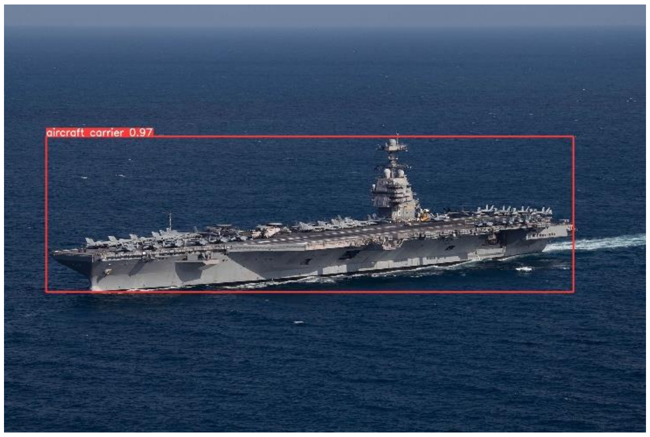


Figure 5 – USS Gerald R. Ford

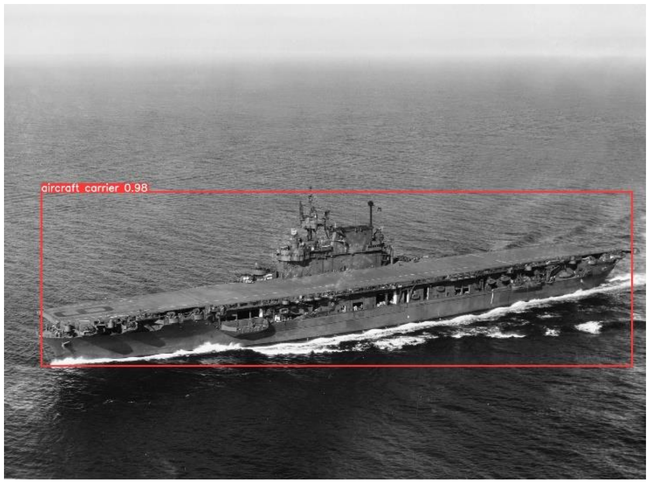


Figure 6 – USS Enterprise (CV-6)

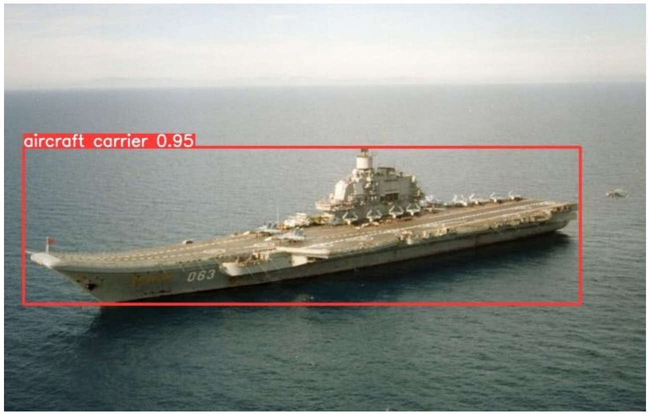


Figure 7 – Admiral Kuznetsov

From all four pictures above, the model labeled the aircraft carriers with confidence of greater than 90%. The next four images are the same image but are tested with a 20 epoch model instead. From the figures below, we can see that the 20 epoch model has a greatly reduced confidence than the model above. Since for every epoch, the model gets more confident in its decisions. If there are not enough epochs, then the model cannot be conclusive to whether the image provided to it is an aircraft carrier or not.

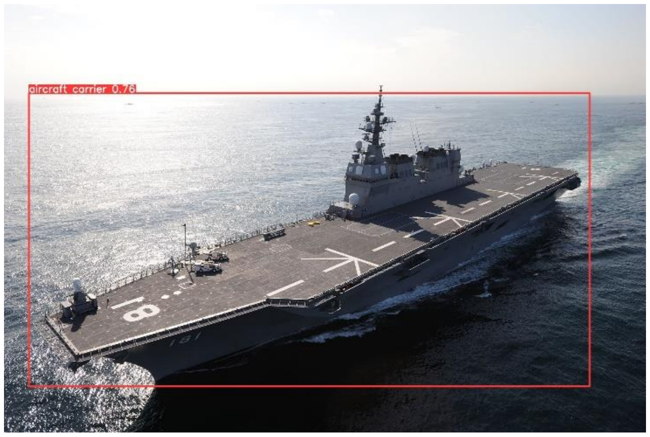


Figure 8 – Helicopter Destroyer Carrier (20 Epochs)

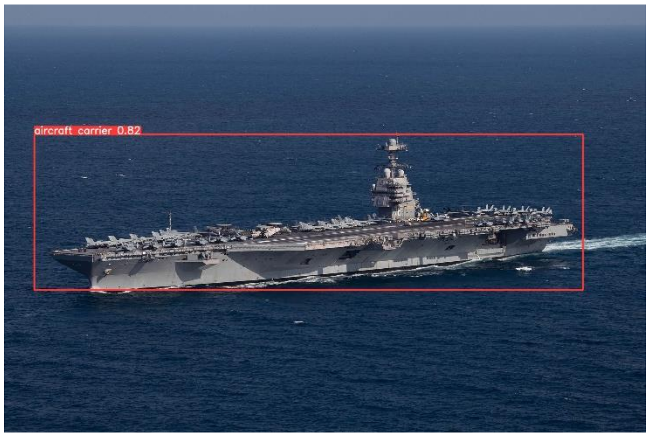


Figure 9 – USS Gerald R. Ford (20 Epochs)

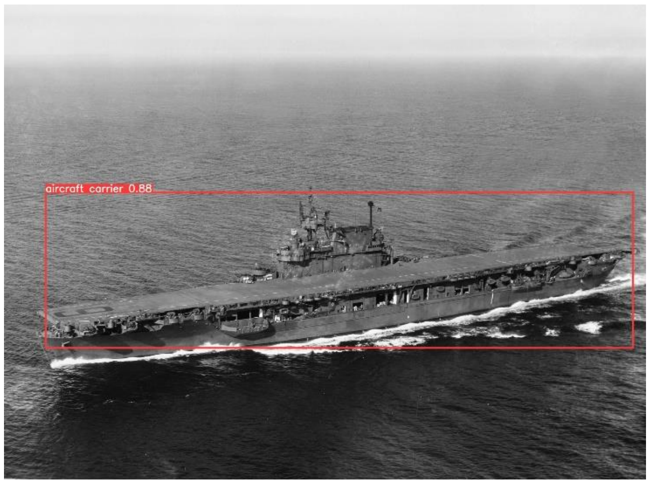


Figure 10 – USS Enterprise (CV-6) (20 Epochs)

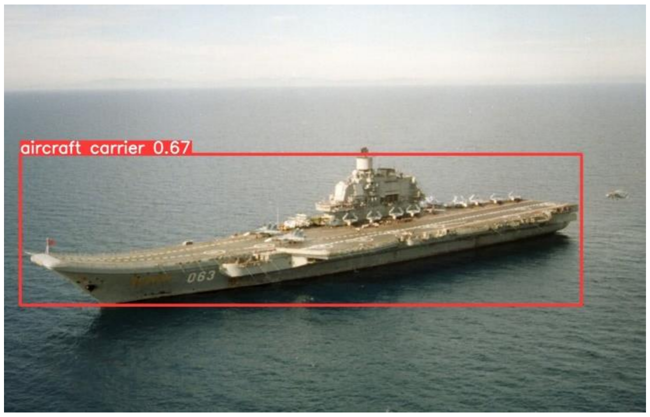


Figure 11 – Admiral Kuznetsov (20 Epochs)

For most of the CPUs trained on 10 epochs, aircraft carriers were detected most of the time with a fairly acceptable confidence score. However, in some images where there were many objects surrounding the aircraft carrier, the object detection struggled, resulting in a low confidence score or not being able to detect the aircraft at all. In figure 8 below, the model trained on 20 epochs mistakenly labeled the resupply ship as an aircraft carrier. The confidence rating of the aircraft carrier is only at 51% while the resupply ship's confidence rating is 34%. This shows that the model is not very confident at making decisions at which ship is an aircraft carrier or not. However, with the more trained 300 epochs model, the issue does not exist, and is also more confident in its label, shown in Figure 13.

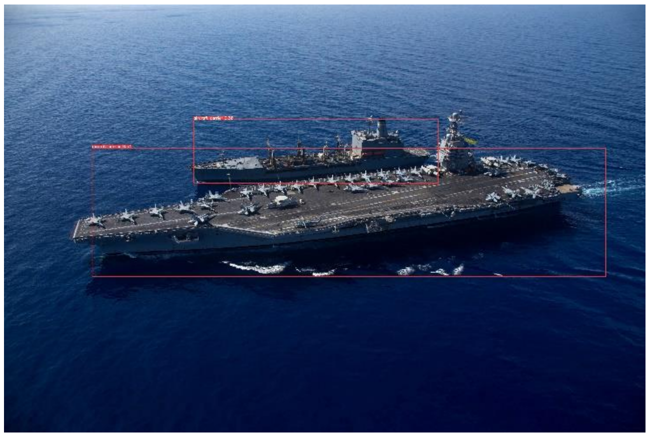


Figure 12 – Aircraft Carrier with Resupply Ship (20 Epochs)

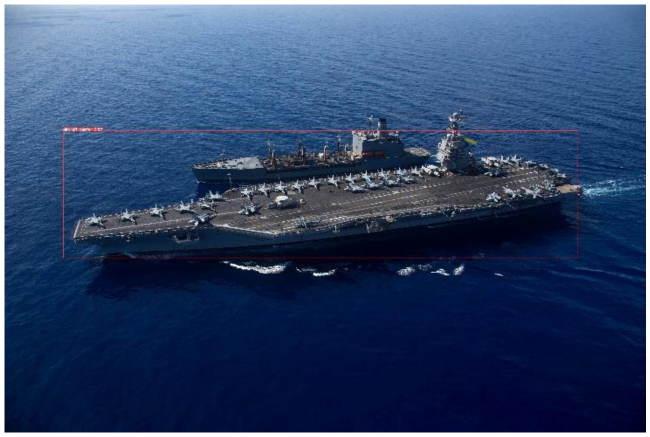


Figure 13 – Aircraft Carrier with Resupply Ship (300 Epochs)

We then tested the precision of the model at differing epoch numbers. The first figure is the 20 epoch model, followed by the 100 epoch model, 160 epoch model, and finally the 300 epoch model model. We can see with the 20 epoch model that there is not enough data to have a precise data. The dotted line showing the average precision of the model shows that the precision only is on average about 75%.  The 100 epoch model on the other hand, is much more precise as it has more epochs to train with. At the end of its training, the precision of the 100 epoch model is around 90%, which is a much greater improvement. However, as we increase the amount of epochs to train with, the model starts to lose precision, in other words, the model is overfitting with the data. We can see this with the 160 epoch model and the 300 epoch model where at some stages it loses a lot of precision drastically. For example the 160 epoch model, it loses its precision near the end of its training. With the 300 epoch model it loses precision slowly over time around 150 to 225 epochs, and then drastically at around 250.

From these figures, we can see that there is slightly or no precision increase from increasing the epoch count when training the model. In fact, there might be even a slight decrease in precision when training the model for long sets. From the four examples, the ideal number of epochs to train with is around 100, given its precision and lack of overfit compared to the larger epochs.

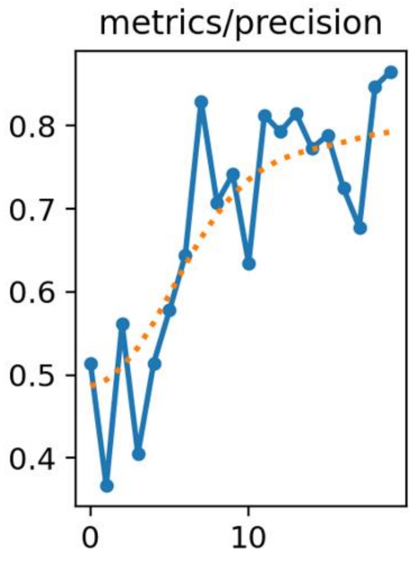


Figure 14 - 20 Epochs

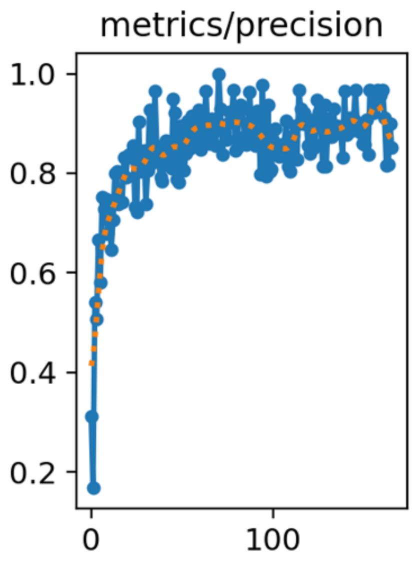


Figure 16 – 160 Epochs

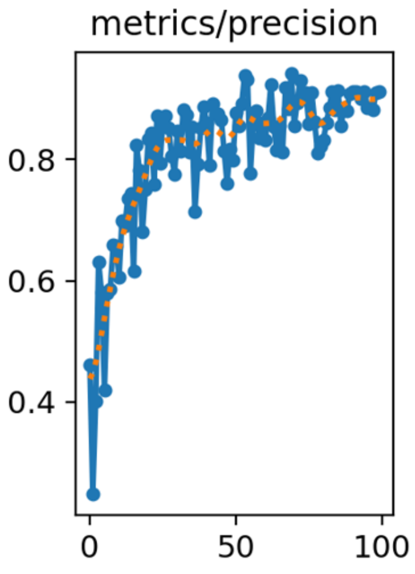


Figure 15 – 100 Epochs

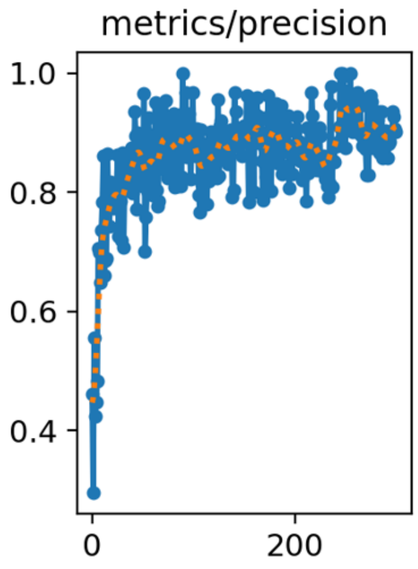


Figure 17 – 300 Epochs

## VII. CONCLUSION

In conclusion, the iterative training process exposed an important pattern: with each epoch, the model's accuracy improves significantly until it reaches a saturation point where following epochs stop improving. But also, too little epochs of course would yield less accurate results. It is critical to strike a balance between training efficacy and resource optimization. According to our findings, the Ryzen 5950X is the best pick for raw performance per epoch, demonstrating its ability to handle the demands of heavy computational tasks.

The I5 1240P, on the other hand, is the best choice for performance per watt, providing a power efficient solution without sacrificing precision. The Ryzen 7 3750H is the preferred option in resource constrained situations, proving efficiency in instances when computing resources are limited (space rovers).

The ideal number of epochs for training a model is around 100 epochs. Too little epochs and the model is underfit and does not have enough data to train and have a high precision. Too many and the model overfits and loses precision because of training the model too much on the same data.

This project provided us with great experience and knowledge on benchmarking, and much was learned.

## References

[1] Ultralytics, "Comprehensive guide to ultralytics yolov5," Ultralytics YOLOv8 Docs, https://docs.ultralytics.com/yolov5/ (accessed November 30, 2023).
[2] new-workspace-xugd1. (2021). n02687172-n03786901 Dataset. Roboflow. https://universe.roboflow.com/new-workspace-xugd1/n02687172-n03786901 (accessed November 29, 2023)
[3] AMD. (2020). AMD Ryzen$^{TM}$ 9 5950X. https://www.amd.com/en/products/specifications/processors (accessed November 30, 2023)