Initial Reading Done

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 **R**eal-**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 [40], video surveillance [26], 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, 9–11, 14, 15, 24, 28, 35, 37] 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.

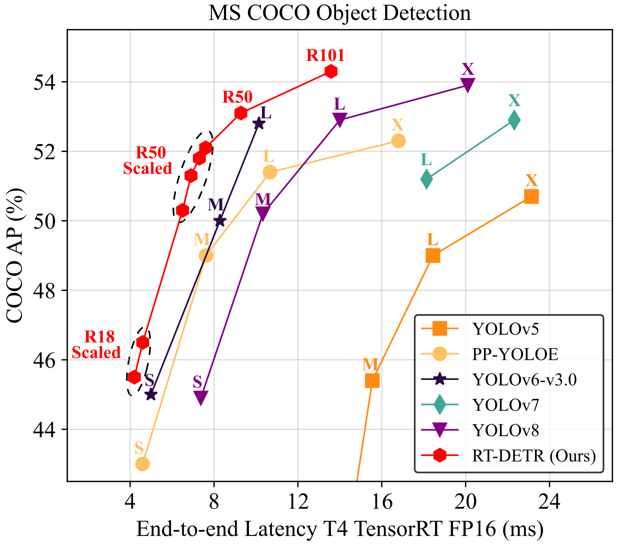

Figure 1. Compared to previously advanced real-time object detectors, our RT-DETR achieves state-of-the-art performance.

Recently, the end-to-end Transformer-based detectors (DETRs) [4, 16, 22, 25, 33, 36, 41, 42] 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.

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 [42], 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 [39, 41, 42] 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 [32], 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 [29] 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 [20, 23] and become the synonymous with the real-time object detector. YOLO detectors can be classified into two categories: anchor-based [1, 10, 14, 24, 27, 28, 34, 35] and anchor-free [9, 11, 15, 37], 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 [42] accelerates training convergence with multi-scale features by enhancing the efficiency of the attention mechanism. DAB-DETR [22] and DN-DETR [16] 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 [39] and Sparse DETR [31] reduce the computational cost by reducing the number of encoder and decoder layers or the number of updated queries. Lite DETR [17] enhances the efficiency of encoder by reducing the update frequency of low-level features in an interleaved way. **Optimizing query initial-**

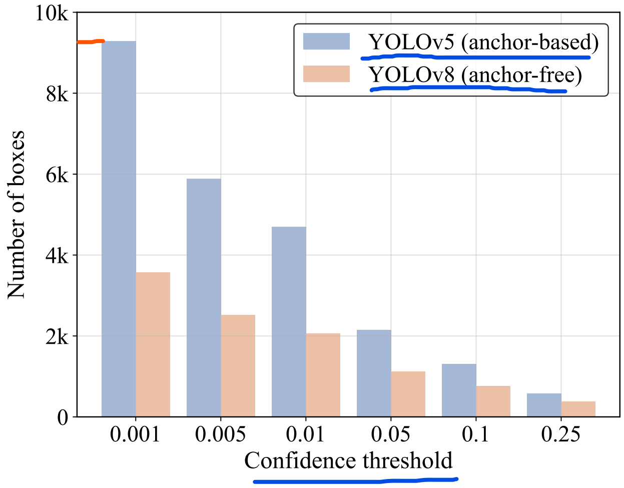

Figure 2. The number of boxes at different confidence thresholds.

ization. Conditional DETR [25] and Anchor DETR [36] decrease the optimization difficulty of the queries. Zhu *et al*. [42] propose the query selection for two-stage DETR, and DINO [41] 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 [10] (anchor-based) and YOLOv8 [11] (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` [19] 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 [10] and YOLOv7 [35], as well as anchor-free detectors PP-YOLOE [37], YOLOv6 [15] and YOLOv8 [11]

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

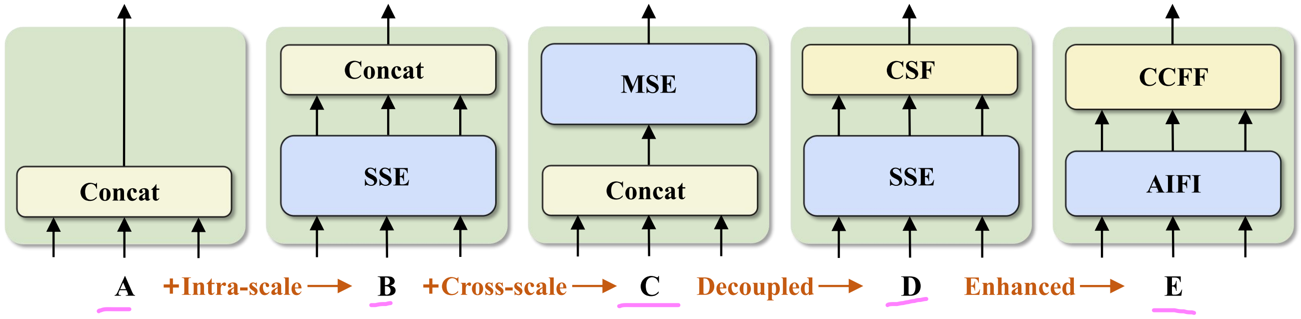

Figure 3. The encoder structure for each variant. **SSE** represents the single-scale Transformer encoder, **MSE** represents the multi-scale Transformer encoder, and **CSF** represents cross-scale fusion. **AIFI** and **CCFF** are the two modules designed into our hybrid encoder.

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 $\{\mathcal{S}_3, \mathcal{S}_4, \mathcal{S}_5\}$ 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 [42]. 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.* [18], 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 → 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 → 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 → 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 [21] structure for the latter.
- D → 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 $\mathcal{S}_5$ 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 $\mathcal{S}_5$ in variant D, and the experimental results are reported in Table 3 (see row $\mathrm{D}_{\mathcal{S}_5}$). Compared to

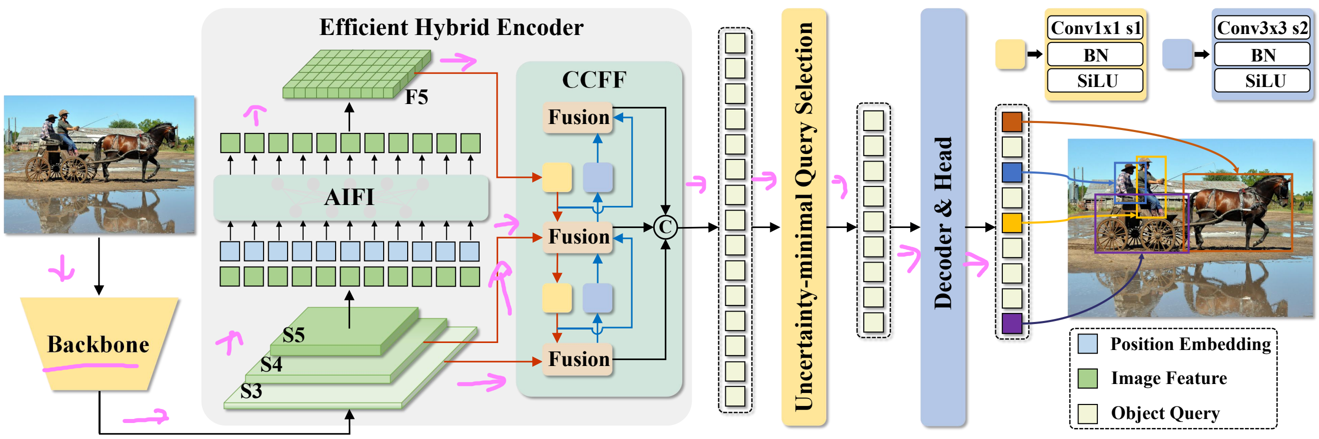

Figure 4. Overview of RT-DETR. We feed the features from the last three stages of the backbone into the encoder. The efficient hybrid encoder transforms multi-scale features into a sequence of image features through the Attention-based Intra-scale Feature Interaction (AIFI) and the CNN-based Cross-scale 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.

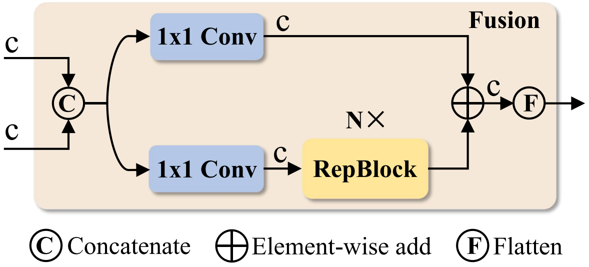

Figure 5. The fusion block in CCFF.

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 1×1 convolutions to adjust the number of channels, N *RepBlock*s composed of RepConv [7] 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:

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

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

4.3. Uncertainty-minimal Query Selection

To reduce the difficulty of optimizing object queries in DETR, several subsequent works [39, 41, 42] 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).

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

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

where $\hat{\boldsymbol{\mathcal{Y}}}$ and $\boldsymbol{\mathcal{Y}}$ denote the prediction and ground truth, $\hat{\boldsymbol{\mathcal{Y}}} = \{\hat{\mathbf{c}}, \hat{\mathbf{b}}\}$, $\hat{\mathbf{c}}$ and $\hat{\mathbf{b}}$ represent the category and bounding box respectively, $\hat{\boldsymbol{\mathcal{X}}}$ 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

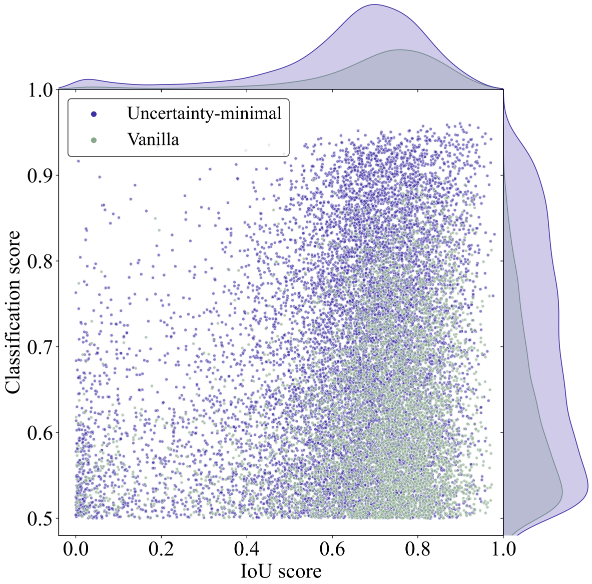

Figure 6. Classification and IoU scores of the selected encoder features. Purple and Green dots represent the selected features from model trained with uncertainty-minimal query selection and vanilla query selection, respectively.

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 [12, 13] 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 [37]) 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 [10], PP-YOLOE [37], YOLOv6v3.0 [15] (hereinafter referred to as YOLOv6), YOLOv7 [35] and YOLOv8 [11]. 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.

Model	Backbone	#Epochs	#Params (M)	GFLOPs	$\text{FPS}_{bs=1}$	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
Real-time Object Detectors											
YOLOv5-L [10]	-	300	46	109	54	49.0	67.3	-	-	-	-
YOLOv5-X [10]	-	300	86	205	43	50.7	68.9	-	-	-	-
PPYOLOE-L [37]	-	300	52	110	94	51.4	68.9	55.6	31.4	55.3	66.1
PPYOLOE-X [37]	-	300	98	206	60	52.3	69.9	56.5	33.3	56.3	66.4
YOLOv6-L [15]	-	300	59	150	99	52.8	70.3	57.7	34.4	58.1	70.1
YOLOv7-L [35]	-	300	36	104	55	51.2	69.7	55.5	35.2	55.9	66.7
YOLOv7-X [35]	-	300	71	189	45	52.9	71.1	57.4	36.9	57.7	68.6
YOLOv8-L [11]	-	-	43	165	71	52.9	69.8	57.5	35.3	58.3	69.8
YOLOv8-X [11]	-	-	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 [36]	R50	50	39	172	-	44.2	64.7	47.5	24.7	48.2	60.6
Anchor-DETR-DC5 [36]	R101	50	-	-	-	45.1	65.7	48.8	25.8	49.4	61.6
Conditional-DETR-DC5 [25]	R50	108	44	195	-	45.1	65.4	48.5	25.3	49.0	62.2
Conditional-DETR-DC5 [25]	R101	108	63	262	-	45.9	66.8	49.5	27.2	50.3	63.3
Efficient-DETR [39]	R50	36	35	210	-	45.1	63.1	49.1	28.3	48.4	59.0
Efficient-DETR [39]	R101	36	54	289	-	45.7	64.1	49.5	28.2	49.1	60.2
SMCA-DETR [8]	R50	108	40	152	-	45.6	65.5	49.1	25.9	49.3	62.6
SMCA-DETR [8]	R101	108	58	218	-	46.3	66.6	50.2	27.2	50.5	63.2
Deformable-DETR [42]	R50	50	40	173	-	46.2	65.2	50.0	28.8	49.2	61.7
DAB-Deformable-DETR [22]	R50	50	48	195	-	46.9	66.0	50.8	30.1	50.4	62.5
DAB-Deformable-DETR++ [22]	R50	50	47	-	-	48.7	67.2	53.0	31.4	51.6	63.9
DN-Deformable-DETR [16]	R50	50	48	195	-	48.6	67.4	52.7	31.0	52.0	63.7
DN-Deformable-DETR++ [16]	R50	50	47	-	-	49.5	67.6	53.8	31.3	52.6	65.4
DINO-Deformable-DETR [41]	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 [41] 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 [41] 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× 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× 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-

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
$\text{D}_{\mathcal{S}_5}$	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-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 AP_S^{val} in the L model (YOLOv8-L) and RT-DETR-R101 is 0.9% AP lower than the highest AP_S^{val} in the X model (YOLOv7-X). We hope that this problem will be addressed in future work.

ID	AP(%)				Latency
	$\textbf{Det}^4$	$\textbf{Det}^5$	$\textbf{Det}^6$	$\textbf{Det}^7$	(ms)
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× configuration.

Discussion. Existing large DETR models [3, 6, 30, 38, 41, 43] have demonstrated impressive performance on COCO `test-dev` [19] 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.

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] 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

[8] 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

[9] 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

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

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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

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

[26] 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

[27] 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

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

[29] 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

[30] 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

[31] 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

[32] 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

[33] 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

[34] 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

[35] 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

[36] 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

[37] 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

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

[39] 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

[40] 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

[41] 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

[42] 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

[43] 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