

# MI-DETR: An Object Detection Model with Multi-time Inquiries Mechanism

Zhixiong Nan<sup>1</sup>, Xianghong Li<sup>1</sup>, Jifeng Dai<sup>2</sup>, Tao Xiang<sup>1\*</sup>

<sup>1</sup> College of Computer Science, Chongqing University, Chongqing, China

<sup>2</sup> Department of Electronic Engineering, Tsinghua University, Beijing, China

nanzx@cqu.edu.cn, 202314021077t@stu.cqu.edu.cn,

daijifeng@tsinghua.edu.cn, txiang@cqu.edu.cn

## Abstract

Based on analyzing the character of cascaded decoder architecture commonly adopted in existing DETR-like models, this paper proposes a new decoder architecture. The cascaded decoder architecture constrains object queries to update in the cascaded direction, only enabling object queries to learn relatively-limited information from image features. However, the challenges for object detection in natural scenes (e.g., extremely-small, heavily-occluded, and confusingly mixed with the background) require an object detection model to fully utilize image features, which motivates us to propose a new decoder architecture with the parallel **Multi-time Inquiries (MI)** mechanism. **MI** mechanism is very simple, enabling object queries to parallelly perform multi-time inquiries to learn more comprehensive information from image features. Our **MI** based model, **MI-DETR**, outperforms all existing DETR-like models on COCO benchmark under different backbones and training epochs, achieving +2.3 AP and +0.6 AP improvements compared to the most representative model DINO and SOTA model Relation-DETR under ResNet-50 backbone.

## 1. Introduction

The first widely-acknowledged object detection model, Viola Jones detector [38], is proposed in 2001. Fueled by the remarkable R-CNN [9] and DETR [3], CNN based works [1, 5, 19, 22, 30, 31, 37] have significantly pushed forward the object detection technique since 2014, and transformer based DETR-like models [2, 6, 11–13, 15, 16, 20, 21, 23, 26, 40–42, 44, 45, 47, 49] further bring the technique to a new landscape since 2020. A DETR-like model consists of a backbone, a transformer encoder, a transformer decoder, and a prediction head. From the perspective of function, the backbone and encoder are responsible for image features extraction, the decoder utilizes image features

to adapt to the object detection task (i.e., “feature utilization”), and the prediction head predicts locations and categories of objects.

This paper attempts to explore a new decoder architecture to optimize feature utilization. Each transformer decoder layer contains a self-attention (SA), a cross-attention (CA), and a feed-forward network (FFN), while object queries are optimized by interacting with image features. Analogously speaking, it is like that a student (*object queries act like a student who “knows” nothing or a little about the image at the beginning decoder layer*) asks a question to a teacher (*image features could be analogous to a teacher since he “knows” all information in the image*) and obtains the image information based on the teacher’s answer. Therefore, the “SA+CA+FFN” architecture in each decoder layer is analogously termed as “inquiry head” in this paper. The transformer decoder of existing DETR-like models adopts the “cascaded inquiries” architecture. Object queries continuously inquire the image features layer by layer to learn the gradually-refined information in the cascaded direction.

The robustness of this decoder architecture has been fully validated in previous DETR-like models. However, from the perspective of feature utilization, this architecture presents a worth-noting character. The representations of object queries in the next decoder layer are directly decided by that in the current decoder layer [4], thus the update of query representations is constrained in the cascaded direction, which indicates that the cascaded inquiries architecture tends to learn the relatively limited information. In addition, excessively refined information learnt in deep decoder layers of the same cascaded direction might be redundant or even negative. More analysis could be found in experiment and discussion sections.

Due to the diversity and complexity of natural scenes, objects might be extremely-small, heavily-occluded, or confusingly mixed with the background. To adapt to diverse scenes, it is required to fully utilize image features to learn comprehensive information. However, based on

\*Corresponding author.

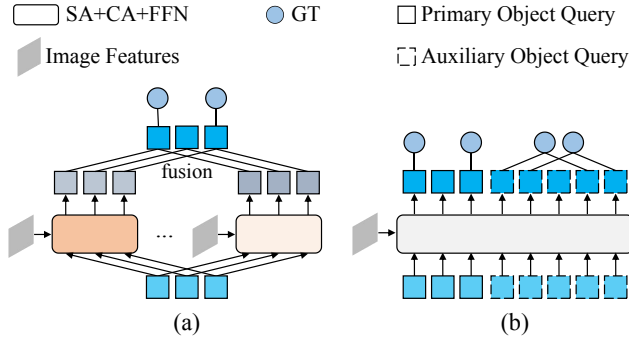


Figure 1. (a) We propose the parallel multi-time inquiries mechanism with parameters-dependent inquiry heads and fusion operation in a decoder layer; (b) In recent proposed DETR-like models, primary and auxiliary queries are concatenated and inputted to the same inquiry head in a decoder layer, thus this kind of parallel architecture is parameters-sharing parallel.

the above analysis, the cascaded multi-time inquiries architecture adopted in existing models tends to learn the relatively limited information. This motivates us to explore a new decoder architecture to optimize feature utilization. Inspired by the development of traditional CNN-based methods, we have noticed that many methods [17, 34–36, 43] enhance feature utilization through reasonable parallel architectures. Therefore, we propose a new decoder architecture with the parallel *Multi-time Inquiries (MI)* mechanism. As shown in Fig. 1a, the architecture is simple, object queries parallelly perform multi-time inquiries through multiple parameters-independent inquiry heads, and the updated object queries are then fused. The parallel *MI* mechanism conducts feature utilization in separate branches, allowing to learn gradually-refined multi-pattern information, and these information can be fused to be comprehensive. Apart from *MI* mechanism, we also design a *U-like Feature Interaction (UFI)* module to further improve feature utilization, which allows image features at each layer of the transformer encoder to serve as Key&Value for the corresponding decoder layer.

It is noted that the proposed decoder architecture is distinct from the parameters-sharing parallel decoder adopted in recent proposed DETR-like models, such as  $\mathcal{H}$ -DETR [15] and Group-DETR [6], whose decoder architecture is illustrated in Fig. 1b. The differences are mainly reflected on two aspects. *First*, from the perspective of motivation, the architecture in Fig. 1b is designed to solve the issue of insufficient positive object queries (*i.e.* the object queries that are supervised by Ground Truths), thus auxiliary object queries and one-to-many supervision are introduced. Differently, our method is proposed to optimize feature utilization by digging multi-pattern information. *Second*, from the perspective of architecture, primary and auxiliary object queries in Fig. 1b are concatenated together and

then inputted into the same inquiry head, thus the parameters are sharing and simultaneously-updating. In our opinion, this type of parameters-sharing parallel architecture, from the perspective of feature utilization, is the pseudo parallel architecture, since sharing parameters make two kinds of queries learn the same pattern of information. In contrast, our parameters-independent parallel architecture allows object queries to learn multi-pattern information to improve feature utilization. More analysis could be found in the discussion section.

The proposed decoder architecture with the parallel *MI* mechanism is simple and friendly, and it can be easily plugged into existing DETR-like models. In the experiments, by plugging into the most representative model DINO [44] and SOTA Relation-DETR [12], impressive performance improvements are obtained, achieving +2.3 AP and +0.6 AP improvements, respectively, under ResNet-50 backbone and the condition achieving convergence. To the best of our knowledge, these are the best results among existing DETR-like models.

The contributions are as follows: *i)* This paper proposes a new decoder architecture to optimize feature utilization by digging multi-pattern information, and our **MI-DETR** (*i.e.*, **Multi-time Inquiries DETR**) achieves the best performance among existing DETR-like models; *ii)* The proposed decoder architecture is simple and friendly to plug into existing DETR-like models.

## 2. Related Work

Original DETR [3] suffers from slow convergence, requiring 500 epochs to achieve 43.3 AP. In recent years, a large amount of works have focused on improving detection performance and accelerating training convergence of DETR, bringing the significant performance breakthrough to 51.7 AP within 12 epochs.

In the early stage, most methods focus on optimizing the representations of object queries. Object queries are learnable vectors that are used to capture information about objects in the image. Some works [11, 41, 44, 48, 49] have noticed that the design of the initial object queries will significantly impact the convergence speed and propose two-stage initialization to select top K encoder features from the last encoder layer to enhance the initial object queries. Furthermore, many studies [14, 20, 23, 26, 46] consider introducing prior knowledge into object queries. In addition, the attention mechanism is also a key area of interest in early researches. Cross-attention performs the interaction between object queries and image features, which enables the object queries to concentrate on related regions of the image. Several methods [7, 8, 40, 42, 49] have optimized the cross-attention in decoder to improve the ability of object queries to quickly locate relevant regions.

In the past two years, many researchers have paid atten-

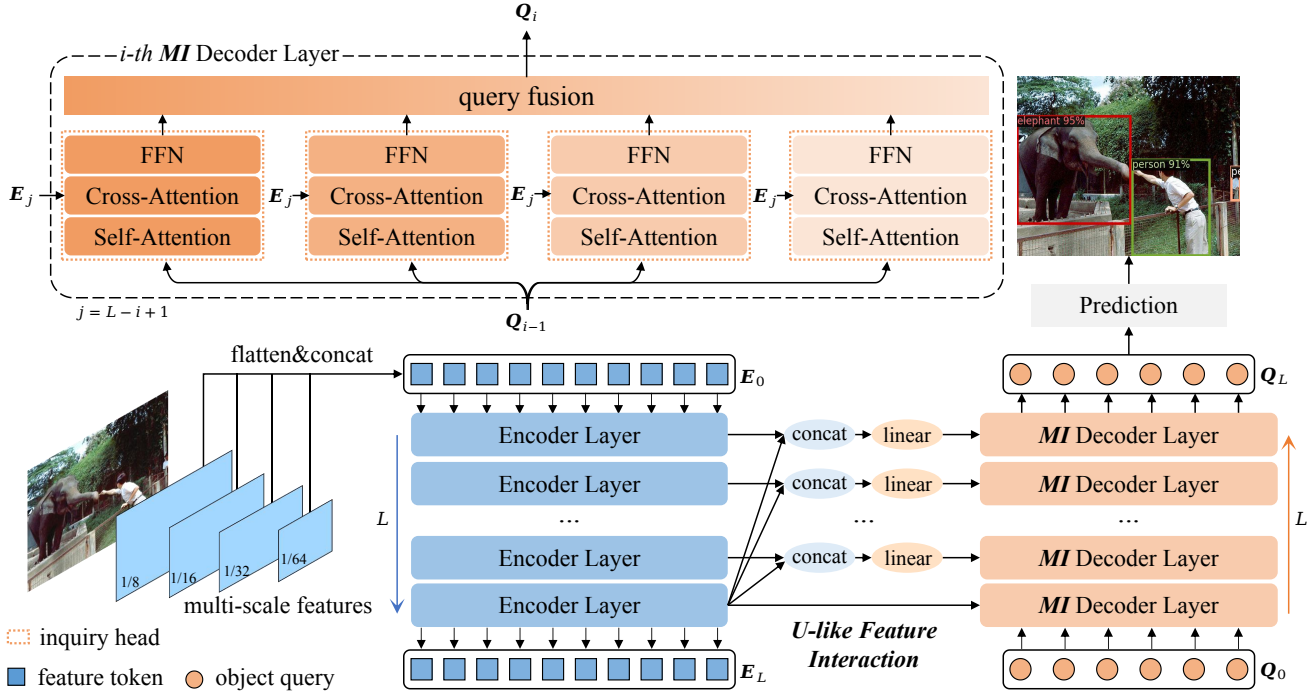


Figure 2. The overview of **MI-DETR**. The main novelty is that **MI-DETR** uses *Multi-time Inquiries (MI)* decoder layers to replace the traditional decoder layers adopted in previous DETR-like models. Backbone and  $L$ -layer transformer encoder extract the image features  $E = \{E_0, \dots, E_L\}$ . For  $i$ -th *MI* decoder layer, the input is object queries  $Q_{i-1}$  and the output is  $Q_i$ . For inquiry heads in  $i$ -th *MI* decoder layer, object queries learn multi-pattern information by interacting with  $E_j$ , where  $j = L - i + 1$ , and  $E_j$  is the corresponding image features after the processing of *U-like Feature Interaction*. The output of the last *MI* decoder layer (i.e.  $Q_L$ ) is used to predict the locations and categories of objects.

tion to the limitations of one-to-one supervision on convergence speed and performance. A few recent studies [6, 15, 28, 39, 47] have proposed some effective one-to-many matching strategies, allowing a GT to match with multiple object queries to accelerate training speed.

### 3. Method

#### 3.1. Preliminaries

**The framework of DETR-like models.** DETR-like models usually present the similar framework with the original DETR [3]. *First*, an image is inputted to the backbone to obtain multi-scale image features, which are flattened into feature tokens, and then concatenated to serve as the input of transformer encoder to refine image features. *Second*, a group of learnable object queries (object candidates are represented in the form of object queries [47]) interact with image features in transformer decoder to obtain the refined object queries. *Third*, the refined object queries are used to predict bounding boxes and categories of objects.

#### 3.2. U-like Feature Interaction

In existing DETR-like models, although the backbone and transformer encoder have effectively extracted image fea-

tures, these features have not been sufficiently utilized. Specifically, only the feature at the last layer of the transformer encoder is leveraged as Key&Value for transformer decoder. Actually, the features from each layer of transformer encoder contain valuable information at different levels. Inspired by the classical U-Net [32], we propose *U-like Feature Interaction (UFI)* to fully make use of features at different transformer encoder layers.

DETR is a typical encoder-decoder architecture. Overall, the learning of DETR is a process of encoding low-level detailed features to high-level abstract features, and then decoding them back into detailed representations. The encoder extracts image features layer by layer. With the number of encoder layers increasing, the extracted features become more global and abstract. The decoder continuously refines the object queries to gradually learn the local and detailed information. Therefore, the interaction between the corresponding layers of the encoder and decoder facilitates to utilize both low-level and high-level information.

The image features at encoder layers are defined as  $E = \{E_j, j = 1, 2, \dots, L\}$ , where  $L$  denotes the number of encoder layers. *UFI* uses the features at the  $j$ -th layer of the transformer encoder (i.e.,  $E_j$ ) as Key&Value for the  $i$ -th decoder layer, where  $j = L - i + 1$ . In detail, *UFI* first fuses

the features at the last layer of the transformer encoder ( $E_L$ ) with the features at other layers  $\{E_1, E_2, \dots, E_{L-1}\}$ ,

$$E_j = \begin{cases} \text{linear}(\text{concat}(E_j, E_L)) & j=1, 2, \dots, L-1 \\ E_L & j=L \end{cases}, \quad (1)$$

and then uses these fused features as the Key&Value of the corresponding decoder layers. The  $i$ -th decoder layer could be formulated as follows:

$$Q_i = \mathcal{D}_i(Q_{i-1}, E_j), \quad j=L-i+1, \quad (2)$$

where  $\mathcal{D}_i$  represents  $i$ -th decoder layer.

### 3.3. Multi-time Inquiries Mechanism

The key idea of parallel **Multi-time Inquiries (MI)** mechanism is to make object queries perform multiple interactions with image features to improve feature utilization. Specifically, we apply **MI** mechanism to original transformer decoder layers, yielding **MI Decoder**. Each layer can be divided into two parts: multi-time inquiries and query fusion.

**Multi-time inquiries.** As shown in Fig. 2, the inputs of the  $i$ -th **MI** decoder layer are the object queries  $Q_{i-1}$  (i.e., the output of  $(i-1)$ -th **MI** decoder layer) and the corresponding image features  $E_j$  defined in Eq. (1). We process the object queries  $Q_{i-1}$  with  $M$  different inquiry heads. Same with the network architecture of traditional transformer decoder layer, each inquiry head is composed of a self-attention layer, a cross-attention layer, and a FFN layer, which is formalized as follows:

$$Q_i^k = \text{FFN}_i^k(\text{CrossAtt}_i^k(\text{SelfAtt}_i^k(Q_{i-1}), E_j)), \quad (3)$$

$$i=1, 2, \dots, L; k=1, 2, \dots, M,$$

where  $Q_i^k$  denotes the object queries outputted from  $k$ -th inquiry head at  $i$ -th **MI** decoder layer.  $\text{SelfAtt}_i^k$ ,  $\text{CrossAtt}_i^k$ , and  $\text{FFN}_i^k$  represent the self-attention layer, cross-attention layer, and FFN layer in  $k$ -th inquiry head of  $i$ -th **MI** decoder layer.  $L$  is the number of decoder layers.

**Query fusion.** Based on Eq. (3),  $M$  groups of object queries  $\{Q_i^1, Q_i^2, \dots, Q_i^M\}$  are computed. Different groups of object queries convey different patterns of information. Therefore, it is necessary to conduct fusion to make them to be mutually cooperative and beneficial. We adopt the classic concatenation fusion, which first concatenates multiple groups of object queries along the feature dimension and then projects the concatenated feature to the original dimensions.

$$Q_i = \text{Linear}(\text{Concat}(Q_i^1, Q_i^2, \dots, Q_i^M)), \quad (4)$$

where  $\text{Concat}$  denotes concatenation operation and  $\text{Linear}$  represents a linear layer.

### 3.4. Lite Multi-time Inquiries

**Lite Multi-time Inquiries (Lite-MI)** is the lightweight version of **MI**. As shown in Fig. 3, the parameters of self-attention layers in different inquiry heads are shared, which

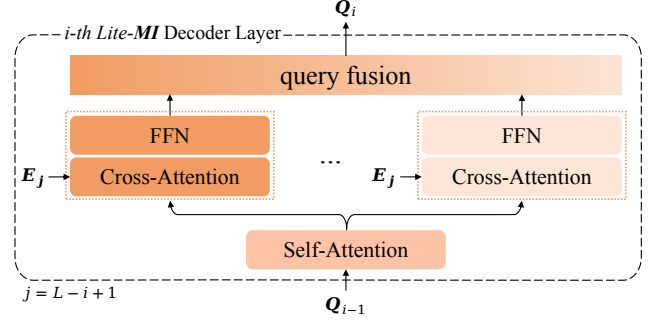


Figure 3. The architecture of *Lite-MI*.

is formulated as follows:

$$Q_i^k = \text{FFN}_i^k(\text{CrossAtt}_i^k(\text{SelfAtt}_i(Q_{i-1}), E_j)), \quad (5)$$

where  $\text{SelfAtt}_i$  represents the shared self-attention layer.

The motivations of *Lite-MI* are two fold. *First*, as studied in [13], the main function of self-attention is to eliminate duplicate candidates. In addition, image features  $E_j$  are not involved in the self-attention layer. Therefore, configuring separate self-attention layers in all inquiry heads might be redundant. *Second*, it is able to reduce parameters.

## 4. Experiments

### 4.1. Settings

We conduct experiments on COCO [18] dataset. Following the common practice, COCO train2017 split (118k images) is for training and COCO val2017 split (5k images) is for validation. AdamW [25] is used for optimization, with the learning rate of  $1 \times 10^{-4}$  and weight decay of  $1 \times 10^{-4}$ . We report the experimental results under the 1x (12 epochs) and 2x (24 epochs) training schedules with two commonly-used backbones, including ResNet-50 [10] pretrained on ImageNet-1k [33] and Swin-L [24] pretrained on ImageNet-22k. The metric is the standard mean Average Precision (AP) under different IoU thresholds and object scales. RTX3090 GPUs are used when the backbone is ResNet-50 and A100 GPUs are used when the backbone is Swin-L. The number of inquiry heads is 4.

### 4.2. Comparison Experiments

For comprehensive comparison, our model is compared with a series of DETR variants [2, 4, 6, 12, 13, 15, 21, 29, 44, 45, 47] that are proposed since 2023, under different training schedules and backbones. The comparison results are summarized in Tab. 1.

**Comparison under ResNet-50.** With the rapid development of DETR variants, the convergence speed has significantly accelerated. Most methods can achieve convergence within 24 epochs, and a few methods (e.g., DINO [44], Rank-DETR [29],  $\mathcal{H}$ -DETR [15], and Group-DETR [6]) require 36 epochs to reach convergence. Therefore, we report

| Method                   | Pub.Year  | Backbone  | Epochs | AP          | $AP_{50}$   | $AP_{75}$   | $AP_S$      | $AP_M$      | $AP_L$      |
|--------------------------|-----------|-----------|--------|-------------|-------------|-------------|-------------|-------------|-------------|
| DINO [44]                | ICLR.2023 | ResNet-50 | 12     | 49.0        | 66.6        | 53.5        | 32.0        | 52.3        | 63.0        |
| Stable-DINO [21]         | ICCV.2023 | ResNet-50 | 12     | 50.4        | 67.4        | 55.0        | 32.9        | 54.0        | 65.5        |
| Rank-DETR [29]           | NIPS.2023 | ResNet-50 | 12     | 50.4        | 67.9        | 55.2        | 33.6        | 53.8        | 64.2        |
| Align-DETR [2]           | BMVC.2024 | ResNet-50 | 12     | 50.2        | 67.8        | 54.4        | 32.9        | 53.3        | 65.0        |
| DDQ-DETR [45]            | CVPR.2023 | ResNet-50 | 12     | 51.3        | 68.6        | 56.4        | 33.5        | 54.9        | 65.9        |
| DAC-DETR [13]            | NIPS.2023 | ResNet-50 | 12     | 50.0        | 67.6        | 54.7        | 32.9        | 53.1        | 64.2        |
| $\mathcal{H}$ -DETR [15] | CVPR.2023 | ResNet-50 | 12     | 48.7        | 66.4        | 52.9        | 31.2        | 51.5        | 63.5        |
| Group-DETR [6]           | ICCV.2023 | ResNet-50 | 12     | 49.8        | -           | -           | 32.4        | 53.0        | 64.2        |
| MS-DETR [47]             | CVPR.2024 | ResNet-50 | 12     | 50.3        | 67.4        | 55.1        | 32.7        | 54.0        | 64.6        |
| Relation-DETR [12]       | ECCV.2024 | ResNet-50 | 12     | 51.7        | 69.1        | 56.3        | 36.1        | 55.6        | 66.1        |
| Ours                     | -         | ResNet-50 | 12     | <b>52.4</b> | <b>69.8</b> | <b>57.0</b> | 35.6        | <b>56.1</b> | <b>67.2</b> |
| <hr/>                    |           |           |        |             |             |             |             |             |             |
| DINO [44]                | ICLR.2023 | ResNet-50 | 36     | 50.9        | 69.0        | 55.3        | 34.6        | 54.1        | 64.6        |
| Stable-DINO [21]         | ICCV.2023 | ResNet-50 | 24     | 51.5        | 68.5        | 56.3        | 35.2        | 54.7        | 66.5        |
| Rank-DETR [29]           | NIPS.2023 | ResNet-50 | 36     | 51.2        | 68.9        | 56.2        | 34.5        | 54.9        | 64.9        |
| SQR-DETR [49]            | CVPR.2023 | ResNet-50 | 36     | 48.9        | 67.5        | 53.2        | 32.0        | 51.8        | 63.7        |
| $\mathcal{H}$ -DETR [15] | CVPR.2023 | ResNet-50 | 36     | 50.0        | 68.3        | 54.4        | 32.9        | 52.7        | 65.0        |
| Group-DETR [6]           | ICCV.2023 | ResNet-50 | 36     | 51.3        | -           | -           | 34.7        | 54.5        | 65.3        |
| Align-DETR [2]           | BMVC.2024 | ResNet-50 | 24     | 51.3        | 68.2        | 56.1        | 35.5        | 55.1        | 65.6        |
| DDQ-DETR [45]            | CVPR.2023 | ResNet-50 | 24     | 52.0        | 69.5        | 56.2        | 35.2        | 54.9        | 65.9        |
| DAC-DETR [13]            | NIPS.2023 | ResNet-50 | 24     | 51.2        | 68.9        | 56.0        | 34.0        | 54.6        | 65.4        |
| MS-DETR [47]             | CVPR.2024 | ResNet-50 | 24     | 51.7        | 68.7        | 56.5        | 34.0        | 55.4        | 65.5        |
| Relation-DETR [12]       | ECCV.2024 | ResNet-50 | 24     | 52.1        | 69.7        | 56.6        | 36.1        | 56.0        | 66.5        |
| Ours                     | -         | ResNet-50 | 24     | <b>52.7</b> | <b>70.4</b> | <b>57.2</b> | <b>36.7</b> | <b>56.7</b> | <b>66.7</b> |
| <hr/>                    |           |           |        |             |             |             |             |             |             |
| DINO [44]                | ICLR.2023 | Swin-L    | 12     | 56.8        | 75.6        | 62.0        | 40.0        | 60.5        | 73.2        |
| DAC-DETR [13]            | CVPR.2023 | Swin-L    | 12     | 57.3        | 75.7        | 62.7        | 40.1        | 61.5        | 74.4        |
| Rank-DETR [29]           | NIPS.2023 | Swin-L    | 12     | 57.6        | 76.0        | 63.4        | 41.6        | 61.4        | 73.8        |
| $\mathcal{H}$ -DETR [15] | CVPR.2023 | Swin-L    | 12     | 56.1        | 75.2        | 61.3        | 39.3        | 60.4        | 72.4        |
| Stable-DINO [21]         | ICCV.2023 | Swin-L    | 12     | 57.7        | 75.7        | 63.4        | 39.8        | 62.0        | 74.7        |
| Relation-DETR [12]       | ECCV.2024 | Swin-L    | 12     | 57.8        | 76.1        | 62.9        | 41.2        | 62.1        | 74.4        |
| Ours                     | -         | Swin-L    | 12     | <b>58.2</b> | <b>76.5</b> | <b>63.4</b> | <b>42.5</b> | <b>62.8</b> | <b>74.6</b> |

Table 1. Comparison experiments under different backbones and training epochs.

experimental results under 12 training epochs (commonly adopted in existing methods) and the condition achieving convergence (*i.e.*, 24 or 36 epochs, depending on the model itself). As shown in Tab. 1, our method demonstrates the advantage under different training conditions. Compared to the second-best method (*i.e.*, Relation-DETR [12]), our method achieves the improvements of +0.7 AP and +0.6 AP with 12 and 24 training epochs, respectively. *Without bells and whistles, our method establishes the best result to date, achieving 52.7 AP without employing extra tricks.*

**Comparison under Swin-L.** As a stronger backbone, Swin-L backbone has demonstrated its excellent performance. However, few methods report the results under Swin-L backbone. To further verify the effectiveness and robustness of our method, the experiments are conducted using Swin-L backbone under the training schedule of 12 epochs, and the results are reported in Tab. 1, from which we can observe that *our method still achieves the best results, obtaining +0.4 AP compared with the existing best-performing model Relation-DETR.*

### 4.3. Diagnostic Experiments

#### 4.3.1. The ablation testing on main components

We conduct a series of diagnostic experiments on main components, including *MI*, *Lite-MI*, and *UFI*. The results

| ID | <i>MI</i> | <i>Lite-MI</i> | <i>UFI</i> | AP   | $AP_{50}$ | $AP_{75}$ |
|----|-----------|----------------|------------|------|-----------|-----------|
| #1 | -         | -              | -          | 49.0 | 66.6      | 53.5      |
| #2 | ✓         |                |            | 49.8 | 67.5      | 54.3      |
| #3 |           | ✓              |            | 49.6 | 67.2      | 54.1      |
| #4 |           |                | ✓          | 49.5 | 67.3      | 54.0      |
| #5 |           | ✓              | ✓          | 50.1 | 67.9      | 54.7      |
| #6 | ✓         |                | ✓          | 50.2 | 68.1      | 54.8      |

Table 2. Diagnostic experiments on *MI* (*Multi-time Inquiries*), *Lite-MI* (*Lite Multi-time Inquiries*), and *UFI* (*U-like Feature Interaction*). The experiments are conducted based on DINO baseline (#1), and we adopt ResNet-50 backbone and 1x (12 epochs) training schedule. The results in Tab. 4, Tab. 5, and Tab. 6 are also obtained under the same experiment settings.

are summarized in Tab. 2, from which we can observe that all components exhibit improvements over baseline (+0.8 AP, +0.6 AP, and +0.5 AP, respectively), demonstrating that each component is effective (comparing #2, #3, and #4 with #1). The combinations “*MI+UFI*” (#5) and “*Lite-MI+UFI*” (#6) represent two kinds of structures of our model. “*MI+UFI*” presents the best result, indicating the effectiveness of our architecture. *Lite-MI* reduces the model complexity by sharing the self-attention layer. Compared to “*MI+UFI*”, “*Lite-MI+UFI*” asks for less parameters at the cost of slight performance degradation.

| Method             | Backbone  | Epochs | AP                | $AP_{50}$         | $AP_{75}$         | $AP_S$            | $AP_M$            | $AP_L$            |
|--------------------|-----------|--------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
| DINO [44]          | ResNet-50 | 12     | 49.0              | 66.6              | 53.5              | 32.0              | 52.3              | 63.0              |
| ours               | ResNet-50 | 12     | <b>50.2(+1.2)</b> | <b>68.1(+1.5)</b> | <b>54.8(+1.3)</b> | <b>33.4(+1.4)</b> | <b>53.6(+1.3)</b> | <b>64.5(+1.5)</b> |
| DINO [44]          | ResNet-50 | 24     | 50.4              | 68.3              | 54.8              | 33.3              | 53.7              | 64.8              |
| ours               | ResNet-50 | 24     | <b>51.2(+0.8)</b> | <b>69.2(+0.9)</b> | <b>55.8(+1.0)</b> | <b>34.6(+1.3)</b> | <b>54.4(+0.7)</b> | <b>65.5(+0.7)</b> |
| DINO [44]          | Swin-L    | 12     | 56.8              | 75.6              | 62.0              | 40.0              | 60.5              | 73.2              |
| ours               | Swin-L    | 12     | <b>57.5(+0.7)</b> | <b>76.3(+0.7)</b> | <b>63.1(+1.1)</b> | <b>41.1(+1.1)</b> | <b>61.5(+1.0)</b> | <b>73.9(+0.7)</b> |
| Relation-DETR [12] | ResNet-50 | 12     | 51.7              | 69.1              | 56.3              | 36.1              | 55.6              | 66.1              |
| ours               | ResNet-50 | 12     | <b>52.4(+0.7)</b> | <b>69.8(+0.7)</b> | <b>57.0(+0.7)</b> | 35.6              | <b>56.1(+0.5)</b> | <b>67.2(+1.1)</b> |
| Relation-DETR [12] | ResNet-50 | 24     | 52.1              | 69.7              | 56.6              | 36.1              | 56.0              | 66.5              |
| ours               | ResNet-50 | 24     | <b>52.7(+0.6)</b> | <b>70.4(+0.7)</b> | <b>57.2(+0.6)</b> | <b>36.7(+0.6)</b> | <b>56.7(+0.7)</b> | <b>66.7(+0.2)</b> |
| Relation-DETR [12] | Swin-L    | 12     | 57.8              | 76.1              | 62.9              | 41.2              | 62.1              | 74.4              |
| ours               | Swin-L    | 12     | <b>58.2(+0.4)</b> | <b>76.5(+0.4)</b> | <b>63.4(+0.5)</b> | <b>42.5(+1.3)</b> | <b>62.8(+0.7)</b> | <b>74.6(+0.2)</b> |

Table 3. Diagnostic experiments of  $MI$  on representative models.

#### 4.3.2. The effects of $MI$ on other models

$MI$  is the core mechanism of our model. Thanks to its simplification,  $MI$  could be friendly plugged into existing DETR-like models. To verify the effectiveness, we examine the performance improvements after plugging  $MI$  into DINO [44] and Relation-DETR [12]. DINO is the most representative model that is widely known in the object detection domain, and Relation-DETR is the newest model presenting SOTA performance. Therefore, we select them as baselines. To avoid the randomness of experiment results, we conduct the experiments under different backbones and training schedules, and the results are presented in Tab. 3.

We can observe from Tab. 3 that: 1) Our method achieves consistent improvements over baselines. Especially, based on the strongest baseline Relation-DETR leveraging one-to-many matching in the decoder, our method can still obtain the performance improvement, surpassing Relation-DETR by +0.7 AP (12 epochs) and +0.6 AP (24 epochs). This demonstrates  $MI$  can effectively strengthen the representations of object queries, even though the object queries have been strengthened by the one-to-many matching mechanism; 2) Plugging  $MI$  into Relation-DETR under the training schedule of 12 epochs obtains better performance than that of Relation-DETR trained with 24 epochs (52.4 AP v.s. 52.1 AP), demonstrating our method can accelerate the training convergence.

#### 4.3.3. Inquiry head number of $MI$

We are interested in how will the number of inquiry heads in  $MI$  decoder layers affect the final performance. Intuitively, the performance will boost with the increasing of inquiry heads, since multi-time inquiries enable the object queries to learn multi-pattern information. With the further increasing of inquiry heads, the performance should drop, since too many patterns of information might be disturbing or even antagonistic. Therefore, we conduct the experiments to examine the effectiveness of inquiry head number on model performance, and the results are available in Tab. 4. When inquiry head number increases from 1 to 4, the performance

improves by +0.7 AP, validating the effectiveness of multiple inquiry heads. However, the performance drops when IHN=5, potentially proving our guess that too many patterns of information might be disturbing or even antagonistic.

| IHN | 1    | 2    | 3    | 4           | 5    |
|-----|------|------|------|-------------|------|
| AP  | 49.5 | 49.6 | 49.9 | <b>50.2</b> | 49.9 |

Table 4. Diagnostic experiments on the number of inquiry heads. ‘‘IHN’’ is short for ‘‘Inquiry Head Number’’.

| Method | IHN | Heads        | AP   | $AP_{50}$ | $AP_{75}$ |
|--------|-----|--------------|------|-----------|-----------|
| DINO   | -   | -            | 49.0 | 66.6      | 53.5      |
| Ours   | 1   | head 1       | 41.0 | 56.7      | 44.8      |
|        |     | head 2       | 42.2 | 58.1      | 45.9      |
|        |     | head 3       | 41.8 | 57.8      | 45.6      |
|        |     | head 4       | 40.4 | 54.9      | 44.0      |
|        | 2   | head 1&2     | 43.8 | 60.4      | 47.7      |
|        |     | head 2&4     | 48.0 | 65.2      | 52.5      |
|        | 3   | head 1&2&3   | 49.7 | 67.5      | 54.4      |
|        |     | head 2&3&4   | 49.4 | 67.2      | 54.0      |
|        | 4   | head 1&2&3&4 | 50.2 | 68.1      | 54.8      |

Table 5. Diagnostic experiments on inquiry head combinations.

#### 4.3.4. Inquiry head combinations of $MI$

The above experiments have validated the effectiveness of  $MI$  and examined the influence of inquiry head number of  $MI$ . To explain why  $MI$  is effective, we further conduct the experiments under different inquiry head combinations. Based on the trained model #6 in Tab. 2, we test the performance when enabling each inquiry head or multiple inquiry heads, and the results are reported in Tab. 5. Specifically, when enabling a certain inquiry head, the object queries from  $k$ -th ( $k=1,2,3$ , or 4) inquiry head are used to predict final results. When enabling multiple inquiry heads (e.g., head 1&2), the object queries from these inquiry heads are fused to predict final results. When inquiry head number is 2 or 3, there are many inquiry head combinations, thus we present the results of two arbitrary combinations.

Overall, the performance of model configured with one inquiry head lags behind by significant margins, and fusing information in all inquiry heads achieves the best performance. Specifically, with the increasing of inquiry heads, the performance are gradually boosting. The reason is straightforward that one-time inquiry only enables object queries to learn the relatively limited information, which is similar to that each convolution kernel in CNNs only extracts the certain channel of feature. Multi-time inquiries allow the model to fuse multi-pattern information learnt from different inquiry heads, though these information might be highly-collaborate (e.g., head 2 and 4) or weakly-collaborate (e.g., head 1 and 2).

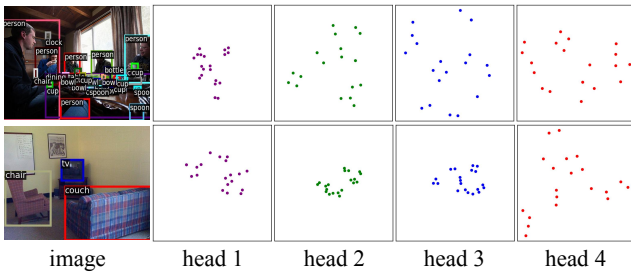


Figure 4. The visualization of object queries in different inquiry heads by T-SNE data visualization tool. More results of Fig. 4, Fig. 5, and Fig. 6 can be found in supplementary material.

### 4.3.5. Visualization analysis

The experiments in Tab. 4 and Tab. 5 have indicated that different inquiry heads are able to learn mutually collaborate information to improve object detection performance when setting reasonable inquiry head number. For deeper analysis, some visualization experiments are also conducted. **Experiment 1 - object queries visualization:** Object detection is performed on object queries at the last decoder layer, thus visualizing object queries at the last decoder layer assists to distinguish what kind of information is finally learnt in individual inquiry head. Therefore, we make use of the T-SNE tool to visualize the distributions of object queries at the last decoder layer. Here, we would like to explain an

object query, unlike CNN feature map, is a vector whose elements do not have the corresponding relations to image pixels, thus T-SNE visualization is adopted. Considering the majority of object queries represent the background and 93.51% of images in COCO contain less than 20 objects, top-20 object queries are visualized in Fig. 4. **Experiment 2 - object detection results visualization:** If object detection results based on individual inquiry head could be visualized, the mutually collaborate mechanism of *MI* could be straightly observed. Therefore, we visualize the object detection results using the single inquiry head and all inquiry heads as shown in Fig. 5.

We can observe from Fig. 4 that object queries in different inquiry heads generally present distinct distributions, some are gathering and others are dispersing. Reflecting on detection results, different inquiry heads tend to focus on different kinds of information. For example, as shown in Fig. 5, the head 2 seems to focus on big objects and head 4 intends to pay attention on small objects. At the same time, some inquiry heads might focus on the similar pattern of information, *since sharing the information in the certain extent is the basis of collaboration*. For example, on the first example in Fig. 5, both head 1 and head 3 seem to struggle detecting objects in all regions of the image. This might partly explain the results in Tab. 4 that continuously increasing inquiry heads will not constantly improve performance. *However, we note that, due to the diversity of images and “black box” characteristic of neural networks, it is challenging to clarify what specific patterns of information are learned through different inquiry heads, and it is also challenging to reflect the patterns constantly on diverse images.*

We can also observe from Fig. 5 that inquiry heads are mutually collaborate. For example, on the first example, the “clock” is not detected in head 2, 3, and 4, and the “cup” in the hand of the leftest person is not detected in head 1, 2, 3, and 4, but these two objects are detected in head 1-4 and the confidences are improved. Thanks to the collaboration of multiple inquiry heads, our model exhibits stronger ability in detecting challenging objects. For example, as shown in Fig. 6a, the occluded “teddy bear” is successfully detected,

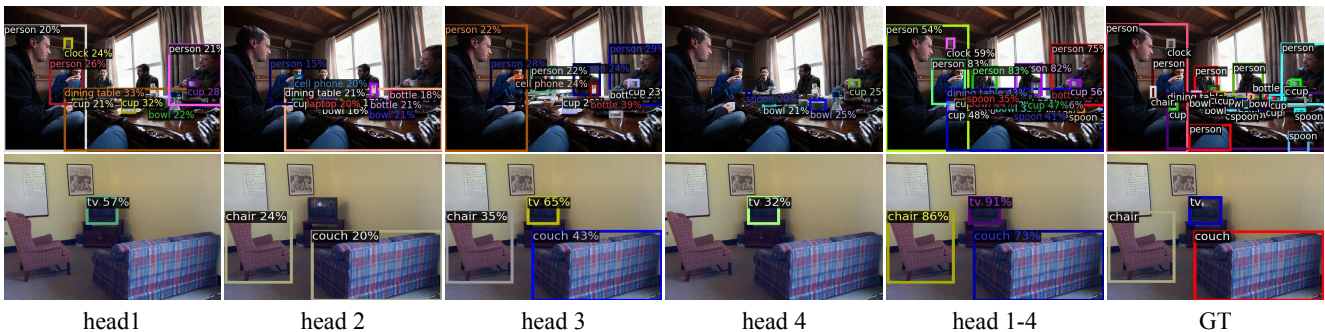


Figure 5. Object detection results based on the single inquiry head and multiple inquiry heads.

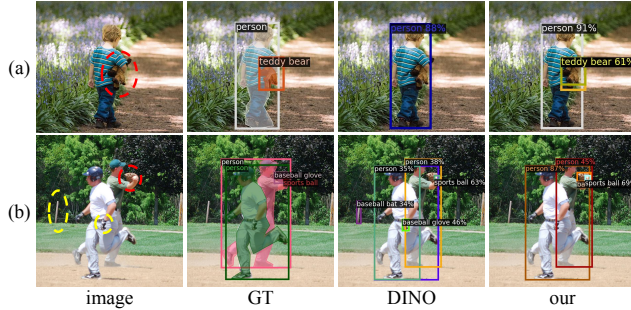


Figure 6. Qualitative comparison between DINO and our method. The red/yellow dashed circle indicates missing/false detection.

while the baseline fails. Some false detections could also be avoided, an example is illustrated in Fig. 6b.

#### 4.4. The Analysis of Complexity

The proposed *MI* is implemented by parallelly adding decoder layers, which easily induce the misunderstanding that the performance improvement might result from increasing the parameters complexity. Therefore, to eliminate the misunderstanding, the models with different numbers of decoder layer and inquiry heads are tested, and the results are summarized in Tab. 6. Note that we do not add *UFI* (with 1M parameters) in the experiments to guarantee the fair comparison with the baseline and eliminate its disturbance on the testing.

As shown in Tab. 6, we can observe that increasing the number of layers does not yield a performance improvement. Instead, a large amount parameters and computation are generated, even leading to performance degradation. The potential reason is as follows. Due to the cascaded architecture of decoder layers, the representations update of object queries are constrained in the cascaded direction. When decoder layers are further increasing, the representations might be redundant or even negative, leading to performance degradation.

Compared to DINO with 12/24 decoder layers, our method with 6 decoder layers and 2/4 inquiry heads asks for less parameters and computation, and achieves better performance (49.5 v.s. 49.0 / 49.8 v.s. 47.3), which demonstrates our proposed *MI* (rather than increasing computation complexity) contributes to the performance improvement.

### 5. Discussions

**Cascade Multi-time Inquiries v.s. Parallel Multi-time Inquiries.** The decoder layers of existing DETR-like models are actually performing cascaded multi-time inquiries. In contrast, our proposed *MI* decoder layers adopt parallel multi-time inquiries. We would like to discuss the differences between them.

| Method | LN | IHN | AP   | GFLOPS | Params |
|--------|----|-----|------|--------|--------|
| DINO   | 6  | 1   | 49.0 | 245    | 47M    |
| DINO   | 12 | 1   | 49.0 | 265    | 58M    |
| Ours   | 6  | 2   | 49.5 | 263    | 57M    |
| DINO   | 24 | 1   | 47.3 | 301    | 78M    |
| Ours   | 6  | 4   | 49.8 | 299    | 75M    |

Table 6. Comparison on performance and complexity with the baseline. “LN” and “IHN” are short for “Layer Number” and “Inquiry Head Number”, respectively.

The essence of an inquiry head is object queries update their representations by interacting with image features. As discussed in the introduction (Sec. 1), one-time inquiry is like that the student asks a question to the teacher, and obtains information about the image based on the teacher’s answer. For cascade multi-time inquiries, the student asks an initial question in the first inquiry head, and only asks the questions *that are related with the initial question* in the following inquiry heads. Differently, parallel multi-time inquiries allow the student to simultaneously ask multiple questions from different views in the first decoder layer and constantly receive the gradually fine-grained answers for these questions in the following decoder layers.

**Real Parallel v.s. Pseudo Parallel.** The number of object queries is tremendously larger than that of object GTs in an image. Therefore, the matching between GTs and object queries is an important issue that existing works are struggling to solve. The mainstream idea is setting up object queries for one-to-many matching [6, 15, 47]. However, to avoid NMS [27] post-processing operation, the object queries for one-to-one matching are also needed. As a result, the object queries for one-to-one matching and the object queries for one-to-many matching are parallelly existing. However, as explained in the introduction (Sec. 1), this parameters-sharing parallel architecture is a type of pseudo parallel. In contrast, our proposed parameters-dependent parallel architecture is real parallel, enabling to learn multi-pattern information. In addition, the query fusion mechanism makes multi-pattern information to be mutually collaborate and complementary.

### 6. Conclusion

This paper introduces an innovative decoder architecture, the core of which is the parallel *Multi-time Inquiries (MI)* mechanism. This mechanism presents the following advantages: **1)** The mechanism improves feature utilization since it enables object queries to learn multi-pattern information from image features; **2)** The mechanism is simple and friendly to plug into existing DETR-like models; **3)** The information learnt in different inquiry heads are mutually collaborate and complementary.

## Acknowledgments

This work was supported by the National Key R&D Program of China under Grant 2022YFB3103500 and in part by Chongqing Science and Health Joint Medical Major Research Project under Grant 2024DBXM003.

## References

- [1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1483–1498, 2019. 1
- [2] Zhi Cai, Songtao Liu, Guodong Wang, Zheng Ge, Xiangyu Zhang, and Di Huang. Align-detr: Improving detr with simple iou-aware bce loss. *arXiv preprint arXiv:2304.07527*, 2023. 1, 4, 5
- [3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the IEEE/CVF European Conference on Computer Vision*, pages 213–229, 2020. 1, 2, 3
- [4] Fangyi Chen, Han Zhang, Kai Hu, Yu-Kai Huang, Chenchen Zhu, and Marios Savvides. Enhanced training of query-based object detection via selective query recollection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23756–23765, 2023. 1, 4, 5
- [5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiao-xiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 1
- [6] Qiang Chen, Xiaokang Chen, Jian Wang, Shan Zhang, Kun Yao, Haocheng Feng, Junyu Han, Errui Ding, Gang Zeng, and Jingdong Wang. Group detr: Fast detr training with group-wise one-to-many assignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6633–6642, 2023. 1, 2, 3, 4, 5, 8
- [7] Xiyang Dai, Yinpeng Chen, Jianwei Yang, Pengchuan Zhang, Lu Yuan, and Lei Zhang. Dynamic detr: End-to-end object detection with dynamic attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2988–2997, 2021. 2
- [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. 2
- [9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 1
- [10] 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. 4
- [11] Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, and Badong Chen. Saliency detr: Enhancing detection transformer with hierarchical saliency filtering refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17574–17583, 2024. 1, 2
- [12] Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, Badong Chen, and Xuguang Lan. Relation detr: Exploring explicit position relation prior for object detection. *arXiv preprint arXiv:2407.11699*, 2024. 2, 4, 5, 6, 1
- [13] Zhengdong Hu, Yifan Sun, Jingdong Wang, and Yi Yang. Dac-detr: Divide the attention layers and conquer. *Proceedings of the Neural Information Processing Systems*, 36, 2024. 1, 4, 5
- [14] Yi-Xin Huang, Hou-I Liu, Hong-Han Shuai, and Wen-Huang Cheng. Dq-detr: Detr with dynamic query for tiny object detection. In *Proceedings of the IEEE/CVF European Conference on Computer Vision*, pages 290–305, 2025. 2
- [15] Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detsr with hybrid matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19702–19712, 2023. 1, 2, 3, 4, 5, 8
- [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
- [17] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6054–6063, 2019. 2
- [18] 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 *Proceedings of the IEEE/CVF European Conference on Computer Vision*, pages 740–755, 2014. 4
- [19] 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. 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. In *Proceedings of the International Conference on Learning Representations*, 2022. 1, 2
- [21] Shilong Liu, Tianhe Ren, Jiayu Chen, Zhaoyang Zeng, Hao Zhang, Feng Li, Hongyang Li, Jun Huang, Hang Su, Jun Zhu, et al. Detection transformer with stable matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6491–6500, 2023. 1, 4, 5
- [22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the IEEE/CVF European Conference on Computer Vision*, pages 21–37, 2016. 1

- [23] Wenze Liu, Hao Lu, Yuliang Liu, and Zhiguo Cao. Box-detr: Understanding and boxing conditional spatial queries. *arXiv preprint arXiv:2307.08353*, 2023. 1, 2
- [24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 4
- [25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, 2019. 4
- [26] 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
- [27] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *Proceedings of the International Conference on Pattern Recognition*, pages 850–855, 2006. 8
- [28] Jeffrey Ouyang-Zhang, Jang Hyun Cho, Xingyi Zhou, and Philipp Krähenbühl. Nms strikes back. *arXiv preprint arXiv:2212.06137*, 2022. 3
- [29] Yifan Pu, Weicong Liang, Yiduo Hao, Yuhui Yuan, Yukang Yang, Chao Zhang, Han Hu, and Gao Huang. Rank-detr for high quality object detection. *Proceedings of the Neural Information Processing Systems*, 36, 2024. 4, 5
- [30] 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. 1
- [31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the Neural Information Processing Systems*, 2015. 1
- [32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015. 3
- [33] 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. 4
- [34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 2
- [35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
- [36] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, 2017. 2
- [37] 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
- [38] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages I–I, 2001. 1
- [39] Wen Wang, Jing Zhang, Yang Cao, Yongliang Shen, and Dacheng Tao. Towards data-efficient detection transformers. In *Proceedings of the IEEE/CVF European Conference on Computer Vision*, pages 88–105, 2022. 3
- [40] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun. Anchor detr: Query design for transformer-based detector. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 2567–2575, 2022. 1, 2
- [41] 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
- [42] Mingqiao Ye, Lei Ke, Siyuan Li, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Cascade-detr: Delving into high-quality universal object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6704–6714, 2023. 1, 2
- [43] Sergey Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 2
- [44] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In *Proceedings of the International Conference on Learning Representations*, 2023. 1, 2, 4, 5, 6
- [45] Shilong Zhang, Xinjiang Wang, Jiaqi Wang, Jiangmiao Pang, Chengqi Lyu, Wenwei Zhang, Ping Luo, and Kai Chen. Dense distinct query for end-to-end object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7329–7338, 2023. 1, 4, 5
- [46] Yingying Zhang, Chuangji Shi, Xin Guo, Jiangwei Lao, Jian Wang, Jiaotuan Wang, and Jingdong Chen. Enhancing detr variants through improved content query and similar query aggregation. *arXiv preprint arXiv:2405.03318*, 2024. 2
- [47] Chuyang Zhao, Yifan Sun, Wenhao Wang, Qiang Chen, Er-rui Ding, Yi Yang, and Jingdong Wang. Ms-detr: Efficient detr training with mixed supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17027–17036, 2024. 1, 3, 4, 5, 8
- [48] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Dets beat yolos on real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16965–16974, 2024. 2
- [49] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *Proceedings of the International Conference on Learning Representations*, 2020. 1, 2