# DETRs Beat YOLOs on Real-time Object Detection

Yian Zhao[1,2†] Wenyu Lv[1†‡] Shangliang Xu[1] Jinman Wei[1] Guanzhong Wang[1]
Qingqing Dang[1] Yi Liu[1] Jie Chen[2 ✉]

[1]Baidu Inc, Beijing, China [2]School of Electronic and Computer Engineering, Peking University, Shenzhen, China

zhaoyian@stu.pku.edu.cn lvwenyu01@baidu.com jiechen2019@pku.edu.cn

## Abstract

*The YOLO series has become the most popular framework for real-time object detection due to its reasonable trade-off between speed and accuracy. However, we observe that the speed and accuracy of YOLOs are negatively affected by the NMS. Recently, end-to-end Transformer-based detectors (DETRs) have provided an alternative to eliminating NMS. Nevertheless, the high computational cost limits their practicality and hinders them from fully exploiting the advantage of excluding NMS. In this paper, we propose the **Re**al-**T**ime **DE**tection **TR**ansformer (RT-DETR), the first real-time end-to-end object detector to our best knowledge that addresses the above dilemma. We build RT-DETR in two steps, drawing on the advanced DETR: first we focus on maintaining accuracy while improving speed, followed by maintaining speed while improving accuracy. Specifically, we design an efficient hybrid encoder to expeditiously process multi-scale features by decoupling intra-scale interaction and cross-scale fusion to improve speed. Then, we propose the uncertainty-minimal query selection to provide high-quality initial queries to the decoder, thereby improving accuracy. In addition, RT-DETR supports flexible speed tuning by adjusting the number of decoder layers to adapt to various scenarios without retraining. Our RT-DETR-R50 / R101 achieves* 53.1% / 54.3% *AP on COCO and* 108 / 74 *FPS on T4 GPU, outperforming previously advanced YOLOs in both speed and accuracy. Furthermore, RT-DETR-R50 outperforms DINO-R50 by* 2.2% *AP in accuracy and about* 21 *times in FPS. After pre-training with Objects365, RT-DETR-R50 / R101 achieves* 55.3% / 56.2% *AP. The project page: https://zhao-yian.github.io/RTDETR.*

## 1. Introduction

Real-time object detection is an important area of research and has a wide range of applications, such as object tracking [43], video surveillance [28], and autonomous driving [2], *etc*. Existing real-time detectors generally adopt the CNN-based architecture, the most famous of which is the YOLO detectors [1, 10–12, 15, 16, 25, 30, 38, 40] due to their reasonable trade-off between speed and accuracy. However, these detectors typically require Non-Maximum Suppression (NMS) for post-processing, which not only slows down the inference speed but also introduces hyperparameters that cause instability in both the speed and accuracy. Moreover, considering that different scenarios place different emphasis on recall and accuracy, it is necessary to carefully select the appropriate NMS thresholds, which hinders the development of real-time detectors.

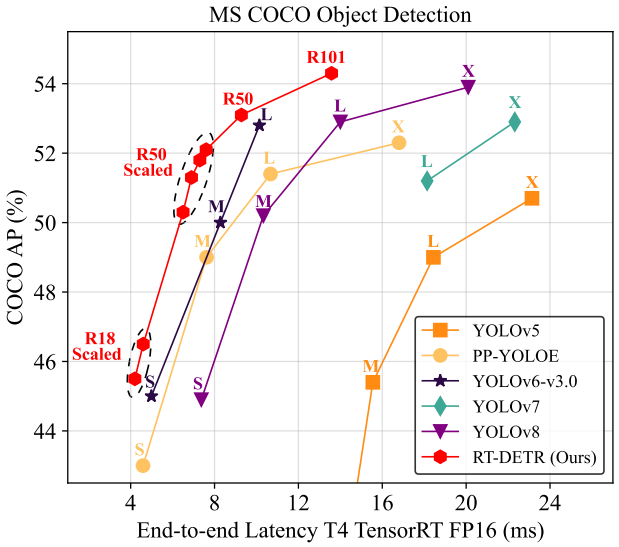



Figure 1. Compared to previously advanced real-time object detectors, our RT-DETR achieves state-of-the-art performance.

Recently, the end-to-end Transformer-based detectors (DETRs) [4, 17, 23, 27, 36, 39, 44, 45] have received extensive attention from the academia due to their streamlined architecture and elimination of hand-crafted components. However, their high computational cost prevents them from meeting real-time detection requirements, so the NMS-free architecture does not demonstrate an inference speed advantage. This inspires us to explore whether DETRs can be extended to real-time scenarios and outperform the advanced YOLO detectors in both speed and accuracy, eliminating the delay caused by NMS for real-time object detection.

To achieve the above goal, we rethink DETRs and conduct detailed analysis of key components to reduce unnecessary

✉Corresponding author. †Equal contribution. ‡ Project leader.

# DETRs在实时目标检测领域超越YOLOs

赵岩[1,2†] 吕文宇[1†‡] 徐尚良[1] 魏金满[1] 王冠中[1] 党青青[1] 刘毅[1] 陈杰[2✉]

[1]Baidu Inc, Beijing, China [2]School of Electronic and Computer Engineering, Peking University, Shenzhen, China
zhaoyian@stu.pku.edu.cn lvwenyu01@baidu.com jiechen2019@pku.edu.cn

## 摘要

*The YOLO series has become the most popular framework for real-time object detection due to its reasonable trade-off between speed and accuracy. However, we observe that the speed and accuracy of YOLOs are negatively affected by the NMS. Recently, end-to-end Transformer-based detectors (DETRs) have provided an alternative to eliminating NMS. Nevertheless, the high computational cost limits their practicality and hinders them from fully exploiting the advantage of excluding NMS. In this paper, we propose the **Re**al-**T**ime **DE**tection **TR**ansformer (RT-DETR), the first real-time end-to-end object detector to our best knowledge that addresses the above dilemma. We build RT-DETR in two steps, drawing on the advanced DETR: first we focus on maintaining accuracy while improving speed, followed by maintaining speed while improving accuracy. Specifically, we design an efficient hybrid encoder to expeditiously process multi-scale features by decoupling intra-scale interaction and cross-scale fusion to improve speed. Then, we propose the uncertainty-minimal query selection to provide high-quality initial queries to the decoder, thereby improving accuracy. In addition, RT-DETR supports flexible speed tuning by adjusting the number of decoder layers to adapt to various scenarios without retraining. Our RT-DETR-R50 / R101 achieves* 53.1% / 54.3% *AP on COCO and* 108 / 74 *FPS on T4 GPU, outperforming previously advanced YOLOs in both speed and accuracy. Furthermore, RT-DETR-R50 outperforms DINO-R50 by* 2.2% *AP in accuracy and about* 21 *times in FPS. After pre-training with Objects365, RT-DETR-R50 / R101 achieves* 55.3% / 56.2% *AP. The project page: https://zhao-yian.github.io/RTDETR.*

## 1. 引言

实时目标检测是一个重要的研究领域，具有广泛的应用场景，如目标跟踪[43]、视频监控[28]和自动驾驶[2]。*etc.*现有的实时检测器通常采用基于CNN的架构，其中最著名的是YOLO系列检测器[1, 10–12, 15, 16, 25, 30, 38, 40]，因其

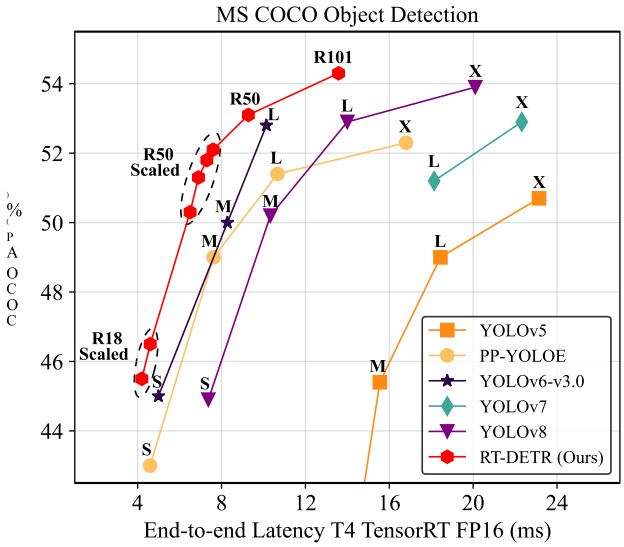



图1. 与先前先进的实时目标检测器相比，我们的RT-DETR实现了最先进的性能。

它们在速度与精度之间实现了合理的平衡。然而，这些检测器通常需要非极大值抑制（NMS）进行后处理，这不仅会降低推理速度，还会引入超参数，导致速度和精度均不稳定。此外，考虑到不同场景对召回率和精确度的侧重不同，必须谨慎选择合适的NMS阈值，这阻碍了实时检测器的发展。

近期，基于端到端Transformer的检测器（DETRs）[4, 17, 23, 27, 36, 39, 44, 45]因其架构简洁、无需手工设计组件而受到学术界广泛关注。然而，其高昂的计算成本使其难以满足实时检测需求，导致这种无需非极大值抑制（NMS）的架构在推理速度上并未显现优势。这促使我们思考：能否将DETRs拓展至实时场景，在速度和精度上均超越先进的YOLO检测器，从而消除NMS带来的延迟，实现真正的实时目标检测。

为了实现上述目标，我们重新思考了DETR架构，并对关键组件进行了细致分析，以减少不必要的{v*}

✉Corresponding author. †Equal contribution. ‡ Project leader.

computational redundancy and further improve accuracy. For the former, we observe that although the introduction of multi-scale features is beneficial in accelerating the training convergence [45], it leads to a significant increase in the length of the sequence feed into the encoder. The high computational cost caused by the interaction of multi-scale features makes the Transformer encoder the computational bottleneck. Therefore, implementing the real-time DETR requires a redesign of the encoder. And for the latter, previous works [42, 44, 45] show that the hard-to-optimize object queries hinder the performance of DETRs and propose the query selection schemes to replace the vanilla learnable embeddings with encoder features. However, we observe that the current query selection directly adopt classification scores for selection, ignoring the fact that the detector are required to simultaneously model the category and location of objects, both of which determine the quality of the features. This inevitably results in encoder features with low localization confidence being selected as initial queries, thus leading to a considerable level of uncertainty and hurting the performance of DETRs. We view query initialization as a breakthrough to further improve performance.

In this paper, we propose the **R**eal-**T**ime **DE**tection **TR**ansformer (RT-DETR), the first real-time end-to-end object detector to our best knowledge. To expeditiously process multi-scale features, we design an efficient hybrid encoder to replace the vanilla Transformer encoder, which significantly improves inference speed by decoupling the intra-scale interaction and cross-scale fusion of features with different scales. To avoid encoder features with low localization confidence being selected as object queries, we propose the uncertainty-minimal query selection, which provides high-quality initial queries to the decoder by explicitly optimizing the uncertainty, thereby increasing the accuracy. Furthermore, RT-DETR supports flexible speed tuning to accommodate various real-time scenarios without retraining, thanks to the multi-layer decoder architecture of DETR.

RT-DETR achieves an ideal trade-off between the speed and accuracy. Specifically, RT-DETR-R50 achieves $53.1\%$ AP on COCO `val2017` and 108 FPS on T4 GPU, while RT-DETR-R101 achieves $54.3\%$ AP and 74 FPS, outperforming $L$ and $X$ models of previously advanced YOLO detectors in both speed and accuracy, Figure 1. We also develop scaled RT-DETRs by scaling the encoder and decoder with smaller backbones, which outperform the lighter YOLO detectors ($S$ and $M$ models). Furthermore, RT-DETR-R50 outperforms DINO-Deformable-DETR-R50 by $2.2\%$ AP ($53.1\%$ AP vs $50.9\%$ AP) in accuracy and by about 21 times in FPS (108 FPS vs 5 FPS), significantly improves accuracy and speed of DETRs. After pre-training with Objects365 [35], RT-DETR-R50 / R101 achieves $55.3\%$ / $56.2\%$ AP, resulting in surprising performance improvements. More experimental results are provided in the Appendix.

The main contributions are summarized as: (i). We propose the first real-time end-to-end object detector called RT-DETR, which not only outperforms the previously advanced YOLO detectors in both speed and accuracy but also eliminates the negative impact caused by NMS post-processing on real-time object detection; (ii). We quantitatively analyze the impact of NMS on the speed and accuracy of YOLO detectors, and establish an end-to-end speed benchmark to test the end-to-end inference speed of real-time detectors; (iii). The proposed RT-DETR supports flexible speed tuning by adjusting the number of decoder layers to accommodate various scenarios without retraining.

## 2. Related Work

### 2.1. Real-time Object Detectors

YOLOv1 [31] is the first CNN-based one-stage object detector to achieve true real-time object detection. Through years of continuous development, the YOLO detectors have outperformed other one-stage object detectors [21, 24] and become the synonymous with the real-time object detector. YOLO detectors can be classified into two categories: anchor-based [1, 11, 15, 25, 29, 30, 37, 38] and anchor-free [10, 12, 16, 40], which achieve a reasonable trade-off between speed and accuracy and are widely used in various practical scenarios. These advanced real-time detectors produce numerous overlapping boxes and require NMS post-processing, which slows down their speed.

### 2.2. End-to-end Object Detectors

End-to-end object detectors are well-known for their streamlined pipelines. Carion *et al*. [4] first propose the end-to-end detector based on Transformer called DETR, which has attracted extensive attention due to its distinctive features. Particularly, DETR eliminates the hand-crafted anchor and NMS components. Instead, it employs bipartite matching and directly predicts the one-to-one object set. Despite its obvious advantages, DETR suffers from several problems: slow training convergence, high computational cost, and hard-to-optimize queries. Many DETR variants have been proposed to address these issues. **Accelerating convergence.** Deformable-DETR [45] accelerates training convergence with multi-scale features by enhancing the efficiency of the attention mechanism. DAB-DETR [23] and DN-DETR [17] further improve performance by introducing the iterative refinement scheme and denoising training. Group-DETR [5] introduces group-wise one-to-many assignment. **Reducing computational cost.** Efficient DETR [42] and Sparse DETR [33] reduce the computational cost by reducing the number of encoder and decoder layers or the number of updated queries. Lite DETR [18] enhances the efficiency of encoder by reducing the update frequency of low-level features in an interleaved way. **Optimizing query initial-**

计算冗余并进一步提升准确性。对于前者，我们观察到，尽管引入多尺度特征有助于加速训练收敛[45]，但它显著增加了输入编码器的序列长度。多尺度特征交互带来的高计算成本使得Transformer编码器成为计算瓶颈。因此，实现实时DETR需要对编码器进行重新设计。对于后者，先前的研究[42,44,45]表明，难以优化的对象查询阻碍了DETRs的性能，并提出用编码器特征替换原始可学习嵌入的查询选择方案。然而，我们发现当前的查询选择直接采用分类分数进行筛选，忽略了检测器需要同时建模对象类别和位置的事实——这两者共同决定了特征的质量。这不可避免地导致定位置信度低的编码器特征被选为初始查询，从而引入显著的不确定性并损害DETRs的性能。我们将查询初始化视为进一步提升性能的突破口。

本文提出了实时检测变换器（RT-DETR），据我们所知，这是首个实时端到端目标检测器。为高效处理多尺度特征，我们设计了一种高效的混合编码器以替代传统Transformer编码器，通过解耦同尺度特征交互与跨尺度特征融合，显著提升了推理速度。为避免定位置信度低的编码器特征被选作目标查询，我们提出不确定性最小化查询选择机制，通过显式优化不确定性为解码器提供高质量初始查询，从而提升检测精度。此外，得益于DETR的多层解码器架构，RT-DETR支持无需重新训练即可灵活调整速度，以适应多样化的实时场景。

RT-DETR在速度与精度之间实现了理想的平衡。具体而言，RT-DETR-R50在COCO val2017数据集上达到53.1% AP，在T4 GPU上实现108 FPS；而RT-DETR-R101则以54.3% AP和74 FPS的表现，在速度和精度上均超越了先前先进的YOLO检测器的$L$和$X$型号，如图1所示。我们还通过采用更轻量级骨干网络对编码器和解码器进行缩放，开发了不同规模的RT-DETR变体，其性能优于轻量级YOLO检测器（$S$和$M$型号）。此外，RT-DETR-R50在精度上以2.2% AP的优势超越DINO-Deformable-DETR-R50（53.1% AP对比50.9% AP），并在FPS上实现约21倍的提升（108 FPS对比5 FPS），显著提升了DETR系列模型的精度与速度。通过在Objects365数据集[35]上进行预训练后，RT-DETR-R50/R101分别达到55.3%/56.2% AP，实现了惊人的性能提升。更多实验结果详见附录。

主要贡献总结如下：(i). 我们提出了首个实时端到端目标检测器RT-DETR，其不仅在速度与精度上超越了先前先进的YOLO检测器，还消除了NMS后处理对实时目标检测造成的负面影响；(ii). 我们定量分析了NMS对YOLO检测器速度与精度的影响，并建立了端到端速度基准以测试实时检测器的端到端推理速度；(iii). 所提出的RT-DETR支持通过调整解码器层数进行灵活的速度调节，无需重新训练即可适应不同场景。

## 2. 相关工作

### 2.1. 实时目标检测器

YOLOv1 [31]是首个基于CNN的单阶段目标检测器，实现了真正意义上的实时目标检测。经过多年持续发展，YOLO系列检测器已超越其他单阶段检测器[21,24]，成为实时目标检测的代名词。YOLO检测器可分为两大类：基于锚框的[1,11,15,25,29,30,37,38]与无锚框的[10,12,16,40]，它们在速度与精度间取得了良好平衡，被广泛应用于各类实际场景。这些先进的实时检测器会产生大量重叠检测框，必须依赖非极大值抑制(NMS)后处理，从而降低了运行速度。

### 2.2. 端到端目标检测器

端到端目标检测器以其流水线式的简洁流程而著称。Carion *et al*。[4]首次提出了基于Transformer的端到端检测器DETR，凭借其独特特性引起了广泛关注。尤其值得注意的是，DETR摒弃了手工设计的锚框和非极大值抑制组件，转而采用二分匹配机制直接预测一对一的目标集合。尽管优势显著，DETR仍存在若干问题：训练收敛速度慢、计算成本高以及查询难以优化。为此，研究者们提出了多种DETR变体以应对这些挑战。 **加速收敛**：Deformable-DETR[45]通过增强注意力机制效率，利用多尺度特征加速训练收敛。DAB-DETR[23]和DN-DETR[17]则通过引入迭代优化方案和去噪训练进一步提升了性能。Group-DETR[5]创新性地采用了分组式一对多分配策略。 **降低计算成本**：Efficient DETR[42]和Sparse DETR[33]通过减少编码器-解码器层数或优化查询更新数量来降低计算开销。Lite DETR[18]采用交错更新低频特征的策略，显著提升了编码器效率。 **优化查询初始**——

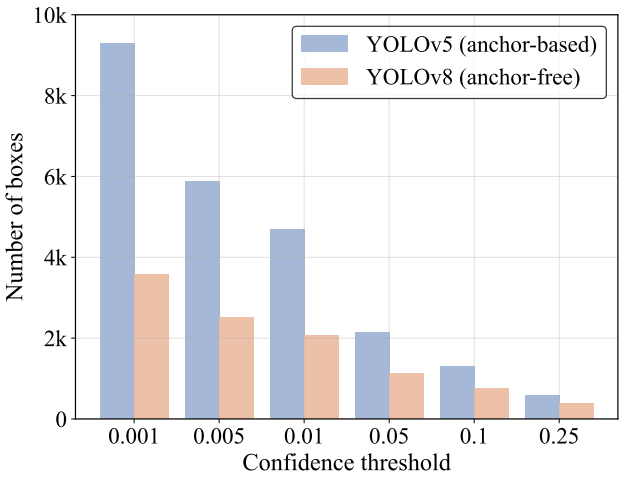



Figure 2. The number of boxes at different confidence thresholds.

**ization.** Conditional DETR [27] and Anchor DETR [39] decrease the optimization difficulty of the queries. Zhu *et al*. [45] propose the query selection for two-stage DETR, and DINO [44] suggests the mixed query selection to help better initialize queries. Current DETRs are still computationally intensive and are not designed to detect in real time. Our RT-DETR vigorously explores computational cost reduction and attempts to optimize query initialization, outperforming state-of-the-art real-time detectors.

## 3. End-to-end Speed of Detectors

### 3.1. Analysis of NMS

NMS is a widely used post-processing algorithm in object detection, employed to eliminate overlapping output boxes. Two thresholds are required in NMS: confidence threshold and IoU threshold. Specifically, the boxes with scores below the confidence threshold are directly filtered out, and whenever the IoU of any two boxes exceeds the IoU threshold, the box with the lower score will be discarded. This process is performed iteratively until all boxes of every category have been processed. Thus, the execution time of NMS primarily depends on the number of boxes and two thresholds. To verify this observation, we leverage YOLOv5 [11] (anchor-based) and YOLOv8 [12] (anchor-free) for analysis.

We first count the number of boxes remaining after filtering the output boxes with different confidence thresholds on the same input. We sample values from 0.001 to 0.25 as confidence thresholds to count the number of remaining boxes of the two detectors and plot them on a bar graph, which intuitively reflects that NMS is sensitive to its hyperparameters, Figure 2. As the confidence threshold increases, more prediction boxes are filtered out, and the number of remaining boxes that need to calculate IoU decreases, thus reducing the execution time of NMS.

Furthermore, we use YOLOv8 to evaluate the accuracy on the COCO `val2017` and test the execution time of the NMS operation under different hyperparameters. Note that the NMS operation we adopt refers to the TensorRT `efficientNMSPlugin`, which involves multiple kernels, including `EfficientNMSFilter`, `RadixSort`, `EfficientNMS`, *etc*., and we only report the execution time of the `EfficientNMS` kernel. We test the speed on T4 GPU with TensorRT FP16, and the input and preprocessing remain consistent. The hyperparameters and the corresponding results are shown in Table 1. From the results, we can conclude that the execution time of the `EfficientNMS` kernel increases as the confidence threshold decreases or the IoU threshold increases. The reason is that the high confidence threshold directly filters out more prediction boxes, whereas the high IoU threshold filters out fewer prediction boxes in each round of screening. We also visualize the predictions of YOLOv8 with different NMS thresholds in Appendix. The results show that inappropriate confidence thresholds lead to significant false positives or false negatives by the detector. With a confidence threshold of 0.001 and an IoU threshold of 0.7, YOLOv8 achieves the best AP results, but the corresponding NMS time is at a higher level. Considering that YOLO detectors typically report the model speed and exclude the NMS time, thus an end-to-end speed benchmark needs to be established.

| IoU thr. (Conf=0.001) | AP (%) | NMS (ms) | Conf thr. (IoU=0.7) | AP (%) | NMS (ms) |
|---|---|---|---|---|---|
| 0.5 | 52.1 | 2.24 | 0.001 | 52.9 | 2.36 |
| 0.6 | 52.6 | 2.29 | 0.01 | 52.4 | 1.73 |
| 0.8 | 52.8 | 2.46 | 0.05 | 51.2 | 1.06 |

Table 1. The effect of IoU threshold and confidence threshold on accuracy and NMS execution time.

### 3.2. End-to-end Speed Benchmark

To enable a fair comparison of the end-to-end speed of various real-time detectors, we establish an end-to-end speed benchmark. Considering that the execution time of NMS is influenced by the input, it is necessary to choose a benchmark dataset and calculate the average execution time across multiple images. We choose COCO `val2017` [20] as the benchmark dataset and append the NMS post-processing plugin of TensorRT for YOLO detectors as mentioned above. Specifically, we test the average inference time of the detector according to the NMS thresholds of the corresponding accuracy taken on the benchmark dataset, excluding `I/O` and `MemoryCopy` operations. We utilize the benchmark to test the end-to-end speed of anchor-based detectors YOLOv5 [11] and YOLOv7 [38], as well as anchor-free detectors PP-YOLOE [40], YOLOv6 [16] and YOLOv8 [12]

https://github.com/NVIDIA/TensorRT/tree/release/8.6/plugin/efficientNMSPlugin

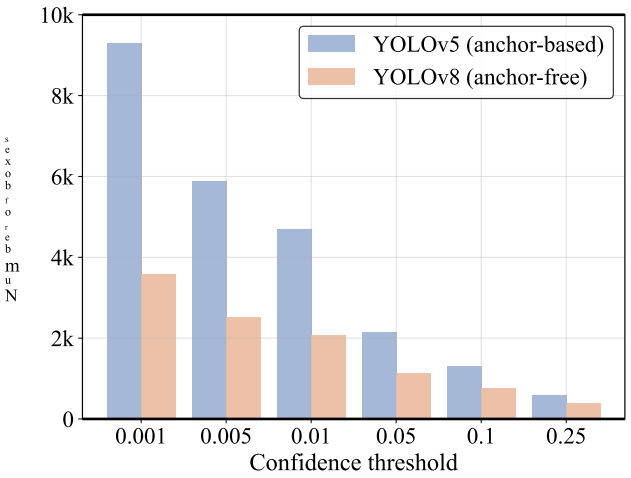



图2. 不同置信度阈值下的框体数量。

化。Conditional DETR [27]和Anchor DETR [39]降低了查询的优化难度。Zhu *et al*。[45]提出了针对两阶段DETR的查询选择方法，而DINO [44]则建议采用混合查询选择以更好地初始化查询。当前的DETR模型仍计算密集，并非为实时检测而设计。我们的RT-DETR大力探索计算成本的降低，并尝试优化查询初始化，性能超越了最先进的实时检测器。

## 3. 检测器的端到端速度 {v*}

### 3.1. NMS分析

NMS是目标检测中广泛使用的后处理算法，用于消除重叠的输出框。NMS需要两个阈值：置信度阈值和IoU阈值。具体而言，分数低于置信度阈值的框会被直接过滤掉；当任意两个框的IoU超过IoU阈值时，分数较低的框将被丢弃。这一过程会迭代进行，直到所有类别的框都被处理完毕。因此，NMS的执行时间主要取决于框的数量及这两个阈值。为验证这一观察，我们采用YOLOv5[11]（基于锚点）和YOLOv8[12]（无锚点）进行分析。

我们首先统计在同一输入上，根据不同置信度阈值过滤输出框后剩余的框数量。从0.001到0.25区间采样置信度阈值，分别计算两种检测器保留的预测框数量，并将其绘制成柱状图。该结果直观反映出NMS对其超参数的敏感性（图2）。随着置信度阈值的提高，更多预测框被过滤掉，需要计算IoU的剩余框数量减少，从而降低了NMS的执行时间。

此外，我们使用YOLOv8评估了在COCO val2017上的准确率，并测试了{v*}的执行时间

| IoU thr. (Conf=0.001) | AP (%) | NMS (ms) | Conf thr. (IoU=0.7) | AP (%) | NMS (ms) |
|---|---|---|---|---|---|
| 0.5 | 52.1 | 2.24 | 0.001 | 52.9 | 2.36 |
| 0.6 | 52.6 | 2.29 | 0.01 | 52.4 | 1.73 |
| 0.8 | 52.8 | 2.46 | 0.05 | 51.2 | 1.06 |

表1. IoU阈值和置信度阈值对准确率及NMS执行时间的影响。

不同超参数下的NMS操作。需注意，我们所采用的NMS操作指的是TensorRT的efficientNMSPlugin，该插件涉及多个内核，包括EfficientNMSFilter、RadixSort、EfficientNMS、*etc*，而我们仅报告EfficientNMS内核的执行时间。测试基于T4 GPU的TensorRT FP16模式，输入数据与预处理保持一致。表1展示了超参数设置及对应结果。从结果可得出结论：EfficientNMS内核的执行时间随置信度阈值降低或IoU阈值升高而增加。这是因为高置信度阈值会直接过滤掉更多预测框，而高IoU阈值则会在每轮筛选中保留更多预测框。附录中还可视化呈现了YOLOv8在不同NMS阈值下的预测效果，结果表明不恰当的置信度阈值会导致检测器出现大量误检或漏检。当置信度阈值为0.001、IoU阈值为0.7时，YOLOv8取得最佳AP结果，但相应的NMS耗时处于较高水平。考虑到YOLO检测器通常报告模型速度时排除NMS时间，因此需要建立端到端的速度基准。

### 3.2. 端到端速度基准测试

为了实现各类实时检测器端到端速度的公平比较，我们建立了一个端到端速度基准。考虑到非极大值抑制（NMS）的执行时间受输入影响，必须选定基准数据集并计算多张图像的平均执行时间。我们选用COCO val2017[20]作为基准数据集，并为YOLO系列检测器附加前述TensorRT的NMS后处理插件。具体而言，我们根据基准数据集上对应精度所采用的NMS阈值测试检测器的平均推理时间，排除I/O和内存复制操作。利用该基准，我们测试了基于锚框的检测器YOLOv5[11]和YOLOv7[38]，以及无锚框检测器PP-YOLOE[40]、YOLOv6[16]和YOLOv8[12]的端到端速度。

https://github.com/NVIDIA/TensorRT/tree/release/8.6/plugin/efficientNMSPlugin

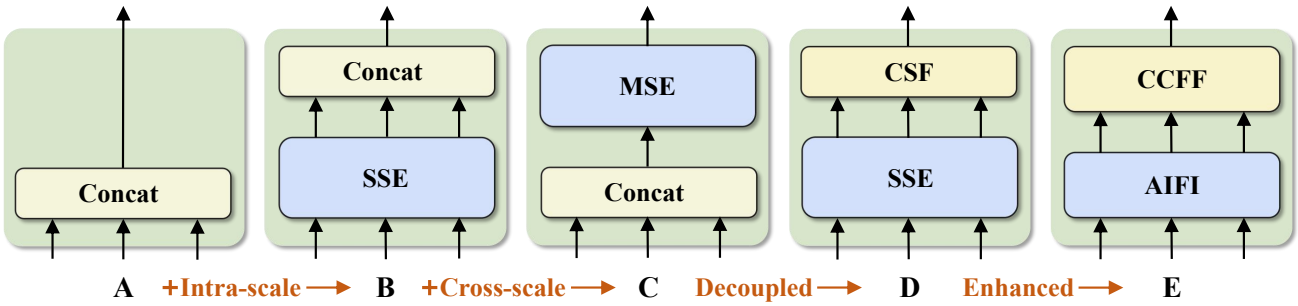



Figure 3. The encoder structure for each variant. **SSE** represents the single-scale Transformer encoder, **MSE** represents the multi-scale Transformer encoder, and **CSF** represents cross-scale fusion. **AIFI** and **CCFF** are the two modules designed into our hybrid encoder.

on T4 GPU with TensorRT FP16. According to the results (*cf. Table 2*), we conclude that *anchor-free detectors outperform anchor-based detectors with equivalent accuracy for YOLO detectors because the former require less NMS time than the latter*. The reason is that anchor-based detectors produce more prediction boxes than anchor-free detectors (three times more in our tested detectors).

# 4. The Real-time DETR

## 4.1. Model Overview

RT-DETR consists of a backbone, an efficient hybrid encoder, and a Transformer decoder with auxiliary prediction heads. The overview of RT-DETR is illustrated in Figure 4. Specifically, we feed the features from the last three stages of the backbone $\{\boldsymbol{\mathcal{S}}_3, \boldsymbol{\mathcal{S}}_4, \boldsymbol{\mathcal{S}}_5\}$ into the encoder. The efficient hybrid encoder transforms multi-scale features into a sequence of image features through intra-scale feature interaction and cross-scale feature fusion (*cf.* Sec. 4.2). Subsequently, the uncertainty-minimal query selection is employed to select a fixed number of encoder features to serve as initial object queries for the decoder (*cf.* Sec. 4.3). Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate categories and boxes.

## 4.2. Efficient Hybrid Encoder

**Computational bottleneck analysis.** The introduction of multi-scale features accelerates training convergence and improves performance [45]. However, although the deformable attention reduces the computational cost, the sharply increased sequence length still causes the encoder to become the computational bottleneck. As reported in Lin *et al*. [19], the encoder accounts for $49\%$ of the GFLOPs but contributes only $11\%$ of the AP in Deformable-DETR. To overcome this bottleneck, we first analyze the computational redundancy present in the multi-scale Transformer encoder. Intuitively, high-level features that contain rich semantic information about objects are extracted from low-level features, making it redundant to perform feature interaction on the concatenated multi-scale features. Therefore, we design a set of variants with different types of the encoder to prove that the simultaneous intra-scale and cross-scale feature interaction is inefficient, Figure 3. Specially, we use DINO-Deformable-R50 with the smaller size data reader and lighter decoder used in RT-DETR for experiments and first remove the multi-scale Transformer encoder in DINO-Deformable-R50 as variant A. Then, different types of the encoder are inserted to produce a series of variants based on A, elaborated as follows (Detailed indicators of each variant are referred to in Table 3):

- A $\rightarrow$ B: Variant B inserts a single-scale Transformer encoder into A, which uses one layer of Transformer block. The multi-scale features share the encoder for intra-scale feature interaction and then concatenate as output.
- B $\rightarrow$ C: Variant C introduces cross-scale feature fusion based on B and feeds the concatenated features into the multi-scale Transformer encoder to perform simultaneous intra-scale and cross-scale feature interaction.
- C $\rightarrow$ D: Variant D decouples intra-scale interaction and cross-scale fusion by utilizing the single-scale Transformer encoder for the former and a PANet-style [22] structure for the latter.
- D $\rightarrow$ E: Variant E enhances the intra-scale interaction and cross-scale fusion based on D, adopting an efficient hybrid encoder designed by us.

**Hybrid design.** Based on the above analysis, we rethink the structure of the encoder and propose an *efficient hybrid encoder*, consisting of two modules, namely the Attention-based Intra-scale Feature Interaction (AIFI) and the CNN-based Cross-scale Feature Fusion (CCFF). Specifically, AIFI further reduces the computational cost based on variant D by performing the intra-scale interaction only on $\boldsymbol{\mathcal{S}}_5$ with the single-scale Transformer encoder. The reason is that applying the self-attention operation to high-level features with richer semantic concepts captures the connection between conceptual entities, which facilitates the localization and recognition of objects by subsequent modules. However, the intra-scale interactions of lower-level features are unnecessary due to the lack of semantic concepts and the risk of duplication and confusion with high-level feature interactions. To verify this opinion, we perform the intra-scale interaction only on $\boldsymbol{\mathcal{S}}_5$ in variant D, and the experimental results are reported in Table 3 (see row $\text{D}_{\boldsymbol{\mathcal{S}}_5}$). Compared to

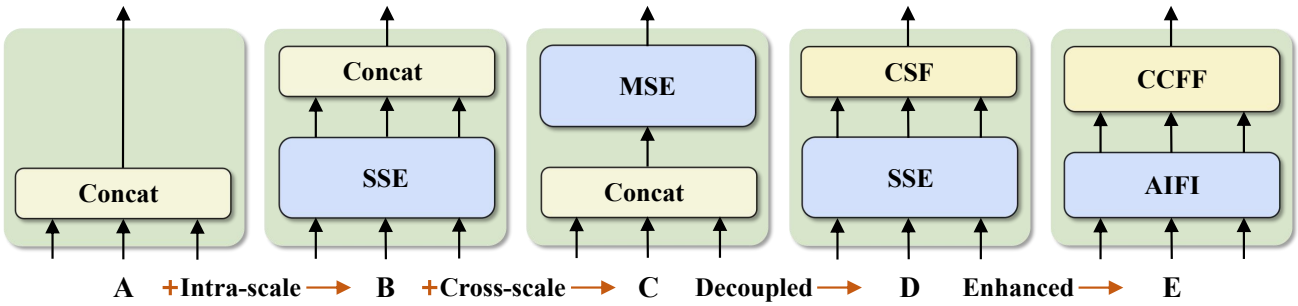



图3. 各变体的编码器结构。SSE代表单尺度Transformer编码器，MSE代表多尺度Transformer编码器，CSF代表跨尺度融合。AIFI与CCFF是我们为混合编码器设计的两个模块。

在T4 GPU上使用TensorRT FP16。根据结果(*cf. Table 2*)，我们得出结论*anchor-free detectors outperform anchor-based detectors with equivalent accuracy for YOLO detectors because the former require less NMS time than the latter*。原因在于基于锚点的检测器比无锚点检测器产生更多的预测框（在我们测试的检测器中多出三倍）。

## 4. 实时DETR

### 4.1. 模型概述

RT-DETR由主干网络、高效混合编码器以及带有辅助预测头的Transformer解码器构成。其整体架构如图4所示。具体而言，我们将主干网络最后三个阶段的特征$\{\boldsymbol{\mathcal{S}}_3, \boldsymbol{\mathcal{S}}_4, \boldsymbol{\mathcal{S}}_5\}$输入编码器。该高效混合编码器通过尺度内特征交互与跨尺度特征融合（*cf.*第4.2节），将多尺度特征转化为图像特征序列。随后采用不确定性最小化查询选择机制，从编码特征中筛选固定数量的特征作为解码器的初始目标查询（*cf.*第4.3节）。最终，配备辅助预测头的解码器通过迭代优化目标查询，生成目标类别与边界框信息。

### 4.2. 高效混合编码器

计算瓶颈分析。多尺度特征的引入加速了训练收敛并提升了性能[45]。然而，尽管可变形注意力机制降低了计算成本，但序列长度的急剧增加仍使编码器成为计算瓶颈。如Lin *et al*[19]所述，在Deformable-DETR中，编码器占据了49%的GFLOPs计算量，却仅贡献了11%的AP指标。为突破这一瓶颈，我们首先分析了多尺度Transformer编码器中存在的计算冗余。直观来看，包含丰富物体语义信息的高层特征是从低层特征提取而来，这使得在拼接的多尺度特征上进行特征交互存在冗余。因此，我们设计了一组采用不同类型编码器的变体模型，以证明同时...

同尺度内及跨尺度的特征交互效率低下，如图3所示。具体而言，我们采用DINO-Deformable-R50模型，配合RT-DETR中更小尺寸的数据读取器和更轻量级的解码器进行实验，首先移除了DINO-Deformable-R50中的多尺度Transformer编码器作为变体A。随后，基于变体A插入不同类型的编码器，生成一系列变体，具体如下（各变体的详细指标参见表3）：

- A → B：变体B在A中插入了一个单尺度Transformer编码器，该编码器使用一层Transformer块。多尺度特征共享该编码器以进行尺度内特征交互，然后拼接作为输出。
- B → C：变体C在B的基础上引入了跨尺度特征融合，并将拼接后的特征输入多尺度Transformer编码器，以实现尺度内与跨尺度的同步特征交互。
- C → D：变体D通过采用单尺度Transformer编码器处理尺度内交互，以及PANet风格[22]结构处理跨尺度融合，从而将两者解耦。
- D → E：变体E在D的基础上增强了尺度内交互与跨尺度融合，采用了我们设计的高效混合编码器。

混合设计。基于上述分析，我们重新思考了编码器的结构，提出了一种*efficient hybridencoder*，由两个模块组成，即基于注意力的同尺度特征交互（AIFI）和基于CNN的跨尺度特征融合（CCFF）。具体而言，AIFI在变体D的基础上进一步降低了计算成本，仅通过单尺度Transformer编码器对$\boldsymbol{\mathcal{S}}_5$进行同尺度交互。这样做的原因是，将自注意力操作应用于具有更丰富语义概念的高层特征，能够捕捉概念实体间的关联，这有助于后续模块对物体的定位与识别。然而，由于缺乏语义概念且存在与高层特征交互重复混淆的风险，对低层特征进行同尺度交互并无必要。为验证这一观点，我们仅在变体D中对$\boldsymbol{\mathcal{S}}_5$实施同尺度交互，实验结果如表3所示（见行D $\boldsymbol{\mathcal{S}}_5$）。相较于

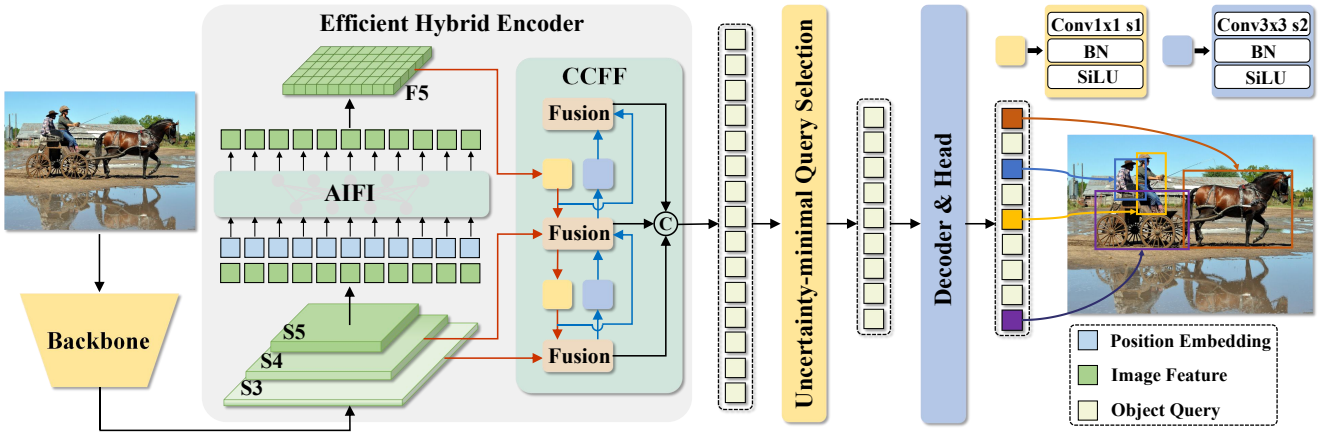



Figure 4. Overview of RT-DETR. We feed the features from the last three stages of the backbone into the encoder. The efficient hybrid encoder transforms multi-scale features into a sequence of image features through the Attention-based Intra-scale Feature Interaction (AIFI) and the CNN-based Cross-scale Feature Fusion (CCFF). Then, the uncertainty-minimal query selection selects a fixed number of encoder features to serve as initial object queries for the decoder. Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate categories and boxes.

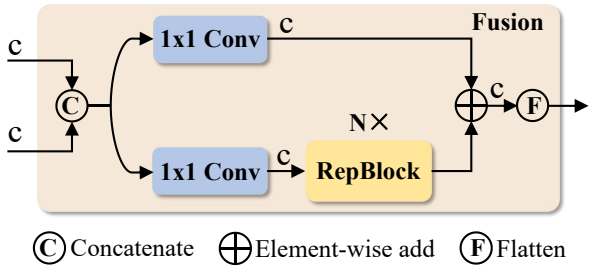



Figure 5. The fusion block in CCFF.

D, $D_{\mathcal{S}_5}$ not only significantly reduces latency (35% faster), but also improves accuracy (0.4% AP higher). CCFF is optimized based on the cross-scale fusion module, which inserts several fusion blocks consisting of convolutional layers into the fusion path. The role of the fusion block is to fuse two adjacent scale features into a new feature, and its structure is illustrated in Figure 5. The fusion block contains two $1 \times 1$ convolutions to adjust the number of channels, $N$ *RepBlocks* composed of RepConv [8] are used for feature fusion, and the two-path outputs are fused by element-wise add. We formulate the calculation of the hybrid encoder as:

$$\begin{aligned} \mathcal{Q} = \mathcal{K} = \mathcal{V} &= \texttt{Flatten}(\mathcal{S}_5), \\ \mathcal{F}_5 &= \texttt{Reshape}(\texttt{AIFI}(\mathcal{Q}, \mathcal{K}, \mathcal{V})), \\ \mathcal{O} &= \texttt{CCFF}(\{\mathcal{S}_3, \mathcal{S}_4, \mathcal{F}_5\}), \end{aligned} \tag{1}$$

where `Reshape` represents restoring the shape of the flattened feature to the same shape as $\mathcal{S}_5$.

### 4.3. Uncertainty-minimal Query Selection

To reduce the difficulty of optimizing object queries in DETR, several subsequent works [42, 44, 45] propose query selection schemes, which have in common that they use the confidence score to select the top $K$ features from the encoder to initialize object queries (or just position queries). The confidence score represents the likelihood that the feature includes foreground objects. Nevertheless, the detector are required to simultaneously model the category and location of objects, both of which determine the quality of the features. Hence, the performance score of the feature is a latent variable that is jointly correlated with both classification and localization. Based on the analysis, the current query selection lead to a considerable level of uncertainty in the selected features, resulting in sub-optimal initialization for the decoder and hindering the performance of the detector.

To address this problem, we propose the uncertainty minimal query selection scheme, which explicitly constructs and optimizes the epistemic uncertainty to model the joint latent variable of encoder features, thereby providing high-quality queries for the decoder. Specifically, the feature uncertainty $\mathcal{U}$ is defined as the discrepancy between the predicted distributions of localization $\mathcal{P}$ and classification $\mathcal{C}$ in Eq. (2). To minimize the uncertainty of the queries, we integrate the uncertainty into the loss function for the gradient-based optimization in Eq. (3).

$$\mathcal{U}(\hat{\boldsymbol{x}}) = \|\mathcal{P}(\hat{\boldsymbol{x}}) - \mathcal{C}(\hat{\boldsymbol{x}})\|, \hat{\boldsymbol{x}} \in \mathbb{R}^D \tag{2}$$

$$\mathcal{L}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{y}) = \mathcal{L}_{box}(\hat{\mathbf{b}}, \mathbf{b}) + \mathcal{L}_{cls}(\mathcal{U}(\hat{\boldsymbol{x}}), \hat{\mathbf{c}}, \mathbf{c}) \tag{3}$$

where $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ denote the prediction and ground truth, $\hat{\boldsymbol{y}} = \{\hat{\mathbf{c}}, \hat{\mathbf{b}}\}$, $\hat{\mathbf{c}}$ and $\hat{\mathbf{b}}$ represent the category and bounding box respectively, $\hat{\boldsymbol{x}}$ represent the encoder feature.

**Effectiveness analysis.** To analyze the effectiveness of the uncertainty-minimal query selection, we visualize the classification scores and IoU scores of the selected features on COCO `val2017`, Figure 6. We draw the scatterplot with classification scores greater than 0.5. The purple and green dots represent the selected features from the model trained with uncertainty-minimal query selection and vanilla query

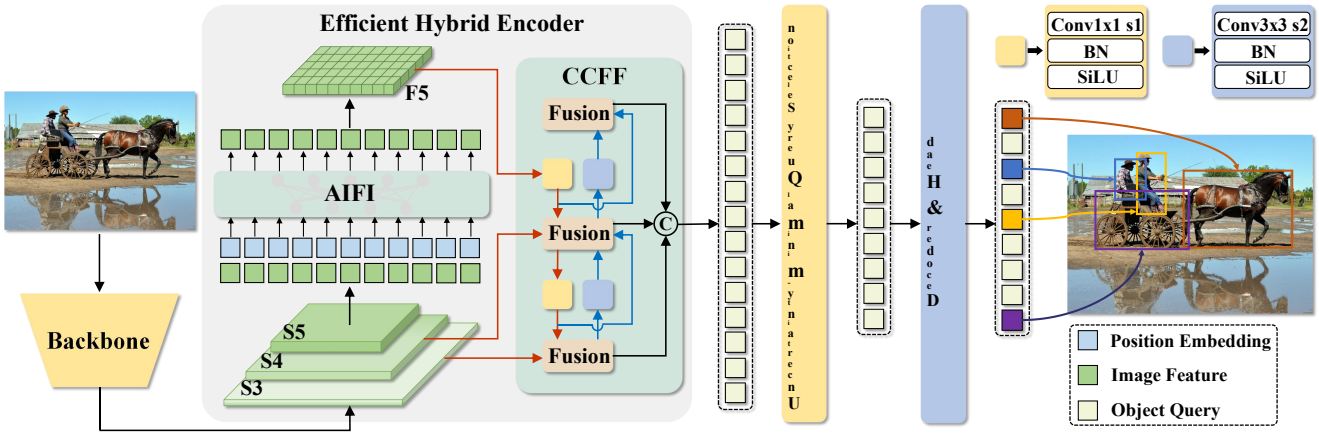



图4. RT-DETR整体架构。我们将主干网络最后三个阶段的特征输入编码器。高效混合编码器通过基于注意力的同尺度特征交互模块（AIFI）和基于CNN的跨尺度特征融合模块（CCFF），将多尺度特征转换为图像特征序列。随后，不确定性最小化查询选择机制筛选固定数量的编码器特征，作为解码器的初始对象查询。最终，配备辅助预测头的解码器通过迭代优化对象查询，生成目标类别与边界框信息。

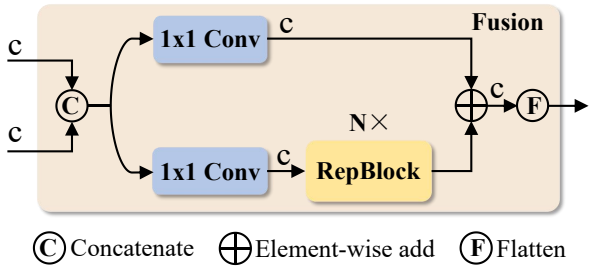



图5. CCFF中的融合模块。

D、$D_{\mathcal{S}_5}$不仅显著降低了延迟（提速35%），还提高了准确率（AP提升0.4%）。CCFF基于跨尺度融合模块进行优化，该模块在融合路径中插入了由卷积层组成的多个融合块。融合块的作用是将两个相邻尺度的特征融合为新的特征，其结构如图5所示。融合块包含两个1×1卷积用于调整通道数，$N$ *RepBlock*s由RepConv[8]构成以实现特征融合，并通过逐元素相加融合双路径输出。我们将混合编码器的计算表述为：

$$\begin{aligned} \mathcal{Q} = \mathcal{K} = \mathcal{V} &= \texttt{Flatten}(\mathcal{S}_5), \\ \mathcal{F}_5 &= \texttt{Reshape}(\texttt{AIFI}(\mathcal{Q}, \mathcal{K}, \mathcal{V})), \\ \mathcal{O} &= \texttt{CCFF}(\{\mathcal{S}_3, \mathcal{S}_4, \mathcal{F}_5\}), \end{aligned} \tag{1}$$

其中Reshape表示将展平后的特征恢复至与$\mathcal{S}_5$相同的形状。

### 4.3. 不确定性最小化的查询选择

为了降低DETR中优化目标查询的难度，后续多项研究[42,44,45]提出了查询选择方案，其共同点在于利用置信度分数从编码器中选取前$K$个特征来初始化目标查询（或仅位置查询）。

置信度分数表示该特征包含前景物体的可能性。然而，检测器需要同时建模物体的类别和位置，这两者共同决定了特征的质量。因此，该特征的性能评分是一个潜在变量，与分类和定位均存在联合相关性。基于此分析，当前查询选择会导致所选特征存在相当程度的不确定性，从而为解码器提供次优的初始化条件，并阻碍检测器的性能提升。

为解决这一问题，我们提出了不确定性最小化查询选择方案，该方案显式构建并优化认知不确定性以建模编码器特征的联合潜在变量，从而为解码器提供高质量查询。具体而言，特征不确定性$\mathcal{U}$被定义为定位$\mathcal{P}$与分类$\mathcal{C}$预测分布在式(2)中的差异。为最小化查询的不确定性，我们将该不确定性整合至式(3)基于梯度的优化损失函数中。

$$\mathcal{U}(\hat{\boldsymbol{x}}) = \|\mathcal{P}(\hat{\boldsymbol{x}}) - \mathcal{C}(\hat{\boldsymbol{x}})\|, \hat{\boldsymbol{x}} \in \mathbb{R}^D \tag{2}$$

$$\mathcal{L}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{y}) = \mathcal{L}_{box}(\hat{\mathbf{b}}, \mathbf{b}) + \mathcal{L}_{cls}(\mathcal{U}(\hat{\boldsymbol{x}}), \hat{\mathbf{c}}, \mathbf{c}) \tag{3}$$

其中$\hat{\boldsymbol{y}}$和$\boldsymbol{y}$分别表示预测值和真实值，$\hat{\boldsymbol{y}} = \{\hat{\mathbf{c}}, \hat{\mathbf{b}}\}$、$\hat{\mathbf{c}}$和$\hat{\mathbf{b}}$代表类别与边界框，$\hat{\boldsymbol{x}}$则指代编码器特征。

效果分析。为了评估不确定性最小化查询选择的有效性，我们在COCO val2017数据集上对所选特征的分类得分与IoU得分进行了可视化展示，如图6所示。我们绘制了分类得分大于0{v*}5的散点图。其中紫色与绿色圆点分别代表采用不确定性最小化查询选择策略训练的模型所选特征，以及基准查询方法所选特征。

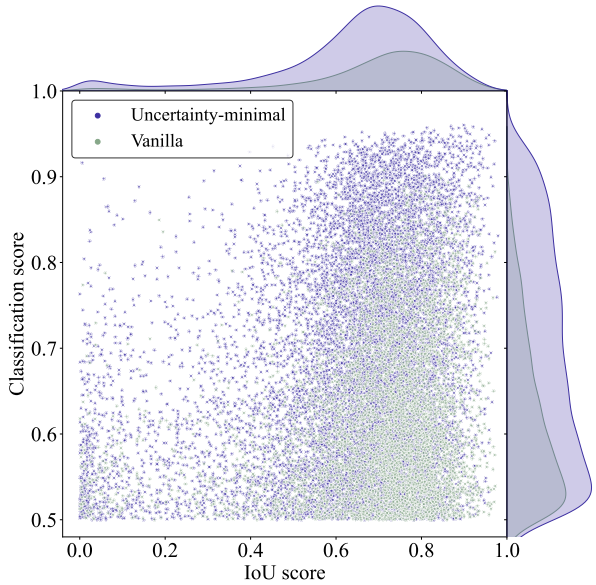



Figure 6. Classification and IoU scores of the selected encoder features. Purple and Green dots represent the selected features from model trained with uncertainty-minimal query selection and vanilla query selection, respectively.

selection, respectively. The closer the dot is to the top right of the figure, the higher the quality of the corresponding feature, *i.e.*, the more likely the predicted category and box are to describe the true object. The top and right density curves reflect the number of dots for two types.

The most striking feature of the scatterplot is that the purple dots are concentrated in the top right of the figure, while the green dots are concentrated in the bottom right. This shows that uncertainty-minimal query selection produces more high-quality encoder features. Furthermore, we perform quantitative analysis on two query selection schemes. There are $138\%$ more purple dots than green dots, *i.e.*, more green dots with a classification score less than or equal to $0.5$, which can be considered low-quality features. And there are $120\%$ more purple dots than green dots with both scores greater than $0.5$. The same conclusion can be drawn from the density curves, where the gap between purple and green is most evident in the top right of the figure. Quantitative results further demonstrate that the uncertainty-minimal query selection provides more features with accurate classification and precise location for queries, thereby improving the accuracy of the detector (*cf.* Sec. 5.3).

### 4.4. Scaled RT-DETR

Since real-time detectors typically provide models at different scales to accommodate different scenarios, RT-DETR also supports flexible scaling. Specifically, for the hybrid encoder, we control the width by adjusting the embedding dimension and the number of channels, and the depth by adjusting the number of Transformer layers and *RepBlock*s. The width and depth of the decoder can be controlled by manipulating the number of object queries and decoder layers. Furthermore, the speed of RT-DETR supports flexible adjustment by adjusting the number of decoder layers. We observe that removing a few decoder layers at the end has minimal effect on accuracy, but greatly enhances inference speed (*cf.* Sec. 5.4). We compare the RT-DETR equipped with ResNet50 and ResNet101 [13, 14] to the $L$ and $X$ models of YOLO detectors. Lighter RT-DETRs can be designed by applying other smaller (*e.g.*, ResNet18/34) or scalable (*e.g.*, CSPResNet [40]) backbones with scaled encoder and decoder. We compare the scaled RT-DETRs with the lighter ($S$ and $M$) YOLO detectors in Appendix, which outperform all $S$ and $M$ models in both speed and accuracy.

## 5. Experiments

### 5.1. Comparison with SOTA

Table 2 compares RT-DETR with current real-time (YOLOs) and end-to-end (DETRs) detectors, where only the $L$ and $X$ models of the YOLO detector are compared, and the $S$ and $M$ models are compared in Appendix. Our RT-DETR and YOLO detectors share a common input size of (640, 640), and other DETRs use an input size of (800, 1333). The FPS is reported on T4 GPU with TensorRT FP16, and for YOLO detectors using official pre-trained models according to the end-to-end speed benchmark proposed in Sec. 3.2. Our RT-DETR-R50 achieves $53.1\%$ AP and 108 FPS, while RT-DETR-R101 achieves $54.3\%$ AP and 74 FPS, outperforming state-of-the-art YOLO detectors of similar scale and DETRs with the same backbone in both speed and accuracy. The experimental settings are shown in Appendix.

**Comparison with real-time detectors.** We compare the end-to-end speed (*cf.* Sec. 3.2) and accuracy of RT-DETR with YOLO detectors. We compare RT-DETR with YOLOv5 [11], PP-YOLOE [40], YOLOv6v3.0 [16] (hereinafter referred to as YOLOv6), YOLOv7 [38] and YOLOv8 [12]. Compared to YOLOv5-L / PP-YOLOE-L / YOLOv6-L, RT-DETR-R50 improves accuracy by $4.1\%$ / $1.7\%$ / $0.3\%$ AP, increases FPS by $100.0\%$ / $14.9\%$ / $9.1\%$, and reduces the number of parameters by $8.7\%$ / $19.2\%$ / $28.8\%$. Compared to YOLOv5-X / PP-YOLOE-X, RT-DETR-R101 improves accuracy by $3.6\%$ / $2.0\%$, increases FPS by $72.1\%$ / $23.3\%$, and reduces the number of parameters by $11.6\%$ / $22.4\%$. Compared to YOLOv7-L / YOLOv8-L, RT-DETR-R50 improves accuracy by $1.9\%$ / $0.2\%$ AP and increases FPS by $96.4\%$ / $52.1\%$. Compared to YOLOv7-X / YOLOv8-X, RT-DETR-R101 improves accuracy by $1.4\%$ / $0.4\%$ AP and increases FPS by $64.4\%$ / $48.0\%$. This shows that our RT-DETR achieves state-of-the-art real-time detection performance.

**Comparison with end-to-end detectors.** We also compare RT-DETR with existing DETRs using the same backbone.

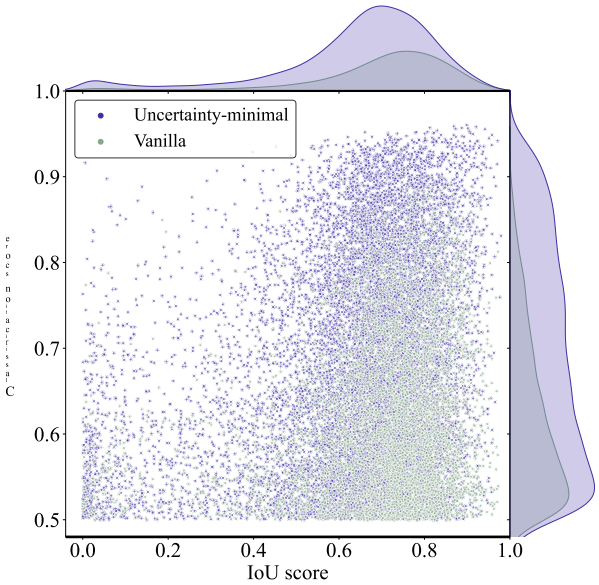



图6. 所选编码器特征的分类与IoU得分。紫色与绿色圆点分别代表采用不确定性最小化查询选择方法和普通查询选择方法训练模型所选取的特征。

选择，分别。图中的点越靠近右上角，对应特征*i.e*的质量越高，预测的类别和边界框越有可能描述真实物体。顶部和右侧的密度曲线反映了两种类型点的数量分布。

散点图最显著的特征在于，紫色点群密集分布于图形右上区域，而绿色点群则集中于右下区域。这表明不确定性最小化的查询选择策略能生成更高质量的编码器特征。我们进一步对两种查询选择方案进行了定量分析：紫色点数量比绿色点多出138%*i.e*；分类分数小于等于0.5的低质量特征中绿色点占比更高；而在两项分数均超过0.5的高质量特征中，紫色点数量比绿色点多出120%。密度曲线同样印证了这一结论——图形右上区域的紫绿点群差异最为明显。定量结果进一步证明，不确定性最小化查询选择能为检测器提供更多具有准确分类和精确定位的查询特征，从而提升检测精度（*cf.*第5.3节）。

### 4.4. 缩放版RT-DETR

由于实时检测器通常提供不同规模的模型以适应不同场景，RT-DETR同样支持灵活的尺度调整。具体而言，对于混合编码器，我们通过调整嵌入维度和通道数来控制宽度，通过调整Transformer层数和{v*}的数量来控制深度。

解码器的宽度和深度可通过调整目标查询数量和解码器层数来控制。此外，RT-DETR的速度支持通过调节解码器层数实现灵活调整。我们观察到，移除末端少量解码层对精度影响甚微，却能显著提升推理速度（*cf.*第5.4节）。我们将配备ResNet50和ResNet101[13,14]的RT-DETR与YOLO检测器的$L$和$X$模型进行对比。通过采用更小规模（*e.g*如ResNet18/34）或可扩展（*e.g*如CSPResNet[40]）的主干网络，并配合缩放编码器与解码器，可设计出更轻量化的RT-DETR。在附录中，我们将缩放后的RT-DETR与轻量级（$S$和$M$）YOLO检测器进行对比，其在速度和精度上均优于所有$S$和$M$模型。

## 5. 实验

### 5.1. 与SOTA的对比

表2将RT-DETR与当前实时（YOLOs）和端到端（DETRs）检测器进行对比，其中仅比较了YOLO检测器的$L$和$X$模型，$S$和$M$模型则在附录中进行对比。我们的RT-DETR与YOLO检测器共享相同的输入尺寸（640, 640），其他DETRs则采用（800, 1333）的输入尺寸。FPS数据基于T4 GPU搭配TensorRT FP16测得，对于YOLO检测器，则根据第3.2节提出的端到端速度基准测试，使用官方预训练模型得出。我们的RT-DETR-R50实现了53.1% AP和108 FPS，而RT-DETR-R101则达到54.3% AP和74 FPS，在速度和精度上均超越了同规模的最先进YOLO检测器及采用相同骨干网络的DETRs。具体实验设置详见附录。

与实时检测器的对比。我们将RT-DETR的端到端速度（*cf.*第3.2节）和准确性与YOLO系列检测器进行比较，包括YOLOv5[11]、PP-YOLOE[40]、YOLOv6v3.0[16]（以下简称YOLOv6）、YOLOv7[38]和YOLOv8[12]。相较于YOLOv5-L/PP-YOLOE-L/YOLOv6-L，RT-DETR-R50在AP准确率上分别提升4.1%/1.7%/0.3%，FPS提高100.0%/14.9%/9.1%，参数量减少8.7%/19.2%/28.8%。对比YOLOv5-X/PP-YOLOE-X，RT-DETR-R101的AP提升3.6%/2.0%，FPS增加72.1%/23.3%，参数量降低11.6%/22.4%。与YOLOv7-L/YOLOv8-L相比，RT-DETR-R50的AP提高1.9%/0.2%，FPS提升96.4%/52.1%。针对YOLOv7-X/YOLOv8-X，RT-DETR-R101的AP增益为1.4%/0.4%，FPS增速达64.4%/48.0%。实验表明，我们的RT-DETR实现了业界领先的实时检测性能。

与端到端检测器的比较。我们还将RT-DETR与使用相同骨干网络的现有DETR模型进行了对比。

| Model | Backbone | #Epochs | #Params (M) | GFLOPs | $FPS_{bs=1}$ | $AP^{val}$ | $AP^{val}_{50}$ | $AP^{val}_{75}$ | $AP^{val}_{S}$ | $AP^{val}_{M}$ | $AP^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Real-time Object Detectors* | | | | | | | | | | | |
| YOLOv5-L [11] | - | 300 | 46 | 109 | 54 | 49.0 | 67.3 | - | - | - | - |
| YOLOv5-X [11] | - | 300 | 86 | 205 | 43 | 50.7 | 68.9 | - | - | - | - |
| PPYOLOE-L [40] | - | 300 | 52 | 110 | 94 | 51.4 | 68.9 | 55.6 | 31.4 | 55.3 | 66.1 |
| PPYOLOE-X [40] | - | 300 | 98 | 206 | 60 | 52.3 | 69.9 | 56.5 | 33.3 | 56.3 | 66.4 |
| YOLOv6-L [16] | - | 300 | 59 | 150 | 99 | 52.8 | 70.3 | 57.7 | 34.4 | 58.1 | 70.1 |
| YOLOv7-L [38] | - | 300 | 36 | 104 | 55 | 51.2 | 69.7 | 55.5 | 35.2 | 55.9 | 66.7 |
| YOLOv7-X [38] | - | 300 | 71 | 189 | 45 | 52.9 | 71.1 | 57.4 | 36.9 | 57.7 | 68.6 |
| YOLOv8-L [12] | - | - | 43 | 165 | 71 | 52.9 | 69.8 | 57.5 | 35.3 | 58.3 | 69.8 |
| YOLOv8-X [12] | - | - | 68 | 257 | 50 | 53.9 | 71.0 | 58.7 | 35.7 | 59.3 | 70.7 |
| *End-to-end Object Detectors* | | | | | | | | | | | |
| DETR-DC5 [4] | R50 | 500 | 41 | 187 | - | 43.3 | 63.1 | 45.9 | 22.5 | 47.3 | 61.1 |
| DETR-DC5 [4] | R101 | 500 | 60 | 253 | - | 44.9 | 64.7 | 47.7 | 23.7 | 49.5 | 62.3 |
| Anchor-DETR-DC5 [39] | R50 | 50 | 39 | 172 | - | 44.2 | 64.7 | 47.5 | 24.7 | 48.2 | 60.6 |
| Anchor-DETR-DC5 [39] | R101 | 50 | - | - | - | 45.1 | 65.7 | 48.8 | 25.8 | 49.4 | 61.6 |
| Conditional-DETR-DC5 [27] | R50 | 108 | 44 | 195 | - | 45.1 | 65.4 | 48.5 | 25.3 | 49.0 | 62.2 |
| Conditional-DETR-DC5 [27] | R101 | 108 | 63 | 262 | - | 45.9 | 66.8 | 49.5 | 27.2 | 50.3 | 63.3 |
| Efficient-DETR [42] | R50 | 36 | 35 | 210 | - | 45.1 | 63.1 | 49.1 | 28.3 | 48.4 | 59.0 |
| Efficient-DETR [42] | R101 | 36 | 54 | 289 | - | 45.7 | 64.1 | 49.5 | 28.2 | 49.1 | 60.2 |
| SMCA-DETR [9] | R50 | 108 | 40 | 152 | - | 45.6 | 65.5 | 49.1 | 25.9 | 49.3 | 62.6 |
| SMCA-DETR [9] | R101 | 108 | 58 | 218 | - | 46.3 | 66.6 | 50.2 | 27.2 | 50.5 | 63.2 |
| Deformable-DETR [45] | R50 | 50 | 40 | 173 | - | 46.2 | 65.2 | 50.0 | 28.8 | 49.2 | 61.7 |
| DAB-Deformable-DETR [23] | R50 | 50 | 48 | 195 | - | 46.9 | 66.0 | 50.8 | 30.1 | 50.4 | 62.5 |
| DAB-Deformable-DETR++ [23] | R50 | 50 | 47 | - | - | 48.7 | 67.2 | 53.0 | 31.4 | 51.6 | 63.9 |
| DN-Deformable-DETR [17] | R50 | 50 | 48 | 195 | - | 48.6 | 67.4 | 52.7 | 31.0 | 52.0 | 63.7 |
| DN-Deformable-DETR++ [17] | R50 | 50 | 47 | - | - | 49.5 | 67.6 | 53.8 | 31.3 | 52.6 | 65.4 |
| DINO-Deformable-DETR [44] | R50 | 36 | 47 | 279 | 5 | 50.9 | 69.0 | 55.3 | 34.6 | 54.1 | 64.6 |
| *Real-time End-to-end Object Detector (ours)* | | | | | | | | | | | |
| RT-DETR | R50 | 72 | 42 | 136 | **108** | **53.1** | **71.3** | **57.7** | 34.8 | 58.0 | 70.0 |
| RT-DETR | R101 | 72 | 76 | 259 | **74** | **54.3** | **72.7** | 58.6 | 36.0 | 58.8 | **72.1** |

Table 2. Comparison with SOTA (only $L$ and $X$ models of YOLO detectors, see Appendix for the comparison with $S$ and $M$ models). We do not test the speed of other DETRs, except for DINO-Deformable-DETR [44] for comparison, as they are not real-time detectors. Our RT-DETR outperforms the state-of-the-art YOLO detectors and DETRs in both speed and accuracy.

We test the speed of DINO-Deformable-DETR [44] according to the settings of the corresponding accuracy taken on COCO `val2017` for comparison, *i.e.*, the speed is tested with TensorRT FP16 and the input size is (800, 1333). Table 2 shows that RT-DETR outperforms all DETRs with the same backbone in both speed and accuracy. Compared to DINO-Deformable-DETR-R50, RT-DETR-R50 improves the accuracy by 2.2% AP and the speed by 21 times (108 FPS vs 5 FPS), both of which are significantly improved.

### 5.2. Ablation Study on Hybrid Encoder

We evaluate the indicators of the variants designed in Sec. 4.2, including AP (trained with $1\times$ configuration), the number of parameters, and the latency, Table 3. Compared to baseline A, variant B improves accuracy by 1.9% AP and increases the latency by 54%. This proves that the intra-scale feature interaction is significant, but the single-scale Transformer encoder is computationally expensive. Variant C delivers a 0.7% AP improvement over B and increases the latency by 20%. This shows that the cross-scale feature fusion is also necessary but the multi-scale Transformer encoder requires higher computational cost. Variant D delivers a 0.8% AP improvement over C, but reduces latency by 8%, suggesting that decoupling intra-scale interaction and cross-scale fusion not only reduces computational cost but also improves accuracy. Compared to variant D, $D_{\mathcal{S}_5}$ reduces the latency by 35% but delivers 0.4% AP improvement, demonstrating that intra-scale interactions of lower-level features are not required. Finally, variant E delivers 1.5% AP improvement over D. Despite a 20% increase in the number of parameters, the latency is reduced by 24%, making the encoder more efficient. This shows that our hybrid encoder achieves a better trade-off between speed and accuracy.

### 5.3. Ablation Study on Query Selection

We conduct an ablation study on uncertainty-minimal query selection, and the results are reported on RT-DETR-R50 with $1\times$ configuration, Table 4. The query selection in RT-DETR selects the top $K$ ($K = 300$) encoder features according to the classification scores as the content queries, and the prediction boxes corresponding to the selected features are used as initial position queries. We compare the encoder features selected by the two query selection schemes on COCO `val2017` and calculate the proportions of classi-

| Model | Backbone | #Epochs | #Params (M) | GFLOPs | $FPS_{bs=1}$ | $AP^{val}$ | $AP^{val}_{50}$ | $AP^{val}_{75}$ | $AP^{val}_{S}$ | $AP^{val}_{M}$ | $AP^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Real-time Object Detectors* | | | | | | | | | | | |
| YOLOv5-L [11] | - | 300 | 46 | 109 | 54 | 49.0 | 67.3 | - | - | - | - |
| YOLOv5-X [11] | - | 300 | 86 | 205 | 43 | 50.7 | 68.9 | - | - | - | - |
| PPYOLOE-L [40] | - | 300 | 52 | 110 | 94 | 51.4 | 68.9 | 55.6 | 31.4 | 55.3 | 66.1 |
| PPYOLOE-X [40] | - | 300 | 98 | 206 | 60 | 52.3 | 69.9 | 56.5 | 33.3 | 56.3 | 66.4 |
| YOLOv6-L [16] | - | 300 | 59 | 150 | 99 | 52.8 | 70.3 | 57.7 | 34.4 | 58.1 | 70.1 |
| YOLOv7-L [38] | - | 300 | 36 | 104 | 55 | 51.2 | 69.7 | 55.5 | 35.2 | 55.9 | 66.7 |
| YOLOv7-X [38] | - | 300 | 71 | 189 | 45 | 52.9 | 71.1 | 57.4 | 36.9 | 57.7 | 68.6 |
| YOLOv8-L [12] | - | - | 43 | 165 | 71 | 52.9 | 69.8 | 57.5 | 35.3 | 58.3 | 69.8 |
| YOLOv8-X [12] | - | - | 68 | 257 | 50 | 53.9 | 71.0 | 58.7 | 35.7 | 59.3 | 70.7 |
| *End-to-end Object Detectors* | | | | | | | | | | | |
| DETR-DC5 [4] | R50 | 500 | 41 | 187 | - | 43.3 | 63.1 | 45.9 | 22.5 | 47.3 | 61.1 |
| DETR-DC5 [4] | R101 | 500 | 60 | 253 | - | 44.9 | 64.7 | 47.7 | 23.7 | 49.5 | 62.3 |
| Anchor-DETR-DC5 [39] | R50 | 50 | 39 | 172 | - | 44.2 | 64.7 | 47.5 | 24.7 | 48.2 | 60.6 |
| Anchor-DETR-DC5 [39] | R101 | 50 | - | - | - | 45.1 | 65.7 | 48.8 | 25.8 | 49.4 | 61.6 |
| Conditional-DETR-DC5 [27] | R50 | 108 | 44 | 195 | - | 45.1 | 65.4 | 48.5 | 25.3 | 49.0 | 62.2 |
| Conditional-DETR-DC5 [27] | R101 | 108 | 63 | 262 | - | 45.9 | 66.8 | 49.5 | 27.2 | 50.3 | 63.3 |
| Efficient-DETR [42] | R50 | 36 | 35 | 210 | - | 45.1 | 63.1 | 49.1 | 28.3 | 48.4 | 59.0 |
| Efficient-DETR [42] | R101 | 36 | 54 | 289 | - | 45.7 | 64.1 | 49.5 | 28.2 | 49.1 | 60.2 |
| SMCA-DETR [9] | R50 | 108 | 40 | 152 | - | 45.6 | 65.5 | 49.1 | 25.9 | 49.3 | 62.6 |
| SMCA-DETR [9] | R101 | 108 | 58 | 218 | - | 46.3 | 66.6 | 50.2 | 27.2 | 50.5 | 63.2 |
| Deformable-DETR [45] | R50 | 50 | 40 | 173 | - | 46.2 | 65.2 | 50.0 | 28.8 | 49.2 | 61.7 |
| DAB-Deformable-DETR [23] | R50 | 50 | 48 | 195 | - | 46.9 | 66.0 | 50.8 | 30.1 | 50.4 | 62.5 |
| DAB-Deformable-DETR++ [23] | R50 | 50 | 47 | - | - | 48.7 | 67.2 | 53.0 | 31.4 | 51.6 | 63.9 |
| DN-Deformable-DETR [17] | R50 | 50 | 48 | 195 | - | 48.6 | 67.4 | 52.7 | 31.0 | 52.0 | 63.7 |
| DN-Deformable-DETR++ [17] | R50 | 50 | 47 | - | - | 49.5 | 67.6 | 53.8 | 31.3 | 52.6 | 65.4 |
| DINO-Deformable-DETR [44] | R50 | 36 | 47 | 279 | 5 | 50.9 | 69.0 | 55.3 | 34.6 | 54.1 | 64.6 |
| *Real-time End-to-end Object Detector (ours)* | | | | | | | | | | | |
| RT-DETR | R50 | 72 | 42 | 136 | **108** | **53.1** | **71.3** | **57.7** | 34.8 | 58.0 | 70.0 |
| RT-DETR | R101 | 72 | 76 | 259 | **74** | **54.3** | **72.7** | 58.6 | 36.0 | 58.8 | **72.1** |

表2. 与SOTA的对比（仅包含YOLO检测器的*L*和*X*版本，与*S*及*M*版本的对比请参阅附录）。由于其他DETR并非实时检测器，我们仅测试了DINO-Deformable-DETR[44]的速度以作比较。我们的RT-DETR在速度和精度上均超越了最先进的YOLO检测器及DETR系列。

我们按照在COCO val2017上取得的相应精度设置测试了DINO-Deformable-DETR[44]的速度以进行对比，*i.e*。速度测试采用TensorRT FP16，输入尺寸为(800, 1333)。表2显示，RT-DETR在相同骨干网络下，速度和精度均优于所有DETR模型。与DINO-Deformable-DETR-R50相比，RT-DETR-R50的精度提升了2.2% AP，速度提高了21倍（108 FPS对比5 FPS），两者均有显著提升。

### 5.2. 混合编码器的消融研究

我们对第4.2节设计的变体指标进行了评估，包括AP（采用1×配置训练）、参数量及延迟，如表3所示。相较于基线A，变体B将准确率提升了1.9% AP，但延迟增加了54%。这证明尺度内特征交互作用显著，但单尺度Transformer编码器的计算代价较高。变体C在B的基础上进一步实现了0.7% AP的提升，同时延迟增加20%。这表明跨尺度特征融合同样必要，但多尺度Transformer编码器需要更高的计算成本。变体D则相比C方案，0.8%的平均精度（AP）提升，同时延迟降低了8%，这表明解耦尺度内交互与跨尺度融合不仅能降低计算成本，还能提高精度。与变体D相比，$D_{\boldsymbol{S}_5}$将延迟降低了35%，同时实现了0.4%的AP提升，证明低层级特征的尺度内交互并非必要。最终，变体E相较D方案取得了1.5%的AP提升。尽管参数量增加了20%，延迟却减少了24%，使编码器效率更高。这表明我们的混合编码器在速度与精度之间实现了更优的平衡。

### 5.3. 查询选择的消融研究

我们对不确定性最小化的查询选择进行了消融研究，结果基于RT-DETR-R50模型的1×配置在表4中呈现。RT-DETR的查询选择机制依据分类分数选取前$K$（$K$ =300）个编码器特征作为内容查询，并将所选特征对应的预测框作为初始位置查询。我们在COCO val2017数据集上对比了两种查询选择方案所筛选的编码器特征，并统计了分类-

| Variant | AP (%) | #Params (M) | Latency (ms) |
|---|---|---|---|
| A | 43.0 | 31 | 7.2 |
| B | 44.9 | 32 | 11.1 |
| C | 45.6 | 32 | 13.3 |
| D | 46.4 | 35 | 12.2 |
| $D_{\mathcal{S}_5}$ | 46.8 | 35 | 7.9 |
| E | 47.9 | 42 | 9.3 |

Table 3. The indicators of the set of variants illustrated in Figure 3.

| Query selection | AP (%) | $\textbf{Prop}_{cls}\uparrow$ (%) | $\textbf{Prop}_{both}\uparrow$ (%) |
|---|---|---|---|
| **Vanilla** | 47.9 | 0.35 | 0.30 |
| **Uncertainty-minimal** | 48.7 | 0.82 | 0.67 |

Table 4. Results of the ablation study on uncertainty-minimal query selection. $\textbf{Prop}_{cls}$ and $\textbf{Prop}_{both}$ represent the proportion of classification score and both scores greater than 0.5 respectively.

fication scores greater than 0.5 and both classification and IoU scores greater than 0.5, respectively. The results show that the encoder features selected by uncertainty-minimal query selection not only increase the proportion of high classification scores (0.82% vs 0.35%) but also provide more high-quality features (0.67% vs 0.30%). We also evaluate the accuracy of the detectors trained with the two query selection schemes on COCO `val2017`, where the uncertainty-minimal query selection achieves an improvement of 0.8% AP (48.7% AP vs 47.9% AP).

### 5.4. Ablation Study on Decoder

Table 5 shows the inference latency and accuracy of each decoder layer of RT-DETR-R50 trained with different numbers of decoder layers. When the number of decoder layers is set to 6, the RT-DETR-R50 achieves the best accuracy 53.1% AP. Furthermore, we observe that the difference in accuracy between adjacent decoder layers gradually decreases as the index of the decoder layer increases. Taking the column RT-DETR-R50-$\text{Det}^6$ as an example, using 5-th decoder layer for inference only loses 0.1% AP (53.1% AP vs 53.0% AP) in accuracy, while reducing latency by 0.5 ms (9.3 ms vs 8.8 ms). Therefore, RT-DETR supports flexible speed tuning by adjusting the number of decoder layers without retraining, thus improving its practicality.

## 6. Limitation and Discussion

**Limitation.** Although the proposed RT-DETR outperforms the state-of-the-art real-time detectors and end-to-end detectors with similar size in both speed and accuracy, it shares the same limitation as the other DETRs, *i.e.*, the performance on small objects is still inferior than the strong real-time detectors. According to Table 2, RT-DETR-R50 is 0.5% AP lower than the highest $\text{AP}_S^{val}$ in the $L$ model (YOLOv8-L) and RT-DETR-R101 is 0.9% AP lower than the highest $\text{AP}_S^{val}$ in the $X$ model (YOLOv7-X). We hope that this problem will be addressed in future work.

| ID | AP(%) | | | | Latency (ms) |
|---|---|---|---|---|---|
| | $\text{Det}^4$ | $\text{Det}^5$ | $\text{Det}^6$ | $\text{Det}^7$ | |
| 7 | - | - | - | 52.6 | 9.6 |
| 6 | - | - | **53.1** | 52.6 | 9.3 |
| 5 | - | 52.9 | 53.0 | 52.5 | 8.8 |
| 4 | 52.7 | 52.7 | 52.7 | 52.1 | 8.3 |
| 3 | 52.4 | 52.3 | 52.4 | 51.5 | 7.9 |
| 2 | 51.6 | 51.3 | 51.3 | 50.6 | 7.5 |
| 1 | 49.6 | 48.8 | 49.1 | 48.3 | 7.0 |

Table 5. Results of the ablation study on decoder. **ID** indicates decoder layer index. $\textbf{Det}^k$ represents detector with $k$ decoder layers. All results are reported on RT-DETR-R50 with $6\times$ configuration.

**Discussion.** Existing large DETR models [3, 6, 32, 41, 44, 46] have demonstrated impressive performance on COCO `test-dev` [20] leaderboard. The proposed RT-DETR at different scales preserves decoders homogeneous to other DETRs, which makes it possible to distill our lightweight detector with high accuracy pre-trained large DETR models. We believe that this is one of the advantages of RT-DETR over other real-time detectors and could be an interesting direction for future exploration.

## 7. Conclusion

In this work, we propose a real-time end-to-end detector, called RT-DETR, which successfully extends DETR to the real-time detection scenario and achieves state-of-the-art performance. RT-DETR includes two key enhancements: an efficient hybrid encoder that expeditiously processes multi-scale features, and the uncertainty-minimal query selection that improves the quality of initial object queries. Furthermore, RT-DETR supports flexible speed tuning without retraining and eliminates the inconvenience caused by two NMS thresholds, facilitating its practical application. RT-DETR, along with its model scaling strategy, broadens the technical approach to real-time object detection, offering new possibilities beyond YOLO for diverse real-time scenarios. We hope that RT-DETR can be put into practice.

**Acknowledgements.** This work was supported in part by the National Key R&D Program of China (No. 2022ZD0118201), Natural Science Foundation of China (No. 61972217, 32071459, 62176249, 62006133, 62271465), and the Shenzhen Medical Research Funds in China (No. B2302037). Thanks to Chang Liu, Zhennan Wang and Kehan Li for helpful suggestions on writing and presentation.

| Variant | AP (%) | #Params (M) | Latency (ms) |
|---|---|---|---|
| A | 43.0 | 31 | 7.2 |
| B | 44.9 | 32 | 11.1 |
| C | 45.6 | 32 | 13.3 |
| D | 46.4 | 35 | 12.2 |
| $D_{\mathcal{S}_5}$ | 46.8 | 35 | 7.9 |
| E | 47.9 | 42 | 9.3 |

表3. 图3所示变体集合的指标{v*}。

| Query selection | AP (%) | $\text{Prop}_{cls}$↑ (%) | $\text{Prop}_{both}$↑ (%) |
|---|---|---|---|
| **Vanilla** | 47.9 | 0.35 | 0.30 |
| **Uncertainty-minimal** | 48.7 | 0.82 | 0.67 |

表4. 不确定性最小化查询选择的消融研究结果。$\text{Prop}_{cls}$和$\text{Prop}_{both}$分别表示分类分数及两者分数均大于0.5的比例。

分类分数大于0.5以及分类和IoU分数均大于0.5的比例。结果表明，通过不确定性最小化查询选择所筛选的编码器特征，不仅提高了高分类分数的比例（0.82%对比0.35%），还提供了更多高质量特征（0.67%对比0.30%）。我们还在COCO val2017上评估了采用两种查询选择方案训练的检测器精度，其中不确定性最小化查询选择实现了0.8%的平均精度提升（48.7% AP对比47.9% AP）。

### 5.4. 解码器消融研究

表5展示了RT-DETR-R50模型在不同解码器层数训练下各层的推理延迟与精度表现。当解码器层数设置为6时，RT-DETR-R50以53.1% AP达到最佳精度。进一步观察发现，随着解码器层序号的增加，相邻层级间的精度差异逐渐缩小。以RT-DETR-R50-$\text{Det}^6$列为例，使用第5层解码器进行推理仅损失0.1% AP精度（53.1% AP vs 53.0% AP），同时降低0.5毫秒延迟（9.3毫秒 vs 8.8毫秒）。因此RT-DETR可通过调整解码器层数实现灵活的速率调节，且无需重新训练，显著提升了实用价值。

## 6. 局限性与讨论

局限性。尽管提出的RT-DETR在速度和精度上均优于同类尺寸的先进实时检测器及端到端检测器，但它与其他DETR模型存在相同局限，*i.e*，即在小物体检测性能上仍逊色于强劲的实时检测器。根据表2数据，RT-DETR-R50的AP值低了0.5%。

| ID | AP(%) | | | | Latency (ms) |
|---|---|---|---|---|---|
| | $\text{Det}^4$ | $\text{Det}^5$ | $\text{Det}^6$ | $\text{Det}^7$ | |
| 7 | - | - | - | 52.6 | 9.6 |
| 6 | - | - | **53.1** | 52.6 | 9.3 |
| 5 | - | 52.9 | 53.0 | 52.5 | 8.8 |
| 4 | 52.7 | 52.7 | 52.7 | 52.1 | 8.3 |
| 3 | 52.4 | 52.3 | 52.4 | 51.5 | 7.9 |
| 2 | 51.6 | 51.3 | 51.3 | 50.6 | 7.5 |
| 1 | 49.6 | 48.8 | 49.1 | 48.3 | 7.0 |

表5. 解码器消融研究结果。ID表示解码器层索引。$\text{Det}^k$代表具有$k$层解码器的检测器。所有结果均在RT-DETR-R50的6×配置下报告。

比$L$模型（YOLOv8-L）中的最高$\text{AP}_S^{val}$还要高，而RT-DETR-R101比$X$模型（YOLOv7-X）中的最高$\text{AP}_S^{val}$低了0.9%。我们希望这个问题能在未来的工作中得到解决。

讨论。现有的大型DETR模型[3,6,32,41,44,46]在COCO test-dev[20]排行榜上展现了卓越性能。所提出的不同规模RT-DETR保持了与其他DETR同构的解码器设计，这使得我们能够利用高精度预训练的大型DETR模型来蒸馏轻量级检测器。我们认为这是RT-DETR相较于其他实时检测器的优势之一，也可能成为未来探索的一个有趣方向。

## 7. 结论

在本工作中，我们提出了一种名为RT-DETR的实时端到端检测器，成功将DETR框架扩展至实时检测场景，并实现了最先进的性能表现。RT-DETR包含两项关键改进：一个高效处理多尺度特征的混合编码器，以及通过不确定性最小化查询选择机制提升初始目标查询质量。此外，RT-DETR支持无需重新训练的灵活速度调节，并消除了双NMS阈值带来的操作不便，显著提升了实际应用便利性。RT-DETR及其模型缩放策略为实时目标检测开辟了新的技术路径，在多样化实时场景中提供了超越YOLO框架的全新可能性。我们希望RT-DETR能够真正投入实际应用。

致谢。本研究部分得到了中国国家重点研发计划（编号2022ZD0118201）、国家自然科学基金（编号61972217、32071459、62176249、62006133、62271465）以及深圳市医学研究基金（编号B2302037）的资助。感谢刘畅、王振南和李可汗在论文撰写与呈现方面提出的宝贵建议。

## References


[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1, 2

[2] Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4488–4499, 2022. 1

[3] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. In *International Conference on Learning Representations*, 2022. 8

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 1, 2, 7

[5] Qiang Chen, Xiaokang Chen, Gang Zeng, and Jingdong Wang. Group detr: Fast training convergence with decoupled one-to-many label assignment. *arXiv preprint arXiv:2207.13085*, 2022. 2

[6] Qiang Chen, Jian Wang, Chuchu Han, Shan Zhang, Zexian Li, Xiaokang Chen, Jiahui Chen, Xiaodi Wang, Shuming Han, Gang Zhang, et al. Group detr v2: Strong object detector with encoder-decoder pretraining. *arXiv preprint arXiv:2211.03594*, 2022. 8

[7] Cheng Cui, Ruoyu Guo, Yuning Du, Dongliang He, Fu Li, Zewu Wu, Qiwen Liu, Shilei Wen, Jizhou Huang, Xiaoguang Hu, Dianhai Yu, Errui Ding, and Yanjun Ma. Beyond self-supervision: A simple yet effective network distillation alternative to improve backbones. *CoRR*, abs/2103.05959, 2021. 1

[8] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021. 5

[9] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3621–3630, 2021. 7

[10] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 1, 2

[11] Jocher Glenn. Yolov5 release v7.0. https://github.com/ultralytics/yolov5/tree/v7.0, 2022. 2, 3, 6, 7

[12] Jocher Glenn. Yolov8. https://github.com/ultralytics/ultralytics/tree/main, 2023. 1, 2, 3, 6, 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6, 1

[14] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 6, 1

[15] Xin Huang, Xinxin Wang, Wenyu Lv, Xiaying Bai, Xiang Long, Kaipeng Deng, Qingqing Dang, Shumin Han, Qiwen Liu, Xiaoguang Hu, et al. Pp-yolov2: A practical object detector. *arXiv preprint arXiv:2104.10419*, 2021. 1, 2

[16] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3.0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*, 2023. 1, 2, 3, 6, 7

[17] 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. 1, 2, 7

[18] Feng Li, Ailing Zeng, Shilong Liu, Hao Zhang, Hongyang Li, Lei Zhang, and Lionel M Ni. Lite detr: An interleaved multi-scale encoder for efficient detr. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18558–18567, 2023. 2

[19] Junyu Lin, Xiaofeng Mao, Yuefeng Chen, Lei Xu, Yuan He, and Hui Xue. Dˆ 2etr: Decoder-only detr with computationally efficient cross-scale attention. *arXiv preprint arXiv:2203.00860*, 2022. 4

[20] 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. 3, 8, 1

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2980–2988, 2017. 2

[22] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018. 4

[23] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. Dab-detr: Dynamic anchor boxes are better queries for detr. In *International Conference on Learning Representations*, 2021. 1, 2, 7

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016. 2

[25] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, et al. Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*, 2020. 1, 2

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 1

[27] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional detr for fast training convergence. In *Proceedings of*

# 参考文献


[1] Alexey Bochkovskiy, Chien-Yao Wang, 与 Hong-Yuan Mark Liao. YOLOv4：目标检测的最佳速度与精度. *arXiv preprint arXiv:2004.10934*, 2020. 1, 2

[2] Daniel Bogdoll, Maximilian Nitsche, 与 J Marius Zöllner. 自动驾驶中的异常检测：综述. 载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 页码 4488–4499, 2022. 1

[3] 蔡宇轩, 周益庄, 韩琦, 孙健健, 孔祥文, 李军, 张翔宇. 可逆柱状网络. 载于 *International Conference on Learning Repre- sentations*, 2022. 8

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, 与 Sergey Zagoruyko. 基于Transformer的端到端目标检测. 载于 *European Conference on Computer Vision*, 页码 213–229. Springer, 2020. 1, 2, 7

[5] 陈强, 陈晓康, 曾刚, 王京东. Group DETR：通过解耦一对多标签分配实现快速训练收敛. *arXiv preprint arXiv:2207.13085*, 2022. 2

[6] 陈强, 王健, 韩楚楚, 张山, 李泽贤, 陈晓康, 陈佳慧, 王晓迪, 韩树明, 张刚, 等. Group DETR v2：基于编码器-解码器预训练的强目标检测器. *arXiv preprint arXiv:2211.03594*, 2022. 8

[7] 崔成, 郭若愚, 杜宇宁, 贺栋梁, 李福, 吴泽武, 刘奇文, 温世磊, 黄继周, 胡晓光, 于佃海, 丁二锐, 马彦君. 超越自监督：一种简单高效的网络蒸馏替代方案以改进骨干网络. *CoRR*, abs/2103.05959, 2021. 1

[8] 丁晓晗, 张翔宇, 马宁宁, 韩军功, 丁贵广, 孙剑. RepVGG：让VGG风格卷积网络重焕光彩. 载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 页码 13733–13742, 2021. 5

[9] 高鹏, 郑明航, 王晓刚, 戴继峰, 李洪胜. 通过空间调制协同注意力实现DETR快速收敛. 载于 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 页码 3621–3630, 2021. 7

[10] 葛政, 刘松涛, 王峰, 李泽明, 孙剑. YOLOX：2021年超越YOLO系列. *arXiv preprint arXiv:2107.08430*, 2021. 1, 2

[11] Jocher Glenn. YOLOv5发布v7.0版. https://github.com/ultralytics/yolov5/tree/v7.0, 2022. 2, 3, 6, 7

[12] Jocher Glenn. YOLOv8. https://github.com/ultralytics/ultralytics/tree/main, 2023. 1, 2, 3, 6, 7

[13] 何恺明, 张翔宇, 任少卿, 孙剑. 深度残差学习在图像识别中的应用. 载于 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 页码 770–778, 2016. 6, 1

[14] 何通, 张智, 张航, 张忠岳, 谢俊元, 李牧. 图像分类技巧集基于卷积神经网络的检测方法。在*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*中，第558–567页，2019年。6, 1

[15] 黄鑫，王欣欣，吕文宇，白夏莹，龙翔，邓凯鹏，党青青，韩书敏，刘奇文，胡晓光等。PP-YOLOv2：一种实用的目标检测器。*arXiv preprint arXiv:2104.10419*，2021年。1, 2

[16] 李楚怡，李璐璐，耿一飞，姜洪亮，程猛，张博，柯再丹，徐晓明，褚相相。YOLOv6 v3.0：全面升级版。*arXiv preprint arXiv:2301.05586*，2023年。1, 2, 3, 6, 7

[17] 李峰，张浩，刘世龙，郭健，倪力铭，张磊。DN-DETR：通过引入查询去噪加速DETR训练。在*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*中，第13619–13627页，2022年。1, 2, 7

[18] 李峰，曾爱玲，刘世龙，张浩，李洪阳，张磊，倪力铭。Lite DETR：一种交错多尺度编码器的高效DETR。在*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*中，第18558–18567页，2023年。2

[19] 林俊宇，毛晓峰，陈月峰，徐磊，何源，薛辉。D2ETR：仅含解码器的DETR与计算高效的跨尺度注意力。*arXiv preprint arXiv:2203.00860*，2022年。4

[20] 林腾毅，Michael Maire，Serge Belongie，James Hays，Pietro Perona，Deva Ramanan，Piotr Dollár，C. Lawrence Zitnick。Microsoft COCO：上下文中的常见物体。在*European Conference on Computer Vision*中，第740–755页。Springer，2014年。3, 8, 1

[21] 林腾毅，Priya Goyal，Ross Girshick，何凯明，Piotr Dollár。密集目标检测的焦点损失。在*Proceedings of the IEEE/CVF International Conference on Computer Vision*中，第2980–2988页，2017年。2

[22] 刘舒，戚鲁，秦海芳，石建平，贾佳亚。实例分割的路径聚合网络。在*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*中，第8759–8768页，2018年。4

[23] 刘世龙，李峰，张浩，杨晓，齐贤标，苏航，朱军，张磊。DAB-DETR：动态锚框作为DETR的更优查询。在*International Conference on Learning Representations*中，2021年。1, 2, 7

[24] 刘伟，Dragomir Anguelov，Dumitru Erhan，Christian Szegedy，Scott Reed，傅成阳，Alexander C. Berg。SSD：单次多框检测器。在*European Conference on Computer Vision*中，第21–37页。Springer，2016年。2

[25] 龙翔，邓凯鹏，王冠中，张阳，党青青，高远，沈辉，任建国，韩书敏，丁二锐等。PP-YOLO：一种高效且有效的目标检测器实现。*arXiv preprint arXiv:2007.12099*，2020年。1, 2

[26] Ilya Loshchilov，Frank Hutter。解耦权重衰减正则化。在*International Conference on Learning Representations*中，2018年。1

[27] 孟德普，陈晓康，范泽佳，曾刚，李厚强，袁宇辉，孙磊，王京东。快速训练收敛的条件DETR。在*Proceedings of*

*the IEEE/CVF International Conference on Computer Vision*, pages 3651–3660, 2021. 1, 3, 7

[28] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, and Xinghuo Yu. Spatiotemporal anomaly detection using deep learning for real-time video surveillance. *IEEE Transactions on Industrial Informatics*, 16(1):393–402, 2019. 1

[29] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017. 2

[30] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 2

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 2

[32] Tianhe Ren, Jianwei Yang, Shilong Liu, Ailing Zeng, Feng Li, Hao Zhang, Hongyang Li, Zhaoyang Zeng, and Lei Zhang. A strong and reproducible object detector with only public datasets. *arXiv preprint arXiv:2304.13027*, 2023. 8

[33] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In *International Conference on Learning Representations*, 2021. 2

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 1

[35] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8430–8439, 2019. 2, 1

[36] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14454–14463, 2021. 1

[37] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13029–13038, 2021. 2

[38] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023. 1, 2, 3, 6, 7

[39] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2567–2575, 2022. 1, 3, 7

[40] Shangliang Xu, Xinxin Wang, Wenyu Lv, Qinyao Chang, Cheng Cui, Kaipeng Deng, Guanzhong Wang, Qingqing Dang, Shengyu Wei, Yuning Du, et al. Pp-yoloe: An evolved version of yolo. *arXiv preprint arXiv:2203.16250*, 2022. 1, 2, 3, 6, 7

[41] Jianwei Yang, Chunyuan Li, Xiyang Dai, and Jianfeng Gao. Focal modulation networks. *Advances in Neural Information Processing Systems*, 35:4203–4217, 2022. 8

[42] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: improving end-to-end object detector with dense prior. *arXiv preprint arXiv:2104.01318*, 2021. 2, 5, 7

[43] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *European Conference on Computer Vision*, pages 659–675. Springer, 2022. 1

[44] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *International Conference on Learning Representations*, 2022. 1, 2, 3, 5, 7, 8

[45] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020. 1, 2, 3, 4, 5, 7

[46] 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. 8

*the IEEE/CVF International Conference on Computer Vision*，第3651–3660页，2021年。1, 3, 7 [28] Rashmika Nawaratne, Damminda Alahakoon, Daswin De Silva, 和 Xinghuo Yu。基于深度学习的实时视频监控时空异常检测。*IEEE Transactions on Industrial Informatics*, 16(1):393–402, 2019年。1 [29] Joseph Redmon 和 Ali Farhadi。YOLO9000：更好、更快、更强。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第7263–7271页，2017年。2 [30] Joseph Redmon 和 Ali Farhadi。YOLOv3：渐进式改进。*arXiv preprint arXiv:1804.02767*, 2018年。1, 2 [31] Joseph Redmon, Santosh Divvala, Ross Girshick, 和 Ali Farhadi。你只需看一次：统一、实时的目标检测。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第779–788页，2016年。2 [32] 任天鹤, 杨建伟, 刘世龙, 曾爱玲, 李峰, 张浩, 李洪阳, 曾朝阳, 张磊。仅用公开数据集构建的强健且可复现的目标检测器。*arXiv preprint arXiv:2304.13027*, 2023年。8 [33] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, 和 Saehoon Kim。稀疏DETR：通过可学习稀疏性实现高效端到端目标检测。载于*International Conference on Learning Representations*, 2021年。2 [34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein等。ImageNet大规模视觉识别挑战赛。*International Journal of Computer Vision*, 115:211–252, 2015年。1 [35] 邵帅, 李泽明, 张天元, 彭超, 余刚, 张翔宇, 李静, 孙剑。Objects365：面向目标检测的大规模高质量数据集。载于*Proceedings of the IEEE/CVF International Conference on Computer Vision*, 第8430–8439页，2019年。2, 1 [36] 孙培泽, 张如风, 姜毅, 孔涛, 徐晨风, 战威, Masayoshi Tomizuka, 李磊, 袁泽寰, 王昌虎等。稀疏R-CNN：基于可学习建议框的端到端目标检测。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第14454–14463页，2021年。1 [37] 王建尧, Alexey Bochkovskiy, 和廖宏远。Scaled-YOLOv4：跨阶段局部网络的缩放优化。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第13029–13038页，2021年。2 [38] 王建尧, Alexey Bochkovskiy, 和廖宏远。YOLOv7：可训练“免费赠品”集合为实时目标检测器树立新标杆。载于*Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 第7464–7475页，2023年。1, 2, 3, 6, 7 [39] 王英明, 张翔宇, 杨桐, 孙剑。Anchor DETR：基于Transformer检测器的查询设计。载于*Proceedings of the AAAI Conference on Artificial Intelligence*, 第2567–2575页，2022年。1, 3, 7 [40] 徐尚亮, 王欣欣, 吕文宇, 常钦尧, 崔程, 邓凯鹏, 王冠中, 青青党，沈裕伟，杜宇宁，等。PP-YOLOE：YOLO的进化版本。*arXiv preprint arXiv:2203.16250*，2022年。1, 2, 3, 6, 7 [41] 杨建伟，李春元，戴熙阳，高剑峰。焦点调制网络。*Advances in Neural Information Processing Systems*，35卷：4203–4217页，2022年。8 [42] 姚竹玉，艾江波，李伯勋，张弛。高效DETR：利用密集先验改进端到端目标检测器。*arXiv preprint arXiv:2104.01318*，2021年。2, 5, 7 [43] 曾繁高，董斌，张远光，王天财，张翔宇，魏一辰。MOTR：基于Transformer的端到端多目标跟踪。载于*European Conference on Computer Vision*，659–675页。Springer，2022年。1 [44] 张浩，李峰，刘世龙，张磊，苏航，朱军，倪立昂，沈向洋。DINO：带改进去噪锚框的DETR用于端到端目标检测。载于*International Conference on Learning Representa- tions*，2022年。1, 2, 3, 5, 7, 8 [45] 朱熙哲，苏伟杰，陆乐威，李斌，王晓刚，戴继峰。可变形DETR：端到端目标检测的可变形Transformer。载于*International Conference on Learning Representations*，2020年。1, 2, 3, 4, 5, 7 [46] 宗卓凡，宋广录，刘宇。协作混合分配训练的DETRs。载于*Proceedings of the IEEE/CVF International Conference on Computer Vision*，6748–6758页，2023年。8

# Appendix of "DETRs Beat YOLOs on Real-time Object Detection"

## 1. Experimental Settings

**Dataset and metric.** We conduct experiments on COCO [20] and Objects365 [35], where RT-DETR is trained on COCO `train2017` and validated on COCO `val2017` dataset. We report the standard COCO metrics, including AP (averaged over uniformly sampled IoU thresholds ranging from 0.50-0.95 with a step size of 0.05), $AP_{50}$, $AP_{75}$, as well as AP at different scales: $AP_S$, $AP_M$, $AP_L$.

**Implementation details.** We use ResNet [13, 14] pretrained on ImageNet [7, 34] as the backbone and the learning rate strategy of the backbone follows [4]. In the hybrid encoder, AIFI consists of 1 Transformer layer and the fusion block in CCFF consists of 3 *RepBlock*s. We leverage the uncertainty-minimal query selection to select top 300 encoder features to initialize object queries of the decoder. The training strategy and hyperparameters of the decoder almost follow DINO [44]. We train RT-DETR with the AdamW [26] optimizer using four NVIDIA Tesla V100 GPUs with a batch size of 16 and apply the exponential moving average (EMA) with $ema\_decay = 0.9999$. The $1\times$ configuration means that the total epoch is 12, and the final reported results adopt the $6\times$ configuration. The data augmentation applied during training includes *random_{color distort, expand, crop, flip, resize}* operations, following [40]. The main hyperparameters of RT-DETR are listed in Table A (refer to RT-DETR-R50 for detailed configuration).

## 2. Comparison with Lighter YOLO Detectors

To adapt to diverse real-time detection scenarios, we develop lighter scaled RT-DETRs by scaling the encoder and decoder with ResNet50/34/18 [13]. Specifically, we halve the number of channels in the *RepBlock*, while leaving other components unchanged, and obtain a set of RT-DETRs by adjusting the number of decoder layers during inference. We compare the scaled RT-DETRs with the $S$ and $M$ models of YOLO detectors in Table B. The number of decoder layers used by scaled RT-DETR-R50/34/18 during training is 6/4/3 respectively, and $\text{Dec}^k$ indicates that $k$ decoder layers are used during inference. Our RT-DETR-R50-$\text{Dec}^{2-5}$ outperform all $M$ models of YOLO detectors in both speed and accuracy, while RT-DETR-R18-$\text{Dec}^2$ outperforms all $S$ models. Compared to the state-of-the-art $M$ model (YOLOv8-M [12]), RT-DETR-R50-$\text{Dec}^5$ improves accuracy by $0.9\%$ AP and increases FPS by $36\%$. Compared to the state-of-the-art $S$ model (YOLOv6-S [16]), RT-DETR-R18-$\text{Dec}^2$ improves accuracy by $0.5\%$ AP and increases FPS by $18\%$. This shows that RT-DETR is able to outperform the lighter YOLO detectors in both speed and accuracy by simple scaling.

| Item | Value |
|---|---|
| optimizer | AdamW |
| base learning rate | 1e-4 |
| learning rate of backbone | 1e-5 |
| freezing BN | True |
| linear warm-up start factor | 0.001 |
| linear warm-up steps | 2000 |
| weight decay | 0.0001 |
| clip gradient norm | 0.1 |
| ema decay | 0.9999 |
| number of AIFI layers | 1 |
| number of *RepBlock*s | 3 |
| embedding dim | 256 |
| feedforward dim | 1024 |
| nheads | 8 |
| number of feature scales | 3 |
| number of decoder layers | 6 |
| number of queries | 300 |
| decoder npoints | 4 |
| class cost weight | 2.0 |
| $\alpha$ in class cost | 0.25 |
| $\gamma$ in class cost | 2.0 |
| bbox cost weight | 5.0 |
| GIoU cost weight | 2.0 |
| class loss weight | 1.0 |
| $\alpha$ in class loss | 0.75 |
| $\gamma$ in class loss | 2.0 |
| bbox loss weight | 5.0 |
| GIoU loss weight | 2.0 |
| denoising number | 200 |
| label noise ratio | 0.5 |
| box noise scale | 1.0 |

Table A. Main hyperparameters of RT-DETR.

## 3. Large-scale Pre-training for RT-DETR

We pre-train RT-DETR on the larger Objects365[35] dataset and then fine-tune it on COCO to achieve higher performance. As shown in Table C, we perform experiments on RT-DETR-R18/50/101 respectively. All three models are pre-trained on Objects365 for 12 epochs, and RT-DETR-R18 is fine-tuned on COCO for 60 epochs, while RT-DETR-R50 and RT-DETR-R101 are fine-tuned for 24 epochs. Experimental results show that RT-DETR-R18/50/101 is improved

# “DETRs在实时目标检测中超越YOLOs”附录

## 1. 实验设置

数据集与评估指标。我们在COCO[20]和Objects365[35]上进行实验，其中RT-DETR在COCO train2017上训练，并在COCO val2017数据集上验证。我们报告了标准COCO指标，包括AP（在0.50至0.95范围内以0.05为步长均匀采样的IoU阈值上的平均值）、$AP_{50}$、$AP_{75}$，以及不同尺度下的AP：$AP_S$、$AP_M$、$AP_L$。

实现细节。我们采用在ImageNet [7, 34]上预训练的ResNet [13, 14]作为主干网络，主干网络的学习率策略遵循[4]。在混合编码器中，AIFI包含1个Transformer层，CCFF中的融合块由3个*RepBlock*组成。我们利用不确定性最小查询选择机制，选取前300个编码器特征来初始化解码器的对象查询。解码器的训练策略及超参数基本遵循DINO [44]方案。使用AdamW [26]优化器在四块NVIDIA Tesla V100 GPU上训练RT-DETR，批量大小为16，并采用$ema_decay$ =0.9999的指数移动平均（EMA）。1×配置表示总训练轮数为12轮，最终报告结果采用6×配置。训练期间应用的数据增强包括*random_*{*color distort, expand, crop, flip,resize*}操作，遵循[40]方法。RT-DETR的主要超参数列于表A（详细配置请参考RT-DETR-R50）。

| Item | Value |
|---|---|
| optimizer | AdamW |
| base learning rate | 1e-4 |
| learning rate of backbone | 1e-5 |
| freezing BN | True |
| linear warm-up start factor | 0.001 |
| linear warm-up steps | 2000 |
| weight decay | 0.0001 |
| clip gradient norm | 0.1 |
| ema decay | 0.9999 |
| number of AIFI layers | 1 |
| number of *RepBlock*s | 3 |
| embedding dim | 256 |
| feedforward dim | 1024 |
| nheads | 8 |
| number of feature scales | 3 |
| number of decoder layers | 6 |
| number of queries | 300 |
| decoder npoints | 4 |
| class cost weight | 2.0 |
| $\alpha$ in class cost | 0.25 |
| $\gamma$ in class cost | 2.0 |
| bbox cost weight | 5.0 |
| GIoU cost weight | 2.0 |
| class loss weight | 1.0 |
| $\alpha$ in class loss | 0.75 |
| $\gamma$ in class loss | 2.0 |
| bbox loss weight | 5.0 |
| GIoU loss weight | 2.0 |
| denoising number | 200 |
| label noise ratio | 0.5 |
| box noise scale | 1.0 |

表A. RT-DETR的主要超参数。

## 2. 与轻量级YOLO检测器的比较

为适应多样化的实时检测场景，我们通过采用ResNet50/34/18[13]对编码器和解码器进行缩放，开发了更轻量化的RT-DETR系列模型。具体而言，我们将*RepBlock*的通道数减半，同时保持其他组件不变，并通过在推理阶段调整解码器层数获得一组RT-DETR变体。表B中将缩放后的RT-DETR与YOLO检测器的$S$、$M$模型进行对比。训练阶段RT-DETR-R50/34/18分别采用6/4/3层解码器，$\text{Dec}^k$表示推理时使用$k$层解码器。我们的RT-DETR-R50-$\text{Dec}^{2-5}$在速度和精度上均超越所有YOLO检测器的$M$模型，而RT-DETR-R18-$\text{Dec}^2$则优于所有$S$模型。相较于最先进的$M$模型（YOLOv8-M[12]），RT-DETR-R50-$\text{Dec}^5$精度提升0.9% AP，帧率提高36%；相比最先进的$S$模型（YOLOv6-S[16]），RT-DETR-R18-$\text{Dec}^2$精度提升0.5% AP，帧率提高18%。这表明通过简单缩放，RT-DETR能在速度和精度上全面超越轻量级YOLO检测器。

## 3. RT-DETR的大规模预训练

我们在更大的Objects365[35]数据集上对RT-DETR进行预训练，随后在COCO上微调以获得更高性能。如表C所示，我们分别对RT-DETR-R18/50/101进行了实验。三个模型均在Objects365上预训练12个周期，其中RT-DETR-R18在COCO上微调60个周期，而RT-DETR-R50和RT-DETR-R101则微调24个周期。实验结果表明，RT-DETR-R18/50/101均实现了性能提升

| Model | #Epochs | #Params (M) | GFLOPs | $FPS_{bs=1}$ | $AP^{val}$ | $AP^{val}_{50}$ | $AP^{val}_{75}$ | $AP^{val}_{S}$ | $AP^{val}_{M}$ | $AP^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| *S and M models of YOLO Detectors* | | | | | | | | | | |
| YOLOv5-S[11] | 300 | 7.2 | 16.5 | 74 | 37.4 | 56.8 | - | - | - | - |
| YOLOv5-M[11] | 300 | 21.2 | 49.0 | 64 | 45.4 | 64.1 | - | - | - | - |
| PPYOLOE-S[40] | 300 | 7.9 | 17.4 | 218 | 43.0 | 59.6 | 47.1 | 25.9 | 47.4 | 58.6 |
| PPYOLOE-M[40] | 300 | 23.4 | 49.9 | 131 | 48.9 | 65.8 | 53.7 | 30.8 | 53.4 | 65.3 |
| YOLOv6-S[16] | 300 | 18.5 | 45.3 | 201 | 45.0 | 61.8 | 48.9 | 24.3 | 50.2 | 62.7 |
| YOLOv6-M[16] | 300 | 34.9 | 85.8 | 121 | 50.0 | 66.9 | 54.6 | 30.6 | 55.4 | 67.3 |
| YOLOv8-S[12] | - | 11.2 | 28.6 | 136 | 44.9 | 61.8 | 48.6 | 25.7 | 49.9 | 61.0 |
| YOLOv8-M[12] | - | 25.9 | 78.9 | 97 | 50.2 | 67.2 | 54.6 | 32.0 | 55.7 | 66.4 |
| *Scaled RT-DETRs* | | | | | | | | | | |
| Scaled RT-DETR-R50-$Dec^2$ | 72 | $36^{\dagger}$ | 98.4 | **154** | **50.3** | **68.4** | 54.5 | **32.2** | 55.2 | **67.5** |
| Scaled RT-DETR-R50-$Dec^3$ | 72 | $36^{\dagger}$ | 100.1 | **145** | **51.3** | **69.6** | **55.4** | **33.6** | **56.1** | **68.6** |
| Scaled RT-DETR-R50-$Dec^4$ | 72 | $36^{\dagger}$ | 101.8 | **137** | **51.8** | **70.0** | **55.9** | **33.7** | **56.4** | **69.4** |
| Scaled RT-DETR-R50-$Dec^5$ | 72 | $36^{\dagger}$ | 103.5 | **132** | **52.1** | **70.5** | **56.2** | **34.3** | **56.9** | **69.9** |
| Scaled RT-DETR-R50-$Dec^6$ | 72 | 36 | 105.2 | 125 | **52.2** | **70.6** | **56.4** | **34.4** | **57.0** | **70.0** |
| Scaled RT-DETR-R34-$Dec^2$ | 72 | $31^{\dagger}$ | 89.3 | 185 | **47.4** | **64.7** | **51.3** | **28.9** | **51.0** | **64.2** |
| Scaled RT-DETR-R34-$Dec^3$ | 72 | $31^{\dagger}$ | 91.0 | 172 | **48.5** | **66.2** | **52.3** | **30.2** | **51.9** | **66.2** |
| Scaled RT-DETR-R34-$Dec^4$ | 72 | 31 | 92.7 | 161 | **48.9** | **66.8** | **52.9** | **30.6** | **52.4** | **66.3** |
| Scaled RT-DETR-R18-$Dec^2$ | 72 | $20^{\dagger}$ | 59.0 | **238** | **45.5** | **62.5** | **49.4** | **27.8** | 48.7 | 61.7 |
| Scaled RT-DETR-R18-$Dec^3$ | 72 | 20 | 60.7 | 217 | **46.5** | **63.8** | **50.4** | **28.4** | 49.8 | **63.0** |

Table B. Comparison with $S$ and $M$ models of YOLO detectors. The FPS of YOLO detectors are reported on T4 GPU with TensorRT FP16 using official pre-trained models according to the proposed end-to-end speed benchmark. † denotes the number of parameters during the training, not inference.

| Model | #Epochs | #Params (M) | GFLOPs | $FPS_{bs=1}$ | $AP^{val}$ | $AP^{val}_{50}$ | $AP^{val}_{75}$ | $AP^{val}_{S}$ | $AP^{val}_{M}$ | $AP^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| RT-DETR-R18 | 60 | 20 | 61 | 217 | 49.2 (↑ **2.7**) | 66.6 | 53.5 | 33.2 | 52.3 | 64.8 |
| RT-DETR-R50 | 24 | 42 | 136 | 108 | 55.3 (↑ **2.2**) | 73.4 | 60.1 | 37.9 | 59.9 | 71.8 |
| RT-DETR-R101 | 24 | 76 | 259 | 74 | 56.2 (↑ **1.9**) | 74.6 | 61.3 | 38.3 | 60.5 | 73.5 |

Table C. Fine-tuning results on COCO `val2017` with pre-training on Objects365.

by 2.7%/2.2%/1.9% AP on COCO `val2017`. The surprising improvement further demonstrates the potential of RT-DETR and provides the strongest real-time object detector for various real-time scenarios in the industry.

## 4. Visualization of Predictions with Different Post-processing Thresholds

To intuitively demonstrate the impact of post-processing on the detector, we visualize the predictions produced by YOLOv8 [12] and RT-DETR using different post-processing thresholds, as shown in Figure A and Figure B, respectively. We show the predictions for two randomly selected samples from COCO `val2017` by setting different NMS thresholds for YOLOv8-L and score thresholds for RT-DETR-R50.

There are two NMS thresholds: confidence threshold and IoU threshold, both of which affect the detection results. The higher the confidence threshold, the more prediction boxes are filtered out and the number of false negatives increases. However, using a lower confidence threshold, *e.g.*, 0.001, results in a large number of redundant boxes and increases the number of false positives. The higher the IoU threshold, the fewer overlapping boxes are filtered out in each round of screening, and the number of false positives increases (the position marked by the red circle in Figure A). Nevertheless, adopting a lower IoU threshold will result in true positives being deleted if there are overlapping or mutually occluding objects in the input. The confidence threshold is relatively straightforward to process predicted boxes and therefore easy to set, whereas the IoU threshold is difficult to set accurately. Considering that different scenarios place different emphasis on recall and accuracy, *e.g.*, the general detection scenario requires the lower confidence threshold and the higher IoU threshold to increase the recall, while the dedicated detection scenario requires the higher confidence threshold and the lower IoU threshold to increase the accuracy, it is neces-

| Model | #Epochs | #Params (M) | GFLOPs | $\text{FPS}_{bs=1}$ | $\text{AP}^{val}$ | $\text{AP}^{val}_{50}$ | $\text{AP}^{val}_{75}$ | $\text{AP}^{val}_{S}$ | $\text{AP}^{val}_{M}$ | $\text{AP}^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| *S and M models of YOLO Detectors* | | | | | | | | | | |
| YOLOv5-S[11] | 300 | 7.2 | 16.5 | 74 | 37.4 | 56.8 | - | - | - | - |
| YOLOv5-M[11] | 300 | 21.2 | 49.0 | 64 | 45.4 | 64.1 | - | - | - | - |
| PPYOLOE-S[40] | 300 | 7.9 | 17.4 | 218 | 43.0 | 59.6 | 47.1 | 25.9 | 47.4 | 58.6 |
| PPYOLOE-M[40] | 300 | 23.4 | 49.9 | 131 | 48.9 | 65.8 | 53.7 | 30.8 | 53.4 | 65.3 |
| YOLOv6-S[16] | 300 | 18.5 | 45.3 | 201 | 45.0 | 61.8 | 48.9 | 24.3 | 50.2 | 62.7 |
| YOLOv6-M[16] | 300 | 34.9 | 85.8 | 121 | 50.0 | 66.9 | 54.6 | 30.6 | 55.4 | 67.3 |
| YOLOv8-S[12] | - | 11.2 | 28.6 | 136 | 44.9 | 61.8 | 48.6 | 25.7 | 49.9 | 61.0 |
| YOLOv8-M[12] | - | 25.9 | 78.9 | 97 | 50.2 | 67.2 | 54.6 | 32.0 | 55.7 | 66.4 |
| *Scaled RT-DETRs* | | | | | | | | | | |
| Scaled RT-DETR-R50-Dec$^2$ | 72 | 36$^\dagger$ | 98.4 | **154** | **50.3** | **68.4** | 54.5 | **32.2** | 55.2 | **67.5** |
| Scaled RT-DETR-R50-Dec$^3$ | 72 | 36$^\dagger$ | 100.1 | **145** | **51.3** | **69.6** | **55.4** | **33.6** | **56.1** | **68.6** |
| Scaled RT-DETR-R50-Dec$^4$ | 72 | 36$^\dagger$ | 101.8 | **137** | **51.8** | **70.0** | **55.9** | **33.7** | **56.4** | **69.4** |
| Scaled RT-DETR-R50-Dec$^5$ | 72 | 36$^\dagger$ | 103.5 | **132** | **52.1** | **70.5** | **56.2** | **34.3** | **56.9** | **69.9** |
| Scaled RT-DETR-R50-Dec$^6$ | 72 | 36 | 105.2 | 125 | **52.2** | **70.6** | **56.4** | **34.4** | **57.0** | **70.0** |
| Scaled RT-DETR-R34-Dec$^2$ | 72 | 31$^\dagger$ | 89.3 | 185 | **47.4** | **64.7** | **51.3** | **28.9** | **51.0** | **64.2** |
| Scaled RT-DETR-R34-Dec$^3$ | 72 | 31$^\dagger$ | 91.0 | 172 | **48.5** | **66.2** | **52.3** | **30.2** | **51.9** | **66.2** |
| Scaled RT-DETR-R34-Dec$^4$ | 72 | 31 | 92.7 | 161 | **48.9** | **66.8** | **52.9** | **30.6** | **52.4** | **66.3** |
| Scaled RT-DETR-R18-Dec$^2$ | 72 | 20$^\dagger$ | 59.0 | **238** | **45.5** | **62.5** | **49.4** | **27.8** | 48.7 | 61.7 |
| Scaled RT-DETR-R18-Dec$^3$ | 72 | 20 | 60.7 | 217 | **46.5** | **63.8** | **50.4** | **28.4** | 49.8 | **63.0** |

T表 B. 与YOLO检测器的$S$和$M$模型对比。YOLO检测器的FPS基于T4 GPU、采用TensorRT FP16及官方预训练模型，按照所提出的端到端速度基准进行报告。†表示训练期间的参数量，而非推理时。

| Model | #Epochs | #Params (M) | GFLOPs | $\text{FPS}_{bs=1}$ | $\text{AP}^{val}$ | $\text{AP}^{val}_{50}$ | $\text{AP}^{val}_{75}$ | $\text{AP}^{val}_{S}$ | $\text{AP}^{val}_{M}$ | $\text{AP}^{val}_{L}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| RT-DETR-R18 | 60 | 20 | 61 | 217 | 49.2 (↑ **2.7**) | 66.6 | 53.5 | 33.2 | 52.3 | 64.8 |
| RT-DETR-R50 | 24 | 42 | 136 | 108 | 55.3 (↑ **2.2**) | 73.4 | 60.1 | 37.9 | 59.9 | 71.8 |
| RT-DETR-R101 | 24 | 76 | 259 | 74 | 56.2 (↑ **1.9**) | 74.6 | 61.3 | 38.3 | 60.5 | 73.5 |

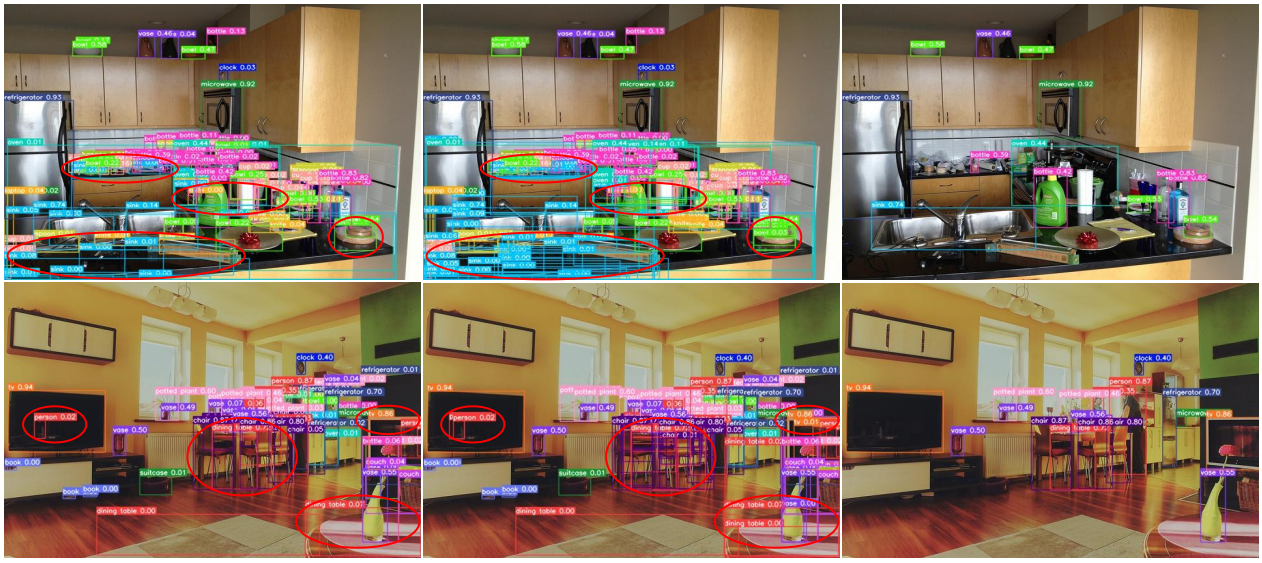
表 C. 在Objects365上预训练后，于COCO val2017数据集上的微调结果。

在COCO val2017上提升了2.7%/2.2%/1.9%的平均精度（AP）。这一显著进步进一步证明了RT-DETR的潜力，并为工业界各类实时场景提供了最强大的实时目标检测器。

## 4. 不同后处理阈值下的预测可视化

为了直观展示后处理对检测器的影响，我们分别在图A和图B中可视化了YOLOv8[12]和RT-DETR在不同后处理阈值下产生的预测结果。通过为YOLOv8-L设置不同的非极大值抑制(NMS)阈值，以及为RT-DETR-R50设置不同的分数阈值，我们展示了从COCO val2017数据集中随机选取的两个样本的预测情况。

有两个NMS阈值：置信度阈值和IoU阈值，两者都会影响检测结果。置信度阈值越高，预测框就越多被过滤掉，假阴性的数量随之增加。然而，若采用较低的置信度阈值，如*e.g*或0.001，则会产生大量冗余框，并增加假阳性的数量。IoU阈值越高，每轮筛选过滤掉的重叠框越少，假阳性数量也随之上升（如图A中红圈标记的位置）。反之，采用较低的IoU阈值时，若输入中存在相互重叠或遮挡的物体，则会导致真正例被误删。置信度阈值处理预测框相对直观，因而易于设定；而IoU阈值则难以精确设置。考虑到不同场景对召回率与准确率的侧重不同——*e.g*，通用检测场景需采用较低置信度阈值与较高IoU阈值以提高召回率，专用检测场景则需较高置信度阈值与较低IoU阈值以提升准确率——因此有必



**Conf_thr = 0.001 IoU_thr = 0.3** | **Conf_thr = 0.001 IoU_thr = 0.7** | **Conf_thr = 0.25 IoU_thr = 0.7**

Figure A. Visualization of YOLOv8-L [12] predictions with different NMS thresholds.

**Score_thr = 0.001** | **Score_thr = 0.3** | **Score_thr = 0.5**

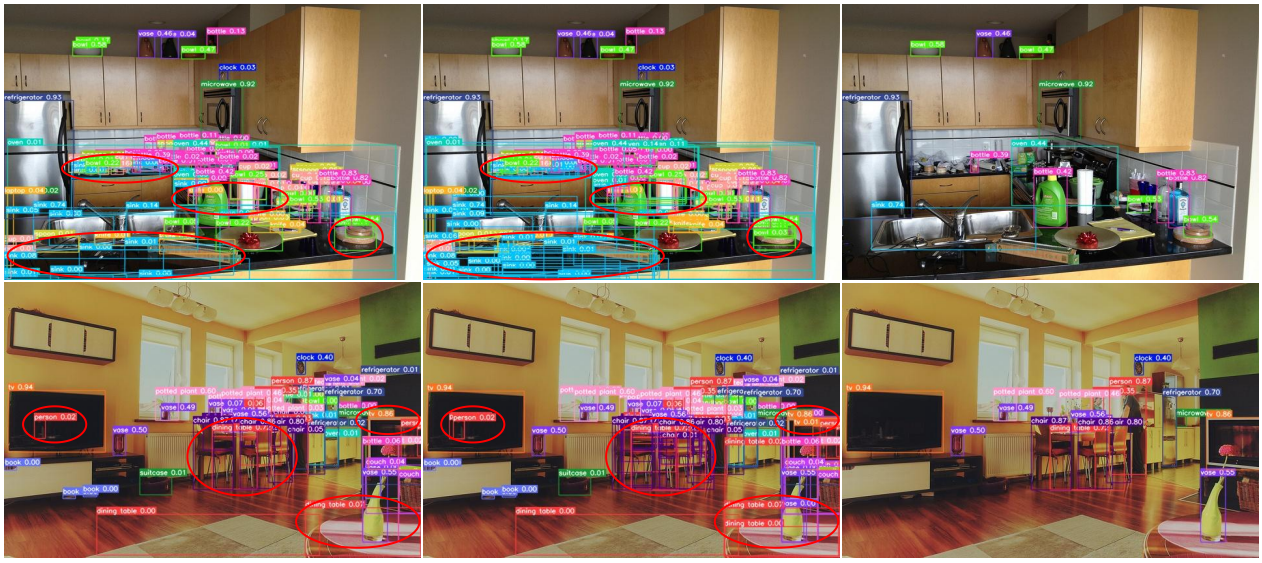
Figure B. Visualization of RT-DETR-R50 predictions with different score thresholds.

sary to carefully select the appropriate NMS thresholds for different scenarios.

RT-DETR utilizes bipartite matching to predict the one-to-one object set, eliminating the need for suppressing overlapping boxes. Instead, it directly filters out low-confidence boxes with a score threshold. Similar to the confidence threshold used in NMS, the score threshold can be adjusted in different scenarios based on the specific emphasis to achieve optimal detection performance. Thus, setting the post-processing threshold in RT-DETR is straightforward and does not affect the inference speed, enhancing the adaptability of real-time detectors across various scenarios.

## 5. Visualization of RT-DETR Predictions

We select several samples from the COCO `val2017` to showcase the detection performance of RT-DETR in complex scenarios and challenging conditions (refer to Figure C



**Conf_thr = 0.001 IoU_thr = 0.3** | **Conf_thr = 0.001 IoU_thr = 0.7** | **Conf_thr = 0.25 IoU_thr = 0.7**

图A. YOLOv8-L [12]在不同NMS阈值下的预测结果可视化。

**Score_thr = 0.001** | **Score_thr = 0.3** | **Score_thr = 0.5**

图 B. 不同分数阈值下RT-DETR-R50预测结果的可视化。

需要根据不同场景仔细选择合适的NMS阈值。

RT-DETR采用二分图匹配技术预测一对一的对象集合，无需抑制重叠框，而是直接通过分数阈值过滤掉低置信度的检测框。与NMS中使用的置信度阈值类似，该分数阈值可根据不同场景的具体需求进行调整，以实现最优的检测性能。因此，设定这一RT-DETR中的后处理阈值设置简单直观，且不影响推理速度，从而增强了实时检测器在不同场景下的适应能力。

## 5. RT-DETR预测结果的可视化

我们从COCO val2017数据集中选取若干样本，以展示RT-DETR在复杂场景及挑战性条件下的检测性能（参见图C）。

Figure C. Visualization of RT-DETR-R101 predictions in complex scenarios (score threshold=0.5).

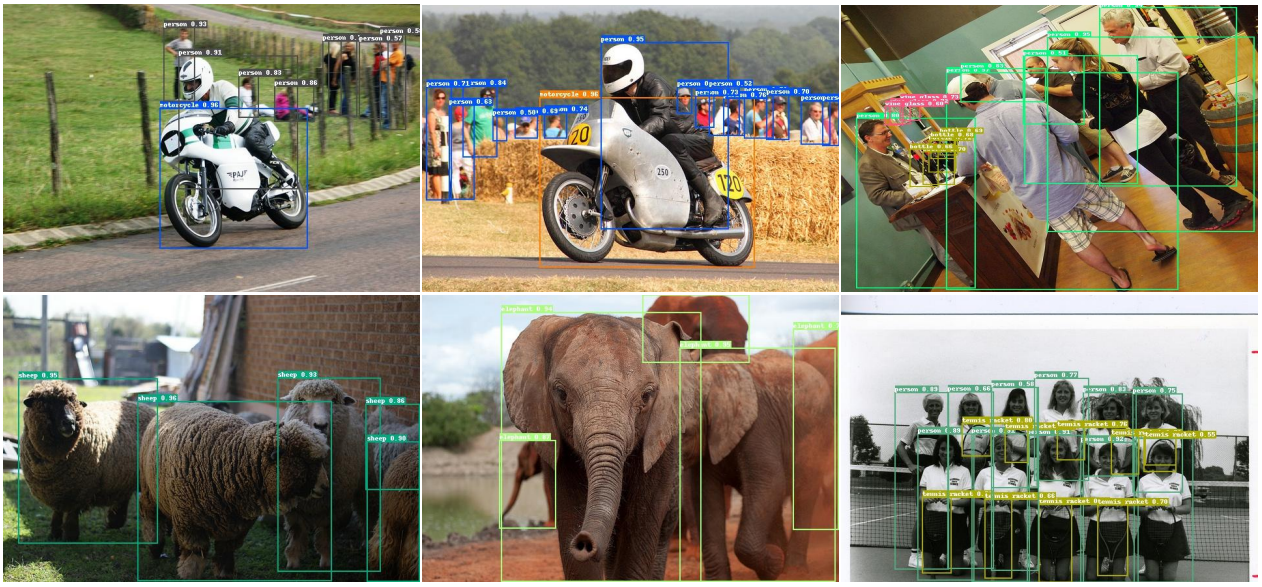


Figure D. Visualization of RT-DETR-R101 predictions under difficult conditions, including motion blur, rotation, and occlusion (score threshold=0.5).

and Figure D). In complex scenarios, RT-DETR demonstrates its capability to detect diverse objects, even when they are small or densely packed, *e.g.*, cups, wine glasses, and individuals. Moreover, RT-DETR successfully detects objects under various difficult conditions, including motion blur, rotation, and occlusion. These predictions substantiate the excellent detection performance of RT-DETR.

图 C. RT-DETR-R101在复杂场景下的预测可视化（得分阈值=0.5）。

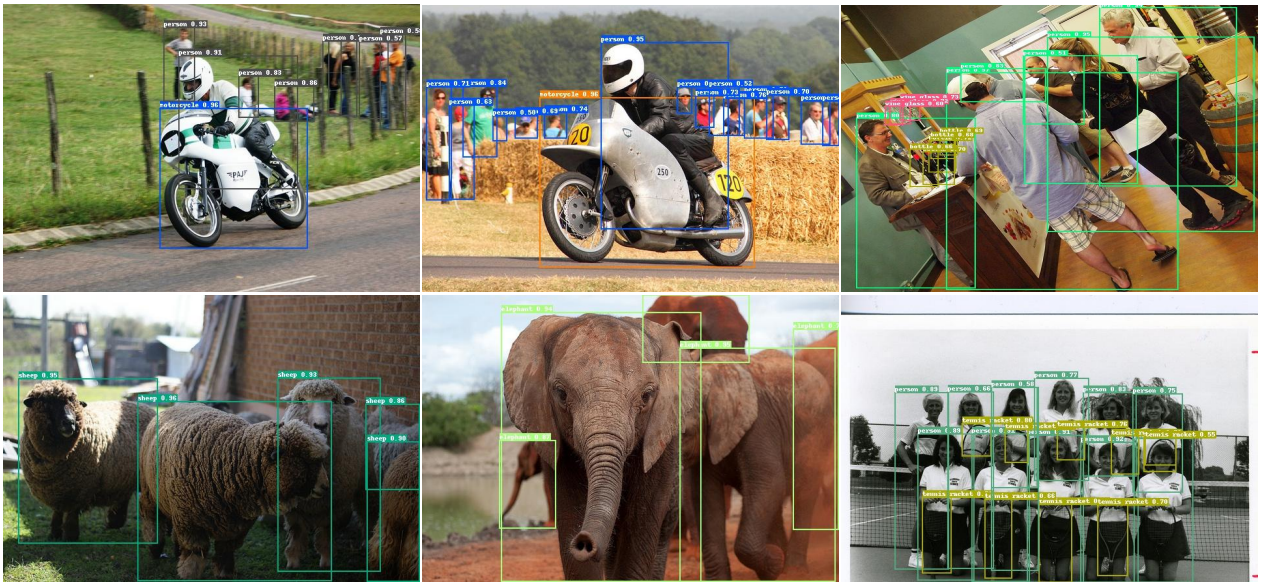


图 D. RT-DETR-R101在困难条件下的预测可视化，包括运动模糊、旋转和遮挡（得分阈值=0.5）。

以及图D）。在复杂场景中，RT-DETR展现了其检测多样化物体的能力，即便这些物体体积微小或排列密集，*e.g*，如杯子、酒杯和人物。此外，RT-DETR能够在多种困难条件下成功检测物体，包括运动模糊、旋转和遮挡。这些预测结果验证了RT-DETR卓越的检测性能。