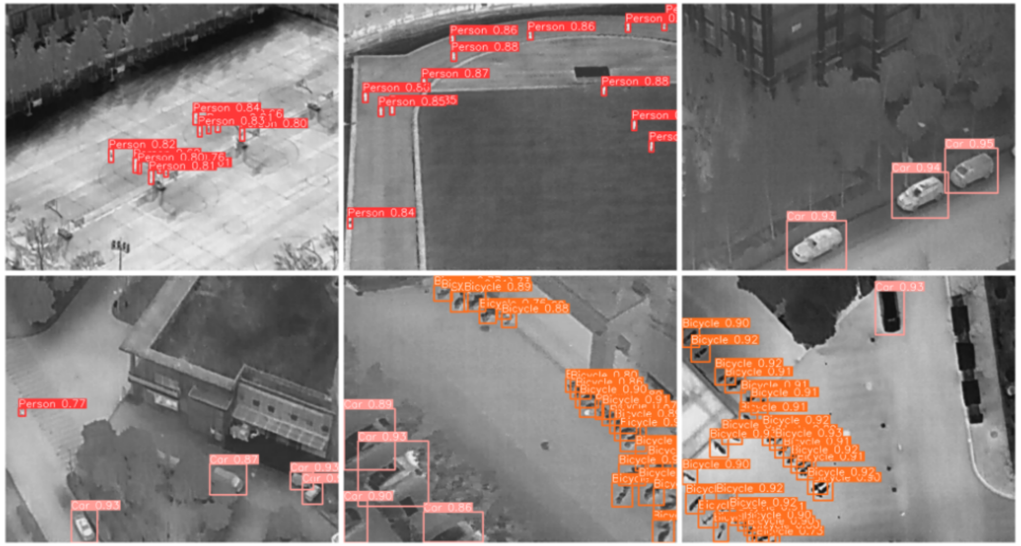


Figure 4: Sample predictions of the YOLO v5 (x).

Edge Intelligence Resource Consumption by UAV-based IR Object Detection

Andrii Polukhin, Yuri Gordienko, Mairo Leier, Gert Jervan, Oleksandr Rokovyi, Oleg Alienin, Sergii Stirenko

Introduction

- AI and ML increasing in value due to applications
- Object detection is a key but challenging task
- CNNs like YOLO show great accuracy and speed
- But deploying CNNs on low-power devices is difficult

Our research aims to:

- Measure YOLO's performance on low-power platforms
- Provide understanding of using YOLO for UAV IR object detection
- Help build CV software for low-power devices

Related Works

Infrared Object Detection from UAVs

- IR images useful for UAV object detection
- But low contrast remains a challenge
- Most methods focus on high-end systems
- Overlooking low-power devices essential for UAVs

Low-Power Devices Computing Resources

- Low-power devices like RPi and OPi used more in IoT, embedded systems, UAVs
- But research on usage and effectiveness lacking
- Consumption and performance crucial to evaluate effectiveness

YOLO for Low-Power Devices

- YOLO great for UAV object detection
- But potential on low-power devices unexplored
- Should explore YOLO for UAV IR object detection using low-power devices

Materials and Methods

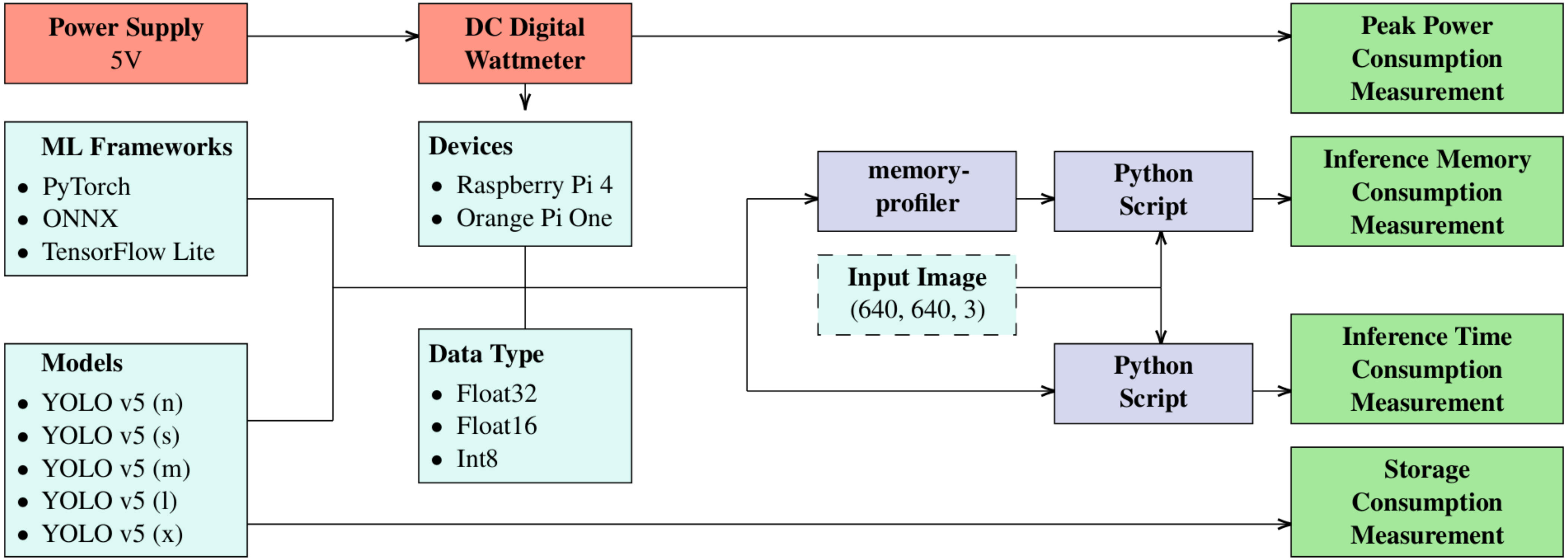
- Measure YOLO v5 on RPi and OPi for:
    - Inference Time (s)
    - Peak Power Consumption (W)
    - Memory Consumption (MB)
    - Inference Energy (J)
    - Storage Consumption (MB)
- Evaluate efficiency and effectiveness
- Train on HIT-UAV dataset
- 2008 train / 287 val / 571 test images

Experimental Design

- Time: 10 runs, account for variations
- Power: Wattmeter on power supply
- Memory: Python memory-profiler
- Energy: Power * Time
- Storage: Weights file size

The approach to understand:

- Performance of YOLO v5 models on RPi and OPi
- Model, device, configuration selection guidelines



Power Supply
5V

DC Digital Wattmeter

Peak Power Consumption Measurement

ML Frameworks
- PyTorch
- ONNX
- TensorFlow Lite

Devices
- Raspberry Pi 4
- Orange Pi One

memory-profiler

Python Script

Inference Memory Consumption Measurement

Input Image
(640, 640, 3)

Models
- YOLO v5 (n)
- YOLO v5 (s)
- YOLO v5 (m)
- YOLO v5 (l)
- YOLO v5 (x)

Data Type
- Float32
- Float16
- Int8

Python Script

Inference Time Consumption Measurement

Storage Consumption Measurement

Figure 3: Resource Consumption Measurement Process Visualization.

Results

- **RPi** clear advantage over **OPi** in inference time and memory
- As model size increases, inference time and power increase
- But YOLO v5 (s) the same power as (n) -> viable for higher accuracy
- Framework affects time and power significantly
  - ONNX most memory efficient
  - TF Lite most energy efficient for smaller models
  - PyTorch consistent balancing of memory and power
- Larger models under TF Lite high demands
- OPi higher variance in inference energy than RPi

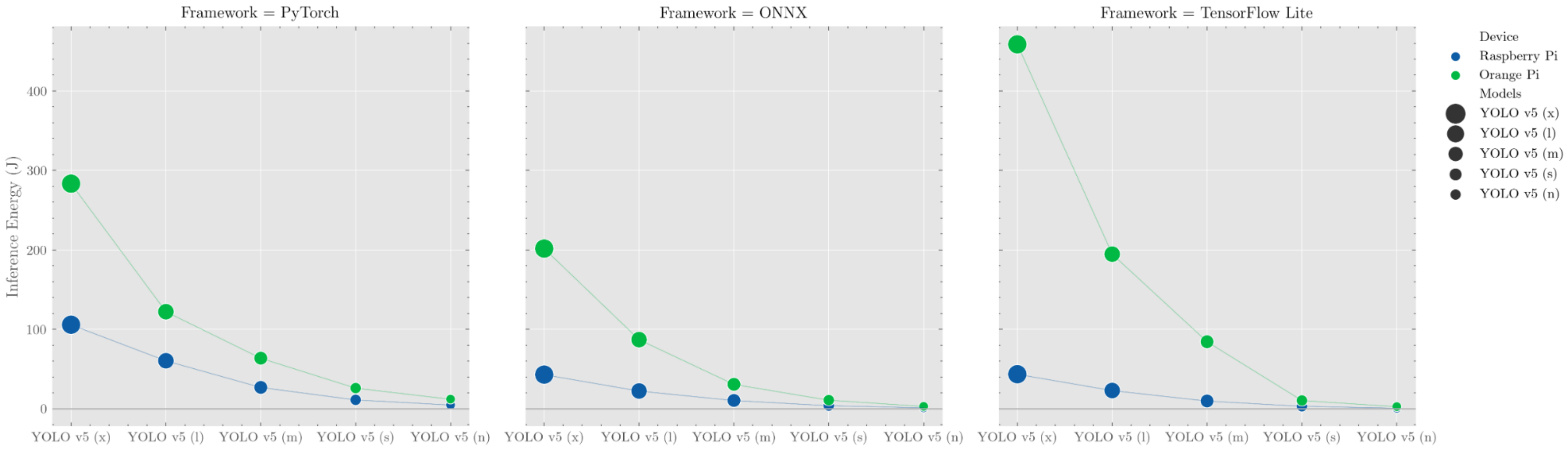


Figure 1: Comparison of Inference Energy (J) vs different model sizes using fp32 data type on Orange Pi and Raspberry Pi.

- Memory consumption optimization crucial
- ONNX and lower precision more efficient
- Performance depends on:
    - Use case
    - Hardware
    - Software optimization
- Future work:
    - Optimize factors above for applications
    - Explore tradeoffs and optimizations

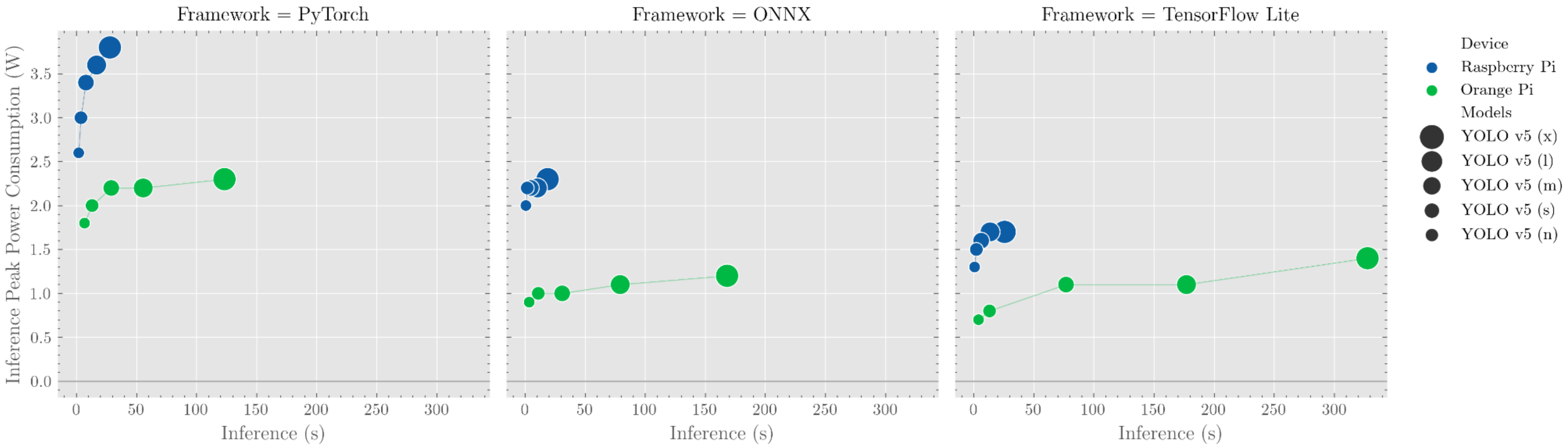


Figure 2: Comparison of Inference Peak Power Consumption (W) vs different Machine Learning Frameworks on Orange Pi and Raspberry Pi.

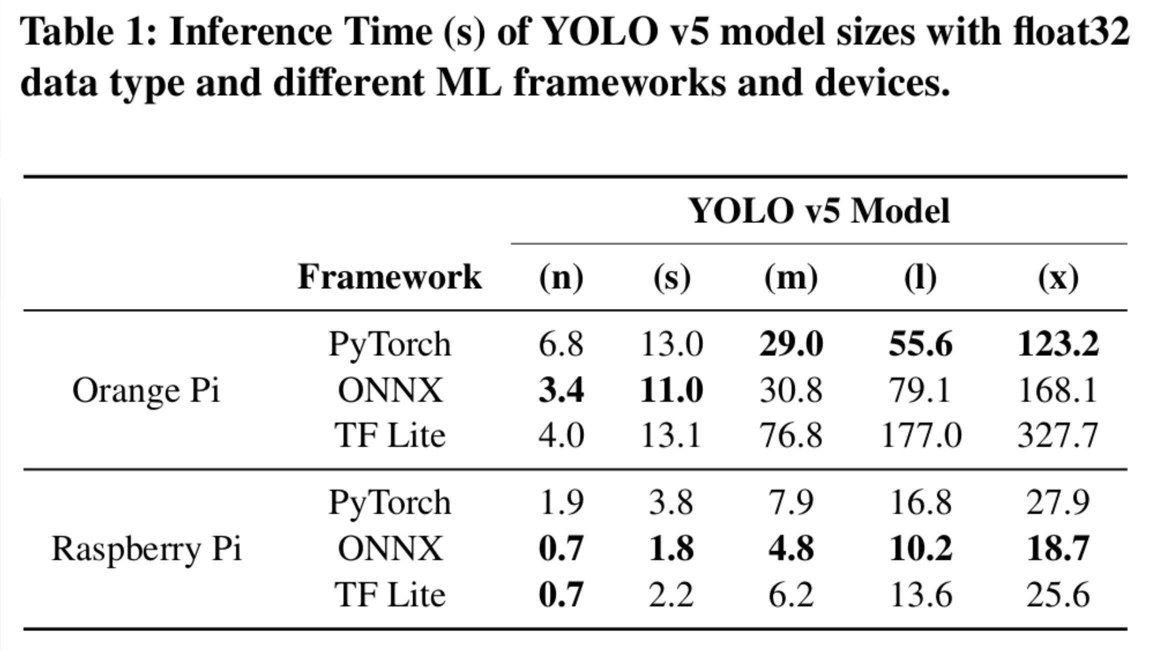


Table 1: Inference Time (s) of YOLO v5 model sizes with float32 data type and different ML frameworks and devices.

		YOLO v5 Model				
	Framework	**(n)**	**(s)**	**(m)**	**(l)**	**(x)**
Orange Pi	PyTorch	6.8	13.0	**29.0**	**55.6**	**123.2**
	ONNX	**3.4**	**11.0**	30.8	79.1	168.1
	TF Lite	4.0	13.1	76.8	177.0	327.7
Raspberry Pi	PyTorch	1.9	3.8	7.9	16.8	27.9
	ONNX	**0.7**	**1.8**	**4.8**	**10.2**	**18.7**
	TF Lite	**0.7**	2.2	6.2	13.6	25.6

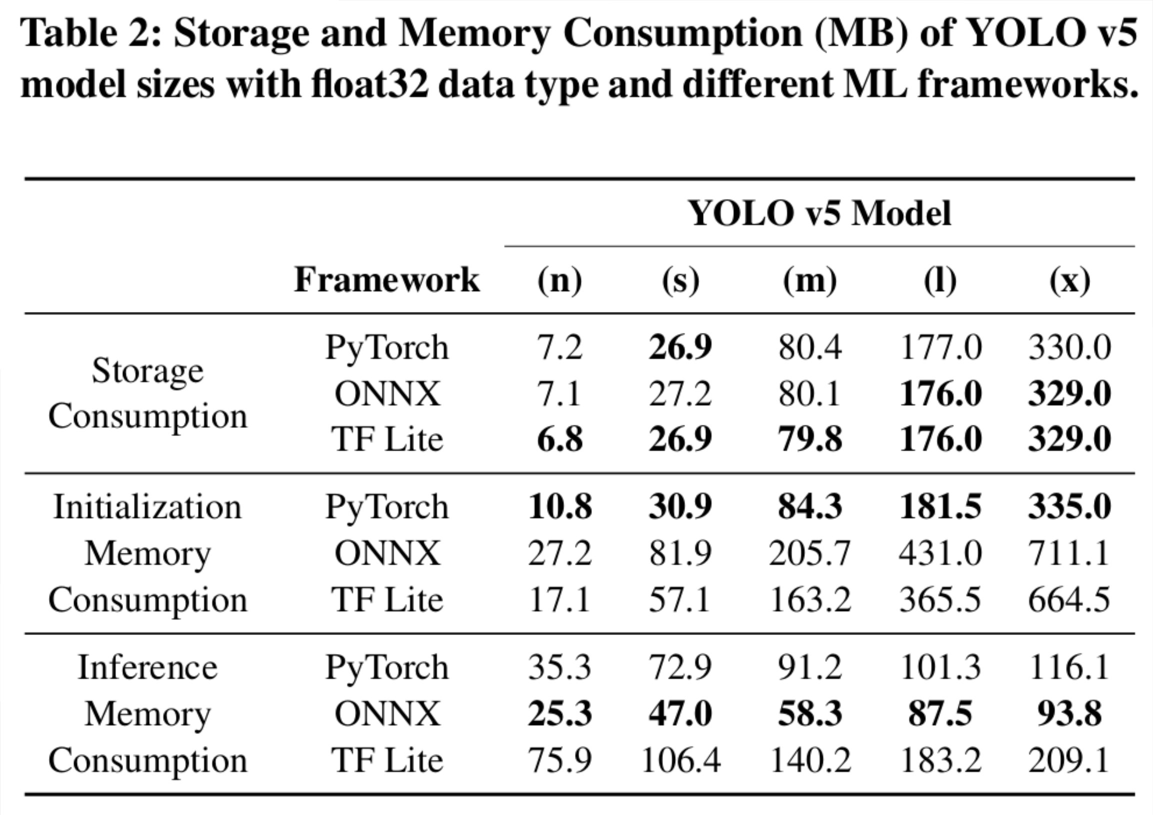


Table 2: Storage and Memory Consumption (MB) of YOLO v5 model sizes with float32 data type and different ML frameworks.

		YOLO v5 Model				
	Framework	**(n)**	**(s)**	**(m)**	**(l)**	**(x)**
Storage Consumption	PyTorch	7.2	**26.9**	80.4	177.0	330.0
	ONNX	7.1	27.2	80.1	**176.0**	**329.0**
	TF Lite	**6.8**	**26.9**	**79.8**	**176.0**	**329.0**
Initialization Memory Consumption	PyTorch	**10.8**	**30.9**	**84.3**	**181.5**	**335.0**
	ONNX	27.2	81.9	205.7	431.0	711.1
	TF Lite	17.1	57.1	163.2	365.5	664.5
Inference Memory Consumption	PyTorch	35.3	72.9	91.2	101.3	116.1
	ONNX	**25.3**	**47.0**	**58.3**	**87.5**	**93.8**
	TF Lite	75.9	106.4	140.2	183.2	209.1

Conclusion

- YOLO feasible for UAV IR object detection on low-power Edge devices
- RPi more energy efficient than OPi
- Framework impacts power and performance significantly
- Smaller models and lower precision more efficient
- Optimization possible through configurations
- Memory and storage management essential

Thank You

This research was in part sponsored by the NATO Science for Peace and Security Programme under grant id. G6032.

Questions