# Evaluating YOLO Object Detection On Various Platforms

CalPolyPomona
College of Engineering
Electrical and Computer Engineering

Kyler Martinez (CpE), Ruben Gutierrez (CpE), Austin Tran (CpE), Nelson Chong (CpE), and Mohamed El-Hadedy

RSCA
Research, Scholarship and Creative Activities

## Abstract

YOLO (You Only Look Once) is an object detection framework using a single convolutional network. YOLOv5 is the 5th iteration of the YOLO series that uses Focus structure with CSPdarknet53 as a backbone. It is a regression-based algorithm that takes in the entire image as opposed to classification-based algorithms that scans the image one pixel at a time. YOLOv5 classifies images using a grid and predicts where to place bounding boxes based on the established objects, classes, and structures. We tested YOLOv5 on PCs, specifically a desktop and laptop, a Raspberry Pi 4, and used it with an ESP-32 microcontroller. The PCs utilize an x86 processor, which is a CISC architecture. On the other hand, the Pi utilizes an ARM-based CPU, which is RISC architecture, and the ESP32 is a single-core RISCV-based microcontroller, but YOLO did not run on it directly. Various tests are performed to measure and capture the performance metrics.

## Introduction

Computer vision has become an increasingly important field with an increase in object detection algorithms of varying capabilities. These algorithms require large amounts of computing and electrical power. These constraints and processor resources must be considered. Many microcontrollers cannot execute algorithms like YOLOv5, but systems can use another device to process the object detection algorithm while the microcontroller performs other tasks. Each solution to this problem has flaws, but machine learning on small devices will become more accessible as technology develops.

## Metrics and Methods

The metrics used are inference time, frames per second, and power consumption. Inference time is the time it takes for forward propagation in milliseconds. The inference time is calculated by dividing FLOPs by FLOPS, where FLOPs is the number of floating-point operation, and FLOPS is the number of floating-point operations per second. The frames per second is the inverse of the inference time. To measure power consumption, a KUMAN Power Meter KW47 is used for the Raspberry Pi and ESP32 microcontroller, and a software called Core Temp is used for the PCs. To measure power consumption, take the difference between the average power and the idle power. The power is measured in Watts.

## Results

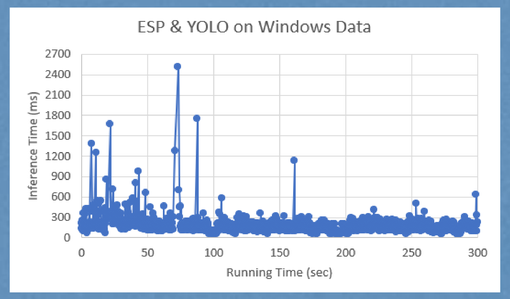
On the PC, the image and the camera test set performs very well. The camera performs with an FPS of about 125 while the image set performs with an FPS of 100.1. In addition, the amount of power needed for both are significantly different with the image test requiring more power than the camera test.

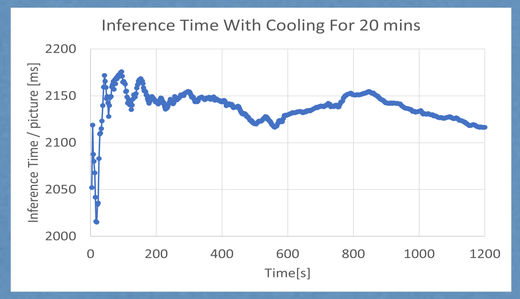
On the laptop, the image test set performs better than a live camera feed with 4.81 FPS, but at the cost of more power consumption.

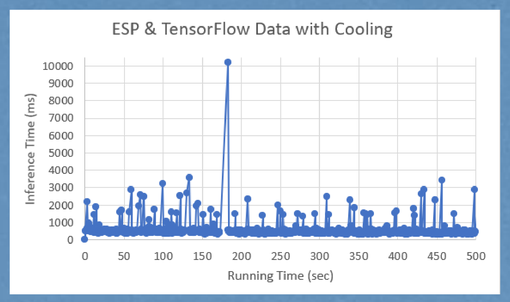
On the Raspberry Pi, YOLOv5 is impractical because it processes images at .46 FPS at best. Greater performance and stability was found with cooling.

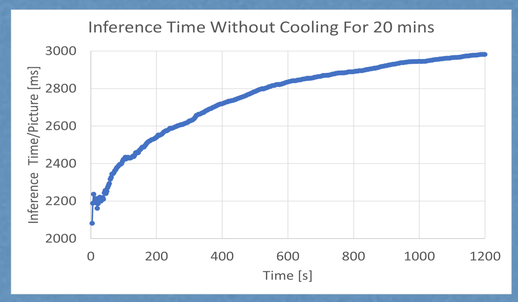
The ESP32 micro with the TensorFlow algorithm performs at an average 1.90 FPS, while the ESP32 micro with YOLOv5 on the desktop PC performs at an average 6.04 FPS.

| YOLOv5 Results Summary: PC | | |
|---|---|---|
| | Process from File | Process from Camera |
| Average FPS | 100.1 | 125 |
| Average IT | 9.99 | 8.0 |
| Average Power | 31.425 | 12.075 |

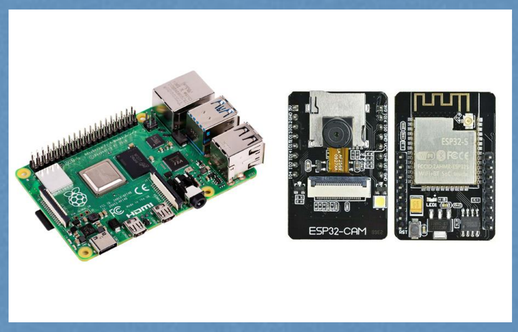







## Summary and Conclusions

YOLOv5 is powerful when applied to problems that require computer vision but is computationally intensive. Modern PCs have the capabilities to process frames and perform inferences fast enough to be useful in many applications. On the other hand, YOLOv5 underperforms on microprocessors such as the Raspberry Pi, and is infeasible for many microcontrollers.

An alternative for these devices is to run a less intensive algorithm such as TensorFlow lite. It is possible to implement YOLOv5 on the microcontrollers by streaming the video feed from the microcontroller and process it on another machine. However, this method accrues additional costs and forces internet constraints that bottlenecks the overall system.

## Future Work

In the future, we want to test similar object detection algorithms using TensorFlow Lite. YOLO may not be cable of running on smaller devices, but devices are in development that contain Tensor Processing Units (TPUs) that will make it easier to create neural networks and other algorithms on tiny devices.

In the future, these devices would be tested similarly to the Raspberry Pi and ESP32 to determine if they are viable solutions for machine learning applications.

## References

Ultralytics (2022) YOLOv5 (Version 7.0) [Source Code]
FreedoomTechWeb (2022) YOLOv5Raspberry Pi 4 [Source Code]
TensorFlow (2022) TensorFlow (Version 2.11.0) [Source Code]
FreedoomTechWeb (2022) ESP32Cam Object Detection [Source Code]
Core Temp. [Online]. [Accessed: 26-Nov-2022].

## Acknowledgments

We thank Dr. Mohamed El-Hadedy for their guidance and feedback throughout the duration of our work and studies.