Performance Comparison: YOLOv11 vs. YOLOv8

Group F

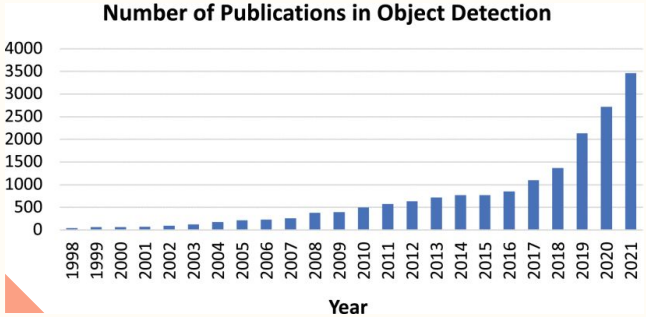
Kenneth Bach, Aaron De Castro
Shrikanth Shivakumar, Tom Solomon

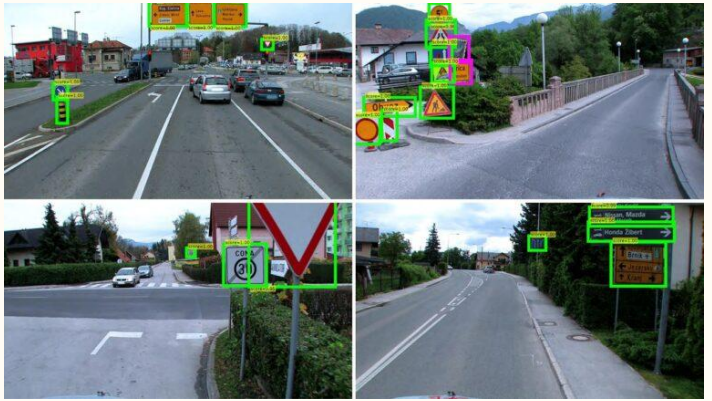
Table of Contents

Presented by: Aaron

Motivation and Introduction

- What is Object Detection?
- The Rise of Object Detection
  - Increased usage in many industries
  - Growth of Research
- Our Purpose
  - Two Models: YOLOv11 and YOLOv8
  - Traffic Sign Recognition





State of the Arts

much of the literature reviewed:

- Discusses comparisons between YOLOv5 and YOLOv8
- Datasets for:
  - Vehicle plates
  - Types of vehicles
  - Wildfire detection
- Conclusions: YOLOv8 performed generally better
- Literature not the same, but informative

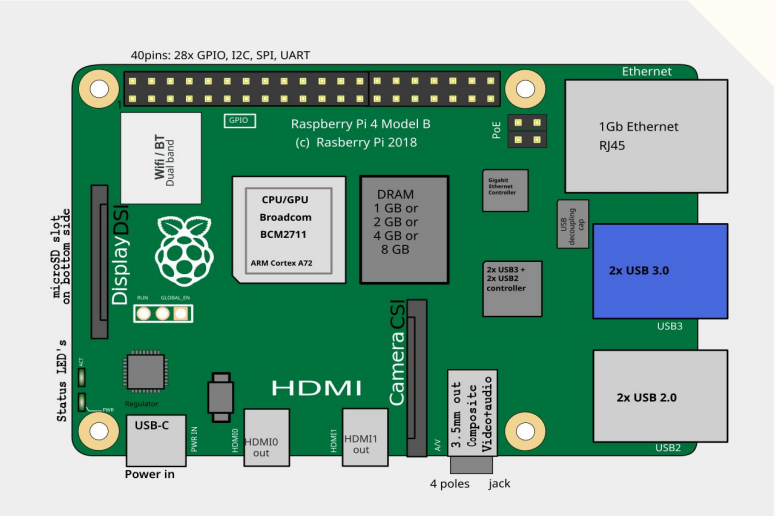
Fig. 1. Sample images (a) Car (b) Bike (c) Track

Presented by:

Our System - Hardware

Raspberry Pi 4

- A powerful single board computer
- Quad-core ARM processor
- Running at 1.5 GHz
- Very compatible with lightweight machine learning frameworks like YOLO



Presented by:

Our System - Software

Google Colab

- Cloud based IDE
- Access to powerful GPUs and TPUs, perfect for machine learning applications
- Free version has some hardware limitations

Perf

- Tool for performance measurement
- Used to measure different parameters:
  - Cycles
  - Instructions
  - CPU time

Our System - Software (cont)

YOLO (You Only Look Once)

- Processes images in real-time
- Predicts bounding boxes and class probabilities directly from the image
- Single layer neural network allows for computational efficiency
- Processes the full image in a single pass

- Limitations:
  - Can struggle with detecting small objects in crowded environments
  - Performance goes down with overlapping or complex shaped objects

Presented by:

Our System - Software

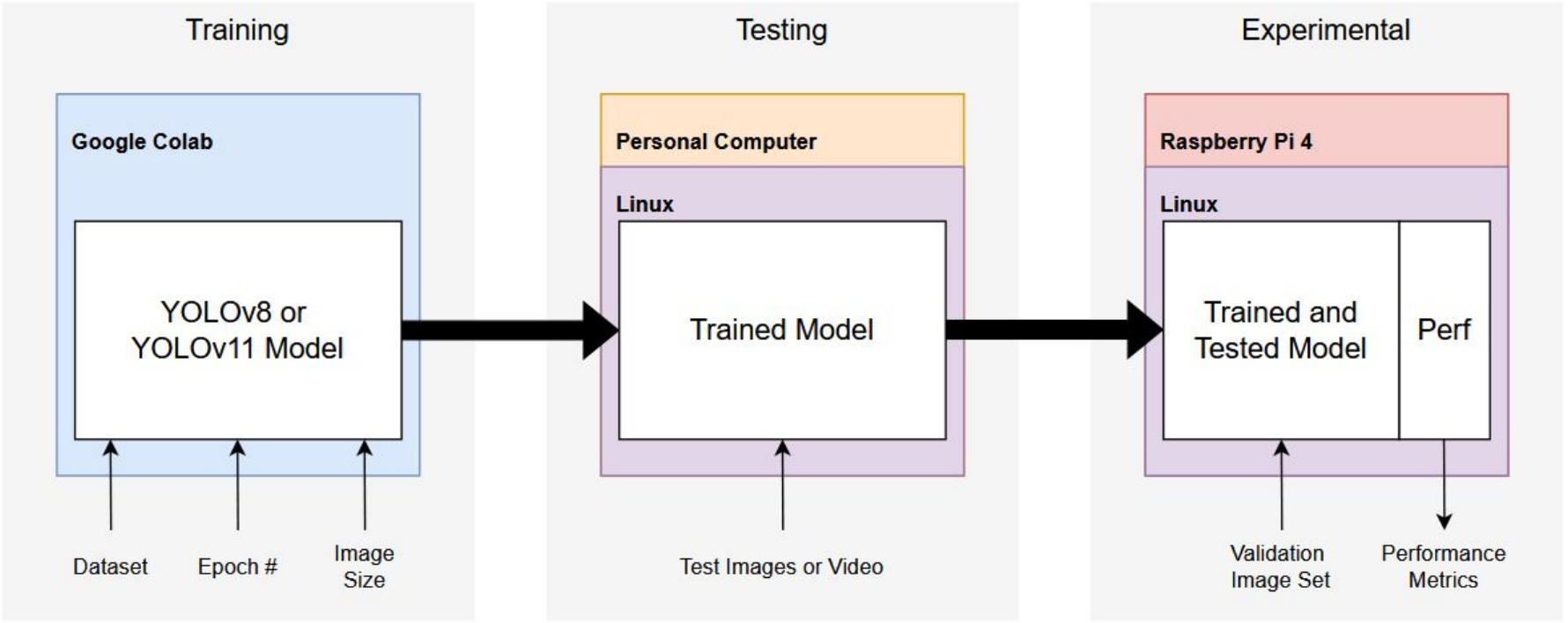
YOLO v8

- Introduced anchor-free detection head
- User friendly training interface and added support for multiple tasks
- Supports:
  - Object detection
  - Image classification
  - Instance segmentation

YOLO v11

- Refines architecture with more efficient backbone and neck design
- Enhances feature representation and detection precision
- Improves accuracy and faster inference times
- Includes pose estimation oriented object detection

Our System - Overview



Our System - Methods

- Download YOLOv8 dataset from Roboflow
- Train using Google Colabs
- Deploy model on personal computer, then Raspberry Pi
- Use Perf for performance analysis
- Repeat with YOLOv11

Performance Analysis

- YOLOv8 takes less time to train
- Both models have similar accuracies and levels of confidence

Model	Time (hours)
YOLOv8	1.690
YOLOv11	2.001

Model	Class	Images	Precision	Recall	mAP50
YOLOv8	All	144	0.897	0.867	0.907
	25 MPH	20	0.940	0.78	0.946
	35 MPH	21	0.941	0.724	0.831
	45 MPH	15	0.918	0.789	0.930
	STOP	25	1	0.958	0.989
YOLOv11	All	144	0.826	0.869	0.899
	25 MPH	20	0.886	0.9	0.897
	35 MPH	21	0.878	0.653	0.929
	45 MPH	15	0.812	0.789	0.915
	STOP	25	1	0.929	0.976

Presented by: Kenneth

Performance Analysis (cont.)

- YOLOv8 is slightly faster at processing frames and slightly more efficient in CPU resource utilization
- YOLOv11 has a higher processing complexity, but is less efficient overall

YOLOv8	Cycles	490,196,308,242	1.729 GHz
	Instructions	413,813,398,747	0.84 insn per cycle
	Cache-Misses	4,802,178,969	5.21% of all cache refs
	Cache-References	92,254,164,579	325.436 M/sec
	Branch-Misses	318,593,074	
	Task-Clock	283,478.49 msec	1.813 CPUs utilized
	Context-Switches	0	0.000 /sec
	Page-Faults	709,206	2.502 K/sec
	Time Elapsed	156.386003770 sec	
	User	275.720335000 sec	
	System	8.089143000 sec	
	FPS = 1/.9571s = 1.045 FPS		

YOLOv11	Cycles	539,804,227,494	1.702 GHz
	Instructions	448,732,835,022	0.83 insn per cycle
	Cache-Misses	4,984,938,173	5.01% of all cache refs
	Cache-References	99,595,990,618	314.072 M/sec
	Branch-Misses	345,711,518	
	Task-Clock	317,112.05 msec	1.881 CPUs utilized
	Context-Switches	0	0.000 /sec
	Page-Faults	1,427,568	4.502 K/sec
	Time Elapsed	168.586001219 sec	
	User	304.288101000 sec	
	System	13.142862000 sec	
	FPS = 1/1.0594s = 0.94429 FPS		

Concluding Remarks

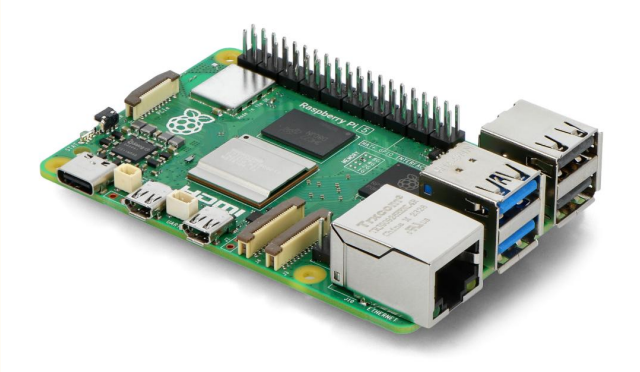
- Performance: YOLOv8 is slightly more efficient in terms of cycles, instructions, and system time.

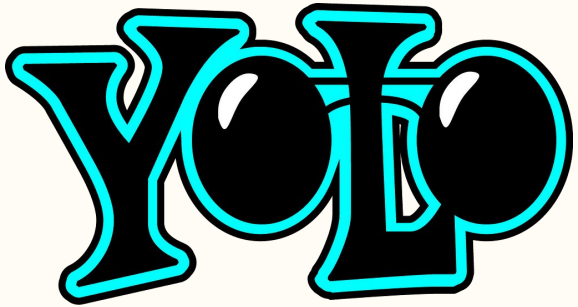
- Resource Utilization: YOLOv11 takes up more CPU resources.
  - It is potentially better suited other more complex tasks but less efficient overall for this task in particular.

- Cache Handling: YOLOv11 has a lower cache miss ratio but produces more page faults.
  - This reflects higher data throughput at the expense of memory efficiency.

Future Work

- The frame rate on the Raspberry Pi 4B is slow
  - 1 FPS
  - Using a Raspberry Pi 5 with the AI acceleration module
- Trying out different models
  - YOLOv5, NCNN framework





References

- Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey," in Proceedings of the IEEE, vol. 111, no. 3, pp. 257-276, March 2023, doi: 10.1109/JPROC.2023.3238524.
- M. H. F. Afonso, E. H. Teixeira, M. R. Cruz., G. P. Aquino and E. C. Vilas Boas, "Vehicle and Plate Detection for Intelligent Transport Systems: Performance Evaluation of Models YOLOv5 and YOLOv8," 2023 IEEE International Conference on Computing (ICOCO), Langkawi, Malaysia, 2023, pp. 328-333, doi: 10.1109/ICOCO59262.2023.10397996.
- A. Swaroop, A. Satsangi, M. Sameer and G. Ahmad, "Performance Evaluation of YOLOv5 and YOLOv8 for Vehicle Detection: A Comparative Study," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-6, doi: 10.1109/ICCCNT61001.2024.10723901.
- E. Casas, L. Ramos, E. Bendek and F. Rivas-Echeverría, "Assessing the Effectiveness of YOLO Architectures for Smoke and Wildfire Detection," in IEEE Access, vol. 11, pp. 96554-96583, 2023, doi: 10.1109/ACCESS.2023.3312217.