Order Matters: Competition-Guided Query Ordering for RNN-Based Object Detection

Shengjian Wu[1,2] **Li Sun**[1,3*] **Yu Shangguan**[1] **Qingli Li**[1]

Abstract

DETR-style detectors use one-to-one bipartite matching during training to assign object queries to ground-truth objects, enabling end-to-end set prediction without non-maximum suppression (NMS). However, without an explicit de-duplication procedure, multiple queries can still produce highly similar hypotheses for the same object, making training unstable and predictions less decisive. Inspired by the sequential ordering of NMS, we propose DETRNN, a plug-and-play module that turns unordered object queries into a competition-aware sequence for recurrent refinement. DETRNN builds an explicit confidence-and-similarity based order from prior predictions, then refines queries with an RNN along this order to model competition inside the decoder. This ordered recurrent refinement reduces redundant predictions, stabilizes optimization, and improves final detection accuracy. Experiments on multiple DETR-style detectors show consistent gains with comparable efficiency. Code repository: `https://github.com/nigelwoo2008-coder/DETRNN`.

1 Introduction

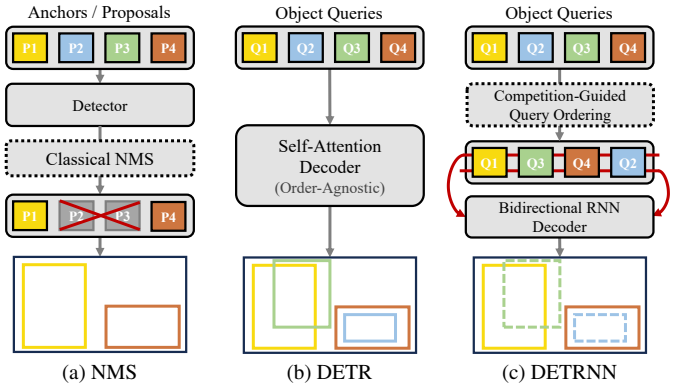
Transformer-based detectors such as DETR [1] unify object detection into an end-to-end set prediction problem. During training, DETR applies one-to-one bipartite matching between object queries and ground-truth objects, which removes post-processing NMS, avoiding manual threshold tuning. Despite this elegant formulation, DETR still suffers from redundant hypotheses, especially in dense scenes (Fig. 1b). Multiple queries can converge to similar regions with close confidence scores, but only one prediction can be matched to each ground-truth object, leaving the others as competing duplicates and slowing convergence.

A useful reference point is classical NMS. NMS resolves redundancy via an explicit sequential procedure: it ranks predictions and greedily removes highly overlapping candidates (Fig. 1a). This sequential order is a strong and simple cue for organizing competition among duplicates. However, using NMS as a hard post-processing step is not always ideal: it is non-learnable, can be threshold-sensitive, and may even hurt in some cases. This motivates a different goal: rather than performing NMS as post-processing, we use an explicit competition order to guide learnable decoder refinement and reduce redundancy.

In DETR, query interaction is mainly handled by self-attention. Recent analyses (e.g., DAC-DETR [2], OneNet [3]) suggest that self-attention, together with one-to-one matching used during training, plays an NMS-like role by suppressing duplicates over decoders. Yet self-attention is parallel and order-agnostic, and it does not provide an explicit competition order. Therefore, prior work typically injects competition priors by modifying attention patterns or biases, such as easing winner-versus-competitor interaction in EASE-DETR [4] or introducing relation priors in Relation-DETR [5], or by applying decoder-wise NMS to discard redundant queries, e.g., DDQ [6]. While effective, these designs remain within parallel attention and rely on carefully crafted attention modifications or query pruning.

*Correspondence to: Li Sun (`sunli@ee.ecnu.edu.cn`). [1] Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University; [2] FinVolution Group; [3] Key Laboratory of Advanced Theory and Application in Statistics and Data Science, East China Normal University.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) NMS (b) DETR (c) DETRNN

Figure 1: **A comparison of de-duplication mechanisms.** Classical NMS de-duplicates by removing highly overlapping proposals with a fixed IoU threshold (a). DETR predicts end-to-end, but its order-agnostic decoder can still yield duplicated results (b). DETRNN internalizes de-duplication into the decoder: CGQO builds a competition-aware query order, and a Bi-RNN refines queries sequentially to softly suppress redundant predictions (dashed boxes) without discarding outputs (c).

We propose a more direct alternative: use an RNN to refine queries along an explicit competition sequence. RNNs are naturally suited for sequential modeling, but DETR queries have an arbitrary order with no competition cue. To bridge this gap, we introduce **DETRNN**, which constructs a competition sequence from prior predictions by sorting queries using confidence and similarity under **Competition-Guided Query Ordering** (CGQO), mimicking the ranking-and-grouping behavior of NMS. A learnable RNN-based query interaction then refines queries along this sequence, making redundant competitors easier to suppress (Fig. 1c). As a result, DETRNN improves training stability and detection quality by suppressing duplicates and making the correct prediction stand out.

To our knowledge, DETRNN is among the first attempts to use an RNN as the query-interaction module in DETR decoders, enabling redundant predictions to be suppressed during decoding. We validate DETRNN on diverse DETR-family baselines, including Def-DETR++ [7], DINO [8], MS-DETR [9], and recent LoRA-DETR [10]. The consistent experimental gains across these architectures show a practical accuracy–efficiency trade-off.

The contributions of this paper are summarized as follows:

- We identify query ordering as a missing inductive bias for de-duplication in end-to-end detection, turning redundancy suppression into an explicit sequential refinement problem.
- We introduce a plug-and-play ordered recurrent query interaction: CGQO builds a competition order from prediction similarity, and an RNN refines object queries along this order to suppress redundant hypotheses.
- Extensive experiments validate CGQO-guided RNN interaction as an effective alternative to decoder self-attention, reducing prediction redundancy and improving AP across diverse detectors with comparable efficiency.

2 Related Work

2.1 End-to-end Object Detectors

DETR [1] introduced a transformer-based end-to-end detector, but its high computational cost, slow convergence, and weak small-object performance limit practical use. Def-DETR [7] alleviates these issues with deformable attention. Other query-based detectors, such as Sparse R-CNN [11] and AdaMixer [12], redesign query decoding with learnable proposals, dynamic convolution, or RoI-based feature extraction. Another line improves convergence and matching stability by enriching query supervision. DN-DETR [13] and DINO [8] introduce denoising queries, while Hybrid-DETR [14], MS-DETR [9], LoRA-DETR [10] and related methods [15, 16, 17, 18, 19, 20] expand query groups or use richer GT–prediction matching to provide more positive supervision. Beyond supervision and matching, stage-wise reuse [21, 22], ranking-aware [23], and dynamic-query methods [24, 25, 26]

improve query propagation, quality, or allocation. These methods improve optimization or query utilization, but they do not explicitly model the order of competition among redundant queries.

OneNet [3] points out that NMS-free detection relies on one-to-one supervision to avoid duplicate predictions. DAC-DETR [2] further shows that decoder self-attention plays a key role in redundancy removal by mediating query–query interaction. However, when decoding hundreds of object queries, parallel self-attention may still struggle to focus on close competitors, often requiring many epochs and multiple decoder stages. Recent methods therefore improve de-duplication in two main ways. Some select or prune queries, e.g., filtering salient queries before decoding [27, 28] or applying NMS-style query pruning across decoder stages [6, 29]. Others reshape parallel query interaction with spatial/IoU or prediction-aware attention priors [30, 12], and recent competition-aware designs such as Relation-DETR [5], EASE-DETR [4], and Dual-R-DETR (formerly Route-DETR) [31]. Rather than further modifying parallel pairwise attention or pruning queries, we study ordered recurrent query interaction, where an RNN exploits an explicit competition order to suppress redundant hypotheses inside the decoder.

2.2 Vision RNNs

Recurrent Neural Networks (RNNs), including LSTM [32] and GRU [33], were originally designed for temporal modeling, and recent variants such as Mamba [34], RWKV [35], TTT [36], and xLSTM [37] further improve efficient long-range sequence modeling. In vision, RNN-style models are often used to process image tokens or enhance feature extractors: Vision Mamba [38] and Vision-RWKV [39] model image tokens sequentially, VMamba [40] explores multidirectional scanning, and Mamba-YOLO [41] integrates SSM blocks into ViTs and FPN. These methods mainly improve visual representations, while DETR's redundancy handling depends on decoder query interaction. A related detector, Recurrent DETR [42], iterates transformer decoder blocks for variable-size prediction, rather than using RNNs for query interaction. In contrast, DETRNN applies recurrent modeling inside the DETR decoder to refine competition-ordered queries and reduce redundancy.

3 Method

3.1 Preliminaries

DETR-style detectors employ a fixed set of learnable object queries $\mathbf{Z} = \{z_1, z_2, \cdots, z_N\} \in \mathbb{R}^{N\times d}$ to produce a set of detection predictions, where each query z_i outputs a classification score and a bounding box. Hungarian bipartite matching enforces a one-to-one assignment between predictions and ground truth objects, enabling end-to-end detection without explicit NMS.

In practice, however, multiple queries may converge to highly overlapping hypotheses around the same object with similar confidence, creating redundant competitors. Since only one prediction can be matched to each ground truth, small score fluctuations may switch the matched prediction during training, making supervision noisy and slowing convergence, especially in crowded scenes.

To mitigate this issue, decoder self-attention is used for query–query interaction. Recent analyses (DAC-DETR [2], OneNet [3]) suggest that self-attention, together with one-to-one matching, implicitly plays an NMS-like role by gradually suppressing duplicates. Nevertheless, self-attention remains parallel and order-agnostic, and does not explicitly organize competitors into a sequential competition procedure.

Recurrent Neural Networks (RNNs) are designed to process sequential data by maintaining hidden states across time steps. Given a sequence $x_1, x_2, ..., x_T$, a standard RNN, denoted as $[o_t, h_t] = \text{RNN}(x_t, h_{t-1})$, computes the hidden state h_t and output o_t at time step t as:

$$h_t = f(\mathbf{W}_h h_{t-1} + \mathbf{W}_x x_t + b) \quad o_t = \mathbf{W}_o h_t \tag{1}$$

where $f(\cdot)$ is a non-linear activation function, e.g., tanh; $\mathbf{W}_x$, $\mathbf{W}_h$ and $\mathbf{W}_o$ are static weight matrices for the input, the hidden state transition and the output, respectively, with b as a bias term.

RNN-style models provide a natural sequential inductive bias by refining representations along an ordered stream. This makes them a suitable candidate for modeling competition in set prediction, provided that the query sequence encodes meaningful competition cues. Recent advances in RNNs integrate transformer-inspired mechanisms that dynamically adjust weight matrices $\mathbf{W}$ based on the current input x_t, enhancing memory and flexibility.

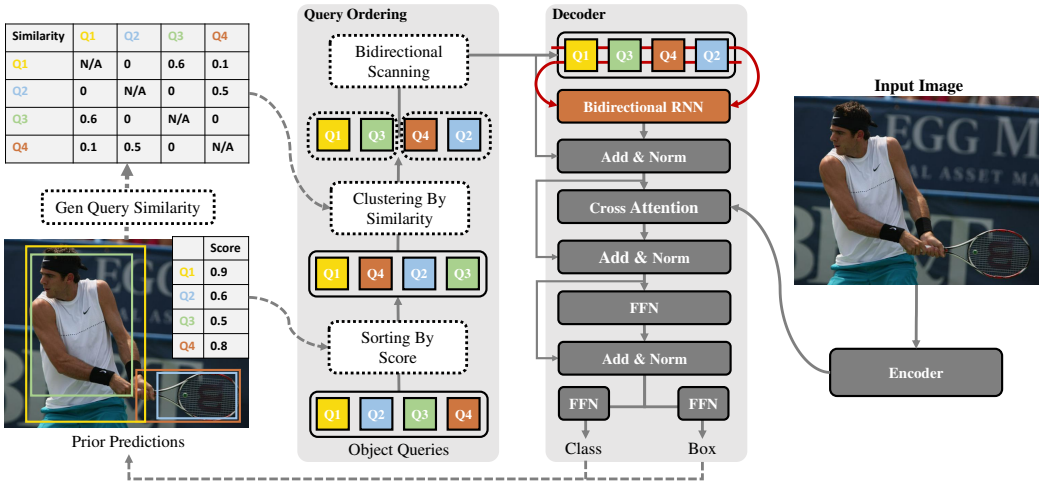


Figure 2: Overview of DETRNN. Given prior predictions from encoder proposals or preceding decoder outputs, Competition-Guided Query Ordering (CGQO) first constructs an explicit competition order among queries. Specifically, queries are ranked by confidence; the top-scoring query is selected as a pivot, and other queries are grouped and reordered by their similarity to the pivot (highly similar competitors are placed nearby). This pivot-and-grouping process is repeated until all queries are ordered, yielding a competition-aware sequence. Bidirectional RNN then replaces decoder self-attention and refines queries sequentially along this order, enabling learnable competition resolution inside the decoder. Unlike NMS, CGQO does not discard queries; it only provides ordering cues that the decoder can internalize for de-duplication.

3.2 Architecture Overview

Building on the observation above, we aim to make query competition resolvable through an explicit order and learnable sequential refinement inside the decoder.

A useful reference point is classical NMS. Beyond being a post-processing rule, its procedure exposes an effective de-duplication ordering cue: it ranks candidates by confidence and organizes close duplicates to follow top-scoring ones. We aim to internalize this ordering cue into the decoder as a learnable refinement process, rather than applying threshold-sensitive post-processing. However, decoder self-attention is parallel and order-agnostic, and does not provide an explicit competition order to exploit.

To this end, we propose DETRNN, a plug-and-play sequential refinement module composed of two steps (Fig. 2). First, Competition-Guided Query Ordering (CGQO) converts the unordered query set into an explicit competition-ordered stream based on prior predictions. Then, an RNN-based interaction refines queries sequentially along this order, suppressing redundant competitors more effectively.

Overall, DETRNN turns de-duplication into an explicit sequential refinement process inside the decoder. We next detail Competition-Guided Query Ordering (CGQO) and the Bi-RNN refinement module.

3.3 Competition-Guided Query Ordering

CGQO is the first step of DETRNN: it converts the unordered query set into an explicit competition-ordered stream using prior predictions (encoder proposals or preceding decoder outputs) (Fig. 2). Unlike post-processing NMS, CGQO keeps *all* queries and only exposes an ordering cue for the decoder to learn de-duplication end-to-end. In practice, we use a pivot-specific threshold to adapt grouping to local crowding; the rule is given in the paragraph *Density-adaptive threshold*.

Sorting by confidence and similarity. CGQO follows a simple pivot-and-group procedure. For each image, we rank queries by confidence and take the top-scoring one as a pivot z_{ref} with box b_{ref} and score vector $s_{\text{ref}} \in \mathbb{R}^C$. For every remaining query z_i (with b_i and s_i), we compute its similarity to the pivot by combining geometric overlap and semantic agreement:

$$\text{Sim}(z_i, z_{\text{ref}}) = (1-\alpha)\,\text{IoU}(b_i, b_{\text{ref}}) + \alpha\,\cos(s_i, s_{\text{ref}}), \tag{2}$$

Algorithm 1 Competition-Guided Query Ordering

Require:
$b \in \mathbb{R}^{B\times N\times 4}$ (bounding boxes), $s \in \mathbb{R}^{B\times N\times (C)}$ (class probabilities),
$c \in \mathbb{R}^{B\times N}$ (confidence scores, $c = \max(s)$), $z \in \mathbb{R}^{B\times N\times d}$ (object queries),
$T \in \mathbb{R}^{B\times N}$ (pivot-wise similarity thresholds), α (IoU/cosine weight)
Definitions:
$all_sim \leftarrow (1-\alpha)\,\mathrm{IoU}(b) + \alpha\,\mathrm{cosine}(s)$ ▷ $all_sim \in \mathbb{R}^{B\times N\times N}$
$reorder_queries \leftarrow \emptyset$
for each batch index $i = 1$ to B **do**
  $order \leftarrow \mathrm{argsort}(c[i])$ descending ▷ rank by confidence
  $clusters \leftarrow \emptyset$
  while $order \neq \emptyset$ **do**
    $top \leftarrow order[0], \quad \tau \leftarrow T[i, top]$ ▷ current pivot and its pivot-specific threshold
    $sim \leftarrow all_sim[i][top,\ order]$
    $inds \leftarrow order[sim \geq \tau]$ ▷ group close competitors and get indices of them
    Append $inds$ to $clusters$
    Remove $inds$ from $order$ ▷ remove grouped queries
  end while
  $ind \leftarrow$ concatenate all $clusters$
  Append $z[i][ind]$ to $reorder_queries$
end for
return $reorder_queries$

where α balances the two terms. Queries with similarity above the pivot threshold are treated as close competitors and placed immediately after the pivot. We then remove the pivot and its grouped competitors and repeat until all queries are ordered (Alg. 1 shown sequentially for clarity; GPU-parallel details in Appendix F.3). This produces a stream where strong candidates appear early and their closest rivals are placed nearby. The gradients are stopped through ordering indices.

Density-adaptive threshold. To avoid using the same grouping threshold across different crowding levels, we assign a **pivot-specific** threshold:

$$T_i = \tau_{\min} + (\tau_{\max} - \tau_{\min}) \cdot \mathrm{norm}\left(\sum_{j\neq i} \mathbf{1}[\rho_{ij} \geq \tau_{\min}]\,\rho_{ij}\right), \tag{3}$$

where ρ_{ij} denotes the similarity in Eq. 2, and $\tau_{\min}$ and $\tau_{\max}$ are the lower/upper bounds. $\mathrm{norm}(\cdot)$ applies a fixed normalization and clamping to map the density score to $[0, 1]$. Intuitively, denser neighborhoods yield larger T_i (finer grouping), while sparser regions yield smaller T_i to avoid over-splitting. Unless specified otherwise, we use fixed hyperparameters for CGQO across all experiments. All hyperparameters and implementation details are provided in Appendix B.

3.4 Sequential Query Refinement with Bi-RNN

Given the competition-ordered query stream, we refine queries using an RNN-based interaction module. Self-attention updates all queries in parallel and does not leverage an explicit order. In contrast, an RNN naturally models an ordered stream, so it can directly use the CGQO order to pass suppression signals from strong candidates to their nearby competitors, reducing redundant hypotheses.

Formally, let $\mathbf{Z} = \{z_1, z_2, \cdots, z_N\} \in \mathbb{R}^{N\times d}$ denote the ordered queries produced by CGQO. An RNN processes the sequence step by step:

$$[o_i, h_i] = \mathrm{RNN}(z_i, h_{i-1}), \tag{4}$$

where h_i is the hidden state and o_i is the refined output at step i. To strengthen information flow, we adopt a bi-directional design by also processing the reversed sequence and combining the two directions. In practice, we use modern RNN-style blocks (e.g., Mamba [34]) for efficient sequence modeling. As shown in Fig. 2, the Bi-RNN replaces the self-attention sub-layer in the decoder, while cross-attention and FFN remain unchanged, making the module plug-and-play for DETR-family detectors. Implementation details of our Bi-RNN instantiation are provided in Appendix C.

Table 1: Results on COCO `val2017` with ResNet-50 and 900 queries. Under unified $1\times$ / $2\times$ schedules, DETRNN consistently improves AP across baselines with comparable FLOPs / FPS.

Method	Backbone	Epochs	Params	FLOPs↓	FPS↑	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Def-DETR++ [7]	R50	12	47.5M	300G	21	47.6	65.6	51.7	30.0	51.0	61.5
		24				49.3	67.9	53.7	32.0	52.6	63.7
Def-DETRNN++ (ours)	R50	12	47.6M	301G	20	**48.7** (+1.1)	67.0	53.2	32.3	52.4	63.2
		24				**50.1** (+0.8)	68.7	54.8	33.7	53.2	64.7
DINO [8]	R50	12	47.5M	300G	21	49.0	66.6	53.5	32.0	52.3	63.0
		24				50.4	68.3	54.8	33.3	53.7	64.8
DINO-RNN (ours)	R50	12	47.6M	301G	20	**50.0** (+1.0)	68.1	54.0	32.9	53.2	64.3
		24				**51.1** (+0.7)	69.0	55.9	34.3	54.2	65.2
MS-DETR [9]	R50	12	47.5M	309G	21	50.0	67.3	54.4	31.6	53.2	64.0
		24				50.9	68.4	56.1	34.7	54.3	65.1
MS-DETRNN (ours)	R50	12	47.6M	310G	21	**50.5** (+0.5)	68.2	55.4	34.2	54.3	64.2
		24				**51.4** (+0.5)	69.2	56.3	33.4	54.9	65.6
LoRA-DETR [10]	R50	12	48.6M	301G	16	52.5	69.8	57.3	37.2	56.5	66.8
LoRA-DETRNN (ours)	R50	12	48.7M	301G	19	**53.0** (+0.5)	70.2	57.6	36.3	56.8	67.5

Table 2: Results with Swin-L backbone.

Method	Backbone	Epochs	Queries	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Def-DETR++ [7]	Swin-L	12	300	54.5	74.0	59.3	37.0	58.6	71.0
Def-DETRNN++ (ours)	Swin-L	12	300	**55.6** (+1.1)	75.1	60.7	38.5	59.8	71.4
LoRA-DETR [10]	Swin-L	12	900	58.2	76.2	63.4	41.8	63.0	74.4
LoRA-DETRNN (ours)	Swin-L	12	900	**58.5** (+0.3)	76.5	64.0	42.5	63.1	74.9

4 Experiments

4.1 Settings

Datasets. We conduct experiments on MS-COCO 2017 [43] and CrowdHuman [44]. For COCO, we train on `train2017` (117K images) and evaluate on `val2017` (5K images) using standard mAP. CrowdHuman is a crowded human detection benchmark with heavy occlusions; we also report results on its validation (4.4K images).

Implementation details. Since implementations of recent detectors can differ substantially, we adapt four widely used baselines for fair comparison: Def-DETR++ [7], MS-DETR [9], DINO [8], and LoRA-DETR [10]. Unless otherwise specified, experiments use the $1\times$ schedule of 12 epochs and follow the training recipe of Def-DETR++. We use a total batch size of 16, learning rates of 2×10^{-5} for the backbone and 2×10^{-4} for the Transformer encoder–decoder, and a weight decay of 2×10^{-5}. For the sequential refinement module, we instantiate the Bi-RNN with Mamba [34] in most experiments. Module configurations, hyperparameter settings, and sensitivity results are provided in Appendix A–E.

4.2 Main Results

We evaluate our plug-and-play module on 4 representative DETR variants. As shown in Tabs. 1 and 2, DETRNN consistently improves AP across all baselines with comparable efficiency. These results show that explicit query-competition modeling is broadly beneficial and transferable, with full-recipe **SOTA** comparison in Appendix G. A detailed analysis of each baseline is provided in the following paragraphs.

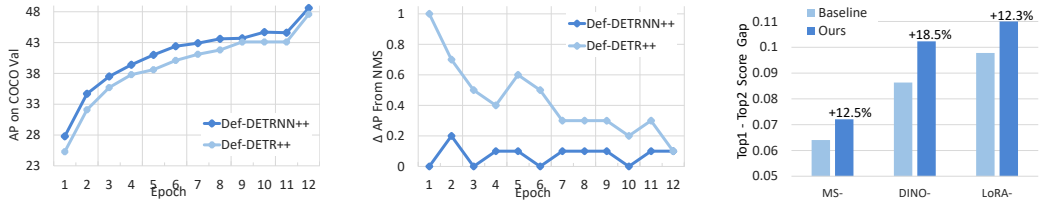
Def-DETR++. Our plug-and-play module consistently improves Def-DETR++ under the same training schedule. With 900 queries and a $1\times$ schedule, it increases AP from 47.6 to 48.7 (+1.1). Under the $2\times$ schedule, it improves the baseline from 49.3 to 50.1 (+0.8), indicating the gain persists when the baseline is further strengthened.

Table 3: **1**-decoder de-duplication stress test (900 queries, 1× schedule). We report AP and +NMS AP (AP after NMS@0.8); ΔAP denotes the NMS gain, where a larger value indicates more residual duplicates. DETRNN improves AP while sharply reducing ΔAP, suggesting stronger decoder-internal de-duplication.

Def-DETR++ (1-decoder)			
	AP	+NMS	ΔAP↓
Base	40.6	43.0	2.4
+Ours	**46.3** (+5.7)	46.6	**0.3** (-2.1)

DINO (1-decoder)			
	AP	+NMS	ΔAP↓
Base	42.2	44.7	2.5
+Ours	**46.8** (+4.6)	47.0	**0.2** (-2.3)

MS-DETR (1-decoder)			
	AP	+NMS	ΔAP↓
Base	40.1	43.1	3.0
+Ours	**43.8** (+3.7)	44.7	**0.9** (-2.1)



(a) (b) (c)

Figure 3: Training dynamics and competition indicators on COCO `val` (**6**-decoder). (a) Validation AP across epochs for Def-DETR++ and ours. (b) AP gain from evaluation-time NMS@0.8; larger values indicate more residual duplicates. (c) Top1–Top2 score gap, where Top1/Top2 are the best and second-best predictions matched to the same GT; a larger gap indicates clearer separation between the correct detection and its closest competitor.

DINO. We observe consistent gains when integrating our module into this baseline. With a 1× schedule, AP increases from 49.0 to 50.0 (+1.0). With a 2× schedule, AP improves from 50.4 to 51.1 (+0.7), showing that our method remains effective across training schedules.

MS-DETR. Despite being a strong baseline, MS-DETR also benefits from our module. At 1×, AP increases from 50.0 to 50.5 (+0.5). At 2×, it improves from 50.9 to 51.4 (+0.5), demonstrating stable gains even with stronger training.

LoRA-DETR. DETRNN is further evaluated on LoRA-DETR [10], a new SOTA DETR baseline. Using its best variant built on Relation-DETR [5], under the 1× schedule, DETRNN improves AP from 52.5 to 53.0 (+0.5). Moreover, by replacing baseline's more complex query interaction with our refinement, we **improve** efficiency with unchanged FLOPs and higher FPS (16→19).

Results with Swin-L backbone. Tab. 2 reports the performance of our variants under the Swin-L [45] backbone. Def-DETRNN++ achieves a clear gain over Def-DETR++ with 300 queries, improving AP from 54.5 to 55.6 (+1.1), with consistent improvements on AP_{50} and AP_{75}. Moreover, LoRA-DETRNN also yields a consistent boost over LoRA-DETR under the 900-query setting, reaching 58.5 AP (+0.3). These results suggest that the proposed refinement generalizes well to stronger backbones and remains compatible with different detector designs.

4.3 De-duplication and Training Dynamics

We verify decoder-internal de-duplication beyond final AP using two complementary checks: a shallow-decoder probe and training-time diagnostics.

Shallow-decoder probe. Tab. 3 reports a **1-decoder** stress test (see Appendix I for settings), where redundancy is strongest because DETR detectors typically rely on multiple decoders to gradually suppress duplicates. We build CGQO from encoder proposals and evaluate the 1st decoder output. DETRNN brings large AP gains (+5.7/+4.6/+3.7) and even outperforms the baseline with NMS. Since NMS mainly removes duplicate predictions, a larger ΔAP indicates more residual redundancy in the outputs. Baselines show large ΔAP (2.4–3.0), while DETRNN reduces it to 0.2–0.9, indicating that far fewer duplicate predictions remain for NMS to remove.

Training dynamics. In the standard **6-decoder** setting, Fig. 3 tracks how de-duplication evolves during training. DETRNN achieves higher AP from early epochs (Fig. 3(a)) and much smaller ΔAP from NMS@0.8, especially early in training (Fig. 3(b)). This indicates stable optimization and earlier suppression of redundant competitors. DETRNN also enlarges the Top1–Top2 score gap (Fig. 3(c)), showing clearer separation between the selected prediction and its closest competitor, which stabilizes one-to-one matching.

Table 4: Ablations on baselines MS- and DINO-. BR denotes replacing self-attention with Bi-RNN, and CGQO denotes Competition-Guided Query Ordering. Using BR or CGQO alone yields negligible gains, while combining them leads to clear improvements, confirming that Bi-RNN can effectively model inter-query dependencies under an explicit competition order.

DINO-

Method	BR	CGQO	AP	AP_{50}	AP_{75}
1			49.0	66.6	53.5
2	✓		49.2 (+0.2)	66.9	53.8
3		✓	49.0	66.6	53.5
4	✓	✓	**50.0** (+1.0)	**68.1**	**54.0**

MS-

Method	BR	CGQO	AP	AP_{50}	AP_{75}
1			50.0	67.3	54.4
2	✓		50.0	67.3	53.3
3		✓	50.0	67.3	54.4
4	✓	✓	**50.5** (+0.5)	**68.2**	**55.4**

Table 5: Ordering variants for Bi-RNN refinement measured by **AP**. Unordered: Method 2 in Tab. 4. Conf-rank: confidence-only ranking. Group-IoU: pivot-and-group with IoU-only similarity. CGQO: Method 4 in Tab. 4. Parentheses report AP changes relative to the Unordered variant.

Detector	Unordered	Random	Conf-rank	Group-IoU	CGQO
DINO-	49.2	48.8 (-0.4)	49.3 (+0.1)	49.8 (+0.6)	**50.0** (+0.8)
MS-	50.0	49.8 (-0.2)	50.0	50.2 (+0.2)	**50.5** (+0.5)

4.4 Ablation Study

Component ablation. Tab. 4 ablates Bi-RNN (BR) and Competition-Guided Query Ordering (CGQO). Bi-RNN alone (Method 2) brings limited benefit because, without an informative order, sequential refinement has little structure to exploit. CGQO alone (Method 3) is also ineffective, since self-attention is permutation-equivariant and processes queries in parallel. In contrast, combining them (Method 4) yields clear gains (e.g., +1.0 on DINO- and +0.5 on MS-), showing that the improvement comes from the interaction between explicit competition ordering and sequential refinement, rather than from either component alone.

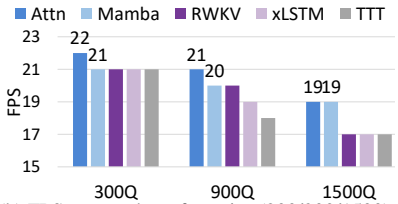
Ordering variants. To isolate the role of ordering itself, Tab. 5 keeps Bi-RNN fixed and varies only the input order. Unordered and CGQO correspond to Methods 2 and 4 in Tab. 4, respectively. Conf-rank keeps only the confidence ranking step in line 3 of Alg. 1, and Group-IoU keeps the pivot-and-group procedure but uses IoU-only similarity in Eq. 2 (α=0). With unsuitable orders, sequential refinement helps little or even hurts: Random ordering degrades AP (-0.4/-0.2), and Conf-rank yields only marginal gains (+0.1/+0.0). A more competition-aware order improves results (Group-IoU), while **CGQO** performs best. This rules out a simple capacity explanation: the same Bi-RNN becomes effective only when close competitors are placed nearby in a meaningful order.

RNN variants and query scaling. Fig. 4 studies whether the gain depends on a specific sequential module. With CGQO fixed, Mamba, RWKV [35], xLSTM [37], and TTT [36] all perform stably (48.4–48.7 AP), consistently outperforming the Attn baseline (47.6 AP). This suggests that the key factor is not a particular RNN design, but the combination of competition-aware ordering and sequential refinement. Fig. 4(b) reports FPS when scaling the number of queries from 300 to 1500.

As N increases, the Mamba variant maintains comparable FPS to Attn in the detector, and Appendix F shows better memory scaling and competitive module runtime under larger query counts.

Detector	Backbone	AP	AP_{50}	AP_{75}
Attn (Base)	R50	47.6	65.6	51.7
Mamba	R50	48.7 (+1.1)	67.0	53.2
RWKV	R50	48.6 (+1.0)	66.8	53.1
xLSTM	R50	48.4 (+0.8)	66.6	52.8
TTT	R50	48.7 (+1.1)	67.3	52.8

(a) AP with different sequential modules



(b) FPS vs. number of queries (300/900/1500)

Figure 4: RNN variants and efficiency scaling on Def-DETRNN++ detector (1× schedule). (a) Different RNN-style modules yield stable gains over baseline. (b) The Mamba variant has controllable FPS overhead, with the gap to Attn narrowing as the number of queries increases.

Table 6: Threshold sensitivity on CrowdHuman dataset. Each entry reports **AP** under a specific setting. NMS is sensitive to the IoU threshold: a lower τ (0.7) can reduce AP, while higher τ brings only marginal gains. In contrast, our method consistently improves accuracy without threshold tuning.

Detector	w/o NMS	+NMS@0.7	+NMS@0.8	+NMS@0.9	+Ours
Def-DETR++	46.5	46.2 (-0.3)	46.7 (+0.2)	46.6 (+0.1)	**47.6** (+1.1)
MS-DETR	48.4	48.3 (-0.1)	48.6 (+0.2)	48.6 (+0.2)	**49.2** (+0.8)

Table 7: COCO consistency check (AP). NMS brings only marginal gains (+0.1–0.2) for the baselines.

Detector	w/o NMS	+NMS@0.7	+NMS@0.8	+NMS@0.9	+Ours
Def-DETR++	47.6	47.6	47.7 (+0.1)	47.7 (+0.1)	**48.7** (+1.1)
MS-DETR	50.0	50.1 (+0.1)	50.2 (+0.2)	50.2 (+0.2)	**50.5** (+0.5)

Table 8: Replacement on competition-aware decoders. DETRNN improves pairwise-interaction methods (EASE-/Relation-) and a query-selection method (DDQ) with higher throughput.

EASE-DETR			
	FLOPs↓	FPS↑	AP
Base	301G	18	49.6
+Ours	301G	**19**	**49.9** (+0.3)

Relation-DETR			
	FLOPs↓	FPS↑	AP
Base	301G	16	52.1
+Ours	300G	**19**	**52.6** (+0.5)

DDQ			
	FLOPs↓	FPS↑	AP
Base	300G	15	52.0
+Ours	300G	**17**	**52.4** (+0.4)

4.5 Crowded-Scene Robustness

Tab. 6 compares our method with applying NMS under the $1\times$ schedule on CrowdHuman, a crowded benchmark with heavy box overlaps, where NMS becomes sensitive to the IoU threshold. With a lower threshold (τ=0.7), NMS is more aggressive and can remove true positives, leading to AP drops of $0.1\sim0.3$ across baselines. With a higher threshold (τ=0.8/0.9), NMS is less destructive but leaves more duplicates, so the gains are limited (typically $+0.1\sim+0.2$). In contrast, our method suppresses redundant competitors inside the decoder and yields consistent improvements of $+0.8\sim+1.1$ AP without NMS or threshold tuning.

Consistency on COCO. Following Tab. 6, Tab. 7 repeats the same NMS-threshold check on COCO. Like CrowdHuman, NMS on COCO provides only marginal gains for the baselines (+0.1–0.2 AP). Our method improves AP by +1.1/+0.5 without relying on post-processing threshold tuning.

4.6 Replacement on Competition-Aware Decoders

Tab. 8 evaluates DETRNN on three representative competition-aware decoders under the $2\times$ schedule: pairwise-interaction methods EASE- [4] and Relation- [5], and the query-selection method DDQ [6]. Because these baselines differ beyond query interaction, we use a replacement setting: *Base* is the original full model, while *+Ours* keeps its training recipe and replaces only the competition-specific decoder operation. DETRNN improves AP on all three baselines (+0.3/+0.5/+0.4) and also increases throughput (18→19, 16→19, and 15→17 FPS) with comparable FLOPs. These results suggest that ordered sequential refinement can serve as an efficient alternative to existing competition-specific designs. More details are provided in Appendix H.

4.7 Stage Placement

Tab. 9 studies where to insert our plug-in module across decoder stages. Def-DETR++ and DINO perform best with the last two stages, whereas MS-DETR and LoRA-DETR favor only the final stage. Restricting the module to these late stages also avoids unnecessary recurrent computation and keeps the additional refinement lightweight, with higher FPS than applying it to all decoder stages. This is consistent with the stage-wise AP/AR100 analysis in Appendix J, where recall largely saturates before the final stage while AP continues to improve. We therefore use last-two-stage insertion for Def-DETRNN++ and DINO-RNN, and last-stage insertion for MS-DETRNN and LoRA-DETRNN. CGQO is recomputed from the current predictions before each inserted stage.

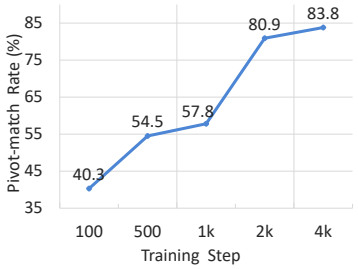
Table 9: Stage placement study across different baselines. "Stage" indicates the decoder stages where our module is inserted.

Def-DETRNN++		
Stage	FPS↑	AP
All	18	48.4
Last 2	20	**48.7**
Last 1	**21**	48.4

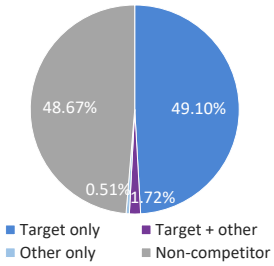
DINO-RNN		
Stage	FPS↑	AP
All	18	49.3
Last 2	20	**50.0**
Last 1	**21**	49.7

MS-DETRNN		
Stage	FPS↑	AP
All	18	49.7
Last 2	20	50.2
Last 1	**21**	**50.5**

LoRA-DETRNN		
Stage	FPS↑	AP
All	16	52.7
Last 2	18	52.9
Last 1	**19**	**53.0**



(a)



(b)

Metric	Value
Pivot-match rate	89.24%
Same-GT coverage	78.85%
Group vs. GT count (r)	0.863

(c)

Figure 5: Mechanistic analysis of CGQO grouping. (a) Pivot-match evolution during the first training epoch. (b) Composition of the Hungarian-matched query's CGQO group. (c) Summary statistics of object-centered grouping.

4.8 Mechanistic Analysis of CGQO Grouping

To examine whether CGQO forms object-centered competition groups, we analyze their evolution, composition, and converged structure on the first 1,600 COCO `val2017` images using Def-DETRNN++ ($1\times$). Fig. 5(a) uses first-epoch checkpoints, while Figs. 5(b,c) use the converged model.

Early grouping evolution. We measure the pivot-match rate, the percentage of GT objects whose Hungarian-matched query is also the pivot of its CGQO group; a higher rate indicates more reliable pivot selection. As shown in Fig. 5(a), it rises from 40.3% at step 100 to 80.9% at 2K and 83.8% at 4K, showing rapid improvement during early training; its converged value is reported in Fig. 5(c).

Group composition. For each GT, we define a *foreground competitor* as a query with box IoU ≥ 0.5 and GT-class score ≥ 0.05, and examine the CGQO group containing its Hungarian-matched query. Among all members of this group, 49.10% are competitors only for the target GT, 1.72% for both the target and other GTs, and only 0.51% exclusively for other GTs; the remaining 48.67% are non-competitor queries, as expected since CGQO retains and groups all queries (Fig. 5(b)). Among foreground competitors, **99.01%** are associated with the target GT, showing very little cross-object mixing.

Object-centered grouping. The Hungarian-matched query is the group pivot in 89.24% of cases, indicating reliable pivot selection. The corresponding CGQO group covers 78.85% of the foreground competitors for the same GT, showing that most same-object competitors are grouped together. Across images, the number of groups containing foreground competitors strongly correlates with the number of GT objects (Pearson $r = 0.863$). These complementary statistics support the object-centered structure of CGQO grouping (Fig. 5(c)).

5 Conclusion

We introduced DETRNN, which turns de-duplication in end-to-end detection into a learnable sequential refinement process. By constructing a competition-aware query order and refining queries along it with an RNN module, DETRNN suppresses redundant predictions while keeping all predictions end-to-end. To our knowledge, this is among the first attempts to treat de-duplication in set-based detection as a sequential refinement problem inside the DETR decoder. Our results across diverse detectors validate a simple message: order matters—a well-structured competition order turns an unordered set into a meaningful sequence, making sequential models a natural fit for resolving duplicates in end-to-end prediction. A brief discussion of limitations is provided in Appendix P.

Acknowledgments and Disclosure of Funding

This work was supported by the Science and Technology Commission of Shanghai Municipality under Grant No.22511105800 and 22DZ2229004, the AI project from the economic and information commission of shanghai (Grant No. 2024-GZL-RGZN-01038), and ECNU Multifunctional Platform for Innovation (001).

The authors declare no competing interests.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[2] Zhengdong Hu, Yifan Sun, Jingdong Wang, and Yi Yang. Dac-detr: Divide the attention layers and conquer. *Advances in Neural Information Processing Systems*, 36:75189–75200, 2023.

[3] Peize Sun, Yi Jiang, Enze Xie, Wenqi Shao, Zehuan Yuan, Changhu Wang, and Ping Luo. What makes for end-to-end object detection? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9934–9944. PMLR, 2021.

[4] Yulu Gao, Yifan Sun, Xudong Ding, Chuyang Zhao, and Si Liu. Ease-detr: Easing the competition among object queries. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17282–17291, 2024.

[5] Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, Badong Chen, and Xuguang Lan. Relation detr: Exploring explicit position relation prior for object detection. In *European conference on computer vision*. Springer, 2024.

[6] Shilong Zhang, Xinjiang Wang, Jiaqi Wang, Jiangmiao Pang, Chengqi Lyu, Wenwei Zhang, Ping Luo, and Kai Chen. Dense distinct query for end-to-end object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7329–7338, June 2023.

[7] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[8] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

[9] Chuyang Zhao, Yifan Sun, Wenhao Wang, Qiang Chen, Errui Ding, Yi Yang, and Jingdong Wang. Ms-detr: Efficient detr training with mixed supervision. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17027–17036, 2024.

[10] Yiwei Zhang, Jin Gao, Hanshi Wang, Fudong Ge, Guan Luo, Weiming Hu, and Zhipeng Zhang. Integrating diverse assignment strategies into detrs. *arXiv preprint arXiv:2601.09247*, 2026.

[11] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, and Ping Luo. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14454–14463, June 2021.

[12] Ziteng Gao, Limin Wang, Bing Han, and Sheng Guo. Adamixer: A fast-converging query-based object detector. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5354–5363, 2022.

[13] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13619–13627, 2022.

[14] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19702–19712, June 2023.

[15] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6748–6758, 2023.

[16] Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

[17] Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. Deim: Detr with improved matching for fast convergence. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[18] Chang-Bin Zhang, Yujie Zhong, and Kai Han. Mr. detr: Instructive multi-route training for detection transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[19] Shengjian Wu, Li Sun, and Qingli Li. Od-detr: Online distillation for stabilizing training of detection transformer. *arXiv*, abs/2406.05791, 2024.

[20] Shilong Liu, Tianhe Ren, Jiayu Chen, Zhaoyang Zeng, Hao Zhang, Feng Li, Hongyang Li, Jun Huang, Hang Su, Jun Zhu, and Lei Zhang. Detection transformer with stable matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6491–6500, October 2023.

[21] Fangyi Chen, Han Zhang, Kai Hu, Yu-Kai Huang, Chenchen Zhu, and Marios Savvides. Enhanced training of query-based object detection via selective query recollection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23756–23765, 2023.

[22] Jing Zhao, Li Sun, and Qingli Li. Recursivedet: end-to-end region-based recursive object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6307–6316, 2023.

[23] Yifan Pu, Weicong Liang, Yiduo Hao, Yuhui Yuan, Yukang Yang, Chao Zhang, Han Hu, and Gao Huang. Rank-detr for high quality object detection. In *NeurIPS*, 2023.

[24] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2968–2977, 2021.

[25] Zhengjian Kang, Jun Zhuang, Kangtong Mo, Qi Chen, Rui Liu, and Ye Zhang. Paq-detr: Learning pattern and quality-aware dynamic queries for object detection. *arXiv preprint arXiv:2603.06917*, 2026.

[26] Yi-Xin Huang, Hou-I Liu, Hong-Han Shuai, and Wen-Huang Cheng. Dq-detr: Detr with dynamic query for tiny object detection. In *European Conference on Computer Vision*, pages 290–305. Springer, 2024.

[27] Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, and Badong Chen. Salience detr: Enhancing detection transformer with hierarchical salience filtering refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17574–17583, 2024.

[28] Guiping Cao, Xiangyuan Lan, Wenjian Huang, Jianguo Zhang, Dongmei Jiang, and Yaowei Wang. Ds-det: Single-query paradigm and attention disentangled learning for flexible object detection. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 2025.

[29] Jeffrey Ouyang-Zhang, Jang Hyun Cho, Xingyi Zhou, and Philipp Krähenbühl. Nms strikes back. *arXiv*, abs/2212.06137, 2022.

[30] Jing Zhao, Shengjian Wu, Li Sun, and Qingli Li. Iou-enhanced attention for end-to-end task specific object detection. In *Asian Conference on Computer Vision*, 2022.

[31] Ye Zhang, Qi Chen, Wenyou Huang, Rui Liu, and Zhengjian Kang. Dual-r-detr: Resolving query competition with pairwise routing in transformer decoders. *arXiv preprint arXiv:2512.13876*, 2025.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[34] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[35] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore, December 2023. Association for Computational Linguistics.

[36] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (Learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 57503–57522. PMLR, 13–19 Jul 2025.

[37] Maximilian Beck, Korbinian Poppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv*, abs/2405.04517, 2024.

[38] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv*, abs/2401.09417, 2024.

[39] Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv*, abs/2403.02308, 2024.

[40] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024.

[41] Zeyu Wang, Chen Li, Huiying Xu, Xinzhong Zhu, and Hongbo Li. Mamba YOLO: A simple baseline for object detection with state space model. In *AAAI*, pages 8205–8213. AAAI Press, 2025.

[42] Hyeong Kyu Choi, Cho Paik, Hyun Woo Ko, Min-Chul Park, and Hyunwoo J. Kim. Recurrent detr: Transformer-based object detection for crowded scenes. *IEEE Access*, 11:78623–78643, 2023.

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[44] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv*, abs/1805.00123, 2018.

[45] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

Technical Appendices and Supplementary Material

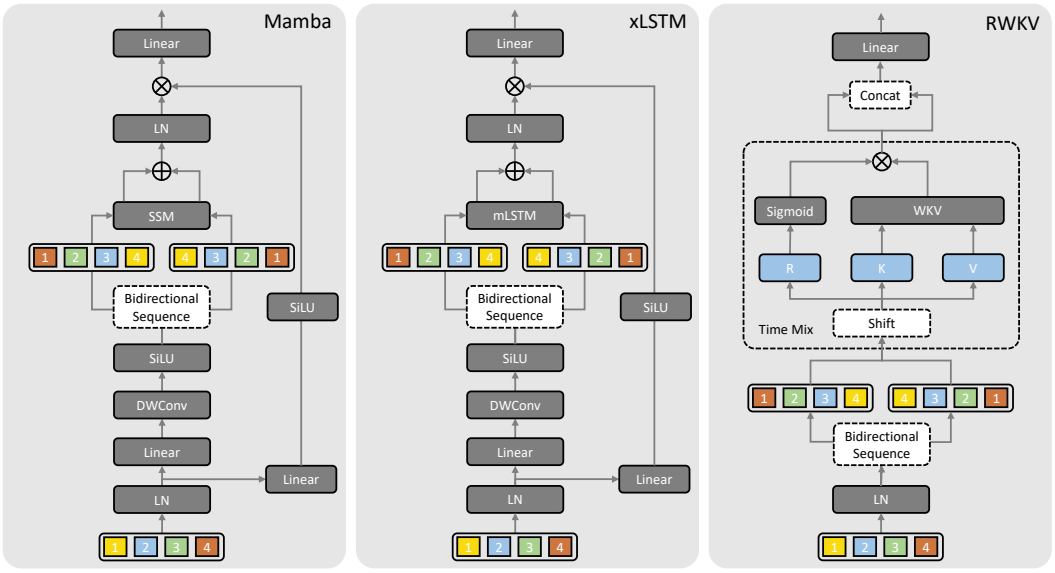


Figure 6: Structures of the three representative RNN variants used in our Bi-RNN module. TTT is also evaluated but omitted from the architecture illustration for space.

A Various RNN architectures

Figure 6 illustrates the three RNN architectures used in our method, each differing in their query interaction mechanisms but all incorporating bidirectional processing to enhance query interactions. Mamba [34] maintains its original structure with bidirectional query processing, utilizing components like SiLU activation, depthwise convolution, and layer normalization. xLSTM [37] extends LSTM with exponential gating and matrix-memory mechanisms such as mLSTM. Similarly, RWKV [35] incorporates bidirectional processing with weighted key-value mixing and temporal shifts. By integrating these bidirectional mechanisms, xLSTM and RWKV further enhance the model's ability to capture relationships within the object query sequence.

B CGQO Hyperparameters

Default settings. We use the same CGQO hyperparameters for all detectors unless explicitly stated. Specifically, we set α=0.3 in Eq. 2, and use $\tau_{\min}$=0.4, $\tau_{\max}$=0.8, and K=18 in Eq. 3. Fig. 7 shows that varying α or K around these defaults only leads to minor changes, and Fig. 8(a,b) shows similar robustness to $\tau_{\min}/\tau_{\max}$.

Normalization in Eq. 3. We implement $\mathrm{norm}(\cdot)$ as a fixed normalization with saturation:

$$\mathrm{norm}(x) = \mathrm{clip}\left(\frac{x}{K}\right), \quad \mathrm{clip}(x) = \min(\max(x, 0), 1).$$

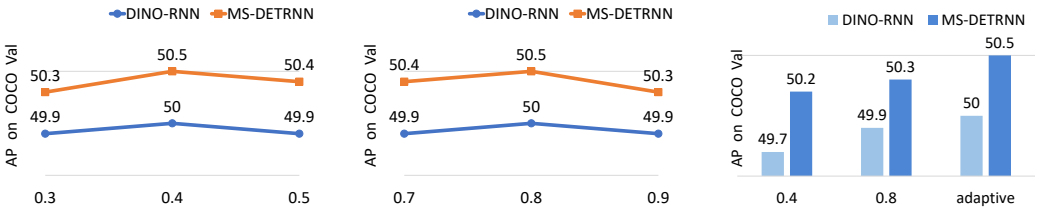
Here K is a fixed saturation constant for the density score, preventing extremely dense neighborhoods from dominating the threshold. As shown in Fig. 7(b), the result is stable when varying K around this default.

Robustness and Benefits of Adaptive Thresholding. Fig. 8 evaluates the threshold design in CGQO on DINO-RNN and MS-DETRNN. Sweeping the lower/upper bounds in Eq. 3 by fixing one of $\tau_{\min}/\tau_{\max}$ and varying the other changes AP only slightly (within 0.1–0.2; Fig. 8a–b), indicating low sensitivity to these hyper-parameters. Moreover, the density-adaptive rule consistently outperforms both fixed thresholds (0.4 and 0.8; Fig. 8c), suggesting a more reliable grouping criterion than a globally fixed τ.



(a) α sweep in Eq. 2. (b) K sweep in Eq. 3.

Figure 7: Sensitivity of CGQO hyperparameters on COCO val (DINO-RNN). (a) Performance varies little across $\alpha \in [0, 0.4]$. (b) Varying the saturation constant K around the default K=18 also leads to minor changes.



(a) $\tau_{\min}$ sweep ($\tau_{\max}$=0.8) (b) $\tau_{\max}$ sweep ($\tau_{\min}$=0.4) (c) Fixed vs. adaptive

Figure 8: Robustness of CGQO thresholding on COCO `val`. (a,b) Sweeping $\tau_{\min}/\tau_{\max}$ (Eq. 3) changes AP only slightly. (c) The density-adaptive rule performs best compared to fixed thresholds.

C Details of Bi-RNN (Mamba) module

Formulation of SSMs. Originating from the Kalman filter, SSMs are linear time-invariant (LTI) systems that map an input $u(t) \in \mathbb{R}$ to an output $y(t) \in \mathbb{R}$ through a hidden state $\mathbf{h}(t) \in \mathbb{R}^N$. Specifically, continuous-time SSMs can be expressed by the following linear ordinary differential equations (ODEs):

$$\begin{aligned} \mathbf{h}'(t) &= A\,\mathbf{h}(t) + B\,u(t), \\ y(t) &= C\,\mathbf{h}(t), \end{aligned} \tag{5}$$

where $A \in \mathbb{R}^{N\times N}$, $B \in \mathbb{R}^{N\times 1}$ and $C \in \mathbb{R}^{1\times N}$ are the weighting parameters.

Discretization. Mamba discretizes the continuous parameters (A, B) into $(\bar{A}, \bar{B})$ using fixed formulas, where $(\bar{A}, \bar{B})$ is called a discretized model. Various rules can be applied, such as the zero-order hold (ZOH) in equation (6), leveraging continuous-time system properties like resolution invariance and ensuring proper normalization.

$$\begin{aligned} \bar{A} &= \exp(\Delta A), \\ \bar{B} &= (\Delta A)^{-1}\big(\exp(\Delta A) - I\big)(\Delta B). \end{aligned} \tag{6}$$

In the original Mamba [34] SSM module, the parameters B, C, and Δ are derived from linear transformations of the same input token (i.e., the object query). In our implementation, we keep the object query as the input for generating B and C, but **replace the input to the Δ branch** with an IOF embedding, so that the discretization step is explicitly conditioned on spatial relations.

IOF Embedding. After NMS-like query sorting, given two adjacent predicted boxes b_i and b_{i+1}, we compute their Intersection over Foreground (IOF) as in (7), and encode it using the sine-cosine scheme in (8), where d_p is the embedding dimension and k is 10000. We follow DETR-style box encodings and first map the bounded IOF value to the angular range by 2π before applying the sine-cosine embedding. The resulting embedding is linearly transformed to produce Δ.

$$\mathrm{IOF}(b_i, b_{i+1}) = \frac{\mathrm{area}(b_i \cap b_{i+1})}{\mathrm{area}(b_{i+1})}. \tag{7}$$

$$\begin{aligned} \mathrm{Embed}(\mathrm{IOF})_{2j} &= \sin\left(\frac{2\pi \cdot \mathrm{IOF}}{k^{2j/d_p}}\right), \\ \mathrm{Embed}(\mathrm{IOF})_{2j+1} &= \cos\left(\frac{2\pi \cdot \mathrm{IOF}}{k^{2j/d_p}}\right), \end{aligned} \tag{8}$$

Table 10: Effect of IOF positional encoding in the default Mamba instantiation on COCO val (R50, 1×).

Detector	Variant	AP	AP_{50}	AP_{75}
Def-DETR++	Attn baseline	47.6	65.6	51.7
	w/o IOF-PE	48.6 (+1.0)	66.9	53.1
	Full	**48.7** (+1.1)	**67.0**	**53.2**
MS-DETR	Attn baseline	50.0	67.3	54.4
	w/o IOF-PE	50.3 (+0.3)	67.7	55.0
	Full	**50.5** (+0.5)	**68.2**	**55.4**

Table 11: Component analysis of DETRNN on Def-DETRNN++/R50/1×.

Variant	CGQO + Bi-RNN	Selected stage	IOF-PE	AP
Baseline (self-attn)	×	×	×	47.6
Ordered recurrent refinement	✓	×	×	48.2 (+0.6)
+ IOF-PE	✓	×	✓	48.4 (+0.8)
+ Selected stage	✓	✓	×	48.6 (+1.0)
Full DETRNN	✓	✓	✓	**48.7 (+1.1)**

IOF is asymmetric and therefore provides a directional overlap cue aligned with the competition order: it measures how much a nearby competitor is covered by a stronger candidate. This directional cue is more suitable than symmetric IoU for order-based refinement. For bidirectional refinement, we compute the cue consistently with the processing direction: the forward pass uses $\mathrm{IOF}(b_i, b_{i+1})$, while the reverse pass swaps the arguments and uses $\mathrm{IOF}(b_{i+1}, b_i)$. For boundary tokens without an adjacent neighbor, we set the missing IOF value to 0 before embedding. By injecting IOF-based embeddings into Δ, we condition the SSM discretization on adjacent spatial relations, providing an explicit order-aware spatial prior for query refinement.

Effect of IOF Embedding. IOF-PE is used as a lightweight spatial conditioning in the default Mamba instantiation, while CGQO ordering and Bi-RNN refinement remain the core components. To isolate its effect, Tab. 10 removes IOF-PE while keeping CGQO and Bi-RNN unchanged. Without IOF-PE, DETRNN still improves the attention baselines by +1.0 AP on Def-DETR++ and +0.3 AP on MS-DETR. Adding IOF-PE further brings +0.1/+0.2 AP. This shows that IOF-PE is a useful auxiliary spatial cue, but the main improvement does not rely on this conditioning.

D Further Component Ablation

We further isolate the contributions of ordered recurrent refinement, IOF-PE, and stage placement on Def-DETRNN++/R50/1× (Tab. 11). Starting from the 47.6 AP self-attention baseline, CGQO-guided Bi-RNN reaches 48.2 AP (+0.6). Adding IOF-PE increases the result to 48.4 AP, while using the selected late-stage placement without IOF-PE reaches 48.6 AP. Combining both gives the full DETRNN result of 48.7 AP (+1.1). Thus, the main gain already comes from ordered recurrent refinement, while IOF-PE and stage placement provide complementary improvements.

We do not treat CGQO alone as a separate interaction module because it only constructs the query order. As shown in Tab. 4, applying CGQO without recurrent interaction brings no clear gain, whereas Tab. 5 shows that, under the same recurrent module, the CGQO order consistently outperforms alternative orders. This further supports the importance of explicit competition-aware ordering.

E Reproducibility of Training and Efficiency Metrics

Training resources. All models are trained with the same settings as their corresponding baselines. Unless otherwise specified, we use 8 NVIDIA RTX 4090 or 8 NVIDIA RTX 3090 GPUs with a total batch size of 16. The standard 1× and 2× schedules correspond to 12 and 24 epochs, respectively.

Efficiency metrics. All FPS measurements are conducted on a single NVIDIA RTX 4090 GPU. We compute FLOPs with a fixed input of size $1333 \times 800 \times 3$ (one image), using the `profile` function from the open-source THOP library. For FPS, we measure throughput with batch size 16 on the first

Table 12: Profiling of query-interaction modules under different query counts N. Runtime is reported in ms/image and rounded to 0.1 ms; extra peak memory is reported in MB. CGQO+Mamba adds small runtime overhead but uses much less memory as N grows, and the overhead is minor relative to the full detector cost ($\sim$50 ms, $\sim$700 MB at 1024$\times$1333).

Module	N=300		N=900		N=1500	
	Time (ms)↓	Mem (MB)↓	Time (ms)↓	Mem (MB)↓	Time (ms)↓	Mem (MB)↓
Self-attn	0.2	6.7	0.2	53.0	0.4	143.2
Mamba	0.4	3.7	0.4	11.3	0.4	19.0
CGQO	0.3	0.5	0.5	3.8	0.7	9.7
CGQO+Mamba	0.7 (+0.5)	4.2 (-2.5)	0.9 (+0.7)	15.1 (-37.9)	1.1 (+0.7)	28.7 (-114.5)

Table 13: Module-level profiling under larger query counts N. Runtime is reported in ms/image and rounded to 0.1 ms; extra peak memory is reported in MB. CGQO+Mamba greatly reduces memory compared with self-attention and has lower isolated module runtime from N=4000 onward.

Module	N=2000		N=4000		N=5000	
	Time (ms)↓	Mem (MB)↓	Time (ms)↓	Mem (MB)↓	Time (ms)↓	Mem (MB)↓
Self-attn	0.8	252.0	3.2	992.2	5.1	1545.4
Mamba	0.4	25.0	0.4	49.9	0.4	62.7
CGQO	0.9	16.9	1.9	65.6	2.3	101.4
CGQO+Mamba	1.3 (+0.5)	41.9 (-210.1)	2.3 (-0.9)	115.5 (-876.7)	2.7 (-2.4)	164.1 (-1381.3)

160 images of COCO `val` and report the average runtime. For module profiling in Tabs. 12 and 13, we use the same RTX 4090 GPU, run warm-up iterations, measure CUDA time with synchronized CUDA events, and report peak allocated memory for the query-interaction module only.

F Profiling and Scaling

F.1 Profiling Results

Tabs. 12 and 13 profile the isolated runtime and extra peak memory of the main query-interaction modules as the number of queries N increases. We measure runtime with synchronized CUDA events after warm-up and round it to 0.1 ms; since individual module runtimes are small, we focus on the total CGQO+Mamba cost and the peak-memory trend. The large-N settings are used as module-level scaling stress tests, while all AP experiments use the standard query counts reported in the main tables. Self-attention shows clear memory growth from its explicit $N\times N$ attention map, reaching 992.2 MB at N=4000 and 1545.4 MB at N=5000. In contrast, CGQO+Mamba requires only 115.5 MB and 164.1 MB at the same query counts. Its runtime is slightly slower at moderate N, but becomes lower in this isolated module profile from N=4000 onward, reducing time from 3.2 to 2.3 ms at N=4000 and from 5.1 to 2.7 ms at N=5000. Since DETRNN is applied only to the last 1–2 decoder stages (Sec. 4.7), the end-to-end overhead remains small for common query counts.

F.2 CGQO Runtime Breakdown

We profile CGQO at 900 queries using the full CUDA implementation and a similarity-only variant with identical inputs and hyperparameters. After 128 warm-up iterations, latency is averaged over 8,000 runs using synchronized CUDA events. Full CGQO takes 0.48 ms, while pairwise similarity alone takes only 0.09 ms (about 19%); the remaining $\sim$0.39 ms (about 81%) comes from grouping and order construction.

Although pairwise similarity has $O(N^2)$ complexity, it is implemented as dense GPU operations over compact box and class-score tensors. Most of the runtime instead comes from the subsequent grouping stage, whose dependent control flow and irregular index updates are less GPU-friendly.

Table 14: Full-recipe comparison with recent DETR-style SOTA on COCO val under comparable ResNet-50 / Swin-L, 1×, and 900-query settings when available. The DETRNN entries use our strongest LoRA-DETRNN full recipes. This is a leaderboard-style context rather than a controlled module comparison, since methods use different recipes.

Method	Backbone	Epochs	Queries	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Def-DETR++ [7]	R50	12	900	47.6	65.6	51.7	30.0	51.0	61.5
DINO [8]	R50	12	900	49.0	66.6	53.5	32.0	52.3	63.0
Salience-DETR [27]	R50	12	900	49.2	67.1	53.8	32.7	53.0	63.1
DAC-DETR [2]	R50	12	900	50.0	67.6	54.7	32.9	53.1	64.2
MS-DETR [9]	R50	12	900	50.0	67.3	54.4	31.6	53.2	64.0
Rank-DETR [23]	R50	12	900	50.4	67.9	55.2	33.6	53.8	64.2
Stable-DETR [20]	R50	12	900	50.4	67.4	55.0	32.9	54.0	65.5
Dual-R-DETR(Route-DETR) [31]	R50	12	900	50.7	68.8	55.0	34.1	54.2	64.6
EASE-DETR [4]	R50	12	900	50.8	68.9	55.3	34.1	54.2	65.1
DDQ-DETR [6]	R50	12	900	51.3	68.6	56.4	33.5	54.9	65.9
Relation-DETR [5]	R50	12	900	51.7	69.1	56.3	36.1	55.6	66.1
Co-DETR [15]	R50	12	900	52.1	69.3	57.3	35.4	55.5	67.2
LoRA-DETR [10]	R50	12	900	52.5	69.8	57.3	**37.2**	56.5	66.8
DETRNN (ours)	R50	12	900	**53.0**	**70.2**	**57.6**	36.3	**56.8**	**67.5**
DINO [8]	Swin-L	12	900	56.8	75.4	62.3	41.1	60.6	73.5
Salience-DETR [27]	Swin-L	12	900	56.5	75.0	61.5	40.2	61.2	72.8
Dual-R-DETR(Route-DETR) [31]	Swin-L	12	900	57.6	76.4	63.2	40.8	61.8	73.6
Rank-DETR [23]	Swin-L	12	900	57.6	76.0	63.4	41.6	61.4	73.8
Stable-DETR [20]	Swin-L	12	900	57.7	75.7	63.4	39.8	62.0	74.7
EASE-DETR [4]	Swin-L	12	900	57.8	**76.7**	63.3	40.7	61.9	73.7
Relation-DETR [5]	Swin-L	12	900	57.8	76.1	62.9	41.2	62.1	74.4
LoRA-DETR [10]	Swin-L	12	900	58.2	76.2	63.4	41.8	63.0	74.4
DETRNN (ours)	Swin-L	12	900	**58.5**	76.5	**64.0**	**42.5**	**63.1**	**74.9**

F.3 Why CGQO+Mamba Scales

Although Algorithm 1 is written as a sequential greedy procedure for clarity, only the pivot selection in Lines 5–10 is inherently order-dependent. The dense pairwise similarity *all_sim* and pivot-wise thresholds are computed once with parallel tensor operations. For each selected pivot, Lines 7–8 read one similarity row and threshold candidate queries in parallel on GPU, and the final ordering is assembled from compact $O(N)$ grouping results. Thus, CGQO preserves the greedy ordering semantics while keeping the expensive pairwise and per-query operations GPU-resident.

CGQO still has $O(N^2)$ pairwise similarity, but it operates on low-dimensional inputs: 4-d boxes for IoU and 80-d class scores for cosine similarity. This gives it a much smaller constant than self-attention over 256-d query features, which forms and normalizes an $N \times N$ attention map. Mamba further scales gently with N due to its linear state-space computation. Together, these design choices improve memory scaling and prevent query ordering from becoming a bottleneck at large N.

G Full-recipe SOTA Context

Beyond the controlled comparisons in Tab. 1 and 2 and the replacement study in Tab. 8, Tab. 14 compares DETRNN with recent DETR-style SOTA methods under their reported full recipes. To reduce recipe mismatch, we focus on the common 1× training and 900-query setting with ResNet-50 and Swin-L backbones when available. Under this leaderboard-style comparison, our DETRNN outperforms recent competition-aware methods such as EASE-DETR, Dual-R-DETR (formerly Route-DETR), DDQ-DETR, and Relation-DETR. It also improves over strong full-recipe baselines such as Co-DETR and LoRA-DETR, achieving the highest AP among the compared DETR-style detectors under both R50 and Swin-L settings. These results complement the controlled replacement study and show that sequential competition refinement remains effective within strong full detector recipes.

H Details for Competition-Aware Baselines

Tab. 8 reports **2**× results for Relation-DETR [5], EASE-DETR [4], and DDQ-DETR [6].

Why replacement is used. Recent competition-aware DETR variants differ not only in how they model query competition, but also in many recipe-level components, such as auxiliary one-to-many supervision, dense/distinct query selection, cross-attention design, training-only branches, loss function and baseline-specific training settings. A full best-recipe comparison would therefore mix the effect of the competition-resolution module with these additional components. Our goal in Tab. 8 is instead to isolate the query-competition mechanism itself. Thus, the *Base* rows report the original full baselines, while the *+Ours* rows keep the remaining training and detector recipe unchanged and replace only the competition-specific decoder query-interaction/pruning operation with DETRNN. This controlled replacement setting tests whether explicit competition ordering with sequential refinement can serve as an efficient alternative to prior competition-specific modules, rather than ranking full SOTA recipes.

Integration rule. For the *+Ours* rows, we remove the decoder-wise competition-specific self-attention/pruning operations and use standard self-attention in those stages, while applying DETRNN (CGQO + Bi-RNN) only at the last decoder stage. All other settings, including the backbone, query setting, training schedule, matching/auxiliary-supervision recipe, and non-competition detector components, are kept the same as the corresponding baseline. This is not an additive stacking comparison; it evaluates whether a lightweight last-stage ordered refinement can replace multi-stage competition-specific operations. The results show that this replacement improves both AP and throughput, suggesting that a single ordered sequential refinement can be a more efficient alternative to multi-stage competition-specific designs.

Relation-DETR. The Base row keeps the original Relation-DETR design and training recipe. In +Ours, we replace its competition-specific relation-attention in the decoder with DETRNN and use standard self-attention in the earlier replaced stages, while keeping the remaining detector recipe unchanged. This reduces computation and improves AP.

DDQ-DETR. The Base row keeps DDQ-DETR(4scale)'s original dense/distinct-query design and stage-wise distinct-query selection/pruning. In +Ours, we replace the per-stage competition-specific pruning operations with standard self-attention in earlier stages and apply DETRNN only at the last decoder, while keeping the remaining DDQ recipe unchanged. This yields higher FPS and AP.

EASE-DETR. Since we did not find an official public implementation of EASE-DETR, we implement an EASE-Deformable-DETR++ baseline following [4]. The Base rows report the corresponding full baselines, using our reproduced/evaluated implementation. The performance matches the originally reported AP. In +Ours, we follow the same replacement rule: only the competition-specific decoder interaction is replaced by DETRNN, while the remaining EASE recipe is kept unchanged.

I Settings for The 1-decoder Stress Test

The **1-decoder** results in Tab. 3 are obtained by training a standard 6-decoder (for both the baseline and our method) but evaluating using the predictions after the first decoder stage. We adopt this setup because training a full 6-decoder model converges faster and is more stable than training a single-decoder from scratch, while still allowing a clean measurement of early-stage redundancy.

Different from our default stage placement (Sec. 4.7), for this stress test we apply DETRNN at *all* decoder stages so that the first-decoder output is also refined by our module. CGQO builds the order from the shared initial predictions produced by the encoder proposal head (available in Def-DETR++, DINO, and MS-DETR). We report AP without NMS and AP after evaluation-time NMS@0.8, where the NMS gain (ΔAP) serves as a redundancy indicator.

Controlled protocol. For all baselines and DETRNN, we use the same training schedule, data processing, and loss setup, and keep the query count fixed. At evaluation, we run the same inference and post-processing code and simply read out the predictions after the first decoder stage. CGQO builds the order from the encoder proposal head predictions (no ground-truth information is used).

Table 15: Stage-wise AP and AR100 of the original Def-DETR++ and MS-DETR baselines on COCO `val2017`. AR100 saturates earlier than AP, indicating that later decoder stages mainly refine already covered objects.

Baseline	Metric	Stage 4	Stage 5	Stage 6
Def-DETR++	AP	47.1	47.4	47.6
	AR100	67.4	67.7	67.6
MS-DETR	AP	49.1	49.5	50.0
	AR100	72.7	73.0	72.6

Table 16: WiderPerson consistency check (AP). NMS brings only marginal gains (+0.1) for the baseline.

Detector	w/o NMS	+NMS@0.7	+NMS@0.8	+NMS@0.9	+Ours
Def-DETR++	46.2	46.3 (+0.1)	46.3 (+0.1)	46.3 (+0.1)	**47.2** (+1.0)

J Stage-wise Analysis for Stage Placement

To better understand the late-stage placement of DETRNN, we examine stage-wise AP and AR100 for the original Def-DETR++ and MS-DETR baselines. For the original Def-DETR++ and MS-DETR baselines, Tab. 15 reports stage-wise AP and AR100 from decoder stages 4–6. In both baselines, AR100 is already close to its peak by stage 4, while AP continues to improve through the final stage. For Def-DETR++, AP increases from 47.1 to 47.6 from stages 4 to 6, whereas AR100 changes only from 67.4 to 67.6. A similar trend appears for MS-DETR, where AP increases from 49.1 to 50.0 while AR100 remains nearly saturated.

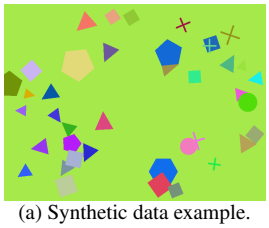
These results suggest that object coverage is largely established before the final decoder stages, while later stages continue to refine the remaining predictions and resolve residual query competition. This is consistent with applying DETRNN only to the last one or two decoder stages, as studied in Tab. 9.

K Additional Crowded Benchmark: WiderPerson

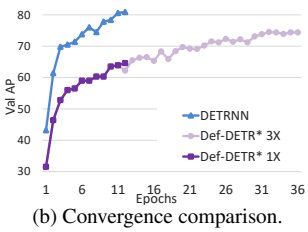
We further evaluate DETRNN on WiderPerson, a crowded pedestrian detection benchmark with dense person instances and substantial occlusion, using the same Def-DETR++/R50/1× setting. As shown in Tab. 16, applying NMS to the baseline brings only a marginal gain of +0.1 AP across thresholds, whereas DETRNN improves AP from 46.2 to 47.2 (+1.0) without post-processing. This is consistent with the CrowdHuman and COCO results in Sec. 4.5, further showing that DETRNN can reduce duplicate competition within the decoder rather than relying on NMS threshold tuning.

L Synthetic Mechanism Validation

To visualize the effect of competition-aware ordering in a controlled setting, we construct a synthetic benchmark (Fig. 9a) that intentionally amplifies object redundancy (dense overlaps) and compare DETRNN with a simplified version of Def-DETR [7] (named Def-DETR*) under the same setup. As shown in Fig. 9b and Fig. 9c, DETRNN reaches 83.6 AP within 12 epochs, while Def-DETR* attains 64.6 AP at 12 epochs and 74.4 AP at 36 epochs. Moreover, since NMS mainly removes duplicates,



(a) Synthetic data example.



(b) Convergence comparison.

Method	Epochs	NMS	AP	FPS
Def-DETR*	12		64.6	124
		✓	81.9 (+17.3)	118
Def-DETR*	36		74.4	124
		✓	83.3 (+8.9)	118
DETRNN (ours)	12		83.6	120
		✓	83.2 (-0.4)	114

(c) Synthetic experiment results.

Figure 9: (a), (b), and (c) show the synthetic data example, convergence comparison on the synthetic val dataset, and experiment results comparing Def-DETR* [7] and DETRNN with and without NMS, respectively. The dataset and model are simplified for rapid validation.

Table 17: Cosine similarity around CGQO group boundaries.

Compared queries	Mean	25th percentile	75th percentile
tail-1 vs. *tail*	**0.734**	**0.610**	**0.906**
tail vs. *next pivot*	0.330	0.133	0.486

the AP change after applying it reflects how many redundant predictions remain. This change is large and positive for Def-DETR*, but near zero or slightly negative for DETRNN, indicating that DETRNN leaves few duplicates for NMS to remove. For DETRNN, the slight AP drop suggests that once redundancy is largely suppressed, hard thresholding may start removing useful boxes. We include this synthetic benchmark only as a fast, controlled sanity check; the main conclusions are still based on MS-COCO and CrowdHuman.

L.1 Settings

We conduct a synthetic experiment on a dataset of 4800 images, each sized 800×600. Every image features a random background and foreground objects drawn from six shape categories (triangle, square, pentagon, hexagon, circle, and cross) in random colors, positions, and sizes, with the number of objects per image varying from 1 to 40 and each object occupying no more than one-tenth of the image size. Our model is a simplified version of Def-DETR [7] with a frozen R50 backbone, a single encoder, and a three-level feature pyramid paired with a single decoder. We further reduce the decoder hidden dimensions to one quarter of the original setting, use 300 object queries, and train with batch size 16. The threshold in CGQO is a simple fixed 0.4. Owing to the small image size, limited dataset, and simplified model, the synthetic experiment runs very fast, making it ideal for rapid hypothesis validation.

M Cross-Group Transitions in the Recurrent Sequence

CGQO concatenates multiple competition groups into a single recurrent sequence. We therefore examine whether transitions between successive groups are distinguishable in the query features.

Using Def-DETRNN++/R50/1$\times$ on the first 1,600 COCO `val2017` images, we refer to the last query of a group as its *tail*, and the preceding query as *tail-1*. We compare the cosine similarity of the last within-group pair (*tail-1* vs. *tail*) with that of the pair crossing the group boundary (*tail* vs. *next pivot*).

As shown in Tab. 17, the mean similarity drops from 0.734 within a group to 0.330 across the boundary. Moreover, the within-group 25th percentile (0.610) remains higher than the boundary 75th percentile (0.486), indicating a clear feature transition between successive competition groups.

N Seed Variance and Reliability

To assess run-to-run variance, we repeat four representative R50/1$\times$ settings with three random seeds. Results are summarized in Tab. 18.

All runs outperform their corresponding baselines, with standard deviations of only 0.06–0.15 AP. This indicates that the improvements are consistent across random seeds and are substantially larger than the observed run-to-run variation.

Table 18: Multi-seed results on COCO. Mean and standard deviation are computed over three runs.

Detector	Baseline	Seed 1	Seed 2	Seed 3	Mean	Std.
Def-DETRNN++	47.6	48.7	48.9	48.6	48.73	0.15
DINO-RNN	49.0	50.0	50.0	49.8	49.93	0.12
MS-DETRNN	50.0	50.5	50.3	50.5	50.43	0.12
LoRA-DETRNN	52.5	53.0	53.0	52.9	52.97	0.06

Table 19: Comparison of order-aware interaction modules with CGQO on DINO/R50/1×.

Method	Params.	AP	AP_{50}	AP_{75}
Self-attention baseline	47.5M	49.0	66.6	53.5
CGQO + Uni-RNN	47.6M	49.5	67.7	53.9
CGQO + rank-bias attn	47.7M	49.7	67.8	**54.1**
CGQO + causal attn	47.5M	49.5	67.6	54.0
CGQO + 1D Conv	47.7M	49.6	67.6	**54.1**
CGQO + Bi-RNN	47.6M	**50.0**	**68.1**	54.0

O Alternative Order-Aware Interaction Modules

To examine how broadly the CGQO order can be exploited, we compare Bi-RNN with several alternative order-aware interaction mechanisms on DINO/R50/1×, while keeping CGQO fixed. Uni-RNN processes the ordered queries in one direction; rank-bias attention injects their relative CGQO ranks into self-attention as bias; causal attention restricts attention according to the query order; and 1D convolution aggregates local neighbors along the ordered sequence.

All four alternatives improve over the self-attention baseline, showing that the CGQO order can benefit different order-aware interaction mechanisms. Bi-RNN achieves the highest overall AP and AP_{50} with a comparable parameter count, supporting its use as our default design.

P Limitations, Future Directions, and Broader Impact

Limitations. CGQO uses a discrete, non-differentiable, hand-designed grouping rule rather than a learned ordering objective, and therefore depends on the quality of preceding predictions. DETRNN is designed for late-stage refinement, and its gains can be smaller on already strong or competition-aware baselines. Our evaluation does not cover some recent detector families such as RT-DETR and D-FINE, while extreme cases including very large or chain-structured competition groups and unusual query distributions remain insufficiently explored.

Future directions. CGQO is not tied to box IoU and only requires a pairwise similarity measure, allowing extensions such as mask IoU for instance segmentation or task-specific similarities for 3D and video detection. Its favorable large-query memory and runtime scaling also motivates further study in dense detection settings such as remote-sensing imagery.

Broader impact. DETRNN may benefit applications such as autonomous perception and assistive vision, while also inheriting general object-detection risks including privacy-sensitive surveillance and biased deployment.

Q Visualization of prediction redundancy

The visual comparison in Fig. 10 highlights the differences in redundancy handling between the models. Without NMS, the baseline predictions contain many overlapping duplicates. Even after applying NMS with τ=0.7 (score threshold $1e{-}1$), noticeable duplicates remain, indicating limited redundancy suppression inside the decoder. In contrast, Def-DETRNN++ predictions show minimal redundancy, whether or not NMS is applied, demonstrating its capability for substantially less redundant end-to-end predictions. This visualization underscores the advantages of Def-DETRNN++ in accelerating convergence and achieving effective query differentiation, especially in the 1-decoder setting.

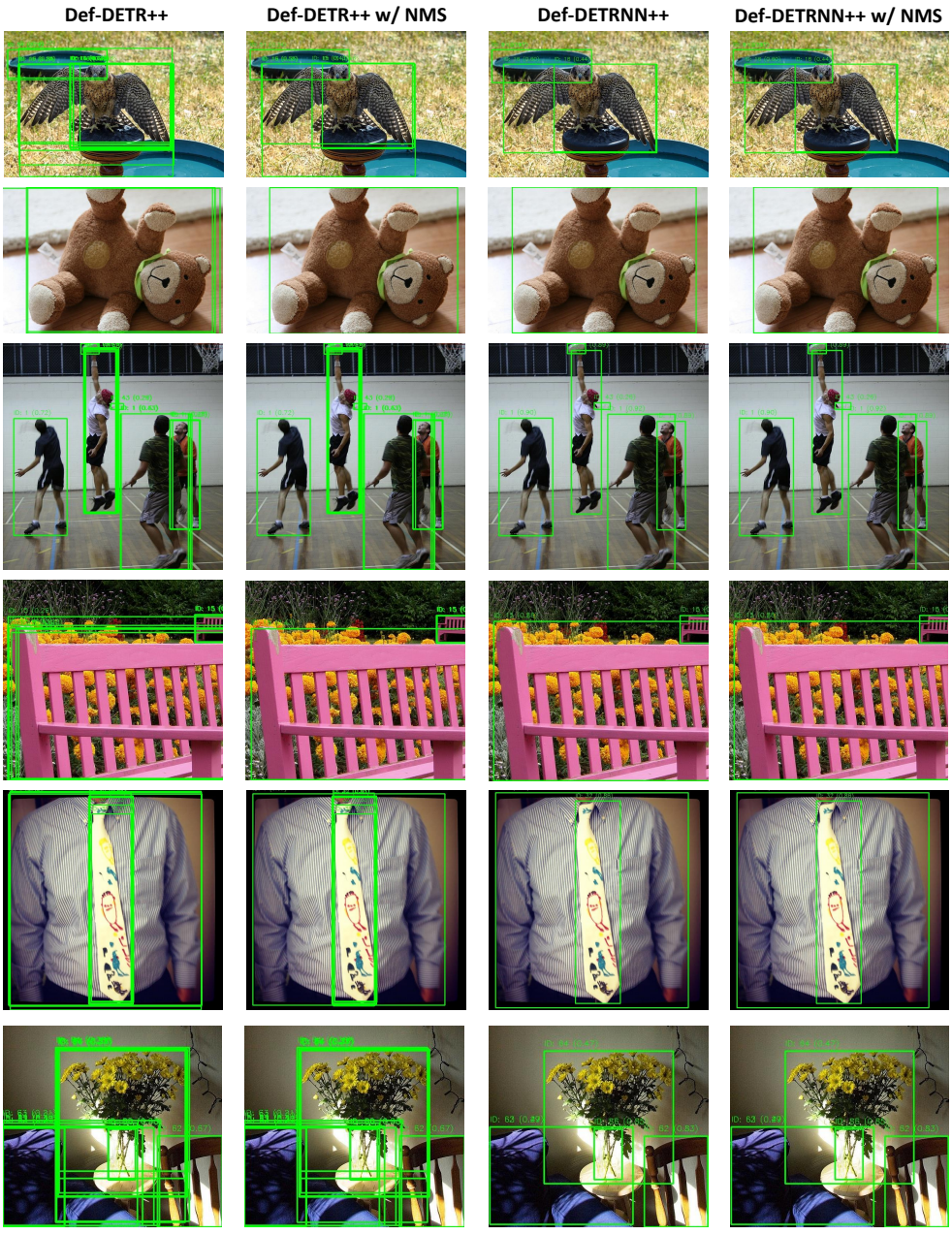


Figure 10: Visual comparison of predictions for Def-DETR++ and Def-DETRNN++, with and without NMS, under the 1-decoder setting. The results highlight Def-DETRNN++'s minimal redundancy and effective end-to-end predictions, while Def-DETR++ shows significant redundancy without NMS.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction align with the paper's actual contributions and evaluated scope. The main claims are directly supported by the benchmark results, ablations, and analysis in Secs. 3–4, and the paper does not make broader claims beyond the tested detection settings.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the proposed method in Appendix P, including the main constraints of the current study and the scope of the reported findings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results requiring assumptions and proofs; it is primarily a method and experimental paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the information needed to reproduce the main experimental results, including the method design, benchmark protocols, training settings, and evaluation details, in the main paper and appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The datasets used in our experiments are public, but we do not release the code with the submission. Nevertheless, the paper and appendix provide the model design, key hyperparameters, and pseudocode needed to reproduce the main experimental setup, and we intend to release the code after publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the main training and test details needed to understand the results, including the data splits, model settings, key hyperparameters, optimizer and schedule, and evaluation protocols. Additional implementation details and the chosen settings are described in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Full error bars are not reported for all experiments due to computational cost. However, we report three-seed results for four representative COCO settings, including mean and standard deviation in Appendix N.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E reports the training GPU resources, batch size, schedules, and the hardware/protocol used for FPS, FLOPs, and module profiling.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, this work conforms to the NeurIPS Code of Ethics. The paper presents a standard algorithmic study and does not involve human participants or sensitive data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix P briefly discusses potential beneficial applications and downstream risks of improved object detection systems.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This question is not applicable because the paper does not release data or models with a high risk for misuse, such as generative models or scraped datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: We properly credit the datasets, models, and prior codebases used in this work through citations in the main paper. However, the current submission does not explicitly list the licenses and terms of use for all such assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper proposes a new method but does not release new datasets, code, or other assets with the submission, so this question is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper is a methodological study on object detection and does not include crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs. Any LLM use was limited to writing or editing assistance and did not affect the methodology, scientific rigor, or originality of the work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.