# YOLO11n vs RFDETRNano – Benchmark Comparison Report

**Environment**

GPU: Tesla P100-PCIE-16GB
CPU: 2 cores (4 logical)
Python: 3.12.12
OS: Linux (6.6.113+)

**Model Size**

- YOLO11n: ~5 MB
- RFDETRNano: 300+ MB

The significantly smaller YOLO11n model size contributes to faster training iterations and lower computational overhead. RFDETRNano, being much larger, is heavier but shows faster convergence in fewer epochs.

**Training Time & Convergence**

YOLO11n:
- Runtime: 10h 54s
- Epochs: 150

RFDETRNano:
- Runtime: 9h 28m 46s
- Epochs: 19

Observation: YOLO11n required substantially more epochs (150) toconverge, while RFDETRNano converged in just 19 epochs. However, YOLO11nremains computationally lighter and faster during inference.

**Core Metrics (Focus: mAP@50, mAP@50-95, Precision, Recall)**

YOLO11n:
- mAP@50: 0.7444
- mAP@50-95: 0.4730
- Precision: 0.8047
- Recall: 0.6599

RFDETRNano (EMA metrics):
- mAP@50: 0.7027

- mAP@50-90: 0.4116
- Recall (AR50-90): 0.5034

**Key Takeaways**

1. YOLO11n achieved higher mAP@50 and stronger precision and recall overall.
2. RFDETRNano converged significantly faster (19 epochs vs 150).
3. YOLO11n is far more lightweight (~5MB vs 300+MB), making it more practical for edge deployment.
4. RFDETRNano is heavier and slower but reaches competitive performance with fewer epochs.
5. YOLO11n offers a strong balance between speed, size, and detection performance.

**Conclusion**

If deployment constraints (size, speed, edge devices) matter, YOLO11n isthe clear winner. If faster convergence with fewer epochs is prioritised and model size is not a constraint, RFDETRNano is a compelling alternative.