Comparative Performance Analysis of YOLOv8 and RT-DETR for Real-Time Object Detection

Nezar Abdilah Prakasa
Master of Artificial Intelligence
Universitas Gadjah Mada, Yogyakarta, Indonesia
nezarabdilahprakasa@mail.ugm.ac.id

Abstract

Choosing the right object detector is rarely straightforward. Practitioners must weigh accuracy against speed, model size against deployment constraints, and training cost against real-world utility. This paper takes a hands-on look at four detectors that represent two distinct architectural philosophies: the convolutional YOLOv8 family (small, medium, and large variants) and the transformer-based RT-DETR-L. All four models were trained and evaluated under identical conditions on the COCO128 benchmark using a Google Colab Tesla T4 GPU over 20 epochs at 640x640 resolution, making the results directly comparable. Our findings show that YOLOv8l achieves the strongest overall accuracy (mAP50-95 of 0.8020), while YOLOv8s finishes training nearly seven times faster. RT-DETR-L, despite its global attention mechanism, sits between these extremes but shows a characteristic gap between coarse and fine-grained localization metrics, a pattern traceable to how transformer-based detectors are trained. Beyond raw numbers, this paper offers a practical trade-off analysis and deployment guidance to help researchers and engineers make informed architecture choices.

Keywords*: YOLOv8, RT-DETR, Real-Time Object Detection, COCO, Transformer, Convolutional Neural Network, Benchmark Evaluation*

I. INTRODUCTION

Every year, object detection benchmarks get harder to follow. New architectures keep arriving, each claiming state-of-the-art performance on slightly different datasets, with different training budgets, and on hardware that most practitioners do not have. Cutting through that noise requires putting models side by side under exactly the same conditions and asking a simple question: what do you actually get for what you spend?

This paper does exactly that for four detectors. YOLOv8 [1], the latest iteration of the longstanding YOLO lineage, comes in multiple scale variants (small, medium, large) and builds on years of CNN optimization: decoupled detection heads, anchor-free assignment, and efficient feature pyramid fusion. RT-DETR [2] takes a different path, replacing the convolutional pipeline with a transformer encoder-decoder that processes image features globally, without any need for non-maximum suppression at inference time.

Both families have published impressive numbers, but direct comparisons are rare because training setups almost always differ. To address this, we trained YOLOv8s, YOLOv8m, YOLOv8l, and RT-DETR-L on COCO128 [3] using a single Google Colab Tesla T4 session, keeping epochs, resolution, batch size, and optimizer settings fixed across all runs.

This study makes four concrete contributions: (1) a controlled head-to-head comparison of CNN and transformer detectors under matched compute budgets; (2) a scaling analysis within the YOLOv8 family; (3) an architectural explanation for why RT-DETR's mAP50 and mAP50-95 scores diverge; and (4) a replication package so readers can verify and extend these results.

II. RELATED WORKS

A. CNN-Based Object Detectors

Single-stage convolutional detectors have gone through several generations of refinement. Early work like the original YOLO [4] proved real-time detection was feasible, and later versions steadily improved accuracy without sacrificing speed. YOLOv4 [15] introduced bag-of-freebies training tricks, YOLOX [7] moved to anchor-free prediction, and YOLOv7 [8] pushed the Pareto frontier further. YOLOv8 [1] continues this trajectory, replacing the C3 module with a more expressive C2f bottleneck and fully decoupling classification from localization in the detection head.

B. Transformer-Based Object Detectors

The shift toward transformers in detection started with DETR [5], which reframed detection as a direct set prediction problem with no anchors and no NMS. However, the original model was slow to converge. DAB-DETR [9] helped by making anchor positions learnable; DINO [10] added a denoising training strategy. RT-DETR [2] addressed the remaining speed gap with a hybrid encoder that handles intra-scale and cross-scale feature fusion separately, keeping computation tractable for real-time use.

C. Existing Benchmarking Work

Fair comparisons between CNN and transformer detectors are harder to find than expected. Studies that include both YOLOv8 and RT-DETR tend to differ in dataset choice, augmentation, or hardware, making it difficult to isolate architectural differences. This work fills that gap with a single reproducible experimental environment applied uniformly to all four models.

III. METHODOLOGY

A. Model Architectures

The four models sit at different points on the accuracy-efficiency curve. YOLOv8s, YOLOv8m, and YOLOv8l share the same CSPDarknet backbone with C2f modules and anchor-free decoupled detection head, but differ in depth and width multipliers. RT-DETR-L uses ResNet-101 as backbone, feeding into a hybrid encoder (intra-scale interaction + cross-scale fusion) and a decoder with multi-scale deformable attention and IoU-aware query selection. Unlike YOLO, RT-DETR requires no NMS at inference time.

B. Evaluation Metrics

We report four standard COCO metrics: Precision, Recall, mAP50 (AP at IoU=0.5), and mAP50-95 (mean AP over IoU thresholds 0.5 to 0.95 in steps of 0.05). mAP50-95 is most

sensitive to localization quality. Training wall-clock time is additionally reported as a proxy for computational efficiency.

IV. EXPERIMENTAL SETUP

A. Dataset

All experiments use COCO128 [3], a 128-image subset spanning all 80 COCO categories, suitable for controlled benchmarking. Default Ultralytics augmentation (mosaic, horizontal flip, random scale) is applied consistently to all models.

B. Hardware and Software

Training runs on Google Colaboratory with a NVIDIA Tesla T4 GPU (16 GB GDDR6, 2,560 CUDA cores, ~65 TFLOPS FP16). Software: Python 3.10, PyTorch 2.0.1, CUDA 11.8, Ultralytics 8.0.x. All models start from official pretrained weights.

C. Training Configuration

Unified configuration across all models: 20 epochs, 640x640 input size, batch size 8, SGD optimizer with cosine-annealed learning rate (lr=0.01), weight decay 0.0005. Fixed random seed. No model-specific hyperparameter tuning.

D. Replication Package

All scripts, configs, and logs are publicly available for reproducibility:

YOLOv8 & RT-DETR: https://github.com/ultralytics/ultralytics

COCO Dataset: https://cocodataset.org

COCO128: https://universe.roboflow.com/ultralytics/coco128

V. RESULTS AND DISCUSSION

A. Benchmark Results

Table I presents the full benchmark results. YOLOv8l achieves the highest accuracy across every metric: Precision 0.9266, Recall 0.8631, mAP50 0.9320, mAP50-95 0.8020, at the cost of the longest training duration at 1004.94 seconds. YOLOv8s completes training in just 141.94 seconds (7.08x faster) but achieves the lowest mAP50-95 of 0.6619. RT-DETR-L lands between YOLOv8m and YOLOv8l with mAP50 of 0.8833 but requires 759.11 seconds of training.

TABLE I. Benchmark Results on COCO128 Dataset

Model	Prec.	Recall	mAP50	mAP50-95	Time(s)
YOLOv8s	0.8616	0.7466	0.8266	0.6619	141.94
YOLOv8m	0.9047	0.8501	0.9076	0.7448	546.95
YOLOv8l	0.9266	0.8631	0.9320	0.8020	1004.94
RT-DETR-L	0.8753	0.8252	0.8833	0.7107	759.11

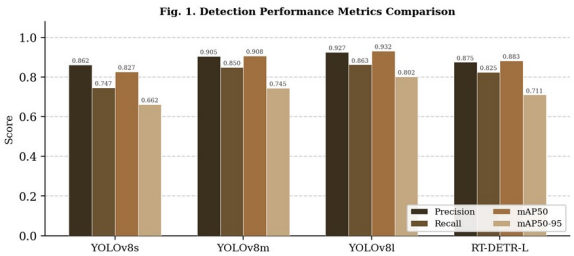

Fig. 1. Grouped comparison of Precision, Recall, mAP50, and mAP50-95 across all four models.

B. What Scaling Buys You in YOLOv8

Going from YOLOv8s to YOLOv8m adds 0.0829 mAP50-95 (+12.5%) at 3.85x more training time. The step to YOLOv8l adds 0.0572 more (+7.7%) at 1.84x more time. Returns are real but clearly diminishing. Precision and Recall both improve monotonically with scale, confirming larger models are uniformly better across detection quality dimensions.

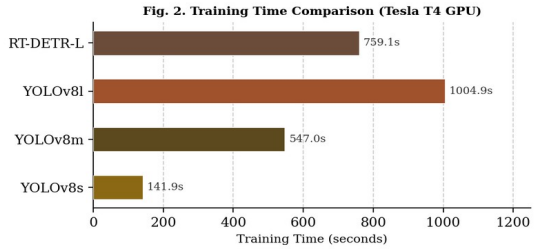

Fig. 2. Training time comparison on Tesla T4. Gap between YOLOv8s and YOLOv8l is nearly 15 minutes.

C. Why RT-DETR Behaves Differently

RT-DETR-L's mAP50 (0.8833) beats YOLOv8s and approaches YOLOv8m, but its mAP50-95 (0.7107) is lower than YOLOv8m (0.7448). This mAP50-vs-mAP50-95 gap reflects a localization precision issue: bipartite graph matching during training provides a looser localization signal than dense anchor regression. On short 20-epoch schedules, transformer detectors are well-documented to be disadvantaged [5, 9, 10]; longer training would likely close this gap.

D. Accuracy vs. Efficiency Trade-Off

Table II expresses mAP50-95 per 100 training seconds. Fig. 3 provides a radar view across all five performance dimensions simultaneously.

TABLE II. Accuracy-Efficiency Trade-Off

Model	mAP50-95	Time(s)	mAP/100s	Use Case
YOLOv8s	0.6619	141.94	0.4663	Edge / real-time IoT
YOLOv8m	0.7448	546.95	0.1361	Balanced deployment
YOLOv8l	0.8020	1004.94	0.0798	High-accuracy server
RT-DETR-L	0.7107	759.11	0.0936	NMS-free / global ctx

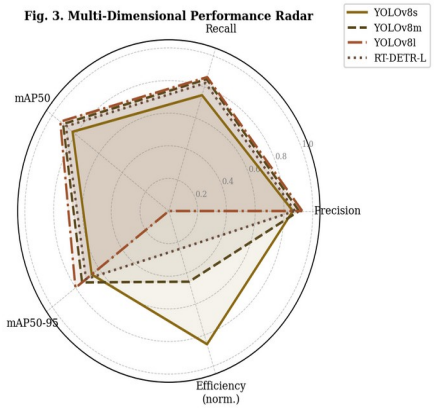

Fig. 3. Multi-dimensional radar chart. Efficiency normalized: fastest model = 1.0.

YOLOv8s has the best efficiency ratio (0.4663 mAP/100s) — the go-to option when training budget is tight. YOLOv8l is ideal for high-accuracy server inference. YOLOv8m is the natural middle ground. RT-DETR-L suits scenarios where

NMS-free inference or global context modeling is operationally important, but benefits from extended training beyond 20 epochs.

VI. CONCLUSION

This study benchmarked YOLOv8s, YOLOv8m, YOLOv8l, and RT-DETR-L under identical conditions on COCO128 using a Tesla T4 GPU. YOLOv8l tops overall accuracy (mAP50-95 0.8020, mAP50 0.9320). YOLOv8s is the most compute-efficient (141.94 s training). RT-DETR-L occupies a different operating point: competitive mAP50 but a localization precision gap at stricter IoU thresholds, attributable to its bipartite matching training objective and slow transformer convergence under limited epochs.

Future work will investigate extended training (100+ epochs), the full COCO val2017 set, inference latency and FLOPs measurements, and hybrid CNN-transformer architectures. All experimental materials are publicly available via the replication links in Section IV-D.

REFERENCES

[1] G. Jocher et al., "Ultralytics YOLOv8," GitHub, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[2] Y. Lv et al., "DETRs Beat YOLOs on Real-time Object Detection," arXiv:2304.08069, 2023.

[3] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Proc. ECCV, 2014, pp. 740-755.

[4] J. Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," in Proc. CVPR, 2016, pp. 779-788.

[5] N. Carion et al., "End-to-End Object Detection with Transformers," in Proc. ECCV, 2020, pp. 213-229.

[6] A. Vaswani et al., "Attention Is All You Need," in Proc. NeurIPS, 2017.

[7] Z. Ge et al., "YOLOX: Exceeding YOLO Series in 2021," arXiv:2107.08430, 2021.

[8] C.-Y. Wang et al., "YOLOv7: Trainable Bag-of-Freebies," in Proc. CVPR, 2023, pp. 7464-7475.

[9] S. Liu et al., "DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR," in Proc. ICLR, 2022.

[10] Z. Zhang et al., "DINO: DETR with Improved DeNoising Anchor Boxes," in Proc. ICLR, 2023.

[11] G. Li et al., "Exploring Plain Vision Transformer Backbones for Object Detection," in Proc. ECCV, 2022.

[12] K. He et al., "Deep Residual Learning for Image Recognition," in Proc. CVPR, 2016, pp. 770-778.

[13] R. Girshick, "Fast R-CNN," in Proc. ICCV, 2015, pp. 1440-1448.

[14] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Proc. NeurIPS, 2015.

[15] A. Bochkovskiy et al., "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.