DETRs Beat YOLOs on Real-time Object Detection

Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui
Yuning Du, Qingqing Dang, Yi Liu
Baidu Inc.
{lvwenyu01, xushangliang, zhaoyian, wangguanzhong} @baidu.com

Abstract

*Recently, end-to-end transformer-based detectors (DETRs) have achieved remarkable performance. However, the issue of the high computational cost of DETRs has not been effectively addressed, limiting their practical application and preventing them from fully exploiting the benefits of no post-processing, such as non-maximum suppression (NMS). In this paper, we first analyze the influence of NMS in modern real-time object detectors on inference speed, and establish an end-to-end speed benchmark. To avoid the inference delay caused by NMS, we propose a **R**eal-**T**ime **DE**tection **TR**ansformer (RT-DETR), the first real-time end-to-end object detector to our best knowledge. Specifically, we design an efficient hybrid encoder to efficiently process multi-scale features by decoupling the intra-scale interaction and cross-scale fusion, and propose IoU-aware query selection to improve the initialization of object queries. In addition, our proposed detector supports flexibly adjustment of the inference speed by using different decoder layers without the need for retraining, which facilitates the practical application of real-time object detectors. Our RT-DETR-L achieves* 53.0% *AP on COCO* `val2017` *and* 114 *FPS on T4 GPU, while RT-DETR-X achieves* 54.8% *AP and* 74 *FPS, outperforming all YOLO detectors of the same scale in both speed and accuracy. Furthermore, our RT-DETR-R50 achieves* 53.1% *AP and* 108 *FPS, outperforming DINO-Deformable-DETR-R50 by* 2.2% *AP in accuracy and by about* 21 *times in FPS. Source code and pretrained models will be available at PaddleDetection*[1].

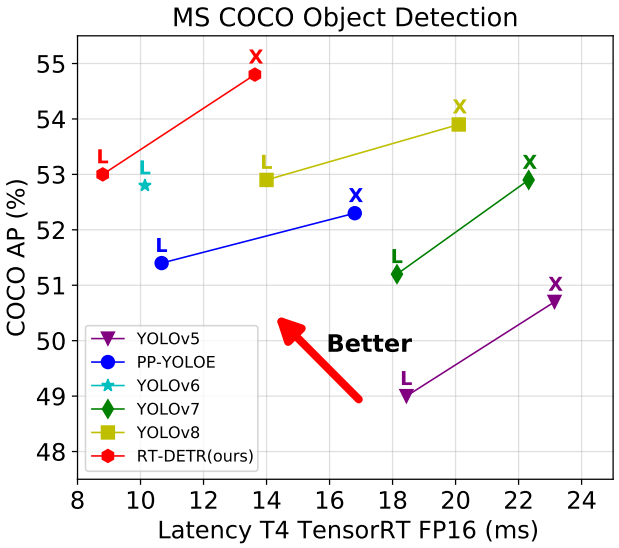

Figure 1: Compared to other real-time object detectors, our proposed detector achieves state-of-the-art performance in both speed and accuracy.

1. Introduction

Object detection is a fundamental vision task that involves identifying and localizing objects in an image. There are two typical architectures for modern object detectors: CNN-based and Transformer-based. Over the past few years, there has been extensive research into CNN-based object detectors. The architecture of these detectors has evolved from the initial two-stage [9, 26, 3] to one-stage [19, 31, 1, 10, 22, 13, 36, 14, 7, 33, 11], and two detection paradigms, anchor-based [19, 22, 13, 10, 33] and anchor-free [31, 7, 36, 14, 11], have emerged. These studies have made significant progress in both detection speed and accuracy. The Transformer-based object detectors (DETRs) [4, 29, 34, 43, 23, 35, 20, 16, 40] have received extensive attention from the academia since it was proposed due to its elimination of various hand-crafted components, such as non-maximum suppression (NMS). This architecture greatly simplifies the pipeline of object detection and realizes end-to-end object detection.

Real-time object detection is a significant research field and has a wide range of applications, such as object tracking [39, 42], video surveillance [24], autonomous driving [2, 38], *etc*. Existing real-time detectors generally

[1] https://github.com/PaddlePaddle/PaddleDetection

adopt a CNN-based architecture, which achieves a reasonable trade-off in detection speed and accuracy. However, these real-time detectors usually require NMS for post-processing, which is usually difficult to optimize and not robust enough, resulting in delays in the inference speed of the detectors. Recently, owing to the efforts of researchers in accelerating training convergence and reducing optimization difficulty, transformer-based detectors have achieved remarkable performance. However, the issue of the high computational cost of DETRs has not been effectively addressed, which limits the practical application of DETRs and results in an inability to take full advantage of their benefits. This means that although the object detection pipeline is simplified, it is difficult to realize real-time object detection due to the high computational cost of the model itself. The above questions naturally inspire us to consider whether we can extend DETR to real-time scenarios, taking full advantage of end-to-end detectors to avoid the delay caused by NMS on real-time detectors.

To achieve the above goals, we rethink DETR and conduct detailed analysis and experiments on its key components to reduce unnecessary computational redundancy. Specifically, we find that although the introduction of multi-scale features is beneficial in accelerating the training convergence and improving performance [43], it also leads to a significant increase in the length of the sequence feed into encoder. As a result, the transformer encoder becomes the computational bottleneck of the model due to the high computational cost. To achieve real-time object detection, we design an efficient hybrid encoder to replace the original transformer encoder. By decoupling the intra-scale interaction and cross-scale fusion of multi-scale features, the encoder can efficiently process features with different scales. Furthermore, previous works [35, 20] showed that the object query initialization scheme of the decoder is crucial for the detection performance. To further improve the performance, we propose IoU-aware query selection, which provides higher quality initial object queries to the decoder by providing IoU constraints during training. In addition, our proposed detector supports flexibly adjustment of the inference speed by using different decoder layers without the need for retraining, which benefits from the design of the decoder in the DETR architecture and facilitates the practical application of the real-time detector.

In this paper, we propose a **R**eal-**T**ime **DE**tection **TR**ansformer (RT-DETR), the first real-time end-to-end object detector to our best knowledge. RT-DETR not only outperforms current state-of-the-art real-time detector in terms of accuracy and speed, but also requires no post-processing, so the inference speed of the detector is not delayed and remains stable, fully exploiting the advantage of end-to-end detection pipeline. Our proposed RT-DETR-L achieves 53.0% AP on COCO `val2017` and 114 FPS on NVIDIA Tesla T4 GPU, while RT-DETR-X achieves 54.8% AP and 74 FPS, outperforming all YOLO detectors of the same scale in both speed and accuracy. Thus, our RT-DETR becomes a new SOTA for real-time object detection, as shown in Fig. 1. Furthermore, our proposed RT-DETR-R50 achieves 53.1% AP and 108 FPS, while RT-DETR-R101 achieves 54.3% AP and 74 FPS. Among them, 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).

The main contributions of this paper are summarized as follows: (i) we propose the first real-time end-to-end object detector, which not only outperforms current state-of-the-art real-time detector in terms of accuracy and speed, but also requires no post-processing, so the inference speed of it is not delayed and remains stable; (ii)we analyze the influence of NMS on real-time detectors in detail and draw a conclusion about CNN-based real-time detectors from a post-processing perspective; (iii)our proposed IoU-aware query selection shows excellent performance improvement in our model, which sheds new light on improving the initialization scheme of object queries; (iv)our work provides a feasible solution for the real-time implementation of end-to-end detectors, and the proposed detector can flexibly adjust the model size and the inference speed by using different decoder layers without the need for retraining.

2. Related work

2.1. Real-time Object Detectors.

Through years of continuous development, the YOLO series [25, 1, 32, 22, 13, 10, 7, 36, 14, 33, 11] have become the synonymous with real-time object detectors, which can be roughly classified into two categories: anchor-based [25, 1, 32, 10, 33] and anchor-free [7, 36, 14, 11]. Judging from the performance of these detectors, anchor is no longer the main factor restricting the development of YOLO. However, the aforementioned detectors produce numerous redundant bounding boxes, requiring the utilization of NMS during the post-processing stage to filter them out. Unfortunately, this leads to performance bottlenecks, and the hyperparameters of NMS have a significant impact on the accuracy and speed of the detectors. We believe this is incompatible with the design philosophy of real-time object detectors.

2.2. End-to-end Object Detectors.

End-to-end object detectors [4, 29, 34, 43, 23, 35, 20, 16, 40] are well-known for their streamlined pipelines. Carion *et al.*[4] first propose the end-to-end object detector based on Transformer, named DETR (DEtection TRansformer). It has attracted significant attention due to its distinctive features. Particularly, DETR eliminates the hand-designed anchor and NMS components in the traditional detection

pipeline. Instead, it employs the bipartite matching and directly predicts the one-to-one object set. By adopting this strategy, DETR simplifies the detection pipeline and mitigates the performance bottleneck caused by NMS. Despite its obvious advantages, DETR suffers from two major issues: slow training convergence and hard-to-optimize queries. Many DETR variants have been proposed to address these issues. Concretely, Deformable-DETR [43] accelerates training convergence with multi-scale features by enhancing the efficiency of the attention mechanism. Conditional DETR [23] and Anchor DETR [35] decrease the optimization difficulty of the queries. DAB-DETR [20] introduces 4D reference points and iteratively optimizes the prediction boxes layer by layer. DN-DETR [16] accelerates training convergence by introducing query denoising. DINO [40] builds upon previous works and achieves state-of-the-art result. Although we are continually improving the components of DETR, our goal is not only to further improve the performance of the model, but also to create a real-time, end-to-end object detector.

2.3. Multi-scale Features for Object Detection.

Modern object detectors have demonstrated the significance of utilizing multi-scale features to improve performance, especially for small objects. FPN [18] introduces a feature pyramid network that fuses features from adjacent scales. Subsequent works [21, 8, 30, 10, 14, 33, 11] extend and enhance this structure, and they are widely adopted in real-time object detectors. Zhu *et al.* [43] first introduce multi-scale features into DETR and improve the performance and convergence speed, but this also leads to a significant increase in the computational cost of DETR. Although the deformable attention mechanism alleviates computational cost to some degree, the incorporation of multi-scale features still results in a high computational burden. To address this issue, some works attempt to design the computationally efficient DETR. Efficient DETR [37] reduces the number of encoder and decoder layers by initializing object queries with dense prior. Sparse DETR [27] selectively updates the encoder tokens that are expected to be referenced by the decoder, thereby reducing the computational overhead. Lite DETR [15] enhances the efficiency of encoder by reducing the update frequency of low-level features in an interleaved way. Although these studies have reduced the computational cost of DETR, the goal of these works is not to promote DETR as a real-time detector.

3. End-to-end Speed of Detectors

3.1. Analysis of NMS

NMS is a widely adopted post-processing algorithm in object detection, employed to eliminate overlapping prediction boxes output by the detector. Two hyperparameters are required in NMS: score threshold and IoU threshold. Specially, prediction boxes with scores below the score threshold are directly filtered out, and whenever the IoU of two prediction 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. Therefore, the execution time of NMS primarily depends on the number of input prediction boxes and two hyperparameters.

To verify this opinion, we leverage YOLOv5 (anchor-based) [10] and YOLOv8 (anchor-free) [11] for experiments. We first count the number of prediction boxes remaining after the output boxes is filtered by different score thresholds with the same input image. We sample some scores from 0.001 to 0.25 as thresholds to count the remaining prediction boxes of two detectors and draw them into a histogram, which intuitively reflects that NMS is susceptible to its hyperparameters, as shown in Fig. 2. Furthermore, we take YOLOv8 as an example to evaluate the model accuracy on the COCO `val2017` and the execution time of the NMS operation under different NMS hyperparameters. Note that the NMS post-processing operation we adopt in our experiments refers to TensorRT `efficientNMSPlugin`, which involves mul-

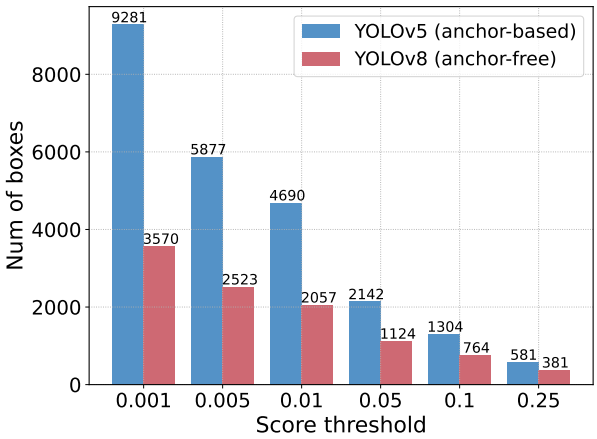

Figure 2: The number of boxes at different score thresholds.

IoU thr. (Score=0.001)	AP (%)	NMS (ms)	Score 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 and score threshold on model accuracy and NMS execution time.

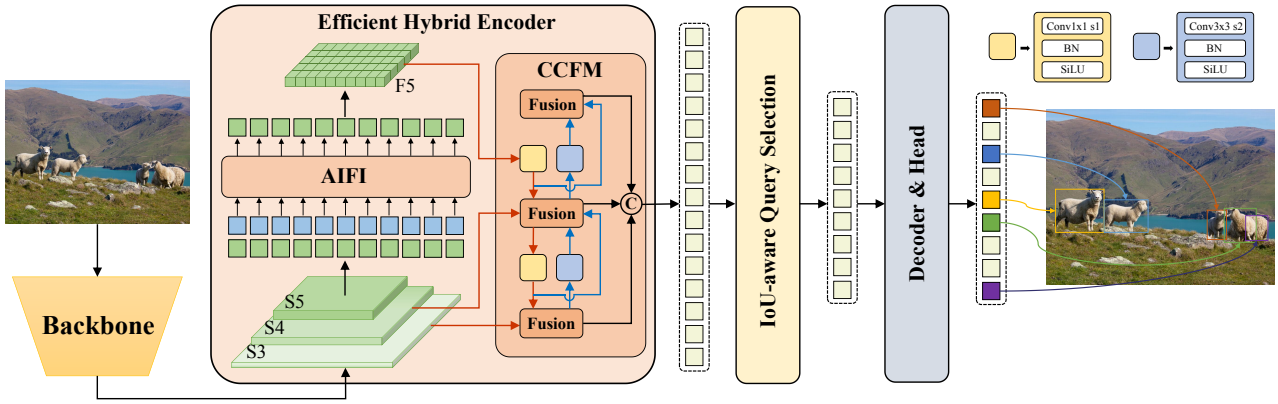

Figure 3: **Overview of RT-DETR.** We first leverage features of the last three stages of the backbone $\{S_3, S_4, S_5\}$ as the input to the encoder. The efficient hybrid encoder transforms multi-scale features into a sequence of image features through intra-scale feature interaction (AIFI) and cross-scale feature-fusion module (CCFM). The IoU-aware query selection is employed to select a fixed number of image features to serve as initial object queries for the decoder. Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries to generate boxes and confidence scores.

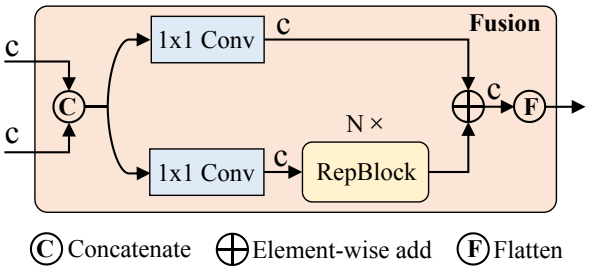

Figure 4: The fusion block in CCFM.

tiple CUDA kernels, including `EfficientNMSFilter`, `RadixSort`, `EfficientNMS`, *etc*., and we only report the execution time of `EfficientNMS` kernel. We test the speed on T4 GPU, and the input image and pre-processing in the above experiments are consistent. The hyperparameters we used and the corresponding results are shown in Tab. 1.

3.2. End-to-end Speed Benchmark

To enable a fair comparison of the end-to-end inference speeds of various real-time detectors, we establish an end-to-end speed test benchmark. Considering that the execution time of NMS can be influenced by the input image, it is necessary to choose a benchmark dataset and calculate the average execution time across multiple images. The benchmark adopts COCO `val2017` as the default dataset, appending the NMS post-processing plugin of TensorRT for real-time detectors that require post-processing. Specifically, we test the average inference time of the detector according to the hyperparameters of the corresponding accuracy taken on benchmark dataset, and excluding IO and Memory-Copy operations. We utilize this benchmark to test the end-to-end speed of anchor-based detectors YOLOv5 [10] and YOLOv7 [33], as well as anchor-free detectors PP-YOLOE [36], YOLOv6 [14] and YOLOv8 [11] on T4 GPU. The test results are shown in Tab. 2. According to the results, we conclude that *for real-time detectors that require NMS post-processing, anchor-free detectors outperform anchor-based detectors with equivalent accuracy because the former takes significantly less post-processing time than the latter*, which was neglected in previous works. The reason for this phenomenon is that anchor-based detectors produce more predicted boxes than anchor-free detectors (three times more in our tested detectors).

4. The Real-time DETR

4.1. Model Overview

The proposed RT-DETR consists of a backbone, a hybrid encoder and a transformer decoder with auxiliary prediction heads. The overview of the model architecture is illustrated in Fig. 3. Specifically, we leverage the output features of the last three stages of the backbone $\{S_3, S_4, S_5\}$ as the input to the encoder. The hybrid encoder transforms multi-scale features into a sequence of image features through intra-scale interaction and cross-scale fusion (described in Sec. 4.2). Subsequently, the IoU-aware query selection is employed to select a fixed number of image features from the encoder output sequence to serve as initial object queries for the decoder (described in Sec. 4.3). Finally, the decoder with auxiliary prediction heads iteratively optimizes object queries

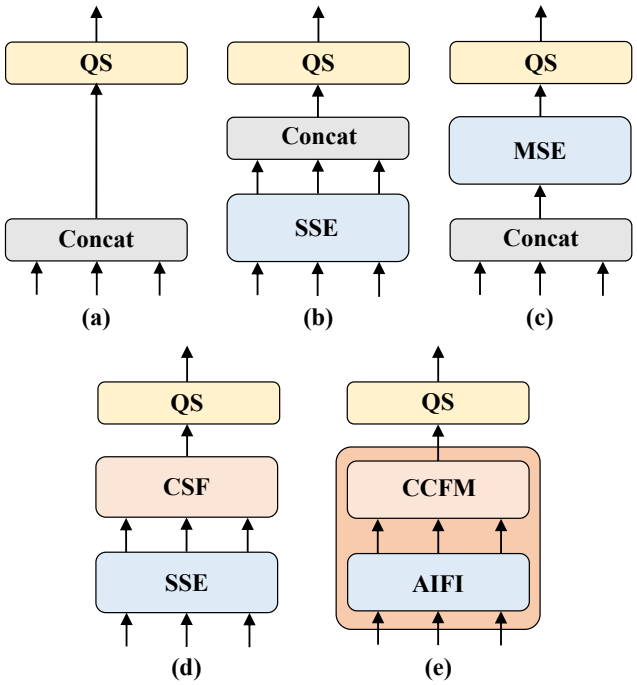

Figure 5: The set of variants with different types of encoders. **QS** represents the query selection, **SSE** represents the single-scale encoder, **MSE** represents the multi-scale encoder, and **CSF** represents cross-scale fusion.

to generate boxes and confidence scores.

4.2. Efficient Hybrid Encoder

Computational bottleneck analysis. To accelerate training convergence and improve performance, Zhu *et al*. [43] suggest introducing multi-scale features and propose the deformable attention mechanism to reduce computation. However, although the improvement in the attention mechanism reduces the computational overhead, the sharply increased length of input sequence still causes the encoder to become a computational bottleneck, hampering the real-time implementation of DETR. As reported in [17], the encoder accounts for 49% of the GFLOPs but contributes only 11% of the AP in Deformable-DETR [43]. To overcome this obstacle, we analyze the computational redundancy present in the multi-scale transformer encoder and design a set of variants to prove that the simultaneous interaction of intra-scale and cross-scale features is computationally inefficient.

High-level features are extracted from low-level features that contain rich semantic information about objects in the image. Intuitively, it is redundant to perform feature interaction on the concatenate multi-scale features. To verify this opinion, we rethink the encoder structure and design a range of variants with different encoders, as shown in Fig. 5. The set of variants gradually improves model accuracy while significantly reducing computational cost by decoupling multi-scale feature interaction into two-step operations of intra-scale interaction and cross-scale fusion (detailed indicators refer to Tab. 3). We first remove the multi-scale transformer encoder in DINO-R50 [40] as baseline A. Next, different forms of encoder are inserted to produce a series of variants based on baseline A, elaborated as follows:

- A $\rightarrow$ B: Variant B inserts a single-scale transformer encoder, which uses one layer of transformer block. The features of each scale share the encoder for intra-scale feature interaction and then concatenate the output multi-scale features.
- B $\rightarrow$ C: Variant C introduces cross-scale feature fusion based on B and feeds the concatenate multi-scale features into the encoder to perform feature interaction.
- C $\rightarrow$ D: Variant D decouples the intra-scale interaction and cross-scale fusion of multi-scale features. First, the single-scale transformer encoder is employed to perform intra-scale interaction, then a PANet-like [21] structure is utilized to perform cross-scale fusion.
- D $\rightarrow$ E: Variant E further optimizes the intra-scale interaction and cross-scale fusion of multi-scale features based on D, adopting an efficient hybrid encoder designed by us (see below for details).

Hybrid design. Based on the above analysis, we rethink the structure of the encoder and propose a novel *Efficient Hybrid Encoder*. As shown in Fig. 3, the proposed encoder consists of two modules, the Attention-based Intra-scale Feature Interaction (AIFI) module and the CNN-based Cross-scale Feature-fusion Module (CCFM). AIFI further reduces computational redundancy based on variant D, which only performs intra-scale interaction on S_5. We argue that applying the self-attention operation to high-level features with richer semantic concepts can capture the connection between conceptual entities in the image, which facilitates the detection and recognition of objects in the image by subsequent modules. Meanwhile, the intra-scale interactions of lower-level features are unnecessary due to the lack of semantic concepts and the risk of the risk of duplication and confusion with interactions of high-level features. To verify this view, we only perform the intra-scale interaction on S_5 in variant D, and the experimental results are reported in Tab. 3, see row D_{S_5}. Compared to the vanilla variant D, D_{S_5} significantly reduces the latency (35% faster) but delivers an improvement in accuracy (0.4% AP higher). This conclusion is crucial for the design of real-time detectors. CCFM is also optimized based on variant D, inserting several fusion blocks composed of convolutional layers

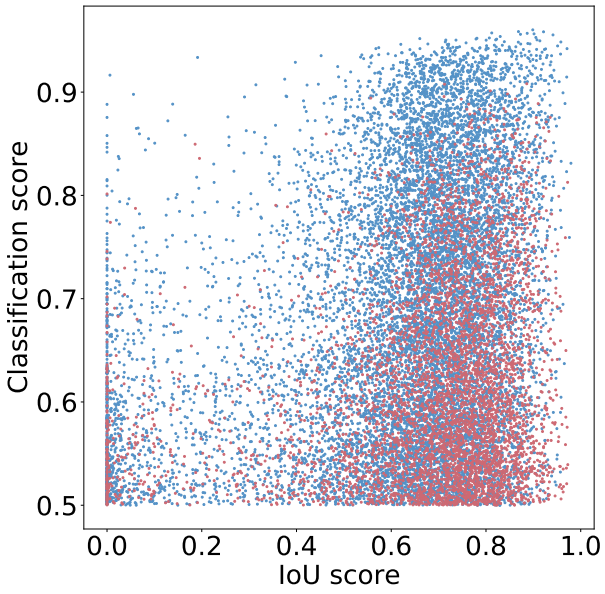

Figure 6: We calculate the classification scores and IoU scores of the encoder features selected by the query selection on `val2017`, and visualize the scatter plot with classification scores greater than 0.5. The red and blue points are calculated from the model trained by applying the vanilla query selection and the proposed IoU-aware query selection, respectively.

into the fusion path. The role of the fusion block is to fuse the adjacent features into a new feature, and its structure is shown in Fig. 4. The fusion block contains N *RepBlock*s, and the two-path outputs are fused by element-wise add. We can formulate this process as follows:

$$\begin{aligned} \mathbf{Q} = \mathbf{K} = \mathbf{V} &= Flatten(S_5) \\ F_5 &= Reshape(Attn(\mathbf{Q}, \mathbf{K}, \mathbf{V})) \\ Output &= CCFM(\{S_3, S_4, F_5\}) \end{aligned} \tag{1}$$

where $Attn$ represents the multi-head self-attention, and $Reshape$ represents restoring the shape of the feature to the same as S_5, which is the inverse operation of $Flatten$.

4.3. IoU-aware Query Selection

The object queries in DETR are a set of learnable embeddings, which are optimized by the decoder and mapped to classification scores and bounding boxes by the prediction head. However, these object queries are difficult to interpret and optimize because they have no explicit physical meaning. Subsequent works [35, 20, 43, 37, 40] improve the initialization of object query and extend it to content query and position query (anchor). Among them, [43, 37, 40] all propose query selection schemes, which have in common that they utilize the classification score to select top K features from the encoder to initialize object queries (or only position queries [40]). However, due to the inconsistent distribution of classification score and location confidence, some predicted boxes have high classification scores but are not close to GT boxes, which results in boxes with high classification scores and low IoU scores being selected, while boxes with low classification scores and high IoU scores are discarded. This impairs the performance of the detector. To address this issue, we propose IoU-aware query selection by constraining the model to produce high classification scores for features with high IoU scores and low classification scores for features with low IoU scores during training. Therefore, the prediction boxes corresponding to the top K encoder features selected by the model according to the classification score have both high classification scores and high IoU scores. We reformulate the optimization objective of the detector as follows:

$$\begin{aligned} \mathcal{L}(\hat{y}, y) &= \mathcal{L}_{box}(\hat{b}, b) + \mathcal{L}_{cls}(\hat{c}, \hat{b}, y, b) \\ &= \mathcal{L}_{box}(\hat{b}, b) + \mathcal{L}_{cls}(\hat{c}, c, IoU) \end{aligned} \tag{2}$$

where $\hat{y}$ and y denote prediction and ground truth, $\hat{y} = \{\hat{c}, \hat{b}\}$ and $y = \{c, b\}$, c and b represent categories and bounding boxes, respectively. We introduce the IoU score into objective function of the classification branch (similar to VFL [41]) to realize the consistency constraint on the classification and localization of positive samples.

Effectiveness analysis. To analyze the effectiveness of the proposed IoU-aware query selection, we visualize the classification scores and IoU scores of the encoder features selected by the query selection on `val2017`, as shown in Fig. 6. Specifically, we first select the top K ($K = 300$ in our experiment) encoder features according to the classification scores, and then visualize the scatter plot with classification scores greater than 0.5. The red and blue points are calculated from the model trained by applying the vanilla query selection and IoU-aware query selection, respectively. The closer the point is to the top right of the figure, the higher the quality of the corresponding feature, *i.e*. the classification label and bounding box are more likely to describe the real object in an image. According to the visualization results, we found that the most striking feature is that a large number of blue points are concentrated in the top right of the figure, while red points are concentrated in the bottom right. This shows that the model trained with IoU-aware query selection can produce more high-quality encoder features.

Furthermore, we quantitatively analyze the distribution characteristics of the two types of points. There are 138% more blue points than red points in the figure, *i.e*. more red points with a classification score less than or equal to 0.5, which can be considered as low-quality features. We then analyze the IoU scores of features with classification scores

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[10]	-	300	46	109	54	49.0	67.3	-	-	-	-
YOLOv5-X[10]	-	300	86	205	43	50.7	68.9	-	-	-	-
PPYOLOE-L[36]	CSPRepResNet	300	52	110	94	51.4	68.9	55.6	31.4	55.3	66.1
PPYOLOE-X[36]	CSPRepResNet	300	98	206	60	52.3	69.9	56.5	33.3	56.3	66.4
YOLOv6-L[14]	-	300	59	150	99	52.8	70.3	57.7	34.4	58.1	70.1
YOLOv7-L[33]	-	300	36	104	55	51.2	69.7	55.5	35.2	55.9	66.7
YOLOv7-X[33]	-	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 [35]	R50	50	39	172	-	44.2	64.7	47.5	24.7	48.2	60.6
Anchor-DETR-DC5 [35]	R101	50	-	-	-	45.1	65.7	48.8	25.8	49.4	61.6
Conditional-DETR-DC5 [23]	R50	108	44	195	-	45.1	65.4	48.5	25.3	49.0	62.2
Conditional-DETR-DC5 [23]	R101	108	63	262	-	45.9	66.8	49.5	27.2	50.3	63.3
Efficient-DETR [37]	R50	36	35	210	-	45.1	63.1	49.1	28.3	48.4	59.0
Efficient-DETR [37]	R101	36	54	289	-	45.7	64.1	49.5	28.2	49.1	60.2
SMCA-DETR [6]	R50	108	40	152	-	45.6	65.5	49.1	25.9	49.3	62.6
SMCA-DETR [6]	R101	108	58	218	-	46.3	66.6	50.2	27.2	50.5	63.2
Deformable-DETR [43]	R50	50	40	173	-	46.2	65.2	50.0	28.8	49.2	61.7
DAB-Deformable-DETR [20]	R50	50	48	195	-	46.9	66.0	50.8	30.1	50.4	62.5
DN-Deformable-DETR [16]	R50	50	48	195	-	48.6	67.4	52.7	31.0	52.0	63.7
DAB-Deformable-DETR++ [16]	R50	50	47	-	-	48.7	67.2	53.0	31.4	51.6	63.9
DN-Deformable-DETR++ [16]	R50	50	47	-	-	49.5	67.6	53.8	31.3	52.6	65.4
DINO-Deformable-DETR [40]	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	R50	72	42	136	**108**	**53.1**	**71.3**	**57.7**	34.8	58.0	70.0
RT-DETR-R101	R101	72	76	259	**74**	**54.3**	**72.7**	58.6	36.0	58.8	**72.1**
RT-DETR-L	HGNetv2	72	32	110	**114**	**53.0**	**71.6**	57.3	34.6	57.3	**71.2**
RT-DETR-X	HGNetv2	72	67	234	**74**	**54.8**	**73.1**	**59.4**	35.7	**59.6**	**72.9**

Table 2: Main results. Real-time detectors and our RT-DETR share a common input size of 640, and end-to-end detectors use an input size of (800, 1333). The end-to-end speed results are reported on T4 GPU with TensorRT FP16 using official pre-trained models followed the method proposed in Sec. 3. (**Note:** We do not test the speed of DETRs, except for DINO-Deformable-DERT for comparison, as they are not real time detectors.)

greater than 0.5, and we find that there are 120% more blue points than red points with IoU scores greater than 0.5. Quantitative results further demonstrate that the IoU-aware query selection can provide more encoder features with accurate classification (high classification scores) and precise location (high IoU scores) for object queries, thereby improving the accuracy of the detector. The detailed quantitative results are presented in Sec. 5.4.

4.4. Scaled RT-DETR

To provide a scalable version of RT-DETR, we replace the ResNet [12] backbone with HGNetv2. We scale the backbone and hybrid encoder together using a depth multiplier and a width multiplier. Thus, we get two versions of RT-DETR with different numbers of parameters and FPS. For our hybrid encoder, we control the depth multiplier and width multiplier by adjusting the number of *RepBlock*s in CCFM and the embedding dimension of the encoder, respectively. It is worth noting that our proposed RT-DETR of different scales maintains a homogeneous decoder, which facilitates the distillation of light detectors using high-precision large DETR models. This would be an explorable future direction.

5. Experiments

5.1. Setups

Dataset. We perform extensive experiments on the Microsoft COCO dataset to validate the proposed detector. For the ablation study, we train on COCO `train2017` and validate on COCO `val2017` dataset. We use the standard COCO AP metric with a single scale image as input.

Implementation Details. We use ResNet [12] and HGNetv2 series pretrained on ImageNet [28] from PaddleClas[2] [5] as our backbone. The AIFI consists of 1 transformer layer and the fusion block in CCMF consists of 3 *RepBlock*s for the base model by default. In IoU-aware query selection, we select the top 300 encoder features to initialize object queries of the decoder. The training strategy and hyperparameters of the decoder almost follow DINO [40]. We train our detectors using

[2] https://github.com/PaddlePaddle/PaddleClas

AdamW optimizer with $base\_learning\_rate = 0.0001$, $weight\_decay = 0.0001$, $global\_gradient\_clip\_norm = 0.0001$, and $linear\_warmup\_steps = 2000$. Learning rates of backbone setting follows [4]. We also use exponential moving average (EMA) with $ema\_decay = 0.9999$. The $1\times$ configuration means that the total epoch is 12, if not specified, all ablation experiments use $1\times$. The final results reported use a $6\times$ configuration. The data augmentation includes *random_*{*colour distort, expand, crop, flip, resize*} operations, follow [36].

5.2. Comparison with SOTA

Tab. 2 compares the proposed RT-DETR with other real-time and end-to-end object detectors. Our proposed RT-DETR-L achieves 53.0% AP and 114 FPS, while RT-DETR-X achieves 54.8% AP and 74 FPS, outperforming all YOLO detectors of the same scale in both speed and accuracy. Furthermore, our proposed RT-DETR-R50 achieves 53.1% AP and 108 FPS, while RT-DETR-R101 achieves 54.3% AP and 74 FPS, outperforming the state-of-the-art end-to-end detector of the same backbone in both speed and accuracy.

Compared to real-time detectors. For a fair comparison, we compare the speed and accuracy of the scaled RT-DETR with current real-time detectors in an end-to-end setting (speed test method refers to Sec. 3.2). We compare the scaled RT-DETR with YOLOv5 [10], PP-YOLOE [36], YOLOv6v3.0 (hereinafter referred to as YOLOv6) [14], YOLOv7 [33] and YOLOv8 [11] in Tab. 2. Compared to YOLOv5-L / PP-YOLOE-L / YOLOv7-L, RT-DETR-L significantly improves accuracy by 4.0% / 1.6% / 1.8% AP, increases FPS by 111.1% / 21.3% / 107.3%, and reduces the number of parameters by 30.4% / 38.5% / 11.1%. Compared to YOLOv5-X / PP-YOLOE-X / YOLOv7-X, RT-DETR-X improves accuracy by 4.1% / 2.5% / 1.9% AP, increases FPS by 72.1% / 23.3% / 64.4%, and reduces the number of parameters by 22.1% / 31.6% / 5.6%. Compared to YOLOv6-L / YOLOv8-L, RT-DETR-L achieves 0.2% / 0.1% AP improvement in accuracy, 15.2% / 60.6% FPS increase in speed, and 45.8% / 25.6% reduction in the number of parameters. Compared to YOLOv8-X, RT-DETR-X achieves a 0.9% AP improvement in accuracy, a 48.0% FPS increase in speed, and a 1.5% reduction in the number of parameters.

Compared to end-to-end detectors. Tab. 2 shows that RT-DETR achieves the state-of-the-art performance in all end-to-end detectors with the same backbone. Compared to DINO-Deformable-DETR-R50 [40], RT-DETR-R50 significantly improves the accuracy by 2.2% AP (53.1% AP vs. 50.9% AP) and the speed by 21 times (108 FPS vs 5 FPS), and reduces the number of parameters by 10.6%. Compared to SMCA-DETR-R101 [6], RT-DETR-R101 significantly improves accuracy by 8.0% AP.

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_{S_5}	46.8	35	7.9
E	47.9	42	9.3

Table 3: Results of analytical experiment that decouple multi-scale feature fusion into two-step operations of intra-scale interaction and cross-scale fusion.

5.3. Ablation Study on Hybrid Encoder

To verify the correctness of our analysis about the encoder and the effectiveness of the proposed hybrid encoder, we evaluate the indicators of the set of variants designed in Sec. 4.2, including AP, number of parameters and latency on T4 GPU. The experimental results are shown in Tab. 3. Variant B delivers a 1.9% AP improvement over A, while increasing the number of parameters by 3% and the latency by 54%. This proves that the intra-scale feature interaction is significant but the vanilla transformer encoder is expensive. Variant C delivers 0.7% AP improvement over B and keeps the number of parameters unchanged, while the latency increases by 20%. This shows that the cross-scale feature fusion is also necessary. Variant D delivers 0.8% AP improvement over C, while increasing the number of parameters by 9% but reducing latency by 8%. This suggests that decoupling intra-scale interaction and cross-scale fusion can reduce computation while improving accuracy. Compared to the vanilla variant D, D_{S_5} reduces the latency by 35% but delivers 0.4% AP improvement. This proves that the intra-scale interactions of lower-level features are unnecessary. Finally, variant E equipped with our proposed hybrid encoder 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 computationally efficient.

5.4. Ablation Study on IoU-aware Query Selection

We conduct an ablation study on IoU-aware query selection, and the quantitative experimental results are shown in 4. The query selection we adopt selects the top K ($K = 300$) encoder features as the content queries according to the classification scores, and the bounding boxes corresponding to these selected features are employed as initial position queries. We compare the encoder features selected by the two query selections on `val2017` and calculate the proportions of classification scores greater than 0.5 and both scores greater than 0.5, corresponding to columns "Prop_{cls}" and "Prop_{both}" respectively. The results show

Query selection	AP(%)	Prop$_{cls}$(%)	Prop$_{both}$(%)
Vanilla	47.9	0.35	0.30
IoU-aware	48.7	0.82	0.67

Table 4: Results of the ablation study on IoU-aware query selection. **Prop**$_{cls}$ and **Prop**$_{both}$ represent the proportion of classification scores greater than 0.5 and both scores greater than 0.5 respectively.

that the encoder features selected by IoU-aware query selection not only increase the proportion of high classification scores (0.82% vs 0.35%), but also provide more features with high classification scores and high IoU scores (0.67% vs 0.30%). We also evaluate the accuracy of the detectors trained with the two types of query selection on `val2017`, where the IoU-aware query selection achieves an improvement of 0.8% AP (48.7% AP vs 47.9% AP).

5.5. Ablation Study on Decoder

Tab. 5 shows the accuracy and speed of each decoder layer of RT-DETR with different decoder layers. When the number of decoder layers is 6, the detector achieves the best accuracy of 53.1% AP. We also analyze the influence of each decoder layer on the inference speed and conclude that each decoder layer consumes about 0.5 ms. Furthermore, we find that the difference in accuracy between adjacent layers of the decoder gradually decreases as the index of the decoder layer increases. Taking the 6-layer decoder as an example, using 5-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 flexibly adjustment of the inference speed by using different decoder layers without the need for retraining for inference, which facilitates the practical application of the real-time detector.

6. Conclusion

In this paper, we propose RT-DETR, the first real-time end-to-end detector to our best knowledge. We first perform a detailed analysis of NMS and establish an end-to-end speed benchmark to verify the fact that the inference speed of current real-time detectors is delayed by NMS. We also conclude from the analysis of NMS that anchor-free detectors outperform anchor-based detectors with the same accuracy. To avoid delays caused by NMS, we design a real-time end-to-end detector that includes two key improved components: a hybrid encoder that can efficiently process multi-scale features and IoU-aware query selection that improves the initialization of object queries. Extensive experiments demonstrate that RT-DETR achieves state-of-the-art performance in both speed and accuracy compared

ID	AP (%)				Latency(ms)
	Det^4	**Det**^{5}	**Det**^{6}	**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** represents the index of the decoder layer and **AP** represents the model accuracy obtained by different decoder layers. **Det**^{k} represents the detector with k decoder layers. Results are reported on RT-DETR-R50 with $6\times$ schedule setting.

to other real-time detectors and end-to-end detectors of similar size. In addition, Our proposed detector supports flexibly adjustment of the inference speed by using different decoder layers without the need for retraining, which facilitates the practical application of real-time object detectors. We hope that this work can be put into practice and provide inspiration for researchers.

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] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 1

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 1, 2, 7, 8

[5] 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. 7

[6] 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, 8

[7] 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

[8] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 3

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[10] Jocher Glenn. Yolov5 release v7.0. https://github.com/ultralytics/yolov5/tree/v7.0, 2022. 1, 2, 3, 4, 7, 8

[11] Jocher Glenn. Yolov8. https://github.com/ultralytics/ultralytics/tree/main, 2023. 1, 2, 3, 4, 7, 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[13] 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

[14] 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, 4, 7, 8

[15] 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. *arXiv preprint arXiv:2303.07335*, 2023. 3

[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, 3, 7

[17] 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. 5

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[20] 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. *arXiv preprint arXiv:2201.12329*, 2022. 1, 2, 3, 6, 7

[21] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. 3, 5

[22] 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

[23] 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, 2, 3, 7

[24] 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

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

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1

[27] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. *arXiv preprint arXiv:2111.14330*, 2021. 3

[28] 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. 7

[29] 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, 2

[30] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 3

[31] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 1

[32] 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

[33] 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. *arXiv preprint arXiv:2207.02696*, 2022. 1, 2, 3, 4, 7, 8

[34] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15849–15858, 2021. 1, 2

[35] 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*, volume 36, pages 2567–2575, 2022. 1, 2, 3, 6, 7

[36] 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, 4, 7, 8

[37] 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. 3, 6, 7

[38] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 1

[39] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 659–675. Springer, 2022. 1

[40] 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 *The Eleventh International Conference on Learning Representations*, 2022. 1, 2, 3, 5, 6, 7, 8

[41] Haoyang Zhang, Ying Wang, Feras Dayoub, and Niko Sunderhauf. Varifocalnet: An iou-aware dense object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8514–8523, 2021. 6

[42] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer, 2022. 1

[43] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 1, 2, 3, 5, 6, 7