YOLO vs RF-DETR

Architecture Comparison

Two Paradigms:

- Grid-based (YOLO)
- Set-based (RF-DETR)

The Problem

Object Detection

Goal

> "What & where?"

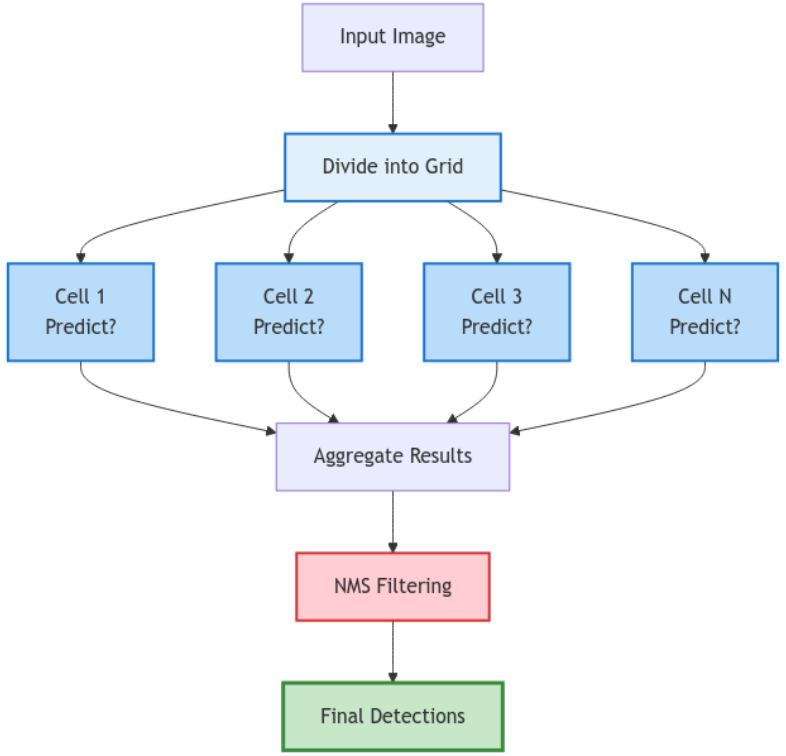
Formulation

$$Y = \{(b_k, c_k)\}_{k=1}^{N}$$

b_k = box, c_k = class, N = count

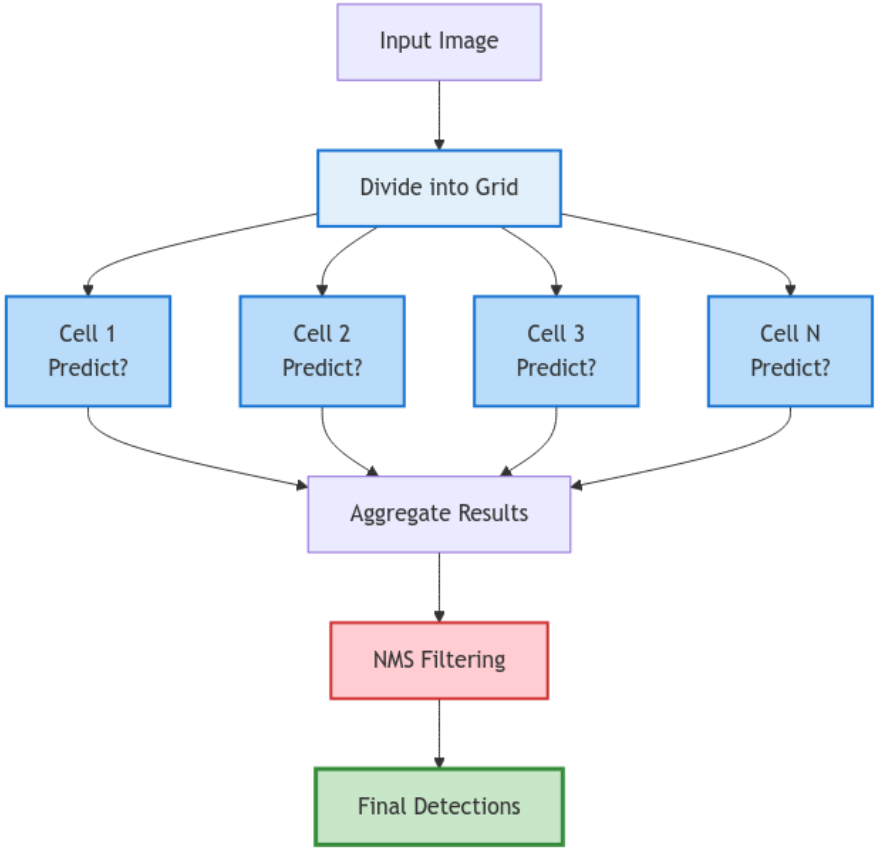
Objective

$$p(Y \mid X; \theta)$$



YOLO Philosophy

Grid-based Predictions



Concept

- Grid cells (p3: 80×80, p4: 40x40, p5: 20x20)
- Independent predictions

Formula

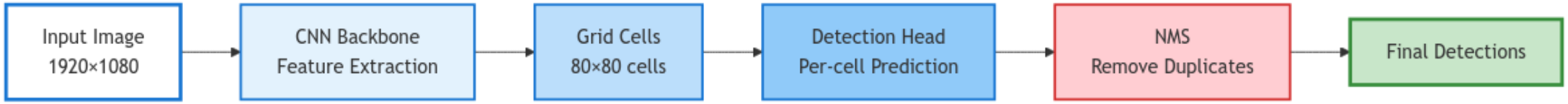
$$p(Y \mid X) = \prod_{i,j} p(y_{ij} \mid F_{ij})$$

Grid factorization, Local F_{ij}

Result

Multiple detections → **NMS needed**

YOLO Architecture Pipeline



Step	Component	Function
1	**Backbone**	CNN features
2	**Grid**	80×80 cells
3	**Head**	Box + Class
4	**NMS**	Remove duplicates

YOLO Characteristics

Strengths ✅

- **Speed**: 30-60 FPS
- **Efficiency**: O(HW)
- **Memory**: Low VRAM
- **Params**: ~43M (YOLOv8l)
- **FLOPs**: ~165G @ 640×1080

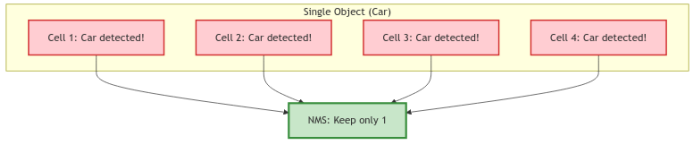
Math

$$\text{Cost} = O(HW)$$

Limitations ❌

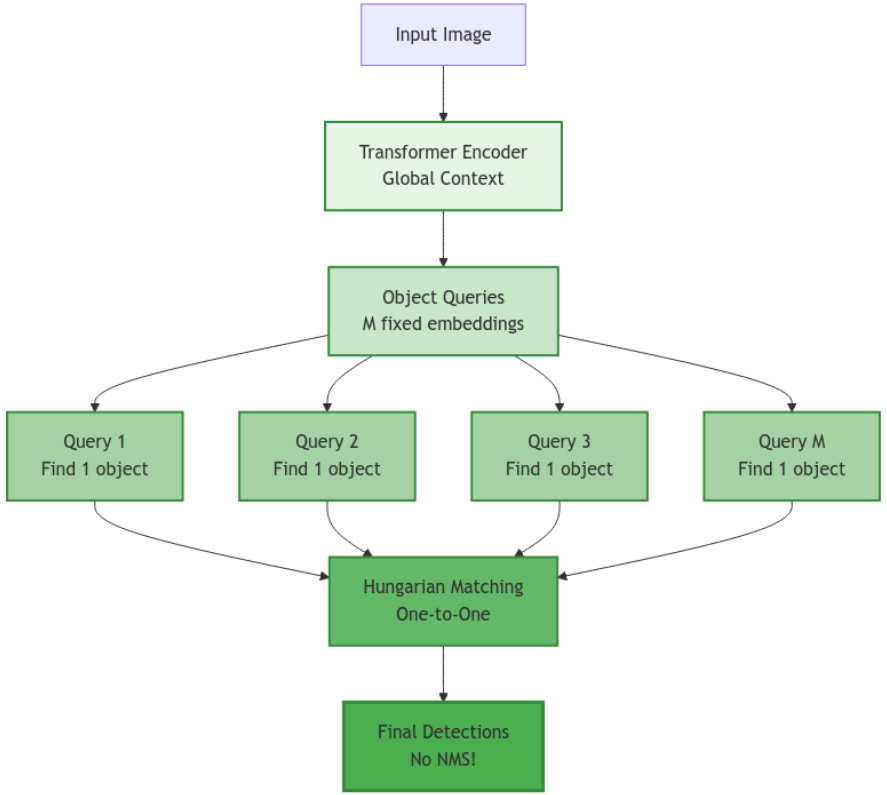
- Multiple predictions
- NMS required
- Limited receptive field
- Dense scenes struggle

Duplication



RF-DETR Philosophy

Set-based Prediction



Concept

- Fixed queries (M=100)
- See entire image

Formula

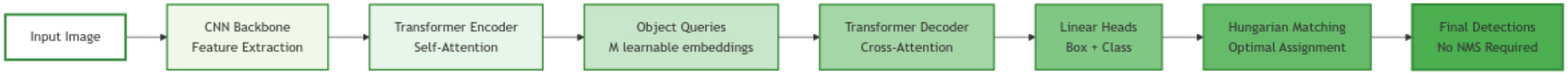
$$p(Y \mid X) = \prod_{k=1}^{M} p(y_k \mid X)$$

Set prediction, Global attention

Result

One-to-one → **No NMS**

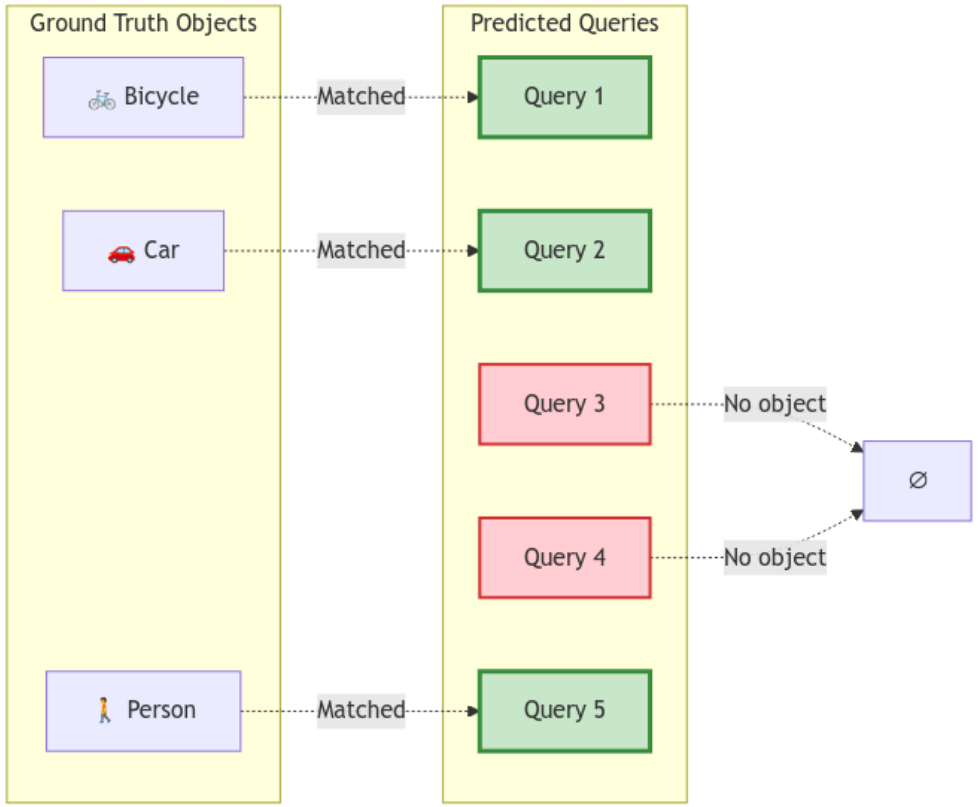
RF-DETR Architecture Pipeline



Step	Component	Function
1	**Backbone**	CNN features
2	**Encoder**	Self-attention
3	**Queries**	M embeddings
4	**Decoder**	Cross-attention
5	**Heads**	Box + Class
6	**Hungarian**	1-to-1 match

Hungarian Matching

One-to-One Assignment



Algorithm

$$\min_{\sigma} \sum_{i} \mathcal{L}(\hat{y}_{\sigma(i)}, y_i)$$

σ = assignment, $\mathcal{L}$ = cost

Properties

✅ No duplicates ✅ No NMS ✅ End-to-end

RF-DETR Characteristics

Strengths ✅

- Global attention
- No NMS
- Dense scenes OK
- Set prediction
- **Params**: ~40M (DETR-R50)
- **FLOPs**: ~86G @ 640×1080

Math

$$\mathcal{R}(y_k) = X$$

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Limitations ❌

- 15-30 FPS
- O((HW)²)
- High VRAM
- GPU required

Cost

$$\text{Cost} = O(N^2 \cdot d)$$

Architecture Comparison

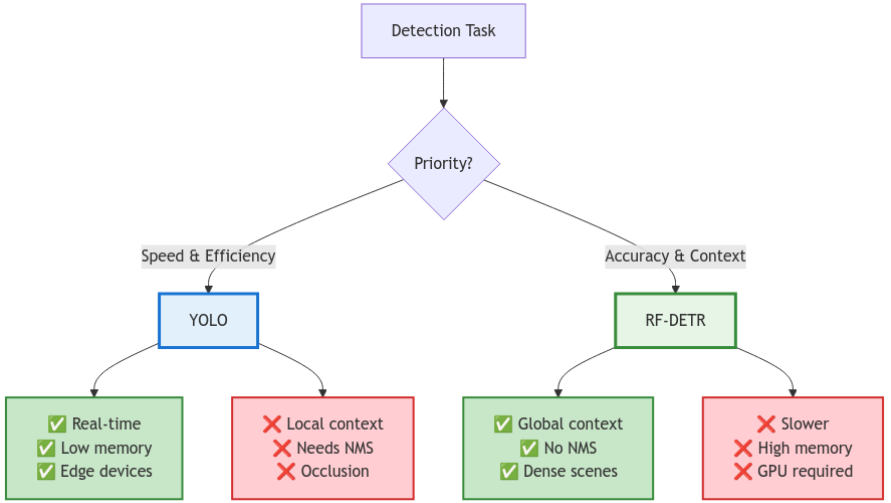
Dimension	YOLO	RF-DETR
Paradigm	Grid-based	Set-based
Context	Local (CNN)	Global (Transformer)
Assignment	Many-to-one	One-to-one
NMS	Required	Not needed
Complexity	O(HW)	$O((HW)^2)$
Speed	30-60 FPS	15-30 FPS
Memory	~0.6 GB	~1.2 GB
Field	Limited	Global

Trade-offs Analysis

Performance

Scenario	YOLO	RF-DETR
Complex scenes	😐	😎 🟢
Speed	😎 🔵	😐
Low hardware	😎 🔵	😐
Accuracy	😐	😎 🟢

Computational Trade-off



Specifications Comparison

Metric	YOLO (YOLOv8l)	RF-DETR (DETR-R50)
Parameters	~43M	~40M
FLOPs @ 640×1080	~165G	~86G
Speed	30-60 FPS	15-30 FPS
Memory	~0.6 GB	~1.2 GB
Complexity	O(HW)	$O((HW)^2)$

Benchmark Results

Model	Size	# Params.	GFLOPS	Latency (ms)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Real-Time Object Detectors w/ NMS										
YOLOv8 (Jocher et al., 2023)	N	3.2M	8.7	2.6	55.0	81.1	59.5	4.8	44.1	48.0
YOLOv11 (Jocher et al., 2024)	N	2.6M	6.5	3.0	55.5	81.3	60.3	4.7	44.4	49.2
YOLOv8 (Jocher et al., 2023)	S	11.2M	28.6	3.1	56.3	82.0	60.9	6.1	45.6	48.6
YOLOv11 (Jocher et al., 2024)	S	9.4M	21.5	3.3	56.4	82.5	61.3	6.5	45.5	48.5
YOLOv8 (Jocher et al., 2023)	M	25.9M	78.9	5.4	56.5	82.3	60.9	6.4	45.7	48.6
YOLOv11 (Jocher et al., 2024)	M	20.1M	68.0	5.1	57.0	82.5	61.9	7.3	46.1	48.6
Open-Vocabulary Object-Detectors (Fully-Supervised Fine-Tuning)										
GroundingDINO (Liu et al., 2023)	T	173.0M	1008.3	309.9*	62.3	88.8	67.8	39.2	57.7	69.5
LLMDet (Fu et al., 2025)	T	173.0M	1008.3	308.4*	62.3	88.3	67.8	39.1	57.6	70.3
End-to-End Real-Time Object Detectors										
LW-DETR (Chen et al., 2024a)	N	12.1M	21.4	1.9	57.1	84.7	61.5	31.2	51.8	65.8
D-FINE (Peng et al., 2024)	N	3.8M	7.3	2.0	58.2	84.4	62.5	32.4	52.9	65.8
RF-DETR (Ours)	N	31.2M	34.5	2.5	57.8	85.1	62.5	30.1	52.2	67.2
RF-DETR w/ Fine-Tuning (Ours)	N	31.2M	34.5	2.5	58.6	85.7	63.0	31.0	53.2	67.6
LW-DETR (Chen et al., 2024a)	S	14.6M	31.8	2.6	57.4	85.0	62.0	32.1	52.1	65.8
D-FINE (Peng et al., 2024)	S	10.2M	25.2	3.5	60.3	85.3	65.4	36.6	56.0	68.4
RF-DETR (Ours)	S	33.5M	62.4	3.7	60.9	87.5	66.1	34.2	55.7	69.6
RF-DETR w/ Fine-Tuning (Ours)	S	33.5M	62.4	3.7	61.2	87.7	66.1	34.9	55.6	69.5
RT-DETR (Zhao et al., 2024)	M	36.0M	100.0	4.3	59.6	85.7	64.6	36.4	54.6	67.3
LW-DETR (Chen et al., 2024a)	M	28.2M	83.9	4.3	59.8	86.8	64.9	34.0	54.4	68.9
D-FINE (Peng et al., 2024)	M	19.2M	56.6	5.6	60.6	85.5	65.8	36.0	56.6	67.5
RF-DETR (Ours)	M	33.5M	86.7	4.6	61.7	88.0	66.9	35.8	56.5	70.0
RF-DETR w/ Fine-Tuning (Ours)	M	33.5M	86.7	4.6	62.0	88.1	67.1	36.2	56.4	70.2
RF-DETR (Ours)	2XL	123.5M	410.2	15.6	63.3	88.9	69.0	38.7	58.2	71.6
RF-DETR (Ours) w/ Fine-Tuning	2XL	123.5M	410.2	15.6	63.5	89.0	69.2	38.9	58.3	71.7

COCO Segmentation Results

- YOLO: Fast, efficient
- RF-DETR: Higher accuracy

Use Cases

YOLO ✅

When:
Real-time, Edge devices, Limited hardware

Apps:
Vehicles, Surveillance, Mobile, Drones

RF-DETR ✅

When:
Accuracy priority, Complex scenes, GPU

Apps:
Medical, Satellite, Crowds, Research

Thank You

YOLO vs RF-DETR: Architecture Comparison

Key Insight:
Both paradigms solve object detection differently - choose based on your constraints and priorities.

Further Reading:

- YOLO: Grid-based detection with local context
- RF-DETR: Set-based detection with global attention
- Trade-off: Speed ↔️ Accuracy